TWO STUDIES ON THE APPLICATION OF LARGE-SCALE DECOMPOSITION ALGORITHMS IN MARITIME MANAGEMENT

YING YANG

PhD

The Hong Kong Polytechnic University

2026

The Hong Kong Polytechnic University

Department of Logistics and Maritime Studies

Two Studies on the Application of Large-Scale Decomposition Algorithms in Maritime Management

Ying Yang

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

November 2025

CERTIFICATE OF ORIGINALITY

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

_____ (Signed)

Ying Yang _____ (Name of student)

Abstract

In recent years, large-scale decomposition optimization algorithms have played an increasingly crucial role in maritime management, particularly in addressing complex and new challenges and enhancing the precision and effectiveness of decision-making. This thesis collectively demonstrates the efficacy of large-scale decomposition algorithms in the field of shipping management by facilitating optimal operational decisions, contributing to reduced operational costs, improved service quality, and increased competitiveness in the fast-evolving maritime industry.

The first study addresses the drone scheduling problem in shore-to-ship delivery (DSP-SSD) to amid growing interest in the integration of drones into maritime logistics. We introduce a mixed-integer programming model with time discretization that incorporates drone-related constraints, moving targets, and the need for multiple drone trips. While commercial solvers can handle this model in small-scale scenarios, we propose a tailored branch-and-price-and-cut (BPC) algorithm for larger and more complex cases. This algorithm integrates a drone-specific backward labeling algorithm, cutting planes, and acceleration methods to boost its effectiveness. Experiments show that the BPC algorithm substantially outperforms the commercial solvers in terms of solution quality and computational efficiency and that the inclusion of acceleration strategies in the algorithm enhances its performance. We also provide detailed sensitivity analyses of critical parameters of the model, such as the time discretization parameter and the number of ships, to gain insights into how our approach could be applied in real-world DSP-SSD operations.

The second study focuses on cruise fleet management, which demands innovative and effective management strategies due to its growing popularity and the diversification of consumer travel preferences. Our approach provides a cohesive framework for the cruise fleet deployment and itinerary schedule problem. We first propose an integer programming model based on a space-time network that captures the movement dynamics of cruises over a planning horizon. To solve this comprehensive model efficiently, we introduce a tailored Benders decomposition approach augmented by the simultaneous Magnanti-Wong method, where a valid and easily computed Magnanti-Wong bound is designed. We validate our approach using extensive numerical experiments on both simulation instances and a real case study. The results demonstrate the effectiveness of our integrated solving scheme and the practical applicability of our advanced decomposition method, marking a significant advancement in the field of cruise fleet management.

Keywords: maritime management; shore-to-ship delivery; drone scheduling; branch-and-price-and-cut; cruise fleet deployment and itinerary scheduling; Benders decomposition; simultaneous Magnanti-Wong method

Publications and working papers

Arising from the thesis

- Yang, Y., Hao, X., Wang, S. (2025). The drone scheduling problem in shore-to-ship delivery: A time discretization-based model with an exact solving approach. *Transportation Research Part B: Methodological*, 191, 103117.
- Yang, Y., Zhang, S., Wang, S. (2025). Integrated cruise fleet deployment and itinerary scheduling problem: An enhanced Benders decomposition approach. *Transportation Research Part B: Methodological*, 201, 103321.

Others

- Yang, Y., Wang, S., Laporte, G. (2025). Improved regression tree models using generalization error-based splitting criteria. *Naval Research Logistics*, 72(8), 1097-1116.
- Yang, Y., Liu, J., Wang, S.. (2025) A bi-objective optimization model for the drone scheduling problem in island delivery. *International Journal of Production Research*, 1-22.
- Yang, Y., Yan, R., Wang, S. (2024). An efficient ranking-based data-driven model for maritime transport optimization. *Transportation Research Part C: Emerging Technologies*, 165, 104731.

- Yang, Y., Yan, R., Wang, S. (2024). Prescriptive analytics models for vessel inspection planning in maritime transportation. *Computers & Industrial Engineering*, 190, 110012.

- Qi, M., Yang, Y., Cheng, C. (2023). Location and inventory pre-positioning problem under uncertainty. *Transportation Research Part E: Logistics and Transportation Review*, 177, 103236.

- Yang, Y., Wu, Z., Hao, X., Liu, H., Qi, M. (2023). Two-layer heuristic for the three-dimensional bin design and packing problem. *Engineering Optimization*, 1-38.

- Qiao, X, Yang, Y.*, Sun, Q., Wang, S.. A branch-and-price approach for optimizing all-electric ships scheduling in waterborne transport. Under major revision in *Transportation Research Part B: Methodological*.

- Jiang, B., Yang, Y., Tian X., Wang, S., Zhen L.. Refined regression trees: A generalizable framework for partitioning the feature space. Under review.

- Yang, Y., Hao, X., Zhang, S., Wang, S.. An exact price-and-cut-and-enumerate algorithm for joint vehicle routing and speed optimization problem. Working Paper.

- Hao, X., Yang, Y., Zhang, S., Xu, Z.. Unlocking the power of column enumeration for split delivery vehicle routing via strategic row aggregation. Working Paper.

- Yang, Y., Jiang, R., Wang, S.. Lagrangian index policies and performance bounds for prenatal care patient acceptance decision-making under uncertainty. Working Paper.

- Zeng, W., Yang, Y., Wang, S., Yang, H.. Cooperative drone scheduling problem in remote area patrol considering signal transmission. Working Paper.

Acknowledgements

Writing with emotion feels unfamiliar after years of prioritizing logic and precision. I suddenly realize that this doctoral journey, spanning less than three years, has truly reshaped me, changing a free-spirited girl into a rigorous researcher. I am grateful to every friend and mentor by my side who has turned a dry academic life into a journey filled with sweetness and wonder.

First and foremost, I extend my deepest gratitude to my supervisor, Professor Shuaian (Hans) Wang. He is the most energetic person I have ever met, like a blazing flame whose boundless energy and fighting spirit inspire everyone around him. I appreciate his rigorous guidance on my manuscripts, his full respect for my research interests, and his unwavering support for my future career. I hope to follow in his footsteps to become a mentor who is enthusiastic, dedicated, and caring.

I also want to thank my mentor during my visiting period at the University of Michigan, Professor Ruiwei Jiang. If Hans is the fire, Ruiwei is the gentle breeze. Every meeting with him gave me the confidence that any problem, whether in research or life, could be solved. Although we only spent a brief six months together, he was incredibly warm and spared no effort in supporting my research and future development. He taught me that a true scholar is both sharp in research and gentle in character.

I would also like to express my sincere thanks to my Master's supervisor, Professor Mingyao Qi. As a lifelong mentor, he continues to offer his utmost support for my devel-

opment and provides invaluable wisdom at every crossroad of my life. I also acknowledge my distinguished co-authors, Professor Gilbert Laporte from HEC Montréal, Dr. Ran Yan from Nanyang Technological University, and others, for their collaboration and valuable advice.

To my friends, thank you for keeping me company throughout my PhD journey. To the seniors and peers in Hans' group, the LMS, and other colleagues, especially Silong and Wenjia, thank you for your guidance and companionship in both life and research. To my "family" in Shenzhen and Hong Kong, I cherish the memory of every mountain we conquered and every delicacy we explored together. To my friends in Prof. Jiang's and Prof. Yin's groups at UMich, you were the unexpected surprise of my PhD journey, gifting me a special and wonderful memory in Ann Arbor.

To my family, although we were thousands of miles apart and met for only a few weeks each year, your silent support has always been my anchor. And to my baby cat, Rookie, thank you for always healing my heart with sweet meows and endless cuteness.

My last tribute is to my partner, Shelton, who knows I am not perfect but still loves me. Thank you for the brave decision to take my hand and walk this long path, as my love, my best friend, and my forever ally, ready to embrace or fight the world together.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The maritime logistics industry, traditionally the backbone of global trade, is undergoing a significant transformation driven by the need for digitalization, sustainability, service quality, and operational efficiency (Yan et al., 2021; Bai et al., 2021). Modern maritime problems are increasingly characterized by their high dimensionality and the integration of multi-level decisions. This complexity is vividly illustrated in two emerging operational contexts addressed in this thesis:

- **Shore-to-ship drone delivery**: Drones have emerged as a sustainable and safer alternative to conventional tender boats for shore-to-ship delivery (SSD) operations (Charles Alcock, 2024; Kumar and Devi, 2023; Muhammad and Gregersen, 2022). This shift is driven by the potential for a markedly smaller carbon footprint and a significant reduction in operational expenditure compared to conventional transport methods. However, unlike traditional vehicle routing, drone routing in this context introduces a unique challenge as customers (ships) are continuously moving along designated lanes. This requires precise synchronization between the aerial vehicle's flight path and the vessel's trajectory under stringent battery and payload

constraints. Furthermore, due to limited fleet sizes, drones must perform multiple trips within a planning horizon, adding another layer of combinatorial difficulty.

- **Integrated cruise operations**: In recent years, the cruise tourism sector has experienced a substantial surge in popularity, driven by growing consumer demand for unique and diverse travel experiences (Forbes, 2024). A critical challenge within this sector lies in the vertical integration of decision-making. Traditionally, tactical fleet deployment, i.e., allocating ships to specific cruise lines, and operational itinerary scheduling, i.e., determining arrival and departure times, are treated sequentially. However, this separation often leads to suboptimal outcomes. Consequently, an integrated approach is required to balance operational costs, such as fuel and deployment, against passenger utility derived from sightseeing time. The complexity of the integrated optimization model is driven by coupling medium-term tactical deployment decisions with micro-scheduling decisions across a vast time-expanded network.

Mathematically, these problems are typically formulated as large-scale mixed integer programming (MIP) or integer programming (IP) models. However, the complicated transport network and complex practical constraints often lead to a combinatorial explosion in the number of variables and constraints, rendering standard commercial solvers (e.g., CPLEX, Gurobi) intractable for realistic instances. Consequently, decomposition-based exact algorithms have become the standard for addressing these NP-hard problems.

1.2 Thesis outline

The thesis is structured as follows:

- **Study 1: Drone scheduling problem in shore-to-ship delivery**

Chapter 2 addresses the drone scheduling problem in shore-to-ship delivery (DSP-SSD), which involves unique challenges like moving targets (ships) and the necessity of multiple drone trips. We develop an MIP model based on a time-expanded network that characterizes the dynamics of moving ships. For practical, large-scale scenarios, we propose a tailored branch-and-price-and-cut (BPC) algorithm. This exact approach integrates a drone-specific backward labeling algorithm for column generation (CG) and specialized cutting planes to enhance efficiency and strengthen solution quality. Computational results confirm that the BPC algorithm substantially outperforms commercial solvers in both efficiency and solution quality, demonstrating its suitability for complex DSP-SSD operations.

- **Study 2: Integrated cruise fleet deployment and itinerary schedule problem**

  Chapter 3 focuses on the integrated optimization of tactical cruise fleet deployment and operational itinerary scheduling. The problem is modeled using an IP framework based on a time-expanded network. For the itinerary scheduling subproblem, we first establish the route-based reformulation and prove that it is totally unimodular (TU), allowing integer variables to be relaxed. Then, we introduce an enhanced Benders decomposition (BD) algorithm, incorporating the simultaneous Magnanti-Wong method. This method utilizes a problem-specific bound to efficiently generate Pareto-optimal cuts. Extensive experiments validate the integrated model and demonstrate that this advanced decomposition method significantly enhances computational efficiency and practical applicability.

Chapter 4 concludes the thesis by summarizing the main contributions and discussing future research directions.

Chapter 2

Study 1: Drone scheduling problem in shore-to-ship delivery

2.1 Introduction

Shore-to-ship delivery by supply or tender boat is a common practice in maritime logistics to transport essential items such as letters, food, and medicine to ships (Squires, 1986). Ships may need to call at non-cargo destinations for replenishment during the voyage. Nevertheless, due to factors such as adverse weather conditions, limited time, crowded traffic, and the ship's excessive size, it may be neither feasible nor economical to dock a ship at a port (Sulligoi et al., 2015; Service, 2023). In such circumstances, tender boats provide the essential connection between the ship and the shore. These deliveries, which provide both logistical support and emergency aid, are crucial for facilitating continuous operations at sea and reducing the need for ships to make port calls for resupply. However, the tender boat is fraught with inefficiencies that raise both safety and environmental concerns (Nguyen et al., 2021). In busy ports, the presence of numerous tender boats darting back and forth can exacerbate congestion, cause operational inefficiencies, and increase the risk of collisions or other accidents. The need to make multiple trips to deliver supplies

results in high fuel consumption, which increases the environmental footprint of maritime operations. Besides, there are mainly two modes of shore-to-ship delivery by tender boat: receiving supplies while the ship is fully anchored or receiving supplies while moving. However, halting a ship, anchoring or drifting, is sometimes impractical, especially in busy coastal areas. Anchoring incurs time and economic costs and is subject to many governmental restrictions (Davis et al., 2016). Drifting, on the other hand, is only feasible in very open waters as it increases the risk of collision (Fowler and Sørgård, 2000). Additionally, restarting and accelerating a ship consumes more energy, reduces stability, and complicates management for the crew compared to when the vessel is moving. Thus, maintaining a steady speed is generally more economical and practical for vessels when drones are conducting deliveries. These issues indicate the need for more safe and sustainable solutions for shore-to-ship delivery.

Recently, unmanned aerial vehicle (UAV) technology, or drones, has become a candidate solution to the longstanding logistical conundrum of shore-to-ship supply transfer (Charles Alcock, 2024; Kumar and Devi, 2023; Muhammad and Gregersen, 2022). The integration of drone technology into shore-to-ship operations promises a significant reduction in operational expenditure due to the lower battery consumption of drones compared with tender boats, and their lower labor costs due to automation. Drones can also travel on direct aerial routes, thus avoiding congested port traffic and complex docking procedures and their associated costs. From an environmental perspective, the deployment of drones for shore-to-ship delivery results in a markedly smaller carbon footprint than conventional modes of transport that are heavily reliant on fossil fuels. There has been some applications of drone-assisted shore-to-ship delivery in Singapore, where there are limited drone resources but many vessels requiring supplies (Airbus Helicopters, 2019).

Shore-to-ship delivery by drones offers great economic and environmental benefits. However, although much attention has been devoted to navigation rule design in this context (Asiimwe and Anvar, 2012; Thompson and Davies, 2021), the drone scheduling prob-

lem (DSP) has attracted much less consideration. We address the pivotal role of this problem in shore-to-ship delivery in ensuring delivery punctuality and reducing operational costs. An illustrative example of the DSP-SSD is given as follows.

Example 1. *Figure 2.1 shows the scenario of six ships sailing through a coastal area in need of supplies. Two drones are scheduled for the delivery task. Drone 1 performs two trips serving four ships, trip 1–1: {ship 1–ship 2} and trip 1–2: {ship 5–ship 6}. Drone 2 delivers supplies to ship 3 and then ship 4 in one trip. The drones depart from a drone station, deliver supplies in sequence, and then return to the drone station for battery replacement, all while the ships are moving along their designated courses. The deliveries are completed within the drone flight boundary, as constrained by the drones' carrying capacity and battery life.*

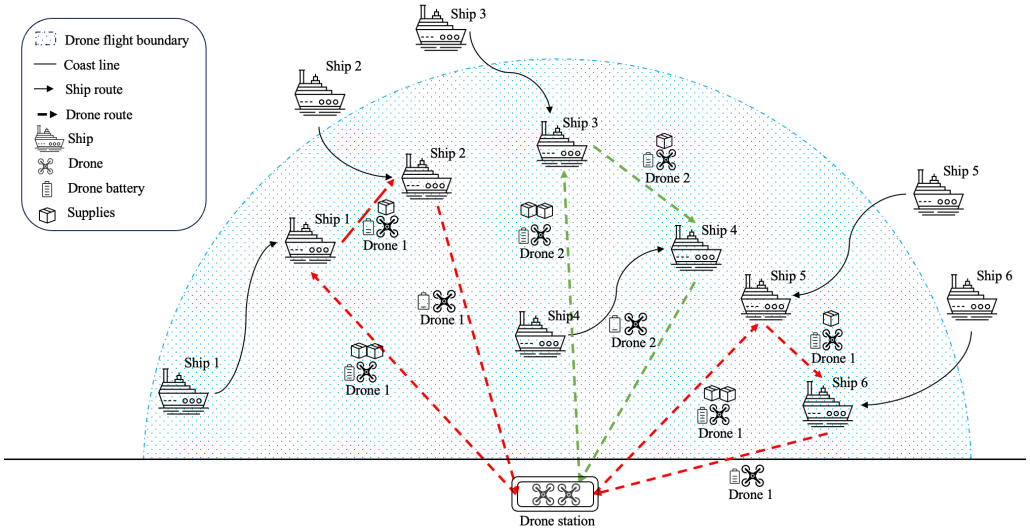


Figure 2.1: Drone scheduling in shore-to-ship delivery

Unlike the typical DSP, which is akin to the conventional vehicle routing problem (VRP), the DSP-SSD involves unique challenges. The complexity arises from two aspects. Firstly, customers in the DSP-SSD, i.e., the vessels being supplied, are not stationary but moving. Therefore, it requires novel modeling and solving schemes that can adapt

to moving targets. Furthermore, the limited number of drones available necessitates the efficient planning of multiple round trips to ensure that drone resources are used optimally to meet demand.

In response to these challenges, the goal of this paper is to provide a comprehensive solution to the DSP-SSD that is practical for real-world applications. Our work makes four main contributions:

- From a modeling perspective, we formulate a MIP model of the DSP-SSD that considers moving targets and multiple trips (Section 2.3). We tackle the challenge of moving targets by constructing a time-expanded network using time discretization. We then use this network to develop the MIP model, which enables drones to perform delivery tasks sequentially using multiple trips. This model can be directly fed into an off-the-shelf solver to provide an optimal solution for drone scheduling in small-scale scenarios.

- From an algorithmic perspective, we develop a BPC algorithm for the DSP-SSD for use at more practical scales than the basic MIP model (Section 2.4). In the BPC algorithm, the original model is reformulated into a route-based model and then solved by a branch-and-bound (B&B) tree in which each node represents a subproblem, and CG is used to add enhanced pricing and cutting plane constraints to tighten the linear relaxation. We propose several specialized acceleration strategies to the BPC algorithm based on the time-expanded network, including three types of tailored cutting planes, an efficient labeling algorithm, and an effective dominance rule, which is specialized under the time-expanded network.

- From a computational perspective, we demonstrate the ability of our MIP model and the BPC algorithm to generate high-quality solutions (Section 2.5.2). Various experiments reveal the superiority of our tailored BPC algorithm compared with an

off-the-shelf solver in handling a complex version of the DSP-SSD. The same comparative experiments also reveal the benefits of our node selection rule, branching strategy, and acceleration strategies.

- From a practical perspective, we offer decision-makers insightful guidance on how to navigate the critical considerations of shore-to-ship delivery scheduling (Section 2.5.3). Sensitivity analysis of the discretization parameter reveals that adjusting the time granularity enables a customizable compromise between solution speed and decision quality. Moreover, our analysis of the relationship between drone fleet size and various operational metrics—such as delivery time, workload, and the number of trips per drone—informs drone deployment strategies.

2.2 Literature review

The paradigm of the DSP-SSD falls within the broader category of the DSP, which bears similarities to the VRP and has gained increasing research attention in recent years. However, the DSP-SSD diverges from the VRP in that ships, unlike traditional customers in the VRP, are moving, a feature that is characterized by a variant known as the vehicle routing problem with floating targets (VRPFT). Besides, since the number of drones is limited, the drones need to take multiple trips during the planning horizon, which is related to the multi-trip vehicle routing problem (MTVRP). Our literature review therefore mainly covers three domains: the DSP, VRPFT, and MTVRP.

2.2.1 Drone scheduling problem

The DSP is a contemporary optimization problem that has gained considerable research attention due to the increasing commercial and logistical applications of drones. Despite its close relationship with the VRP, the complexity of the DSP is amplified by constraints that

are unique to drones, such as limited flight durations and payload capacity, and the need to navigate through changing airspace. A number of heuristic and exact methods have been developed to handle the DSP in this context, such as the vehicle routing problem with drones (VRPD), which coordinates the efforts of several trucks with a fleet of drones (Dorling et al., 2017; Poikonen et al., 2017; Wang and Sheu, 2019); the flying sidekick traveling salesman problem (FSTSP), in which a drone is launched, operated, and retrieved by its associated truck (Murray and Chu, 2015; Murray and Raj, 2020); and the traveling salesman problem with drones (TSPD), which aims to determine the most efficient path for a drone covering a group of customers by considering the complex constraints to which drones are subject (Ha et al., 2018; Roberti and Ruthmair, 2021). A state-of-the-art review of these methods is provided by Macrina et al. (2020). While these variants can be applied to various operational scenarios, our DSP-SSD features the unique difficulty that the customers are moving during delivery, which therefore requires specialized modeling and algorithmic approaches.

In practical terms, drone scheduling has applications that span a multitude of industries and services, including the logistics sector, where drones transform traditional delivery models and enable last-mile delivery solutions that significantly reduce time and costs (Dorling et al., 2017; Poikonen et al., 2017; Garg et al., 2023); the field of agriculture, where drones perform regular surveillance to improve yields and resource management (Puri et al., 2017; Rejeb et al., 2022); and disaster operations, where drones are used for search and rescue support (Pereira, 2021; Betti Sorbelli et al., 2022). A few studies have explored the DSP in the maritime field for scheduling drones to monitor pollution emissions in emission control areas (Xia et al., 2019; Shen et al., 2020; Liu et al., 2023). Specifically, Xia et al. (2019) apply the DSP in inspecting ship emissions, and use a Lagrangian relaxation-based method to provide a near-optimal solution. Based on their work, Liu et al. (2023) further develop an exact approach for vessel emission monitoring with a heterogeneous fleet of drones. However, there is a notable scarcity of research addressing

the DSP in the shore-to-ship context.

2.2.2 Vehicle routing problem with floating targets

The VRPFT is tailored to address scenarios in which targets or destinations are not fixed but can change location dynamically, and is an advanced variation of the classic VRP. Solving the VRPFT requires sophisticated algorithms that can adapt routes to the real-time location of customers while ensuring the optimal deployment of resources and accommodating the movement of the delivery target. Some studies focus on a simpler version of this problem where the targets' locations are restricted to a set of discrete points, for example Zhen et al. (2023); Ozbaygin et al. (2017). When the targets are constantly moving, the solving complexity of the problem increases dramatically. There are two main types of model construction for the VRPFT: the coordination-based model, where time and location are continuous, and the arc flow model, which relies on time discretization. Studies applying the coordination-based model include Gambella et al. (2018), who propose a mixed-integer second-order cone optimization that can be solved by a branch-and-price (BP) algorithm, and Zhang et al. (2023), who design a new decomposition algorithm that solves the continuous model as a one-stop subproblem. However, these continuous models are unable to handle more complex multi-trip scenarios, and also rest on the stringent assumption that drones can cooperate with their customers' vehicles. This assumption is impractical in the DSP-SSD scenario, as ships lack the freedom to maneuver, particularly in coastal areas. The other type of model, the arc flow model based on time discretization, is also an effective approach to solving the VRPFT. For example, Stieber et al. (2015) study the multiple traveling salesmen problem with moving targets, in which the movement direction and speed are fixed. Xia et al. (2019) and Liu et al. (2023) focus on the multi-trip drone scheduling problem in vessel monitoring scenarios. These discrete models have greater flexibility when dealing with multi-trip scenarios.

2.2.3 Multi-trip vehicle routing problem

The MTVRP has gained increasing popularity in recent years. Azi et al. (2007) first present a two-phase exact algorithm addressing a problem where a single vehicle completes multiple routes to serve a set of customers with specific time windows. Azi et al. (2010) extend their previous work by considering the problem with multiple vehicles and propose a BP approach to tackle it. Macedo et al. (2011) then propose a new exact algorithm based on a pseudo-polynomial network flow model for this problem, which can solve more instances to optimality compared to that proposed by Azi et al. (2010). Mingozzi et al. (2013) formally propose the MTVRP and present an exact method to solve the MTVRP based on two set-partitioning-like formulations of the problem. Hernandez et al. (2016) also consider the MTVRP with time windows, where there are no restrictions on the duration but all customers should be visited. They develop a BP algorithm with tailored route-generating approaches. Paradiso et al. (2020) summarize prior studies on the MTVRP and identify four distinct variants. They develop an exact solution framework that incorporates column generation, route enumeration, and cutting plane, capable of solving all four MTVRP variants and surpassing previous state-of-the-art methods. Yang (2023) follow the novel superstructure-based formulation proposed by Paradiso et al. (2020) and develop an exact price-cut-and-enumerate method. Yang (2025) further put forward a variable fixing strategy that largely improves the efficiency of the CG and can be used in the MTVRP, capacitated vehicle routing problem (CVRP), and vehicle routing problem with time windows.

Some studies consider the practical applications of MTVRP, including multi-trip multi-repairman problem (Liu et al., 2018), drone delivery (Cheng et al., 2020), urban waste collection (Huang et al., 2021, 2024b), worker teams routing for airport ground (Dall'Olio and Kolisch, 2023), non-emergency patient transportation services (Huang et al., 2024a).

Our DSP-SSD presents a novel variant of the MTVRP with time windows, incorpo-

rating the unique challenge of floating customers, which adds the solving difficulty to the problem. Consequently, the existing state-of-the-art methods from previous studies cannot be directly applied to our problem.

2.2.4 Summary

This paper fills the research gap by exploring a novel DSP for shore-to-ship delivery to moving ships and providing a comprehensive solving scheme based on multiple drone trips. An arc-flow model is constructed inspired by the time discretization strategy in previous DSP in the maritime sector (Xia et al., 2019; Liu et al., 2023). However, our approach differs from their work in three main aspects. Firstly, the shore-to-ship delivery problem differs from the emissions inspection problem because it features payload capacity and nonlinear battery consumption constraints, adding new resources to the route generation process. Secondly, the time discretization granularity is modeled as a parameter to create the versatility needed to optimize solving efficiency and solution quality. Last but not least, Xia et al. (2019) propose a Lagrangian relaxation-based method that gives a feasible approximation to the optimal solution, and Liu et al. (2023) propose a structure enumeration-based exact approach whose bottleneck is reflected in a particular gap between upper and lower bounds. By contrast, we propose a tailored BPC algorithm that provides optimal solutions to large-scale instances with considerably higher quality in a short computation time.

2.3 Time discretization and model construction

This section begins with a clear definition of the DSP-SSD. A time-expanded network is then established based on time discretization as the foundation of the basic MIP model for the DSP-SSD.

2.3.1 Problem description

Within a planning period T, a set of ships (denoted by $\mathcal{V}$) requiring supplies are to be served by a set of identical drones (denoted by $\mathcal{D}$). The drones have a constant travel speed $S^{\mathcal{D}}$, deadweight $W^{\mathcal{D}}$, load capacity $C^{\mathcal{D}}$, and battery capacity $Q^{\mathcal{D}}$ that restrict the range that they can fly when carrying a specified payload. The drones are located in the same station with coordinates $(0, 0)$ and take off with supplies for multiple ships and return to the drone station after finishing their scheduled deliveries. When they return to the base station, their batteries are replaced, which takes τ_0 minutes. According to Dorling et al. (2017), the battery consumption $\bar{Q}$ of a drone during a certain period can be presented linearly to its flying time $\bar{T}$ and total weight $\bar{W}$, i.e., the sum of the drone's deadweight and the weight of its payload. Therefore, by introducing the parameter θ representing the battery consumption per weight over one unit of time, we can define the relationship by $\bar{Q} = \theta \bar{W} \bar{T}$ (Meng et al., 2023; Jeong et al., 2019; Cheng et al., 2020). The delivery must be finished in a coastal area within the drone flight boundary, which is defined by the maximum flying distance of a drone without payload, i.e., in a circle centered on the drone base station of radius R determined by $R = Q^{\mathcal{D}} S^{\mathcal{D}} / (2\theta W^{\mathcal{D}})$.

Each ship $v \in \mathcal{V}$ requiring supplies weighing a total of $w_v \leq C^{\mathcal{D}}$ enters the drone flight boundary and sails along its route at a constant speed of $S^{\mathcal{V}}$, where $S^{\mathcal{V}} < S^{\mathcal{D}}$. Each ship needs to be served exactly once. We define $T_v = [t_{ve}, t_{vl}]$ as the time interval during which ship v is sailing within the drone flight boundary, where the subscripts e and l represent the earliest and latest time, respectively. This time interval can be regarded as the time window for serving ship v. We assume that the navigational routes of the ships around the coastal area are relatively fixed due to the need to adhere to established shipping lanes, which are designed to ensure safe passage through potentially congested and hazardous waters. Therefore, the location coordinates of each ship $v \in \mathcal{V}$ at any time $t \in [t_{ve}, t_{vl}]$, denoted by $(\alpha_v(t), \beta_v(t))$, are known a priori. Based on the coordinates of each ship and

Euclidean distance, the flying time of each drone from ship v at time t to another ship v', denoted by $\tau_{v,v'}(t)$, can be calculated by (A.1) in A. Similarly, the expression of traveling time between the drone station and ships is also provided in A. For calculation purposes, we assume that the location of ship v at time $t \leq t_{ve}$ is $(\alpha_v(t_{ve}), \beta_v(t_{ve}))$ and that at time $t \geq t_{vl}$ is $(\alpha_v(t_{vl}), \beta_v(t_{vl}))$. Based on this assumption, we always have a nonnegative delivery time for a drone to reach a ship as presented and proved in B.

In our shore-to-ship problem, each drone takes off from the base station, sequentially delivers supplies to the ships following a predetermined route, and returns to the base station for battery replacement. This constitutes a single cycle, referred to as a trip. Throughout the planning period, a drone's itinerary may encompass several such trips. The aim is to finish all of the delivery tasks within the designated planning period and optimize the scheduling of the limited drone resources at the base station by minimizing the total activity time (battery replacement time included) of the drones.

2.3.2 Time-expanded network

Given the coordinates of each ship at different time points, it is appropriate to design a time-expanded network such that each node represents a ship or drone station at a certain time point. The time can be discretized into $K+1$ points with intervals of Δ minutes, where the positive discretization parameter $\Delta \in \mathbb{N}_{>0}$ and $K = \lceil T/\Delta \rceil$. Then, the set of time points is denoted by $\mathcal{T} = \{0, \Delta, 2\Delta, \cdots, K\Delta\}$. The node set $\mathcal{N}$ of this time-expanded network includes station-time nodes $\mathcal{N}^0 = \{(0,t) | t \in \mathcal{T}\}$ and ship-time nodes $\mathcal{N}^{\mathcal{V}} = \{(v,t) | v \in \mathcal{V}, t \in \mathcal{T}_v\}$, where $\mathcal{T}_v$ is the set of time points obtained by discretizing T_v into $\lfloor t_{vl}/\Delta \rfloor - \lceil t_{ve}/\Delta \rceil + 1$ elements with intervals of Δ minutes, i.e., $\mathcal{T}_v = \{\lceil t_{ve}/\Delta \rceil \Delta, \cdots, \lfloor t_{vl}/\Delta \rfloor \Delta\}$. We denote u_i and t_i as the corresponding ship (or station) and time of node i ($i \in \mathcal{N}$), respectively. The weight of supplies required by node i is w_{u_i}. As the weight of the required supplies of the station-time nodes is 0, we have

$w_{u_i} = 0$ if $i \in \mathcal{N}^0$.

Using these nodes, the ship-ship arcs, denoted by the set $\mathcal{A}^{\mathcal{V}}$, are constructed by connecting ship node (v, t) and ship node (v', t') if $\tau_{v,v'}(t)$ lies in $(t' - t - \Delta, t' - t]$. Similarly, the ship-station arcs, denoted by the set $\mathcal{A}^0$, are constructed by connecting station node $(0, t)$ and ship node (v, t') if $\tau_{0,v}(t)$ lies in $(t' - t - \Delta, t' - t]$, and by connecting ship node (v, t) and station node $(0, t')$ if $\tau_{v,0}(t) + \tau_0$ lies in $(t' - t - \Delta, t' - t]$. The battery replacement time τ_0 is included in the ship-station arc to reduce the total number of arcs in the network, which implies that there is no station-station arc. We denote each arc $a_{i,j}$ in $\mathcal{A} = \mathcal{A}^0 \cup \mathcal{A}^{\mathcal{V}}$ as the arc from node i to node j.

The complete time-expanded graph can be presented as $\mathcal{G}_\Delta = (\mathcal{N}, \mathcal{A})$, where the parameter Δ determines the degree of segmentation of the network. We denote z_t as the number of available drones in the station at time $t \in \mathcal{T}$ (if there are drones landing or taking off at time t, z_t reflects the number of drones available immediately after this time point). All of the drones located in the station have a full battery prior to flight, and the battery is replaced with a full battery on returning to the station at the end of the planning horizon. Therefore, we have $z_{-\Delta} = |\mathcal{D}|$ and $z_{K\Delta} = |\mathcal{D}|$, where $z_{-\Delta}$ is a constant that represents the initial number of available drones in the station at virtual time point $-\Delta$. The original problem can thus be conceptualized as selecting the appropriate arcs to finish all of the delivery tasks while meeting the payload capacity and battery consumption constraints of the drones.

2.3.3 Original model

The important notation for model construction can be summarized as follows.

- **Ship-related parameters:**

$\mathcal{V}$: the set of ships, indexed by v.

$S^{\mathcal{V}}$: the sailing speed of the ships.

$\mathcal{T}_v$: the set of discretized time points within the time window for serving ship v.

w_v: the weight of supplies required by ship v.

• **Drone-related parameters:**

$\mathcal{D}$: the set of drones, indexed by d.

$S^{\mathcal{D}}$: the traveling speed of the drones.

$W^{\mathcal{D}}$: the deadweight of the drones.

$C^{\mathcal{D}}$: the load capacity of the drones.

$Q^{\mathcal{D}}$: the battery capacity of the drones.

• **Network-related parameters:**

$\mathcal{T}$: the set of discretized time points during the time planning period T.

Δ: the discretization parameter.

$\mathcal{N}$: the set of nodes in the time-expanded network, which includes the station-time node set $\mathcal{N}^0$ and the ship-time node set $\mathcal{N}^{\mathcal{V}}$.

u_i: the ship (or station) of node $i \in \mathcal{N}$.

t_i: the time of node $i \in \mathcal{N}$.

$\mathcal{A}$: the set of arcs in the time-expanded network, which includes the ship-ship arc set $\mathcal{A}^{\mathcal{V}}$ and the ship-station arc set $\mathcal{A}^0$.

$a_{i,j}$: the arc from node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$.

• **Decision variables:**

$x_{i,j}$: a binary variable. $x_{i,j} = 1$ if arc $a_{i,j}$ is included in the drones' itineraries; and $x_{i,j} = 0$ otherwise.

y_i: a continuous variable that indicates the loaded weight of the drone when arriving at node $i \in \mathcal{N}^{\mathcal{V}}$.

q_i: a continuous variable that indicates the remaining battery level of the drone when arriving at node $i \in \mathcal{N}^{\mathcal{V}}$.

z_t: an integer variable that captures the number of available drones with a full battery in the station at time $t \in \mathcal{T}$.

Based on the discretized time-expanded network, we can formulate the following MIP model:

[Original model]

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{q},\boldsymbol{z}} \sum_{a_{i,j}\in\mathcal{A}} (t_j - t_i)x_{i,j} \tag{2.1a}$$

$$\text{s.t.} \sum_{j\in\mathcal{N}^{\mathcal{V}}} x_{i,j} - \sum_{j\in\mathcal{N}^{\mathcal{V}}} x_{j,i} = z_{t_i-\Delta} - z_{t_i} \qquad \forall i \in \mathcal{N}^0, \tag{2.1b}$$

$$\sum_{i\in\mathcal{N}} x_{i,j} - \sum_{i\in\mathcal{N}} x_{j,i} = 0 \qquad \forall j \in \mathcal{N}^{\mathcal{V}}, \tag{2.1c}$$

$$\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}^{\mathcal{V}}:u_j=v} x_{i,j} = 1 \qquad \forall v \in \mathcal{V}, \tag{2.1d}$$

$$y_i - w_{u_i}x_{i,j} + C^{\mathcal{D}}(1 - x_{i,j}) \geq y_j \qquad \forall i \in \mathcal{N}^{\mathcal{V}}, \forall j \in \mathcal{N}^{\mathcal{V}}, \tag{2.1e}$$

$$y_i - w_{u_i}x_{i,j} \geq 0 \qquad \forall i \in \mathcal{N}^{\mathcal{V}}, \forall j \in \mathcal{N}^0, \tag{2.1f}$$

$$Q^{\mathcal{D}} - \theta(t_j - t_i)(y_j + W^{\mathcal{D}})x_{i,j} \geq q_j \qquad \forall i \in \mathcal{N}^0, \forall j \in \mathcal{N}^{\mathcal{V}}, \tag{2.1g}$$

$$q_i - \theta(t_j - t_i)(y_j + W^{\mathcal{D}})x_{i,j} + Q^{\mathcal{D}}(1 - x_{i,j}) \geq q_j \qquad \forall i \in \mathcal{N}^{\mathcal{V}}, \forall j \in \mathcal{N}^{\mathcal{V}}, \tag{2.1h}$$

$$q_i - \theta(t_j - t_i - \tau_0)W^{\mathcal{D}}x_{i,j} + Q^{\mathcal{D}}(1 - x_{i,j}) \geq 0 \qquad \forall i \in \mathcal{N}^{\mathcal{V}}, \forall j \in \mathcal{N}^0, \tag{2.1i}$$

$$x_{i,j} \in \{0, 1\} \qquad \forall a_{i,j} \in \mathcal{A}, \tag{2.1j}$$

$$0 \leq y_i \leq C^{\mathcal{D}} \qquad \forall i \in \mathcal{N}^{\mathcal{V}}, \tag{2.1k}$$

$$0 \leq q_i \leq Q^{\mathcal{D}} \qquad \forall i \in \mathcal{N}^{\mathcal{V}}, \tag{2.1l}$$

$$z_{-\Delta}, z_{K\Delta} = |\mathcal{D}|, \tag{2.1m}$$

$$z_t \in \{0, 1, \ldots, |\mathcal{D}|\} \qquad \forall t \in \mathcal{T}. \tag{2.1n}$$

The objective of the model is to minimize the total time taken to deliver the supplies over the planning horizon. The battery replacement time is included in this horizon. Constraints (2.1b) are the flow balance restriction of the station-time nodes, which determine the number of available drones at each station-time node by relating the decision variable

$x_{i,j}$ to z_t. Constraints (2.1c) are the flow balance restriction on each ship-time node. Constraints (2.1d) mandate that each ship must be served. Constraints (2.1e)–(2.1f) enforce the load capacity of the drones when arriving at each node. Constraints (2.1g)–(2.1i) specify the battery depletion during drone service and the remaining battery level of the drone when arriving at each node. Constraints (2.1j)–(2.1n) define the domains of the variables.

As the battery consumption constraints (2.1g) and (2.1h) are nonlinear due to the term $y_j x_{i,j}$, we introduce the new continuous variables $\gamma_{i,j} = y_j x_{i,j}$ ($i \in \mathcal{N}, j \in \mathcal{N}^{\mathcal{V}}$) to linearize it. Specifically, constraints (2.1g) and (2.1h) are transformed as follows:

$$Q^{\mathcal{D}} - \theta\left(t_j - t_i\right)\left(\gamma_{i,j} + W^{\mathcal{D}} x_{i,j}\right) \geq q_j \qquad \forall i \in \mathcal{N}^0, \forall j \in \mathcal{N}^{\mathcal{V}}, \qquad (2.2a)$$

$$q_i - \theta\left(t_j - t_i\right)\left(\gamma_{i,j} + W^{\mathcal{D}} x_{i,j}\right) + Q^{\mathcal{D}}\left(1 - x_{i,j}\right) \geq q_j \qquad \forall i \in \mathcal{N}^{\mathcal{V}}, \forall j \in \mathcal{N}^{\mathcal{V}}, \qquad (2.2b)$$

$$\gamma_{i,j} \geq y_j - C^{\mathcal{D}}(1 - x_{i,j}) \qquad \forall i \in \mathcal{N}, j \in \mathcal{N}^{\mathcal{V}}, \qquad (2.2c)$$

$$\gamma_{i,j} \geq 0 \qquad \forall i \in \mathcal{N}, \forall j \in \mathcal{N}^{\mathcal{V}}. \qquad (2.2d)$$

This model is a variant of the CVRP, which is a classic NP-hard problem. There are no known algorithms that can solve all instances of the problem in polynomial time, which reflects the computational difficulty of the problem.

2.4 Branch-and-price-and-cut algorithm

In this section, a BPC algorithm is presented to solve the DSP-SSD problem. We first reformulate the original model into a set covering-like model by Dantzig-Wolfe decomposition. Then, we demonstrate the specific procedures of the CG and B&B schemes, where a set of cuts are proposed to enhance the solving efficiency. Finally, the overall framework of the BPC algorithm is given.

2.4.1 Dantzig-Wolfe decomposition

The original model (2.1) resembles the classic CVRP and can be formulated as a set covering-like model by Dantzig-Wolfe decomposition. The following notation is defined:

• **Parameters:**

$\mathcal{P}$: the set of feasible trips for the drones, indexed by p. Trip p is defined as a single cycle $(i_0, i_1, \cdots, i_n)$ in which the drone takes off from the station-time node i_0, sequentially delivers supplies to ship-time nodes $i_1, \cdots, i_{n-1}$, and returns to the station-time node i_n for battery replacement.

$\mathcal{A}_p$: the set of arcs contained in trip p.

$\mathcal{V}_p$: the set of vessels visited in trip p.

$\mathcal{T}_p$: the set of time points contained in trip p. For the trip $p = (i_0, i_1, \cdots, i_n)$, we assume that $\mathcal{T}_p = \{t \mid t = k\Delta, t_{i_0} \leq t < t_{i_n}, k \in \mathbb{N}_0\}$.

$c_{i,j}$: the time cost of arc $a_{i,j}$, i.e., $t_j - t_i$.

c_p: the total time cost of trip p, i.e., $t_{i_n} - t_{i_0}$, which includes the total flying time and the time required to replace the battery once.

$\mathbb{1}_X(x)$: an indicator function that takes the value of 1 if $x \in X$, and 0 otherwise.

• **Decision variable:**

χ_p: a binary variable. $\chi_p = 1$ if the trip is included in the drones' itineraries; and $\chi_p = 0$ otherwise.

The original model can be reformulated to select the appropriate trips to serve all ships with a minimized total drone activity time, written as follows:

[Set covering-like model]

$$\min_{\boldsymbol{\chi}} \sum_{p \in \mathcal{P}} c_p \chi_p \tag{2.3a}$$

$$\text{s.t.} \sum_{p \in \mathcal{P}} \mathbb{1}_{\mathcal{V}_p}(v) \chi_p \geq 1 \qquad \forall v \in \mathcal{V}, \tag{2.3b}$$

$$\sum_{p \in \mathcal{P}} \mathbb{1}_{\mathcal{T}_p}(t) \chi_p \leq |\mathcal{D}| \qquad \forall t \in \mathcal{T}, \tag{2.3c}$$

$$\chi_p \in \mathbb{N}_0 \qquad \forall p \in \mathcal{P}. \tag{2.3d}$$

The objective function (2.3a) attempts to minimize the total activity time of the drones including battery replacement. Constraints (2.3b) ensure that every vessel is served. We note that the constraints are relaxed to greater than or equal to 1, whereas under the optimal solution, the left-hand side will naturally be 1. Constraints (2.3c) restrict the available number of drones. Constraints (2.3d) regulate the variables are nonnegative integers.

Proposition 1. *Constraints* (2.3c) *are equivalent to constraints* (2.1b) *and* (2.1n).

The proof of Proposition 1 is given in C.

In the model, each column, corresponding to the variable χ_p, represents a feasible trip that a drone can undertake. The number of columns increases dramatically as the problem size grows. In this case, CG is crucial, as it introduces only the most promising columns iteratively. This technique significantly reduces the computational burden, allowing the practical solution of large-scale problems that would otherwise be intractable (Desrochers et al., 1992). The linear relaxation of the master model is presented in Section 2.4.2, where the candidate columns are generated by resolving a subproblem through dynamic programming. Typically, the resulting solution offers superior relaxation, which is then integrated into the B&B framework introduced in Section 2.4.3 to produce integer solutions.

2.4.2 Column generation

In this section, the two-phase CG scheme is introduced. In the first phase, the optimal solution of the linear master model is obtained, and in the second phase promising columns with negative reduced costs are generated.

2.4.2.1 Restricted master problem

The CG starts with a restricted master problem (RMP), which relaxes model (2.3) by linearizing the binary variables and only introducing a subset of all possible columns. The RMP can be written as follows:

[Restricted master problem]

$$\min_{\boldsymbol{\chi}} \sum_{p\in\mathcal{P}'} c_p\chi_p \tag{2.4a}$$

$$\text{s.t.} \sum_{p\in\mathcal{P}'} \mathbb{1}_{\mathcal{V}_p}(v)\chi_p \geq 1 \qquad \forall v \in \mathcal{V}, \tag{2.4b}$$

$$\sum_{p\in\mathcal{P}'} \mathbb{1}_{\mathcal{T}_p}(t)\chi_p \leq |\mathcal{D}| \qquad \forall t \in \mathcal{T}, \tag{2.4c}$$

$$\chi_p \geq 0 \qquad \forall p \in \mathcal{P}', \tag{2.4d}$$

where $\mathcal{P}' \subseteq \mathcal{P}$ and constraints (2.4d) are obtained by relaxing variable constraints (2.3d) into nonnegative continuous variables.

As an initial step, a dummy column corresponding to χ_0 is added into the RMP, where $\mathbb{1}_{\mathcal{V}_0}(v)$ for each ship v is set to 1, the occupied number of drones for each time point t is set to $|\mathcal{D}|$, and c_0, which represents a huge cost in this column (also known as big-M), is set to an empirical value of $10^4 \cdot |\mathcal{V}| \cdot 2R/S^{\mathcal{D}}$. Theoretically, we will have $\chi_0 = 0$ after a certain number of iterations by adding more promising columns with smaller costs; otherwise, the model is infeasible.

2.4.2.2 Cutting planes

To accelerate the convergence, we augment model (2.4) by incorporating the rounded capacity inequalities (RCIs), 2-path inequalities (TPIs), and subset-row inequalities (SRIs). The first two inequalities are robust cuts that do not change the structure of the pricing problem, while SRIs belong to nonrobust cuts which are based on a subset of the con-

straints in the RMP (Costa et al., 2019). We generate these inequalities at the early stage of the BPC algorithm, for example, only the root node of the B&B tree.

Rounded capacity inequalities. We employ the RCIs first proposed by Augerat (1995) to strengthen the RMP, which is to enforce drone capacity constraints more effectively by considering the total demand of a subset of vessels and ensuring that it cannot be served by a given number of drones unless their combined capacity is sufficient. For notation convenience, we denote $\mathcal{RC}$, indexed by rc, as the set of RCIs. Specifically, for a subset of vessels $\mathcal{V}^{rc}$, an RCI states that the total number of drones required to serve $\mathcal{V}^{rc}$ must be at least $\lceil \sum_{v \in \mathcal{V}^{rc}} w_v / C^{\mathcal{D}} \rceil$.

In this paper, we implement an exact separation procedure initially proposed by Fukasawa et al. (2006) to effectively generate the RCIs. Specifically, given a solution $\boldsymbol{\chi}^*$ to the RMP, we can obtain $x^*_{i,j}$ for each arc $a_{i,j} \in \mathcal{A}$. Then, we define a support graph $\bar{\mathcal{G}}_\Delta = \{\mathcal{N}, \bar{\mathcal{A}}\}$, where $\bar{\mathcal{A}} = \{a_{i,j} \in \mathcal{A} : x^*_{i,j} > 0\}$. We denote binary variable r_v, $v \in (\mathcal{V} \cup \{0\})$, that represents whether a vessel v (0 is the station) is included in $\mathcal{V}^{rc}$ and variable $s_{i,j}$, $a_{i,j} \in \bar{\mathcal{A}}$, that equals 1 if arc $a_{i,j}$ satisfies $u_i \in \mathcal{V}^{rc}$ and $u_j \notin \mathcal{V}^{rc}$. A given parameter M represents the number of drones that cannot serve the subset of vessels $\mathcal{V}^{rc}$. Then, for each value of M in $\{1, \cdots, \lceil \sum_{v \in \mathcal{V}} w_v / C^{\mathcal{D}} \rceil - 1\}$, we can solve the following model:

$$\min_{\boldsymbol{s},\boldsymbol{r}} \sum_{a_{i,j} \in \bar{\mathcal{A}}} x^*_{i,j} s_{i,j} \tag{2.5a}$$

$$\text{s.t.} s_{i,j} \geq r_{u_i} - r_{u_j} \qquad \forall a_{i,j} \in \bar{\mathcal{A}}, \tag{2.5b}$$

$$s_{i,j} \geq r_{u_j} - r_{u_i} \qquad \forall a_{i,j} \in \bar{\mathcal{A}}, \tag{2.5c}$$

$$\sum_{v \in \mathcal{V}} w_v r_v \geq M \cdot C^{\mathcal{D}} + 0.001, \tag{2.5d}$$

$$r_0 = 0, \tag{2.5e}$$

$$r_v \in \{0, 1\} \qquad \forall v \in \mathcal{V}, \tag{2.5f}$$

$$s_{i,j} \geq 0 \qquad \forall a_{i,j} \in \bar{\mathcal{A}}. \tag{2.5g}$$

If the optimal objective $\sum_{a_{i,j}\in\bar{\mathcal{A}}} x^*_{i,j}s^*_{i,j} < 2(M+1)$, the capacity restriction over set $\mathcal{V}^{rc} = \{v \in \mathcal{V} : r^*_v = 1\}$ is violated and we add the following cut into the RMP:

$$\sum_{p\in\mathcal{P}'} \sum_{a\in\mathcal{A}^{rc}} \mathbb{1}_{\mathcal{A}_p}(a)\chi_p \geq M+1, \tag{2.6}$$

where $\mathcal{A}^{rc} = \{a_{i,j} \in \mathcal{A} : u_i \in \mathcal{V}^{rc}, u_j \notin \mathcal{V}^{rc}\}$ is the set of arcs flowing out of the subset of vessels $\mathcal{V}^{rc}$.

2-path inequalities. Since considering capacity alone is not strong enough, we also apply the TPIs to our problem. These cuts mitigate solutions involving subsets of vessels that cannot be served by a single drone due to capacity, battery consumption, and time-window constraints. Similarly, we denote $\mathcal{TP}$, indexed by tp, as the set of TPIs. Firstly, we use the heuristic proposed by Kohl et al. (1999) to generate candidate subsets $\mathcal{V}^{tp}$. For each candidate subset, we check whether this subset of vessels can be served by one drone (in one trip) satisfying all resource constraints. As the RCI has separated all subsets of vessels that cannot be served by one drone due to capacity constraints, only energy consumption and time window have to be checked here. In traditional TPIs, this is equivalent to checking whether there is a feasible solution to the TSPTW defined on $\mathcal{V}^{tp} \cup \{0, |\mathcal{V}^{tp}| + 1\}$ (with origin 0 and destination $|\mathcal{V}^{tp}| + 1$). While this problem can be solved by a classic dynamic programming (DP) method (Dumas et al., 1995), our problem differs as the vessels are floating and nodes are defined on a time-expanded network. Therefore, we propose a tailored DP algorithm specified in D to check the feasibility of serving the subset of vessels with one drone. It should be noted that this DP algorithm is similar to but much simpler than the labeling algorithm that will be proposed in Section 2.4.2.3 as we only need to check the feasibility property here. Then, the TPI can be

written as below.

$$\sum_{p\in\mathcal{P}'}\sum_{a\in\mathcal{A}^{tp}}\mathbb{1}_{\mathcal{A}_p}(a)\chi_p \geq 2, \tag{2.7}$$

where $\mathcal{A}^{tp} = \{a_{i,j} \in \mathcal{A} : u_i \in \mathcal{V}^{tp}, u_j \notin \mathcal{V}^{tp}\}$ is the set of arcs flowing out of the subset of vessels $\mathcal{V}^{tp}$.

Partial reduced cost. The RMP is solved to optimality, yielding dual prices $\boldsymbol{\pi} \geq 0$ and $\boldsymbol{\lambda} \leq 0$ associated with constraints (2.4b) and (2.4c), respectively. We also denote $\boldsymbol{\delta} \geq 0$ and $\boldsymbol{\xi} \geq 0$ as the dual variables corresponding to RCIs and TPIs, respectively. Therefore, we let b_p be the partial reduced cost of trip path p (we call b_p a "partial" reduced cost because it does not consider the nonrobust cuts), which can be written as

$$b_p = c_p - \sum_{v\in\mathcal{V}}\mathbb{1}_{\mathcal{V}_p}(v)\pi_v - \sum_{t\in\mathcal{T}}\mathbb{1}_{\mathcal{T}_p}(t)\lambda_t - \sum_{rc\in\mathcal{RC}}\sum_{a\in\mathcal{A}^{rc}}\mathbb{1}_{\mathcal{A}_p}(a)\delta_{rc} - \sum_{tp\in\mathcal{TP}}\sum_{a\in\mathcal{A}^{tp}}\mathbb{1}_{\mathcal{A}_p}(a)\xi_{tp}. \tag{2.8}$$

Then, the partial reduced cost for each arc $a_{i,j} \in \mathcal{A}_p$ can be written as

$$b_{i,j} = c_{i,j} - \sum_{v\in\mathcal{V}}\mathbb{1}_{u_i}(v)\pi_v - \sum_{t\in\mathcal{T}}\mathbb{1}_{\mathcal{T}_{(i,j)}}(t)\lambda_t - \sum_{rc\in\mathcal{RC}}\mathbb{1}_{\mathcal{A}^{rc}}(a_{i,j})\delta_{rc} - \sum_{tp\in\mathcal{TP}}\mathbb{1}_{\mathcal{A}^{tp}}(a_{i,j})\xi_{tp}, \tag{2.9}$$

where $\mathcal{T}_{(i,j)} = \{t \mid t = k\Delta, t_i \leq t < t_j, k \in \mathbb{N}_0\}$.

Subset-row inequalities. We concentrate on the inequalities specified for three vessels because they can be separated by straightforward enumeration (Jepsen et al., 2008; Tilk et al., 2018). We denote the index set of vessel subsets as $\mathcal{SR}$, where $|\mathcal{SR}| = \binom{|V|}{3} = \frac{|V|!}{3!(|V|-3)!}$, and the subsets of all three vessels as $\mathcal{V}^{sr}$, $sr \in \mathcal{SR}$. For each subset $\mathcal{V}^{sr}$, an

SRI is added to the RMP, which can be written as follows:

$$\sum_{p\in\mathcal{P}'} \lfloor \frac{\sum_{v\in\mathcal{V}} \mathbb{1}_{\mathcal{V}^{sr}\cap\mathcal{V}_p}(v)}{2} \rfloor \chi_p \leq 1. \tag{2.10}$$

By incorporating inequality (2.10) for each $sr \in \mathcal{SR}$ into the RMP, we can cut out many fractional solutions so as to lift the LB of the B&B tree. These SRIs require us to make specific modifications to our pricing problems. We denote $\mu_{sr} \leq 0$ as the dual price associated with $sr \in \mathcal{SR}$. The value μ_{sr} is subtracted from the reduced cost of path p if the path includes two or three vessels in subset $\mathcal{V}^{sr}$. Therefore, the full reduced cost of p, denoted by $\bar{c}_p$, can be computed as follows:

$$\bar{c}_p = b_p - \sum_{sr\in\mathcal{SR}} \mathbb{1}_{\mathcal{SR}(p)}(sr)\mu_{sr}, \tag{2.11}$$

where $\mathcal{SR}(p) = \{sr \in \mathcal{SR} : |\mathcal{V}_p \cap \mathcal{V}^{sr}| \geq 2\}$.

2.4.2.3 Pricing problem

The pricing problem in the CG is to identify new columns with a negative reduced cost, i.e., promising trip path $p \in \mathcal{P}\backslash\mathcal{P}'$ for the drones. The binary decision variable $x_{i,j}$ takes the value of 1 if arc $a_{i,j}$ is included in the new path, and 0 otherwise. We further define $\hat{\mathcal{V}} := \{u_i \in \mathcal{V} : x_{i,j} = 1, i \in \mathcal{N}^{\mathcal{V}}, j \in \mathcal{N}\}$ and $\hat{\mathcal{SR}} := \{sr \in \mathcal{SR} : |\hat{\mathcal{V}} \cap \mathcal{V}^{sr}| \geq 2\}$. Then, given the dual prices obtained from the RMP, the pricing problem in our case can be presented as follows.

[Pricing problem]

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{q}} \sum_{a_{i,j}\in\mathcal{A}} b_{i,j}x_{i,j} - \sum_{sr\in\mathcal{SR}} \mathbb{1}_{\hat{\mathcal{SR}}}(sr)\mu_{sr} \tag{2.12a}$$

$$\text{s.t.} \sum_{a_{i,j}\in\mathcal{A}^0:u_i=0} x_{i,j} = 1, \tag{2.12b}$$

$$\sum_{a_{j,i}\in\mathcal{A}^0:u_i=0} x_{j,i} = 1, \tag{2.12c}$$

$$(2.1c), (2.1e) - (2.1l).$$

The classic pricing problem, which is always reduced to an elementary shortest path problem with resource constraints (ESPPRC), is NP-hard (Dror, 1994; Garey and Johnson, 1983). The labeling algorithm, a DP approach, has been demonstrated to be an effective method for solving the ESPPRC (Feillet et al., 2004; Righini and Salani, 2006). Our model (2.12) is a variant of the ESPPRC where the depot is differentiated by distinct time points and battery consumption constraints are imposed. As a drone's battery consumption is associated with its total loaded weight, i.e., the total load required by the subsequent ships that the drone visits, we propose a backward labeling algorithm to generate the trip path.

Let $L_p = (\mathcal{V}_p, y^p, q^p, \mathcal{T}_p, i^p, b_p, \bar{c}_p)$ be the label vector of the elementary backward path $p = (i_0, i_1, \cdots, i^p)$, which means that the drone flies to vessel-time nodes $i^p, \cdots, i_1$ sequentially, and finally arrives at the station-time node i_0. $\mathcal{V}_p$ is the visited vessel on path p; y^p is the total weight of supplies on path p; q^p is the battery consumption of a drone traveling from i^p to i_0; $\mathcal{T}_p$ is the set of time points covered on path p; i^p is the last node of backward path p; b_p is the partial reduced cost of path p, including costs defined on every single arc and excluding the cost contributed by the dual variables corresponding to the SRIs; $\bar{c}_p$ is the full reduced cost corresponding to path p. Due to the load capacity and battery consumption constraints, a label L_p is feasible only if:

$$y^p \leq C^{\mathcal{D}}, \quad q^p \leq Q^{\mathcal{D}}. \tag{2.13}$$

The feasible backward path is generated following F.2. Denote $\bar{\mathcal{V}}_p$ as the set of vessels that path p can further access (satisfying capacity, battery consumption, and time-window

constraints). To speed up the solution process and limit the number of labels generated, the following dominance rule can be employed, the proof of which is presented in E.

Proposition 2. *Given two labels* $L_{p_1} = (\mathcal{V}_{p_1}, y^{p_1}, q^{p_1}, \mathcal{T}_{p_1}, i^{p_1}, b_{p_1}, \bar{c}_{p_1})$ *and* $L_{p_2} = (\mathcal{V}_{p_2}, y^{p_2}, q^{p_2}, \mathcal{T}_{p_2}, i^{p_2}, b_{p_2}, \bar{c}_{p_2})$ *in* $List_L$ *or* $List_{L^*}$, L_{p_2} *is dominated by* L_{p_1} *if the following conditions are satisfied: (i)* $\bar{\mathcal{V}}_{p_2} \subseteq \bar{\mathcal{V}}_{p_1}$, *(ii)* $y^{p_1} \leq y^{p_2}$, *(iii)* $q^{p_1} \leq q^{p_2}$, *(iv)* $\mathcal{T}_{p_1} \subseteq \mathcal{T}_{p_2}$, *(v)* $i^{p_1} = i^{p_2}$, *(vi)* $b_{p_1} \leq b_{p_2}$, *and (vii)* $(\mathcal{V}^{sr} \cup \mathcal{V}_{p_1}) \subseteq (\mathcal{V}^{sr} \cup \mathcal{V}_{p_2}), \forall sr \in \mathcal{SR}$.

Heuristic labeling method. To avoid generating too many labels with poorly reduced costs, we selectively retain only the first Num_L partial labels with the most negative reduced costs during the label generation process. Notably, when no complete label with a negative reduced cost can be generated through the selected partial labels, we extend all of the partial labels to check again whether there are still columns with negative reduced costs. An indicator parameter "flag" is used to control whether to extend all of the partial labels. Additionally, when returning columns after each round of label generation, we return the top Num_{L^*} columns with the most negative reduced costs (if there are fewer than Num_{L^*} columns with negative reduced costs, we return as many as there are). The new columns corresponding to the backward paths are then added to the RMP.

2.4.3 Branch-and-bound

The RMP with SRIs is solved by the CG algorithm until no more columns with negative reduced costs can be generated. If the solution to the RMP is not integral, the B&B scheme is introduced by selecting a branching variable and creating two new subproblems, each with an additional constraint that forces the variable to take an integer value on one branch. In this section, we first discuss how to obtain the upper bound (UB) and lower bound (LB) of the B&B tree. We then demonstrate the specific branching strategy.

2.4.3.1 Upper and lower bounds

In a B&B tree, the UB is the value of the best feasible solution found so far, which serves as a benchmark for evaluating the potential of other solutions. To obtain a UB at the early stage of the B&B tree, we apply two methods in the root node. Firstly, we adopt the feasibility pump (FP) algorithm proposed by Fischetti et al. (2005), which is a heuristic algorithm designed to quickly find a feasible solution by repeatedly adjusting a non-integer solution to the RMP. The main idea of the algorithm is to round the fractional variables to the nearest integer and then solve a modified version of the RMP to minimize the distance from this rounded solution. This is called a pumping action between continuous and discrete spaces, which seeks to strike a balance between satisfying the problem's constraints and adhering to the integrality requirements of the variables. The pseudocode of the FP algorithm is given in F.1. If no feasible solution is found by FP, we solve the IP form of the RMP in the root node, which includes a small scale of columns and can be solved quickly by the state-of-the-art solver (Yang, 2023). If a feasible integer solution is found, the global UB is set to the objective value of this solution; otherwise, the global UB is set to $+\infty$. The global UB is updated when a new feasible integer solution is found to have an objective value smaller than the global UB.

The local LB is obtained by solving the RMP (2.4) with inequalities after CG, which significantly improves the objective value compared with solving the simple linearized version of the set covering-like model (Jepsen et al., 2008). The global LB is updated to the smallest local LB of all leaf nodes. Notably, we will stop the CG prematurely if the algorithm has reached the time limit and will compute an LB based on the LP value of the current RMP and the current reduced cost (Vanderbeck and Wolsey, 1996).

2.4.3.2 Branching scheme

In our B&B tree, we adopt a branching strategy by branching on arcs, which has been proven to be powerful for the VRP in expediting convergence and simplifying the problem structure through straightforward implementation (Desaulniers et al., 2006). Specifically, we adhere to the procedures outlined below.

Node selection: When there are multiple nodes with fractional solutions to be extended, we follow the best-bound-first strategy proposed by Toth and Vigo (2014); Land and Doig (1960). This strategy aims to speed up the convergence process by concentrating on the region that is most likely to contain the optimal solution. In our minimization problem, this strategy selects the currently unexplored node with the smallest estimated bound for branching, where the estimated bound for each node is set to the objective value of its parent node. When multiple nodes have the same estimated bound, e.g., they are child nodes branching from the same parent node, we randomly choose one of them.

Arc selection: Given a solution $\boldsymbol{\chi}^*$ to the RMP of the chosen node, we can calculate the value of $x^*_{i,j}$ for each arc $a_{i,j} \in \mathcal{A}$. When the solution contains multiple fractional arcs, we follow the arc selection rule proposed by Desrochers et al. (1992) and look for arc $a_{i,j}$ such that $x^*_{i,j}$ is closest to 0.5 for branching. Ties are broken by selecting the arc with the smallest cost $c_{i,j}$.

Branching on arc: The chosen arc is fixed at 0 and 1 in each child node. In a node where arc $a_{i,j}$ is fixed at 0, the arc is directly removed from the pricing subproblem of the node. The columns corresponding to the path that contains arc $a_{i,j}$ are removed from the RMP. In the other nodes where arc $a_{i,j}$ is fixed at 1, we remove the arcs in

$$\bar{\mathcal{A}}_1(a_{i,j}) = \{(k,l) \in \mathcal{A} | (u_k = u_i, k \neq i) \vee (k = i, l \neq j) \vee (u_l = u_i, l \neq i)\} \text{ if } i \in \mathcal{N}^{\mathcal{V}}, \tag{2.14}$$

and

$$\bar{\mathcal{A}}_2(a_{i,j}) = \{(k,l) \in \mathcal{A} | (u_k = u_j, k \neq j) \vee (l = j, k \neq i) \vee (u_l = u_j, l \neq j)\} \text{ if } j \in \mathcal{N}^{\mathcal{V}}. \tag{2.15}$$

Specifically, the three conditions in equation (2.14) forbid three kinds of arcs, i.e., the arcs starting from the other nodes of the same vessel of node i, the arcs starting from node i but not connecting to node j, and the arcs ending at other nodes of the same vessel of node i. The three conditions in equation (2.15) forbid three kinds of arcs, i.e., the arcs starting from the other nodes of the same vessel of node j, the arcs ending at node j but not starting to node i, and the arcs ending at other nodes of the same vessel of node j. Therefore, the arc $a_{i,j}$ must be chosen as long as the vessel u_i is served. Similarly, columns with paths that use specific arcs are removed from the RMP according to (2.14)–(2.15).

The overall pseudocode of the BPC algorithm is presented in F.3.

2.5 Numerical experiments

In this section, we generate several testing instances based on vessel data from real ports and drone-related data obtained from UAV companies to implement our proposed BPC algorithm and analyze the critical parameters. All of the experiments are carried out on three computers running the Windows operating system with a 3.70-GHz Intel Core i7 processor with 64GB of RAM, using 12 threads to take advantage of the multiple cores available. The models and algorithms are implemented by Java and solved by CPLEX 12.7.

2.5.1 Experiment setting

This section introduces the data used to generate the instances.The drone-related data are mostly sourced from DJI (2024) with the parameter θ adjusted to meet the shore-to-ship delivery duration and the ship-related parameters are generated following the procedures in Xia et al. (2019).

• **Drones.** Each drone travels at a constant speed ($S^{\mathcal{D}}$) of 0.72 km/min. The maximum payload of a drone $(C^{\mathcal{D}})$ is 30 kg, where the dead weight $(W^{\mathcal{D}})$ is 65 kg. The battery's capacity ($Q^{\mathcal{D}}$) is $38{,}000$ mAh and the battery consumption ratio (θ) is 4.5 mAh/(kg·min). Battery replacement takes 12 minutes (τ_0).

• **Ships.** The speed of the ships is randomly generated from a uniform distribution ranging from 5 to 7 knots (0.154 to 0.216 km/min). The demand of each ship (w_v) is generated from a uniform distribution over the interval $[5, 8]$ kg. The earliest and latest service points of each ship are determined by generating random coordinates in a semi-circle of radius 32 km, which is the boundary of the service range that a fully loaded drone can cover, centered on the drone station. The ships are assumed to sail along a straight line within the service boundary. The earliest serving time, t_{ve}, for each ship is obtained from a uniform distribution over the interval $[0, 300]$ minutes, and the latest serving time, t_{vl}, is then computed based on the earliest service time, the ship's speed, and the earliest and latest service points.

• **Instances.** We denote each instance as $V - D$, where V is the number of ships and D is the number of drones. The number of drones is selected from $\{5, 10, 15\}$, and the number of ships is chosen from $\{10, 20, 30, 40, 50\}$. For each combination of V and D, we generate 10 instances. For a particular instance at each scale, we add the suffix i to represent the ith instance in this scale, denoted as $V - D - i$. To ensure the feasibility of the instance as far as possible, we set the value of T to $\max\{t_{vl}, \forall v \in \mathcal{V}\} + R/S^{\mathcal{D}} + \tau_0$, which is the latest delivery time of the ship plus the maximum travel time of a drone from

the ship to the station plus the battery replacement time.

• **Algorithm parameters.** The discretization granularity, Δ, in all of the experiments except that in Section 2.5.3.1 is set to 5 minutes. The time limit is designated as 3600 seconds for small- and medium-scale instances where $V < 30$, and 7200 seconds for large-scale instances where $V = 30$. The optimality gap is set to 0.001%. The label reduction parameters Num_L and Num_{L^*} are set to 2000 and 200, respectively. The maximum iteration number is set to 100 and the number of variables to be flipped is set to $|\boldsymbol{\chi}^*|/6$ in the FP algorithm.

2.5.2 Computational Results

In this section, we solve the instances using CPLEX, the BPC algorithm, and variants of the BPC algorithm to demonstrate the superiority of our proposed BPC algorithm in terms of computational efficiency and effectiveness.

2.5.2.1 Performance comparison of CPLEX and the BPC algorithm

Comparison of the average performance of CPLEX and the BPC algorithm for different scales of instances are provided in Table 2.1. The following details are provided.

• **#Opt**: the number of instances that are solved to optimality at each scale within the time limit.

• **#Fea**: the number of instances where feasible integer solutions are obtained at each scale within the time limit but the algorithm has not converged to the optimality gap.

• **#Unk**: the number of instances where no feasible integer solution is obtained at each scale within the time limit, i.e., the solution status is unknown.

• $\overline{\textbf{Gap}}$**(%)**: the average gap of the instances where feasible solutions are obtained at each scale. The gap(%) of each instance is calculated by $(\text{UB} - \text{LB})/\text{UB} \times 100$.

• $\textbf{Gap}_{\text{max}}$**(%)**: the maximum (worst) gap of the instances where feasible solutions are

obtained at each scale.

- **CPU(s)**: the total computational CPU time in seconds.
- **Avg.**: the average results of each metric for all instances at different scales.

Table 2.1: The results of CPLEX and BPC on different instances

Instance	CPLEX						BPC				
	#Opt	#Fea	#Unk	$\overline{\text{Gap}}$(%)	$\text{Gap}_{\max}$(%)	CPU(s)	#Opt	#Fea	$\overline{\text{Gap}}$(%)	$\text{Gap}_{\max}$(%)	CPU(s)
10−5	10	0	0	0.00	0.00	841.10	10	0	0.00	0.00	1.00
20−5	0	10	0	38.67	47.60	3600.00	6	4	0.62	3.10	1823.13
20−10	0	10	0	41.90	50.10	3600.00	8	2	0.56	3.50	1103.83
30−5	0	1	9	60.40	60.40	7200.00	2	8	1.62	3.80	6103.45
30−10	0	9	1	50.58	61.10	7200.00	3	7	1.01	2.70	5259.07
40−10	0	4	6	63.80	72.90	7200.00	1	9	3.02	12.80	6767.17
40−15	0	6	4	69.62	75.30	7200.00	0	10	2.14	9.03	7200.00
50−15	0	2	8	71.65	75.70	7200.00	0	10	5.21	20.20	7200.00
Avg.	1.25	5.25	3.50	49.58	55.39	5505.14	3.75	6.25	1.77	6.89	4432.20

For small-scale instances such as 10−5, both algorithms achieve optimal solutions with a zero gap. However, the BPC algorithm demonstrates a clear advantage in computational efficiency, solving the problem in an average of 1 second compared with CPLEX's 841.10 seconds. As the problem size increases to instances like 20−5 and 20−10, CPLEX struggles to find optimal solutions, hitting the maximum computational time cap of 3600 seconds with high average gaps exceeding 38%. It fails to find an optimal solution in any of the instances and the worst gaps of 20−5 and 20−10 soar to 47.6% and 50.1%, respectively. In contrast, BPC manages to adapt more effectively to these moderate problem sizes, resolving many instances with substantially lower gaps and in considerably less time, achieving a worst-case gap of just 3.1% and 3.5% in the 20−5 and 20−10 instances, respectively.

As the problem complexity further increases (for instances like 30−5 and 30−10), CPLEX's performance deteriorates with no optimal solutions and worst-case gaps reaching up to 61.1% in the 30−10 instance. The computational time reaches 7200 seconds for these instances, highlighting significant limitations in scalability. The BPC algorithm, on the other hand, demonstrates superior performance across all metrics, successfully finding feasible solutions in all instances. It maintains a remarkably low average gap of 1.62% and

1.01% for instance 30−5 and 30−10, respectively, indicating its effectiveness at closely approximating or achieving an optimal solution. The BPC algorithm's average CPU time is also less than 7200 seconds, which is significantly lower than that of CPLEX.

In the largest and most complex instances, such as 40−10, 40−15, and 50−15, the performance differential between CPLEX and BPC is markedly pronounced. CPLEX fails to find feasible solutions for more than half of the instances and encounters the worst-case gap peaking at 75.70% for the 50−15 instances, while consistently exhausting the computational limit of 7200 seconds. BPC, while also reaching maximum computational time, maintains significantly lower average gaps, with a comparatively small worst-case gap of 20.20% in the 50−15 instance. Moreover, it can always find feasible solutions for all instances.

In summary, our proposed BPC algorithm displays better performance not only in consistently finding optimal solutions to more instances but also in maintaining lower gap percentages. Moreover, this superiority is evident despite the BPC algorithm's higher CPU time in complex cases, as it still offers much lower average and worst gaps than CPLEX, suggesting that the BPC algorithm is more effective at navigating and solving large-scale problems.

2.5.2.2 Effectiveness of the branching and node selection strategies

In this section, we compare our branching and node selection strategies with other methods commonly used in the literature.

Firstly, we test our best-bound-first strategy with depth-first, width-first, and random-selection strategies on each instance on a scale of 20−5, the results of which are presented in Table 2.2. It can be observed that the best-bound-first strategy demonstrates significant efficacy by consistently achieving the lowest optimality gaps across multiple instances, surpassing other strategies. The average gap for best-bound-first is 0.62%, considerably lower than depth-first (1.25%), width-first (1.14%), and random (1.26%). In terms of com-

putational efficiency, best-bound-first also outperforms the other methods with an average CPU time of 1823.11 seconds, which is lower than the other strategies, reflecting not only its ability to find closer solutions to the optimal but also its efficiency in computational expenditure. Furthermore, the number of nodes processed by best-bound-first (average of 145.4) illustrates a balanced approach, effectively navigating between exploring many paths and focusing on promising ones.

Table 2.2: The results of different node selection rules

Instance	Best-bound-first			Depth-first			Width-first			Random-selection		
	Gap(%)	CPU(s)	#Node	Gap(%)	CPU(s)	#Node	Gap(%)	CPU(s)	#Node	Gap(%)	CPU(s)	#Node
20−5−1	0.20	3600.00	410	2.10	3600.00	136	1.70	3600.00	174	1.70	3600.00	168
20−5−2	0.00	627.69	18	2.00	3600.00	162	1.90	3600.00	224	1.90	3600.00	228
20−5−3	0.00	23.77	0	0.00	24.74	0	0.00	24.24	0	0.00	24.20	0
20−5−4	3.10	3600.00	500	3.70	3600.00	440	3.70	3600.00	578	3.70	3600.00	558
20−5−5	0.00	278.03	10	0.00	530.66	18	0.00	540.44	18	0.00	673.36	24
20−5−6	0.40	3600.00	130	0.70	3600.00	150	0.10	3600.00	74	1.10	3600.00	70
20−5−7	0.00	2710.94	230	1.40	3600.00	194	1.40	3600.00	296	1.40	3600.00	258
20−5−8	0.00	153.93	0	0.00	156.87	0	0.00	155.36	0	0.00	160.61	0
20−5−9	0.00	36.74	0	0.00	36.83	0	0.00	36.60	0	0.00	37.49	0
20−5−10	2.50	3600.00	156	2.60	3600.00	78	2.60	3600.00	110	2.80	3600.00	106
Avg.	0.62	1823.11	145.4	1.25	2234.91	117.8	1.14	2235.66	147.4	1.26	2249.57	141.2

We also validate our branching strategy by comparing it to the strategy of branching on the number of drones and then on the arc. The specific procedure of branching on the number of drones is given in H and the performances of both methods are shown in Table 2.3. These results highlight the efficiency and effectiveness of our strategy. Firstly, our branching rule achieves consistently lower optimality gaps, with a remarkable zero gap in 6 instances and an average gap of 0.62% versus 1.02% for the alternative approach. Moreover, this strategy often requires less computational time, as evidenced by quicker resolutions in instances like 20−5−2 and 20−5−5, and processes fewer nodes on average (145.4 compared to 160.8), suggesting a more efficient and focused search path.

2.5.2.3 Benefits of acceleration strategies for the BPC algorithm

In this section, we verify the effect of acceleration strategies on the solving efficiency of the BPC algorithm. We denote BPC-V1 as the BPC algorithm without performing the FP

Table 2.3: The results of different branching strategies

Instance	Branch on arc			Branch on drone number then on arc		
	Gap(%)	CPU(s)	#Node	Gap(%)	CPU(s)	#Node
20−5−1	0.20	3600.00	410	1.40	3600.00	172
20−5−2	0.00	627.69	18	1.60	3600.00	282
20−5−3	0.00	23.77	0	0.00	24.94	0
20−5−4	3.10	3600.00	500	3.40	3600.00	640
20−5−5	0.00	278.03	10	0.00	557.65	18
20−5−6	0.40	3600.00	130	0.10	3600.00	80
20−5−7	0.00	2710.94	230	1.10	3600.00	300
20−5−8	0.00	153.93	0	0.00	162.75	0
20−5−9	0.00	36.74	0	0.00	37.80	0
20−5−10	2.50	3600.00	156	2.60	3600.00	116
Avg.	0.62	1823.11	145.4	1.02	2238.31	160.8

algorithm or solving the IP form of RMP in the root node, BPC-V2 as the BP algorithm without SRIs, RCIs, and TPIs, and BPC-V3 as the BPC algorithm without the heuristic labeling method. The original BPC algorithm and the variants with the acceleration strategies are tested on medium-scale instances (20−5), the detailed results of which are provided in Table 2.4. Note that the UB and LB are in hours.

Table 2.4: The acceleration effects of BPC on solution quality and solving efficiency

Instance	BPC				BPC-V1				BPC-V2				BPC-V3			
	UB	Gap(%)	CPU(s)	#Node	UB	Gap(%)	CPU(s)	#Node	UB	Gap(%)	CPU(s)	#Node	UB	Gap(%)	CPU(s)	#Node
20−5−1	13.50	0.20	3600.00	410	13.50	1.30	3600.00	176	13.50	2.30	3600.00	5132	13.50	1.70	3600.00	90
20−5−2	13.00	0.00	627.69	18	/[1]	100.00	3600.00	290	13.00	1.60	3600.00	4902	13.00	1.70	3600.00	162
20−5−3	13.50	0.00	23.77	0	13.50	0.00	24.62	0	13.50	0.00	2.71	0	13.50	0.00	155.22	0
20−5−4	14.67	3.10	3600.00	500	/	100.00	3600.00	634	14.75	5.80	3600.00	7250	14.75	4.00	3600.00	644
20−5−5	13.50	0.00	278.03	10	13.50	0.00	561.48	18	13.50	1.40	3600.00	4868	13.50	0.00	1196.24	14
20−5−6	13.83	0.40	3600.00	130	14.83	6.90	3600.00	126	13.83	0.60	3600.00	2802	13.83	0.20	3600.00	24
20−5−7	15.25	0.00	2710.94	230	/	100.00	3600.00	326	15.25	0.70	3600.00	4172	15.42	2.20	3600.00	190
20−5−8	12.08	0.00	153.93	0	12.08	0.00	160.80	0	12.08	0.00	121.72	70	12.08	0.00	813.49	0
20−5−9	14.75	0.00	36.74	0	14.75	0.00	37.43	0	14.75	0.00	282.54	328	14.75	0.00	324.10	0
20−5−10	12.58	2.50	3600.00	156	/	100.00	3600.00	116	12.58	2.40	3600.00	1810	13.25	7.50	3600.00	30
Avg.	13.67	0.62	1823.11	145.4	/	40.82	2238.43	168.6	13.67	1.48	2560.70	3133.4	13.76	1.73	2408.91	115.4

[1] : It means no upper bound is found in the time limit and the gap is 100%.

As shown in Table 2.4, from the perspective of solution quality, the original BPC algorithm demonstrates robust performance, achieving optimal solutions in most instances with an average gap of just 0.62%. This result indicates that using a combination of acceleration strategies with the BPC is highly effective. BPC-V1, which does not use IP form or the FP algorithm to derive the UBs in the root node, fails to find a UB in four of the ten instances and has a much larger gap of 40.82%, which suggests that the importance of recognizing a feasible integer solution at the early stages of the BPC algorithm. The second

variant, which removes the inequalities used to cut out fractional solutions, results in a significant drop in performance, with the average gap increasing to 1.48%. The increased gap indicates that the inequalities are vital for the BPC algorithm's ability to efficiently solve these problems, and their absence leads to looser bounds and reduced overall efficiency. Similar to BPC-V2, excluding the heuristic labeling method causes the average gap and average UB to increase to 1.73% and 13.76, respectively. The degradation in the optimality gaps and recognized UBs demonstrates the importance of heuristic labeling in achieving efficient solutions. Heuristic labeling appears to be crucial for navigating the solution space effectively and ensuring that solutions remain optimal.

Table 2.4 also compares the original BPC algorithm with its three variants on the CPU time in seconds (CPU(s)) and the number of nodes explored (#Node). It can be observed that BPC-V1 explores a greater number of nodes in five of the ten instances, indicating that the lack of effective UBs results in inefficient pruning in the B&B tree, as it cannot discard suboptimal branches early in the process. Therefore, it also has a larger average computational time of 2238.43 seconds than the BPC. BPC-V2 often explores significantly more nodes, indicating less effective pruning. Despite exploring more nodes, BPC-V2 also shows a larger average CPU time than both the original BPC algorithm and the other two variants. This result suggests that BPC-V2, by not including the inequalities, is likely to achieve faster processing times per node because it deals with a less constrained and thus simpler pricing problem at each node. However, it leads to poorer relaxations in each node, which makes it harder for the algorithm to converge. BPC-V3 explores fewer nodes but has a larger computation time on average than the original BPC algorithm, which suggests that each node's processing is computationally more intensive due to the presence of too many labels in each iteration of the CG procedure.

2.5.2.4 Detailed analysis on different cuts

To assess the impact of each type of cut on our algorithm, we solve the medium-scale instances (20−5) using algorithms that exclusively utilize either SRI, RCI, or TPI. The detailed results are shown in Table 2.5. Our findings illustrate that employing a combination of all cuts significantly enhances performance over using any single type of cut. For instance, the average optimality gap with all cuts is considerably lower at 0.62%, as opposed to 1.37% for SRI-only, 1.34% for RCI-only, and 1.31% for TPI-only. Furthermore, both the computation time and the number of nodes processed are substantially reduced when all types of cuts are employed. The average computation time for the algorithm incorporating all cuts stands at 1823.11 seconds, significantly lower than the times observed when only one type of cut is used, which range from 2541.72 to 2881.00 seconds, indicating a more efficient search process and potentially superior pruning within the B&B tree.

When comparing the performances of employing each type of cut individually, notable differences in efficiency and effectiveness emerge. The SRI-only algorithm exhibits a slightly higher average optimality gap at 1.37%, closely followed by RCI-only and TPI-only at 1.34% and 1.31%, respectively. Interestingly, both the SRI-only and TPI-only algorithms achieve the optimal solution in three out of ten instances, whereas the RCI-only algorithm obtains the optimal solution in only two. The relatively smaller optimality gap of applying only TPI may stem from its effectiveness in cutting out the infeasible solutions by including the capacity, battery consumption, and time-window constraints. In terms of computational efficiency, SRI yields the longest CPU time per node. Specifically, the average CPU times for the algorithms with each type of cut are all above 2500 seconds. However, algorithms employing only RCI and TPI explore a significantly larger average number of nodes (3419.4 and 3169.6, respectively) compared to those using SRI-only (2220). This suggests that the SRI may be the least effective at reducing the gap when

used alone, but it is crucial in closing the gap for certain instances.

Table 2.5: The results of applying different cuts

Instance	All cuts			SRI only			RCI only			TPI only		
	Gap(%)	CPU(s)	#Node	Gap(%)	CPU(s)	#Node	Gap(%)	CPU(s)	#Node	Gap(%)	CPU(s)	#Node
20−5−1	0.20	3600.00	410	1.60	3600.00	2390	0.90	3600.00	3126	1.60	3600.00	8290
20−5−2	0.00	627.69	18	1.50	3600.00	3032	1.70	3600.00	7284	1.70	3600.00	7308
20−5−3	0.00	23.77	0	0.00	24.27	0	0.00	1.68	0	0.00	1.66	0
20−5−4	3.10	3600.00	500	4.80	3600.00	4114	4.30	3600.00	7704	4.30	3600.00	8040
20−5−5	0.00	278.03	10	1.70	3600.00	3928	1.70	3600.00	7414	1.50	3600.00	5836
20−5−6	0.40	3600.00	130	1.10	3600.00	2632	1.10	3600.00	1176	1.10	3600.00	32
20−5−7	0.00	2710.94	230	0.40	3600.00	4066	1.00	3600.00	2622	0.30	3600.00	1940
20−5−8	0.00	153.93	0	0.00	155.65	0	0.10	3600.00	2168	0.00	3600.00	0
20−5−9	0.00	36.74	0	0.00	37.31	0	0.00	8.34	2	0.00	35.81	2
20−5−10	2.50	3600.00	156	2.60	3600.00	2038	2.60	3600.00	2698	2.60	3600.00	248
Avg.	0.62	1823.11	145.4	1.37	2541.72	2220.0	1.34	2881.00	3419.4	1.31	2883.75	3169.6

In addition, we also compare the algorithm efficiency by applying the SRI in different layers of the B&B tree with results detailed in Table 2.6. The outcomes reveal marked improvements in solving efficiency with the increased number of layers applying SRI. Specifically, when SRI is applied only at the root node, the average optimality gap is 1.45%, which progressively decreases to 0.62% as SRI is extended to all nodes, highlighting a significant tightening of the solution space. Concurrently, the average CPU time decreases from 2541.94 seconds to 1823.11 seconds, and the number of nodes processed dramatically reduces from 2822.6 to just 145.4 on average, indicating more effective pruning and faster convergence towards optimal solutions when SRI is more extensively applied. It is worth noting that the number of layers applying the SRI can serve as a hyperparameter, which should be chosen reasonably for different datasets.

Table 2.6: The results of applying SRI in different nodes

Instance	Only in root node			First 3 levels of the B&B tree			First 5 levels of the B&B tree			In all nodes		
	Gap(%)	CPU(s)	#Node	Gap(%)	CPU(s)	#Node	Gap(%)	CPU(s)	#Node	Gap(%)	CPU(s)	#Node
20−5−1	2.40	3600.00	5032	1.60	3600.00	4006	1.60	3600.00	4316	0.20	3600.00	410
20−5−2	1.80	3600.00	4304	1.50	3600.00	4094	0.00	616.59	238	0.00	627.69	18
20−5−3	0.00	23.20	0	0.00	23.03	0	0.00	24.03	0	0.00	23.77	0
20−5−4	5.20	3600.00	5442	4.80	3600.00	4972	4.80	3600.00	8428	3.10	3600.00	500
20−5−5	1.40	3600.00	4634	1.40	3600.00	4308	0.00	280.87	10	0.00	278.03	10
20−5−6	0.60	3600.00	2660	1.10	3600.00	2844	1.10	3600.00	2524	0.40	3600.00	130
20−5−7	0.70	3600.00	4064	0.60	3600.00	5414	0.60	3600.00	9458	0.00	2710.94	230
20−5−8	0.00	158.41	0	0.00	151.82	0	0.00	151.26	0	0.00	153.93	0
20−5−9	0.00	37.84	0	0.00	35.56	0	0.00	35.28	0	0.00	36.74	0
20−5−10	2.40	3600.00	2090	2.60	3600.47	2162	2.60	3600.00	1996	2.50	3600.00	156
Avg.	1.45	2541.94	2822.6	1.36	2541.09	2780.0	1.07	1910.80	2697.0	0.62	1823.11	145.4

2.5.2.5 Summary of the algorithm comparison

Overall, the experiments in Section 2.5.2 reveal the superiority of the BPC in terms of effective exploration of the solution space and handling of the complexities inherent in large-scale instances. This superiority may be due to the advantages of a trip-based model structure and algorithm design. Firstly, the BPC algorithm, with its CG technique, dynamically generates and incorporates only the most promising trips into the model, efficiently handling this complexity. In contrast, the arc-flow model used by CPLEX involves variables for every arc in the time-expanded network, leading to a large number of variables and constraints. Secondly, our design and choice of branching and node selection strategies are appropriate for this problem. Thirdly, the BPC algorithm is enhanced by integrating several acceleration strategies, which results in the algorithm handling complex problems more effectively than CPLEX in specific problem contexts. For instance, implementing problem-specific inequalities and heuristic labeling methods allows the algorithm to quickly find high-quality feasible solutions, guiding the search process more effectively than the general-purpose heuristics used in CPLEX.

2.5.3 Sensitivity analysis

2.5.3.1 Discretization granularity

In the following experiments, instances at scales of $20-5$ are tested with different discretization granularity values Δ, varying from $\{1, 2, 5, 10\}$ (minutes). The average numbers of time-space nodes and arcs with different discretization granularity are presented in Table 2.7. As the granularity increases from 1 to 10, there is a notable reduction in both the average number of time-space nodes and arcs across the instance $20-5$. Specifically, when Δ is set at 1, the network is most dense with 3765.4 time-space nodes and 29195.6 arcs. However, increasing Δ to 10 results in a significant decrease, reducing the number of time-space nodes and arcs to 379.0 and 2920.7, respectively. This trend suggests that

while finer granularity enhances the network's detail and potentially its modeling accuracy, it also substantially increases computational complexity. Based on the constructed time-expanded network, the average performances of CPLEX and BPC are reported in Table 2.8.

Table 2.7: The scale of the expanded-time network under different discretization granularity

Instance	Δ	#Avg. time-space nodes	#Avg. arcs
20−5	1	3765.4	29195.6
	2	1884.6	14614.5
	5	755.6	5844.4
	10	379.0	2920.7

Table 2.8: The results of CPLEX and BPC with different discretization granularity

Instance	Δ	CPLEX						BPC					
		#Opt	#Fea	#Unk	$\overline{\text{Gap}}$(%)	UB	CPU(s)	#Opt	#Fea	#Infea[1]	$\overline{\text{Gap}}$(%)	UB	CPU(s)
20−5	1	0	10	0	48.57	13.06	3600.00	7	3	0	0.84	11.97	1597.38
	2	0	10	0	44.03	13.81	3600.00	7	3	0	0.60	12.56	1426.49
	5	0	10	0	38.67	14.47	3600.00	6	4	0	0.62	13.67	1823.13
	10	0	7	3	36.89	17.48	3600.00	7	1	2	0.28	16.29	484.73

[1] : It presents that the model is infeasible. Notably, it differs from "#Unk" where the model may be feasible but no feasible integer solution is found by the algorithm.

From the perspective of algorithm efficiency, we can observe that the BPC algorithm consistently finds a larger number of optimal solutions across different granularities, while CPLEX shows a general increase in the number of optimal solutions as Δ increases. BPC experiences a notable reduction in CPU time as Δ increases (from 1597.38 seconds at $\Delta = 1$ minute to 484.73 seconds at $\Delta = 10$ minutes), suggesting that a coarser time granularity substantially reduces computational complexity, while the CPU time of CPLEX remains at 3600 seconds, indicating notably worse solving efficiency than BPC even with finer discretizations.

From the standpoint of decision quality, increasing the value of Δ simplifies the problem, leading to shorter solution times but at the cost of solution quality (increased drone delivery times). When the value of Δ increases to 10 minutes, the model may even become

infeasible. Therefore, selecting an appropriate Δ value is crucial for balancing model accuracy and computational feasibility in network-based modeling. For real-world applications requiring precise drone scheduling decisions, a finer granularity is preferable despite the higher computational cost. However, when computational resources or time are constrained, a coarser granularity might be a viable compromise.

2.5.3.2 Number of ships

We fix the number of drones at 5 but vary the number of ships to 10, 20, and 30 to compare the total number of trips and computation time, where the feasible integer solutions are obtained by the BPC algorithm.

In Figure 2.2, the x-axis represents the total number of ships, i.e., $|\mathcal{V}|$. The left y-axis depicts the total number of trips taken by the drones over the planning horizon, while the right y-axis shows the CPU time in seconds. The histogram displays these two metrics for each instance, while the line graph presents the average values of these metrics across instances of the same scale.

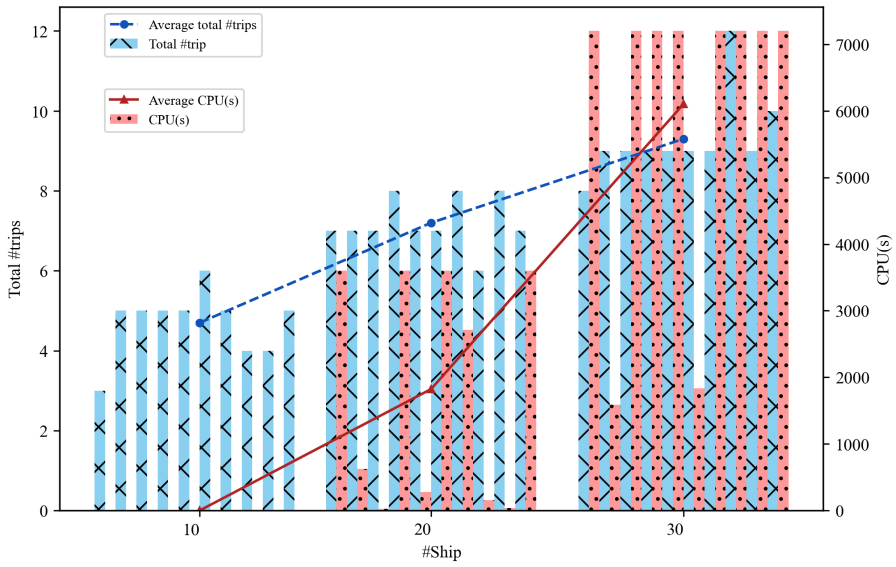


Figure 2.2: Different number of ships delivered by the same number of drones

It can be seen that scalability issues become apparent as the number of ships increases,

as evidenced by significant increases in the computation time. For instances with 30 ships, the computation time is notably high, often reaching the maximum time limit of 7200 seconds. For instances with 10 ships, the computation time is drastically smaller, mostly approximately 0 seconds. In terms of the total number of drone trips, the results reveal a more steady increasing trend with the number of ships. For instances with 20 or 30 ships, the number of trips varies from 6 to 12, indicating multiple trips made by each drone. This feature is less pronounced in instances of 10 ships, with the number of trips ranging from 3 to 6.

2.5.3.3 Summary of the sensitivity analysis

The sensitivity analysis conducted on the discretization granularity parameter demonstrates that a finer granularity (smaller Δ) enhances the solution quality at the expense of an increased computation time. Decision-makers are thus presented with the choice of balancing these factors according to their strategic priorities. Furthermore, the experiments reveal that a coarser granularity (larger Δ) can lead to model infeasibility, mainly attributable to the inability to complete services within the required time window after rough time discretization. The experiment on the number of ships highlights the capability of drones to undertake multiple trips in scenarios of high delivery demand versus limited resources. These insights are invaluable for decision-makers determining the optimal number of drones to deploy to ensure a balance between service efficiency and drone workload for effective shore-to-ship delivery operations.

2.6 Conclusions

Motivated by the lack of attention paid to drone-assisted shore-to-ship delivery, we put forward the DSP-SSD and provide a comprehensive solution to this problem. Firstly, we develop the MIP model, which is tailored for scenarios involving dynamic targets and mul-

tiple trips and can be solved directly by off-the-shelf solvers in small-scale instances. This model leverages a time-expanded network created through time discretization, enabling drones to execute sequential delivery tasks effectively.

To apply the model to larger, more practical scenarios, we introduce a tailored BPC algorithm. This advanced algorithm transforms the original model into a route-based framework, which is then processed using a B&B strategy. The algorithm incorporates CG for dynamic variable integration and employs cutting planes to strengthen the linear relaxations. Our computational tests confirm that the BPC algorithm outperforms standard solvers, particularly in managing the frequent demands of the DSP-SSD, and our acceleration techniques further enhance its efficiency.

We offer decision-makers detailed analyses of how changes to the time discretization parameter can optimize the trade-off between solution speed and decision accuracy, and how varying the number of ships affects delivery times, workloads, and trip frequencies, thereby aiding optimal drone fleet management. Moreover, our model and algorithms are capable of handling regulatory constraints, such as low-altitude airspace safety rules and maritime navigation limitations, which can be integrated into the model to enhance feasibility in practical deployment.

Chapter 3

Study 2: Integrated cruise fleet deployment and itinerary schedule problem

3.1 Introduction

Maritime transportation plays a vital role in global trade and tourism, encompassing diverse sectors such as container shipping and cruise shipping, each with distinct operational and strategic priorities. While container shipping management has been widely investigated in both industry and the literature (Fransoo and Lee, 2013; Ding and Chou, 2015; Legros et al., 2019), the cruise shipping sector remains relatively underexplored. Following the upward trend in cruise tourism, the cruise industry has experienced a surge in popularity, distinguishing itself from container shipping through its unique tactical considerations, such as itinerary design, sailing frequency, and passenger-driven demand dynamics (Wang et al., 2017). This section begins by outlining the key challenges faced by the cruise industry and then presents the primary focus and contributions of this article.

3.1.1 Problem background

In recent years, there has been a noticeable surge in the popularity of cruise tourism, driven by increasing consumer interest in unique and diverse travel experiences (Forbes, 2024). This rise in demand necessitates a strategic reevaluation of how cruise companies manage their fleets and design their itineraries, an area that has traditionally been overshadowed in both academia and the industry compared to cargo fleet management (Wang et al., 2016). As the market for cruise travel continues to expand, the imperative for effective cruise fleet management becomes increasingly critical, which not only helps in reducing operational costs but also plays a pivotal role in tapping into new market opportunities and maximizing profitability. Moreover, sophisticated cruise deployment and itinerary scheduling strategies can enhance passenger satisfaction, thereby fostering continuous profit growth. This paper specifically addresses the burgeoning issue of cruise fleet management.

The decision-making process in cruise fleet management can be categorized into three distinct levels: strategic, tactical, and operational, which are presented in Figure 3.1 (Papathanassis et al., 2011). Each level addresses different aspects of fleet and itinerary management, playing a critical role in the overall efficiency and profitability of cruise operations. At the *strategic level*, decisions focus on the long-term aspects of cruise fleet planning, usually lasting 5 to 10 years. This includes determining the optimal size of the acquired cruise fleet and renting berths, by assessing market trends, forecasting future demand, and making rational financial investments. The goal at this level is to set up a fleet that is capable of delivering sustained growth and profitability over the years. At the *tactical level*, the focus shifts to cruise deployment, which concerns the specific allocation of the fleet across various locations. This medium-term planning, typically operating over 1 to 3 years, is crucial for optimizing the use of the fleet by positioning cruises in different locations based on seasonal demand, regional popularity, and other logistical considerations. Tactical decisions ensure that the right cruises are available in the right places to

meet anticipated passenger loads, thereby maximizing operational efficiency and revenue. At the *operational level*, the planning becomes more granular, focusing on the specifics of itinerary planning. This includes the schedule of cruise arrivals and departures at each port. Operational decisions are concerned with the day-to-day management of the fleet, ensuring that each cruise operates on a schedule that enhances passenger satisfaction and operational turnover.

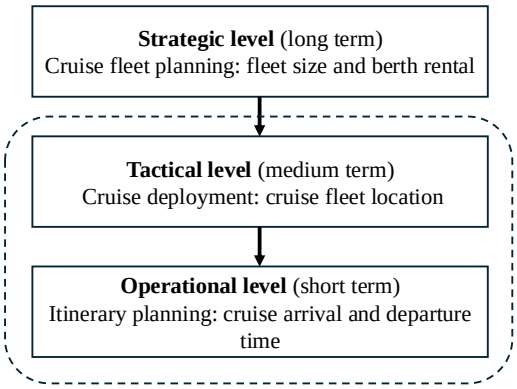


Figure 3.1: The decision levels for cruise management

In the previous studies, researchers always focus on optimizing a single level of decision (Wang et al., 2017; Di Pillo et al., 2021), despite that each of these levels is interdependent, forming a comprehensive framework for decision-making in cruise fleet management (Papathanassis, 2017). Based on the long-term goals and decisions, the cruise company optimizes medium-term and short-term decisions, thus ensuring a harmonious balance between profitability, customer satisfaction, and efficient fleet management. In fact, integrated optimization has been proven to be quite effective in reducing costs and improving supply-demand interactions in many industries, such as transportation (Birolini et al., 2021; Yin et al., 2023a), production (Özer and Uncu, 2013), and health care (Liu et al., 2019; Chen et al., 2019). Therefore, in this paper, we focus on both medium-term and short-term decisions, providing an integrated solution method for the *cruise fleet deployment and itinerary scheduling problem* (CFD-ISP).

Specifically, the integrated CFD-ISP can be defined below. In a given planning hori-

zon, usually several months, we consider a cruise network with a set of potential cruise lines, each containing different visited ports. We first need to decide the specific number of cruises to be deployed on each cruise line. There are usually hundreds of potential cruise lines, and a cruise line can be either a *bi-directional itinerary* (with two home ports) or a *round-trip itinerary* (with one home port) (Goldjoy Holidays, 2024). As shown in Figure 3.2, there is a bi-directional cruise: "Shanghai–Jeju Island–Nagasaki–Fukuoka–Beppu–Shimizu–Tokyo" with home ports Shanghai and Tokyo, and a round-trip cruise: "Shanghai–Jeju Island–Fukuoka–Yeosu–Busan–Shanghai" with one home port Shanghai. Notably, while the visiting ports of a bi-directional cruise in both directions are identical in the illustrative example, they can vary without loss of generalization. The visited port sequence of each line is given while the specific departure and arrival time at each home port has to be determined. For cruise companies, the main objective is to optimize the deployed number of cruises on each cruise line, thereby trying to minimize the sum of cruise deployment cost, fuel cost, and opportunity cost associated with unmet demand. In addition to these explicit cost-related considerations, the sailing itinerary scheduling should align with passenger expectations, specifically accommodating the majority preference to engage in daytime sightseeing and return to the cruise for nocturnal rest.

However, effectively managing cruise schedules is highly challenging given the vastness of the cruise network, the complexity of costs, and the varied demands of customers. Traditional policy-based cruise deployment decisions and manual schedule methods fall short of addressing these complexities, highlighting the need for more sophisticated decision-making approaches. The itinerary scheduling problem with a fixed visiting sequence is widely studied in logistics but remains computationally intractable due to its large scale and high time granularity requirements (Wang et al., 2017; Yin et al., 2023b; Chai et al., 2024). Consequently, integrating cruise deployment with itinerary scheduling presents a formidable challenge, rendering the integrated problem unsolvable by commercial solvers directly.

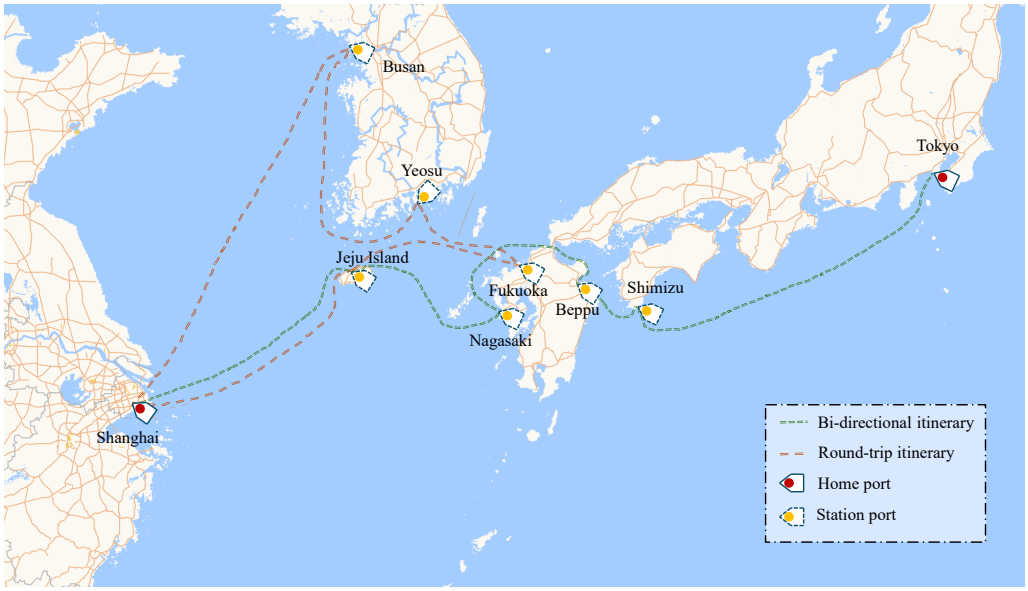


Figure 3.2: Two types of cruise itineraries

3.1.2 Main contributions

From a *modeling* perspective, we propose an IP model for the integrated CFD-ISP based on the time-expanded network, which formulates the movement of cruises over a given planning horizon. Besides, we propose and apply multiple model reformulation techniques that streamline the time-expanded network and enhance the model solvability. From an *algorithmic* standpoint, we employ a tailored BD algorithm, enhanced with an advanced simultaneous Magnanti-Wong method (Perrykkad et al., 2022), which can generate Pareto-optimal cuts in each iteration by solving an augmented version of the subproblem (SP) only once, thus accelerating the solving process. Extensive numerical experiments with simulation instances validate our model and demonstrate the effectiveness of incorporating the simultaneous Magnanti-Wong method in the BD algorithm. This advancement not only enhances the efficiency of solving the CFD-ISP but also broadens the applicability of the simultaneous Magnanti-Wong method in addressing integration problems. From a *practical* perspective, we utilize the cruise network obtained through Goldjoy Holidays (2024) as a case study and incorporate in-depth sensitivity analysis on various types of

cost components, which verifies the importance of integrating the tactical and operational decisions, providing instructive implications on cruise management.

In particular, the multifaceted contributions of this paper are outlined as follows:

- We establish an IP model for the CFD-ISP that integrates medium-term tactical cruise deployment decisions with short-term operational schedules.
- Our innovative route-based reformulation of the itinerary scheduling problem streamlines the time-expanded network, which is also expected to offer enhanced solving efficiency to similar problems, such as rolling stock and aircraft schedules.
- Utilizing the cumulative-flow-based variables, we prove that the itinerary scheduling problem is TU and the integer variables can be relaxed.
- We establish a tailored BD algorithm based on the simultaneous Magnanti-Wong method that significantly outperforms commercial solvers and can handle large-scale instances with up to 100 cruise lines in minutes.
- Sensitivity analysis on various cost components shows the superiority of our integrated models compared to two policy-based sequential models, achieving an over 90% reduction in costs under benchmark parameters.
- A case study in the Asia-Pacific area provides four major management insights: (i) the necessity of integrating tactical and operational decisions, (ii) the need for dynamic adjustments in cruise deployment amidst operational cost fluctuations, (iii) the strategic benefit of focused investments under an unstable market environment, and (iv) the advantage of decentralizing cruise deployment in scenarios where utility is emphasized.

The remainder of this paper is structured as follows. Section 3.2 provides an overview of relevant work on cruise management and the BD method. In Section 3.3, we describe

the CFD-ISP problem in detail and develop its mathematical model. Section 3.4 elucidates the specific procedure for incorporating the simultaneous Magnanti-Wong method to solve the proposed CFD-ISP model. Section 3.5 presents comprehensive numerical experiments to validate our models and algorithms. Finally, Section 3.6 offers a summary and discusses potential future research directions.

3.2 Literature review

The CFD-ISP falls within the broader category of cruise management and belongs to the integrated optimization problem of fleet assignment and itinerary scheduling, which is well-suited for the BD solution method. To improve the computational efficiency of the BD algorithm, various acceleration methods have been proposed (Golari et al., 2017; Sudermann-Merx et al., 2021; Wu et al., 2023). Consequently, our literature review focuses primarily on three key areas: advancements in cruise management, common integrated problems, and recent enhancements of the BD algorithm.

3.2.1 Cruise management

While container shipping management has been extensively studied (Fransoo and Lee, 2013; Ding and Chou, 2015; Legros et al., 2019), the domain of cruise shipping remains notably underexplored in the relevant literature. Wang et al. (2016) provide a comprehensive review on cruise shipping, analyzing current trends in the cruise sector and demonstrating that it is relatively young with significant potential for improvement and growth.

Early research regarding cruise management usually adopts data analysis and empirical study methods, where some qualitative conclusions regarding the global cruise network are usually drawn (Soriani et al., 2009; Véronneau and Roy, 2009; Gui and Russo, 2011; Rodrigue and Notteboom, 2013; Santos et al., 2021).

Recently, cruise management has gained increasing attention, with researchers actively

employing optimization models to enhance cruise management strategies. Biehn (2006) argues that cruise ship cannot be regarded as a floating hotel and develops a deterministic linear programming (LP) model to optimize revenue for cruise ships. Based on this study, Maddah et al. (2010) develop a discrete-time dynamic capacity control model for cruise ships that addresses multiple capacity constraints, specifically cabin and lifeboat capacities. Wang et al. (2017) develop a dynamic programming approach to solve the cruise itinerary scheduling problem by enumerating all port sequences and optimizing timings to maximize net utility. Zhen et al. (2018) propose a MIP model designed to schedule the voyage considering fuel cost optimization under emission control area regulations, where a tabu search based solution method is developed. Wang et al. (2018a) establish both deterministic and stochastic models for managing waste disposal in the cruise industry under various scenarios: static and dynamic itineraries. They develop a polynomial-time solution algorithm, which is adaptable across different models. Di Pillo et al. (2021) address the cruise itinerary scheduling problem at the tactical level and develop a MIP model, which successfully maximizes revenue for cruise ships by determining optimal cruise schedules within specific maritime areas and season windows, validated through real-world applications by a major cruise company. Nevertheless, in these previous studies, the decision types are limited, typically encompassing only a single level of decision-making and focusing on the narrow objective of minimizing fuel costs.

3.2.2 Related integrated problems

Although integrated decision-making in CFD-ISP remains largely unexplored, it falls within the scope of scheduled service network design with resource acquisition and management (SSND-RAM) (Crainic and Hewitt, 2021), which integrates tactical decisions related to service network design and strategic decisions related to resource acquisition and allocation. In this section, we provide a brief overview of representative methodolog-

ical approaches and studies of SSND-RAM in the transportation sector. We then highlight why current methodologies developed for SSND-RAM in urban rail transit cannot be directly applied to the CFD-ISP context.

Due to the NP-hard nature of the SSND-RAM problem, previous studies have emphasized the need for solution methods capable of efficiently generating near-optimal solutions for practical-sized instances, often embedding heuristics and metaheuristics in the solution algorithms. Crainic et al. (2016) are the first to introduce the integrated SSND-RAM formulation in freight transportation. Their model uses a time–space network to jointly optimize service selection, commodity routing, and resource allocation under terminal capacity constraints. To solve this complex problem, they propose a hybrid approach that combines column generation, metaheuristics, and exact optimization methods, and validate its performance through computational experiments against both a commercial solver and a column generation-based heuristic. Building on this foundation, Crainic et al. (2018) propose a matheuristic solution method for the SSND-RAM problem and conduct extensive computational experiments to explore the interactions between resource planning and network design under varying demand and cost scenarios. Addressing uncertainty, Hewitt et al. (2019) formulate a stochastic model for the SSND-RAM problem that incorporates demand variability. A column generation-based matheuristic is developed to efficiently solve the model, demonstrating strong performance on realistic freight and postal network instances. Recently, a few studies have used exact methods to obtain higher quality solutions for SSND-RAM (Fontaine et al., 2021; Li et al., 2023b), which, however, differ significantly from the specific problem setting and decision-making content of this paper, as they often address practical and context-dependent challenges. More recently, several studies have adopted exact methods to obtain higher quality solutions for SSND-RAM (Fontaine et al., 2021; Li et al., 2023b). However, these works typically focus on specific, context-dependent applications and differ significantly from the problem scope and decision-making structure considered in this paper. For comprehensive reviews of the

SSND-RAM problem, particularly in the context of transit network design and scheduling, we refer readers to Guihaire and Hao (2008) and Farahani et al. (2013).

A subset of SSND-RAM studies more closely aligned with our context involves applications in urban rail transit systems. Such studies always consider the integrated optimization of fleet size and rolling stock circulation by establishing IP models (Pu et al., 2022; Yin et al., 2023a). However, these studies are generally limited to smaller scales, such as instances with only 4 lines and 7 depots in Pu et al. (2022) by commercial solvers or 11 lines and 13 depots in Yin et al. (2023a) by the BD algorithm. Meanwhile, in cruise management, complexity escalates as companies have to manage more than 50 cruise lines, thus increasing network complexity and computational challenges.

3.2.3 Accelerations in Benders decomposition

We provide a concise overview of various strategies in the literature that aim to enhance and expedite the BD within the MIP framework. According to Rahmaniani et al. (2017), there are four main categories of acceleration strategies, among which, enhancing the cut-generation process is mostly discussed (Magnanti and Wong, 1981; Shen et al., 2017; Rahmaniani et al., 2018; Perrykkad et al., 2022) and widely used in various problems (Wu et al., 2023; Yin et al., 2023a; Li et al., 2023a).

One branch of studies proposes some valid inequalities to complement the feasibility and optimality cuts (Shen et al., 2017; Rahmaniani et al., 2018), whereas another branch follows the idea of Magnanti and Wong (1981) by generating Pareto-optimal cuts. Magnanti and Wong (1981) first introduce the concept of Pareto-optimal cuts when a feasible master problem solution is associated with multiple optimal solutions (i.e., extreme points) from the dual SP. In their approach, often referred to as the Magnanti-Wong method, an additional dual SP, called Magnanti-Wong SP, using a key from the master, is solved to identify a Pareto-optimal cut. The introduction of Pareto-optimal cuts substantially en-

hanced the BD algorithm by effectively leveraging the lower bound of the master problem. Based on their study, Papadakos (2008) demonstrate that certain alternative points, which are easily computed, could serve as substitutes for core points in solving the SP to produce Pareto-optimal cuts. Fischetti et al. (2010) offer a distinct approach to generating Pareto-optimal cuts by redefining the SP as a feasibility problem, where cuts, both for optimality and feasibility, are derived by identifying minimal infeasible subsystems. Later, Sherali and Lunday (2013) advance these techniques to create maximal nondominated cuts. While the Magnanti-Wong method is widely used, Perrykkad et al. (2022) point out that the main shortcomings of this method include (i) two LPs to be solved and (ii) loss of SP structure. Therefore, they propose a simultaneous Magnanti-Wong method to address the aforementioned points. The key to this method is to obtain a *Magnanti-Wong bound* that can be computed in the pre-processing stage, under which solving one simultaneous Magnanti-Wong SP is equivalent to solving the SP and the Magnanti-Wong SP sequentially, thereby protecting the original structure of the SP and avoiding solving two LPs in each iteration. Although well-defined in their problem, it is worth noting that determining a Magnanti-Wong bound is as difficult as solving the two phases separately, which is unlikely to be possible for all LPs (Sherali and Soyster, 1983), thereby restricting its applicability to broader problems.

3.2.4 Research gap

First, despite the increasing body of research focused on cruise schedule optimization using mathematical models, the current literature predominantly focuses on optimization at the operational level, such as the cruise itinerary scheduling problem, with a notable lack of comprehensive approaches that integrate broader strategic and tactical considerations. Besides, most existing optimization models for cruise management consider only a narrow range of factors, primarily focusing on elements like fuel costs. To fill this gap, this paper

addresses the integrated CFD-ISP that optimizes both tactical and operational decisions in cruise management considering various cost- and revenue-related elements, where we derive essential management implications through a real case study.

Second, while similar integrated optimization problems have been explored in other sectors, such as the railway and aircraft industries, their model and solution methods are restricted on limited scales and cannot perfectly suit our specific needs in the cruise industry. Consequently, we propose an IP model for our CFD-ISP, where some critical model properties (TU property of the coefficient matrix) and reformulation methods (route-based reformulation) are drawn, which largely reduce the complexity of the model and can also be applied to similar problems in other fields such as rolling stock.

Third, while the simultaneous Magnanti-Wong method stands out as an effective acceleration technique for the BD algorithm, the absence of a generalized approach for computing the Magnanti-Wong bound hinders its broader application. To tackle our integrated model on a practical scale, we develop a BD algorithm based on the simultaneous Magnanti-Wong method, where a problem-specific Magnanti-Wong bound is innovatively designed.

3.3 Problem description

We first describe the CFD-ISP in Section 3.3.1. Then, we construct the time-expanded network of the CFD-ISP in Section 3.3.2. The mathematical formulation of this problem is given in Section 3.3.3.

3.3.1 Integrated CFD-ISP

The cruise deployment problem is to deploy a total of $Z_{\max}$ identical cruises to different home ports and then allocate to the lines starting from the home ports. Notably, the cruise deployed to different lines cannot be shared during the planning horizon. We denote $\mathcal{L}^b$

and $\mathcal{L}^r$ as the set of bi-directional and round-trip itineraries, respectively. For each line $l \in \mathcal{L} = \mathcal{L}^b \cup \mathcal{L}^r$, we denote the set of visited ports by $\mathcal{P}_l$, which includes the set of home port(s) ($\mathcal{H}_l$) and the set of ports of call ($\mathcal{S}_l$), also referred to as station ports. Notably, for a bi-directional line $l \in \mathcal{L}^b$, the set of station ports is $\mathcal{S}_l = \mathcal{S}_l^0 \cup \mathcal{S}_l^1$, where $\mathcal{S}_l^0$ and $\mathcal{S}_l^1$ are the station ports in direction 0 and 1, respectively. For example, in Figure 3.2, there is a bi-directional cruise line "Shanghai–Jeju Island–Nagasaki–Fukuoka–Beppu–Shimizu–Tokyo," with two directions: Shanghai to Tokyo (direction 0) and Tokyo to Shanghai (direction 1). Then, each station port $s \in \mathcal{S}_l^0$ representing a port in direction 0 shares the same physical port with a port $s' \in \mathcal{S}_l^1$ in the other direction 1 (Wang et al., 2018b). For notation convenience, we define $\mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2 \cdots \cup \mathcal{H}_{|\mathcal{L}|}$, $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cdots \cup \mathcal{S}_{|\mathcal{L}|}$, and $\mathcal{P} = \mathcal{H} \cup \mathcal{S}$. Different lines may share the same home port(s).

Based on the cruise deployment decisions, the itinerary scheduling problem is to determine the specific timetable for each cruise line during the planning horizon. We assume that the sailing time between two ports (denoted by $t_{p,q}^{\text{sail}}$, where $p, q \in \mathcal{P}_l$ are two successive ports along the same line $l \in \mathcal{L}$), the touring time at each station port (denoted by $t_{s,l}^{\text{opp}}$, $s \in \mathcal{S}_l, l \in \mathcal{L}$), and the layover time at each home port (denoted by $t_{h,l}^{\text{layover}}$, $h \in \mathcal{H}_l, l \in \mathcal{L}$) are given. These assumptions align with real-world applications, as the cruise company already offers several optional cruise lines with predefined tourism plans but needs to determine the times to provide these services.

Passenger demand, measured by the number of tourists, varies considerably among cruise lines and times of year. This variation can be described along two dimensions: cruise line and travel period. First, demand is associated with a specific cruise line $l \in \mathcal{L}$, which defines a fixed sequence of port visits that tourists choose as their travel itinerary. Second, demand is also linked to a preferred period $k \in \mathcal{K}$, indicating when tourists wish to take their trip. For instance, the number of passengers tends to increase during holidays and on Tokyo lines during the cherry blossom season. To account for these seasonal fluctuations, the entire planning horizon is divided into a set of periods, such as weeks or

months, represented by $\mathcal{K}$. The length of each period may vary, depending on the granularity needed to accurately reflect demand patterns. Together, each combination (l, k) thus defines a particular demand, specifying both the route and the time of the desired trip. Based on historical data, the expected number of passengers starting their journey on the round-trip line $k \in \mathcal{K}$ of a round trip $l \in \mathcal{L}^r$ is denoted by $d_{l,k}$. Similarly, for a bidirectional line $l \in \mathcal{L}^b$, we denote the required numbers of cruises for each direction in period k as $d^0_{l,k}$ and $d^1_{l,k}$, respectively.

For cruise companies, the objective is to jointly optimize strategic-level cruise deployment decisions and tactical-level itinerary scheduling decisions to minimize total costs while maintaining service quality. Total costs encompass not only direct expenditures, such as deployment costs and sailing costs associated with cruise operations, but also the opportunity costs incurred due to unmet demand. Beyond these cost-related factors, the chosen schedule also significantly influences the overall cruise experience. In particular, the timing of port visits plays a vital role in passenger satisfaction. For example, schedules that allow for sightseeing during the day and provide rest periods at night are generally preferred by tourists. To capture this aspect of service quality, we incorporate a measure of passenger satisfaction into the objective function by translating it into a utility-based revenue component, following the framework of Wang et al. (2017). The alignment between scheduled port visit times and passenger preferences thus directly contributes to the overall profitability of the service, balancing cost minimization with utility-driven revenue maximization.

3.3.2 Time-expanded network for CFD-ISP

To model the movement of cruises, we divide the planning horizon into distinct timestamps, represented by $\mathcal{T} = \{0, 1, \cdots, T\}$, where the timestamp 0 represents the start of the planning horizon, T represents the end of the planning horizon. The unit of time is

generally an hour, as this granularity suffices for the planning of cruise itineraries (Wang et al., 2017). The decision-makers can choose this discretization parameter appropriately to achieve a trade-off between model accuracy and solving efficiency. Note that the set of time periods $\mathcal{K}$ and timestamps $\mathcal{T}$ serve distinct roles in the modeling framework. Time periods characterize the temporal structure of demand, capturing broader seasonal or cyclical patterns relevant to the planning horizon (e.g., weekly, or monthly variations). In contrast, timestamps represent the discretized time units of the time-expanded network that define the resolution of the operational timeline. They are used to track and schedule specific events and decisions at a finer temporal granularity, often on the scale of minutes or hours. We adopt $\mathcal{T}_t = \{t' \in \mathcal{T} | t' \leq t\}$ to represent the set of timestamps at or before time t, which is indexed by τ. We also denote the set of timestamps contained in a time period $k \in \mathcal{K}$ as $\widetilde{\mathcal{T}}_k = \{\underline{\tau}(k), \cdots, \overline{\tau}(k)\}$, where $\underline{\tau}(k)$ and $\overline{\tau}(k)$ represent the first and last timestamps of time period k, respectively.

After breaking down the planning horizon into these timestamps, we then create a time-expanded network $G(\mathcal{N}, \mathcal{A})$, where each node $n_{p,t} \in \mathcal{N}$ denotes a port $p \in \mathcal{P}$ at the specific timestamp $t \in \mathcal{T}$, and $\mathcal{A}$ denotes set of arcs in the network. We have $\mathcal{A} = \cup_{l \in \mathcal{L}} \mathcal{A}_l$, where $\mathcal{A}_l$ is the arc set of cruise line l including the following three types of arcs:

- *station-station port arc*: An arc $(n_{s,t}, n_{s',t'}, l)$, where $s, s' \in \mathcal{S}_l$ are two successive station ports in the same line l. And we have $t' - 1 < t + t^{\text{sail}}_{s,s'} + t^{\text{opp}}_{s',l} \leq t'$.

- *station-home port arc*: $(n_{s,t}, n_{h,t'}, l)$, where $s \in \mathcal{S}_l$ stands for the last station port and $h \in \mathcal{H}_l$ represents the destination home port of line l. And we have $t' - 1 < t + t^{\text{sail}}_{s,h} + t^{\text{layover}}_{h,l} \leq t'$.

- *home-station port arc*: $(n_{h,t}, n_{s,t'}, l)$, where $h \in \mathcal{H}_l$ stands for the origin home port and $s \in \mathcal{S}_l$ represents the first station port of line l. And we have $t' - 1 < t + t^{\text{sail}}_{h,s} + t^{\text{opp}}_{s,l} \leq t'$.

It should be noted that in our model, we assign a capacity of one to each arc $a \in \mathcal{A}$, signifying that only one cruise can leave port p to port q on the same line l at any given timestamp t. This constraint, also seen in similar problems such as rolling stock (Yin et al., 2023a), is due to various resource limitations, including berth availability at ports and staffing constraints. For modeling convenience, we define $o(a)$ and $d(a)$ as the origin and destination port of arc a, respectively; $t_1(a)$ and $t_2(a)$ as the time of the origin and destination node of arc a, respectively.

Then, we can further present the following arcs:

- *outflow arc of* $n_{p,t}$ *for line* l: $\delta^+(n_{p,t}, l) = \{a \in \mathcal{A}_l | o(a) = p, t_1(a) = t\}$, where $p \in \mathcal{P}, t \in \mathcal{T}$.
- *inflow arc of* $n_{p,t}$ *for line* l: $\delta^-(n_{p,t}, l) = \{a \in \mathcal{A}_l | d(a) = p, t_2(a) = t\}$, where $p \in \mathcal{P}, t \in \mathcal{T}$.

Notably, $\delta^+(n_{p,t}, l)$ ($\delta^-(n_{p,t}, l)$, respectively) is either a singleton or empty, which indicates that there is either one or no outflow (inflow, respectively) arc of $n_{p,t}$ for line l. This signifies that only one cruise for a line l can leave or arrive at a port p at a certain timestamp t, which is consistent with our previous assumption due to various resource limitations, including berth availability at ports and staffing constraints. Besides, the outflow arcs of a home port h of line $l \in \mathcal{L}^r$ during time period k can be presented as $\mathcal{A}_{l,k} = \{a \in \mathcal{A}_l | o(a) = h, t_1(a) \in \widetilde{\mathcal{T}}_k\}$. Similarly, for a bi-directional line, we use $\mathcal{A}^0_{l,k}$ and $\mathcal{A}^1_{l,k}$ to present the set of outflow arcs of the origin home port h in direction 0 and 1, respectively.

Example 2. *We give an illustrative example of the time-expanded construction. Suppose that there is a round-trip line* l_r*: "A–B–C–D–A" and a bi-directional line* l_b*: "E–F–G–A" (direction 0), "A–G′–F′–E" (direction 1). Ports G and F in direction* 0 *share the same physical port with G′ and F′ in the other direction* 1*, respectively. We discretize*

the planning horizon into $\mathcal{T} = \{0, 1, \cdots, 8\}$. *The sailing times between two successive ports are set to 0.5, and the touring times of all station ports are 0.5. We assume the layover times* $t^{layover}_{A,l_r} = 1.5$, $t^{layover}_{A,l_b} = 0.5$, *and* $t^{layover}_{E,l_b} = 2.5$. *Then, we can construct a time-expanded network shown in Figure 3.3.*

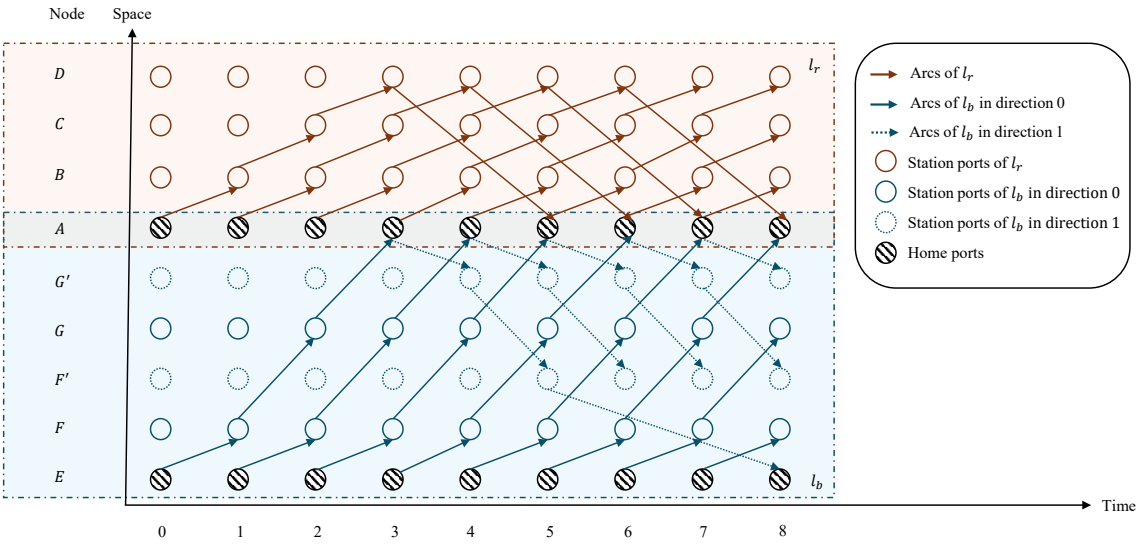


Figure 3.3: An example of the time-expanded network

3.3.3 Mathematical model

As our previous discussion revealed, we jointly consider a cruise company's medium-term fleet deployment alongside its short-term itinerary scheduling decisions within the context of the predefined strategic framework. Therefore, the decision variables of CFD-ISP include:

- z_h: nonnegative integer, indicating the number of cruises allocated at home port h.
- $u_{h,l}$: nonnegative integer, indicating the number of cruises allocated at home port h and assigned to line l.
- x_a: binary, taking 1 if arc a is included in the cruises' itineraries and 0, otherwise.

- $y_{l,k}$: nonnegative integer, indicating the number of unsatisfied demands of a round-trip itinerary $l \in \mathcal{L}^r$ at time period k.

- $y^i_{l,k}$: nonnegative integer, indicating the number of unsatisfied demands of a bi-directional itinerary $l \in \mathcal{L}^b$ at time period k in direction $i \in \{0, 1\}$.

In the model, we use $\mathbb{N}_0$ to represent the set of nonnegative integers. Besides, we let $\boldsymbol{z} = (z_h)_{h\in\mathcal{H}}$, $\boldsymbol{u} = (u_{h,l})_{h\in\mathcal{H},l\in\mathcal{L}}$, $\boldsymbol{x} = (x_a)_{a\in\mathcal{A}}$, and $\boldsymbol{y} = ((y_{l,k})_{l\in\mathcal{L}^r,k\in\mathcal{K}}, (y^i_{l,k})_{l\in\mathcal{L}^b,k\in\mathcal{K},i\in\{0,1\}})$ be the vector of each kind of decision variable.

The objective of CFD-ISP comprises two parts, tactical-level and operational-level costs. The first part is the fixed cost of cruise deployment, which can be presented by

$$C_1 = \sum_{h\in\mathcal{H}} c^{\text{loc}}_h z_h, \tag{3.1}$$

where c^{loc}_h is the cost of locating a cruise at home port h. The second part is the overall cost of the itinerary schedule, which consists of the cost of cruise sailing, the opportunity cost, and the minus revenue of service utility. We let c^{fuel}_a be the fuel cost of traveling through arc a, $c^{\text{opp}}_{l,k}$ be the penalty of unsatisfied demand of line l during time period k, and r^{uti}_a be the utility of sailing through arc a. Notably, r^{uti}_a corresponds to visiting port $d(a)$ during a certain time period, usually reflecting the passengers' preference for daytime sightseeing activities and nighttime rest back on the cruise. We give an example of how to calculate the utility of arc a as follows.

Example 3. *Given an arc $a \in \mathcal{A}_l$ such that $d(a) \in \mathcal{S}_l$, we present the utility distribution of passengers visiting port $d(a)$ during different times of a day in Figure 3.4. The touring time at port $d(a)$, i.e., $t^{opp}_{d(a),l}$, is 10 hours, lasting from 8:00 to 18:00. Therefore, the utility of arc a can be calculated by summing up the utility from 8:00 to 18:00, which equals 149 dollars.*

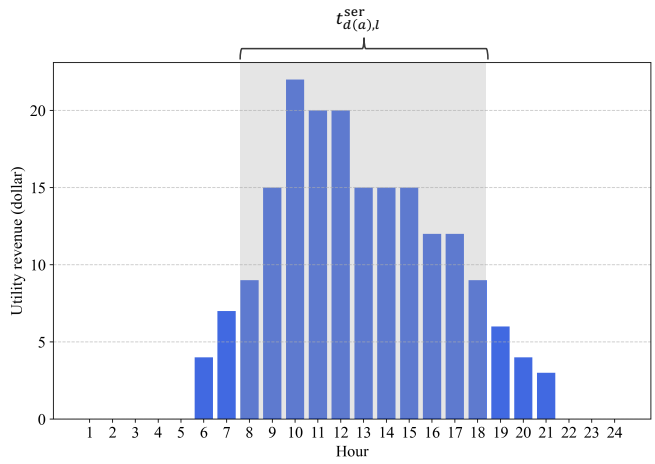


Figure 3.4: The utility revenue distribution and touring time at one port

Then, the second part of the overall operational cost can be written as

$$C_2 = \sum_{a\in\mathcal{A}}(c_a^{\text{fuel}} - r_a^{\text{uti}})x_a + \sum_{l\in\mathcal{L}^r}\sum_{k\in\mathcal{K}} c_{l,k}^{\text{opp}} y_{l,k} + \sum_{l\in\mathcal{L}^b}\sum_{i\in\{0,1\}}\sum_{k\in\mathcal{K}} c_{l,k}^{\text{opp}} y_{l,k}^{i}. \tag{3.2}$$

Key notation for parameters, sets, and decision variables used in the model formulation are provided in G. The mathematical model of CFD-ISP is presented below.

$$[\text{OP}] \quad \min_{\boldsymbol{z},\boldsymbol{u},\boldsymbol{x},\boldsymbol{y}} : C_1 + C_2 \tag{3.3a}$$

$$\text{s.t.} \sum_{h\in\mathcal{H}} z_h \le Z_{\max}, \tag{3.3b}$$

$$\sum_{l\in\mathcal{L}_h} u_{h,l} \le z_h \qquad \forall h \in \mathcal{H}, \tag{3.3c}$$

$$\sum_{\tau\in\mathcal{T}_t}\Big(\sum_{a\in\delta^+(n_{h,\tau},l)} x_a - \sum_{a\in\delta^-(n_{h,\tau},l)} x_a\Big) \le u_{h,l} \qquad \forall l \in \mathcal{L}, h \in \mathcal{H}_l, t \in \mathcal{T}\backslash\{T\}, \tag{3.3d}$$

$$\sum_{\tau\in\mathcal{T}}\Big(\sum_{a\in\delta^+(n_{h,\tau},l)} x_a - \sum_{a\in\delta^-(n_{h,\tau},l)} x_a\Big) = 0 \qquad \forall l \in \mathcal{L}, h \in \mathcal{H}_l, \tag{3.3e}$$

$$\sum_{a\in\delta^+(n_{s,t},l)} x_a - \sum_{a\in\delta^-(n_{s,t},l)} x_a = 0 \qquad \forall l \in \mathcal{L}, s \in \mathcal{S}_l, t \in \mathcal{T}, \tag{3.3f}$$

$$\sum_{a\in\mathcal{A}_{l,k}} x_a + y_{l,k} = d_{l,k} \qquad \forall l \in \mathcal{L}^r, k \in \mathcal{K}, \tag{3.3g}$$

$$\sum_{a \in \mathcal{A}^i_{l,k}} x_a + y^i_{l,k} = d^i_{l,k} \qquad \forall i \in \{0,1\}, l \in \mathcal{L}^b, k \in \mathcal{K}, \tag{3.3h}$$

$$z_h, u_{h,l} \in \mathbb{N}_0 \qquad \forall h \in \mathcal{H}, l \in \mathcal{L}_h, \tag{3.3i}$$

$$y_{l,k}, y^i_{l',k} \in \mathbb{N}_0 \qquad \forall i \in \{0,1\}, l \in \mathcal{L}^r, l' \in \mathcal{L}^b, k \in \mathcal{K}, \tag{3.3j}$$

$$x_a \in \{0,1\} \qquad \forall a \in \mathcal{A}. \tag{3.3k}$$

Objective (3.3a) minimizes the sum of tactical and operational costs. Constraint (3.3b) restricts that the total number of deployed cruises should not exceed the limit $Z_{\max}$ defined by strategic-level decision. Constraints (3.3c) regulate that the number of cruises allocated to all lines associated with each home port should not surpass the available number of cruises deployed there. Constraints (3.3d) represent the used cruises along each line should be no larger than the deployed number of cruises on this line. Constraints (3.3e) control that at the end of the planning horizon, all cruises on a cruise line should go back to their home port. Constraints (3.3f) are the flow-balance constraints for each time-expanded node of station ports. Constraints (3.3g) and (3.3h) are the demand constraints for round-trip and bi-directional itineraries, respectively. Constraints (3.3i)–(3.3k) define the domains of the variables. We note that the strategic decision on the number of cruises is set as a parameter in our model, as they are often long-term and predetermined in practice; however, our model can readily incorporate this decision by treating $Z_{\max}$ as a variable and associating a corresponding cost with it through minor modifications.

3.4 Benders decomposition algorithm

This section first introduces the BD for the proposed model of CFD-ISP, where we reformulate the itinerary scheduling problem by employing cumulative-flow-based variables and introducing route-based representation. Then, the BD algorithm with tailored accel-

erating strategies for CFD-ISP is developed.

3.4.1 Model decomposition

We apply the BD algorithm to our CFD-ISP and decompose the original model, i.e., OP, into the *fleet deployment problem*, i.e., RMP, and the *itinerary scheduling problem*, i.e., SPs. Notably, given the assigned number of cruises of each line l, its itinerary scheduling problem can be solved separately. Therefore, we can further decouple the itinerary scheduling problem into $|\mathcal{L}|$ independent SPs. Specifically, for each iteration, the assigned number of cruises of each line at different home ports $u_{h,l}$ is fixed as $\hat{u}_{h,l}$. Each SP is an IP model characterized by the vector $\hat{\boldsymbol{u}}$. For a round-trip $l \in \mathcal{L}^r$, the $\text{SP}_l(\hat{\boldsymbol{u}})$ can be presented as below.

$$[\text{SP}_l(\hat{\boldsymbol{u}})] \quad \min_{\boldsymbol{x},\boldsymbol{y}} : \sum_{a\in\mathcal{A}_l}(c_a^{\text{fuel}} - r_a^{\text{uti}})x_a + \sum_{k\in\mathcal{K}} c_{l,k}^{\text{opp}} y_{l,k} \tag{3.4a}$$

$$\text{s.t.} \sum_{\tau\in\mathcal{T}_t}\Big(\sum_{a\in\delta^+(n_{h,\tau},l)} x_a - \sum_{a\in\delta^-(n_{h,\tau},l)} x_a\Big) \le \hat{u}_{h,l} \quad \forall h\in\mathcal{H}_l, t\in\mathcal{T}\backslash\{T\}, \tag{3.4b}$$

$$\sum_{\tau\in\mathcal{T}}\Big(\sum_{a\in\delta^+(n_{h,\tau},l)} x_a - \sum_{a\in\delta^-(n_{h,\tau},l)} x_a\Big) = 0 \quad \forall h\in\mathcal{H}_l, \tag{3.4c}$$

$$\sum_{a\in\delta^+(n_{s,t},l)} x_a - \sum_{a\in\delta^-(n_{s,t},l)} x_a = 0 \quad \forall s\in\mathcal{S}_l, t\in\mathcal{T}, \tag{3.4d}$$

$$\sum_{a\in\mathcal{A}_{l,k}} x_a + y_{l,k} = d_{l,k} \quad \forall k\in\mathcal{K}, \tag{3.4e}$$

$$y_{l,k} \in \mathbb{N}_0 \quad \forall k\in\mathcal{K}, \tag{3.4f}$$

$$x_a \in \{0,1\} \quad \forall a\in\mathcal{A}_l. \tag{3.4g}$$

Similarly, for a bi-directional itinerary $l \in \mathcal{L}^b$, we also have model $\text{SP}_l(\hat{\boldsymbol{u}})$ except that the objective (3.4a) is modified into:

$$\min_{\boldsymbol{x},\boldsymbol{y}} : \sum_{a \in \mathcal{A}_l} (c_a^{\text{fuel}} - r_a^{\text{uti}}) x_a + \sum_{i \in \{0,1\}} \sum_{k \in \mathcal{K}} c_{l,k}^{\text{opp}} y_{l,k}^i, \tag{3.5}$$

and the constraints (3.4e) and (3.4f) are replaced by:

$$\sum_{a \in \mathcal{A}_{l,k}^i} x_a + y_{l,k}^i = d_{l,k}^i \qquad \forall i \in \{0,1\}, k \in \mathcal{K}, \tag{3.6}$$

$$y_{l,k}^i \in \mathbb{N}_0 \qquad \forall i \in \{0,1\}, k \in \mathcal{K}. \tag{3.7}$$

For simplicity, in the following descriptions, we only discuss the round-trip cruise line $l \in \mathcal{L}^r$ and its itinerary scheduling problem $\text{SP}_l(\hat{\boldsymbol{u}})$, as the problem structure and solving method for the bi-directional cruise line $l \in \mathcal{L}^b$ remain similar.

Remark 1. *Despite the above decomposition, the original problem remains difficult to solve for the following reasons:*

- *The $SP_l(\hat{\boldsymbol{u}})$ is an IP model that cannot be dualized and thus cannot be used directly to generate Benders cuts.*
- *Solving the $SP_l(\hat{\boldsymbol{u}})$ takes a considerable amount of time since there are many integer variables and summation operations over each time point in constraints* (3.4b)–(3.4c)*. For example, if we consider a planning horizon of 3 months with time discretized into 1 hour, there will be 2160 timestamps, leading to computational burden-related issues.*

Considering the complexity of the SPs, we reformulate them by employing the cumulative flow variables, utilizing techniques similar to those discussed in previous studies on other network flow models (Shi and Zhou, 2015; Mahmoudi et al., 2019; Yin et al., 2023a).

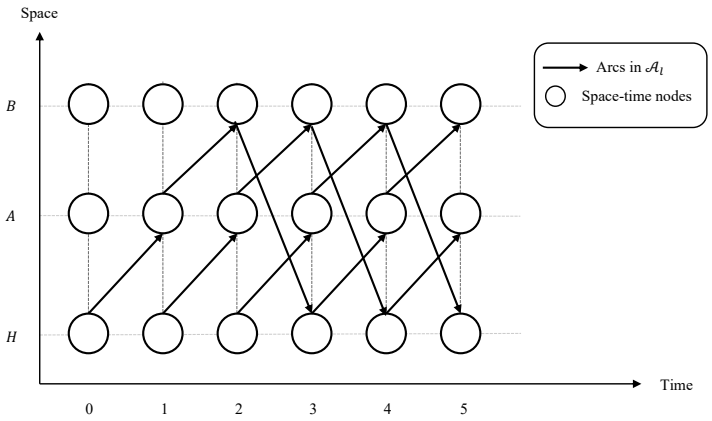
Moreover, we further propose a route-based model, where the numbers of time-expanded nodes and arcs are greatly reduced. Based on the reformulated model, we prove that the new SPs can be solved equivalently by relaxing the integer variables, whose coefficient matrix is proved to be TU. The specific model reformulations and model property analysis are given in the subsequent section.

3.4.1.1 Subproblem reformulation

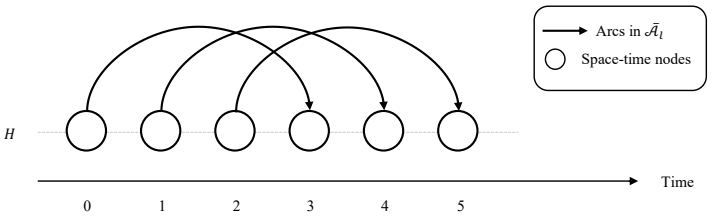
Route-based reformulation Considering that the visiting sequence and touring time of each cruise line is given in advance, we transform the original arc-based itinerary scheduling problem into a route-based one. Specifically, provided that the cruise leaves its home port h for itinerary l at time t, we can denote its visiting arcs as $\mathcal{A}_l(h,t)$. The total traveling time of a cruise line l can be calculated by $T_l = \sum_{a\in\mathcal{A}_l(h,t)} t^{\text{sail}}_{o(a),d(a)} + \sum_{s\in\mathcal{S}_l} t^{\text{opp}}_{s,l} + t^{\text{layover}}_{h,l}$. Then, the total cost for a cruise to take an itinerary l at time t can be calculated based on the fuel cost and utility reward of each arc $a \in \mathcal{A}_l(h,t)$, i.e., the route cost denoted by $c^{\text{route}}_{l,t} = \sum_{a\in\mathcal{A}_l(h,t)}(c^{\text{fuel}}_a - r^{\text{uti}}_a)$. Thus, we can delete all time-expanded nodes and arcs of station ports and construct the outflow arc of each home port connecting directly to its target home port node. We denote $\bar{\mathcal{A}}_l$ as the set of arcs for line l in the route-based time-expanded network. An illustrative example of the route-based time-expanded network is provided below.

Example 4. *Suppose that the arc-based time-expanded network of a cruise line l "H-A-B-H" is presented as Figure 3.5(a). We can reformulate it into a route-based one as shown in Figure 3.5(b).*

Cumulative flow variable For each line l, we introduce a set of new variables $\chi_a \in \mathbb{N}_0, \forall a \in \bar{\mathcal{A}}_l$, signifying the cumulative flow occurring at and before arc a with the same origin and destination port. Specifically, for each $a \in \bar{\mathcal{A}}_l$, we define $\mathcal{A}(a) = \{a' \in \bar{\mathcal{A}}_l | o(a') = o(a), d(a') = d(a),\ t_1(a') \le t_1(a)\}$. Then, we have $\chi_a = \sum_{a'\in\bar{\mathcal{A}}(a)} x_{a'}$.



(a) Arc-based time-expanded network for line l



(b) Route-based time-expanded network for line l

Figure 3.5: An illustrative example of the reformulated time-expanded network

Additionally, we define χ_{a^-} as the cumulative flow between the same pair of ports occurring at least 1 timestamp before arc a, i.e., $\chi_{a^-} = \sum_{a' \in \mathcal{A}^-(a)} x_{a'}$, where $\mathcal{A}^-(a) = \{a' \in \bar{\mathcal{A}}_l | o(a') = o(a), d(a') = d(a), t_1(a') \leq t_1(a) - 1\}$. In simple terms, a^- is the adjacent arc one timestamp before arc a. Recall the definition of $\widetilde{\mathcal{T}}_k = \{\underline{\tau}(k), \cdots, \overline{\tau}(k)\}$, which is the set of timestamps contained in a time period k. We can present the cumulative flow of a home port h of line l before the start of time period k as χ_{a^-}, where $o(a) = h, t_1(a) = \underline{\tau}(k)$ and at the end of time period k as $\chi_{a'}$, where $o(a') = h, t_1(a') = \overline{\tau}(k)$.

Additionally, for modeling rigor, we clarify the following boundary cases:

- $\chi_{a^-} = 0$ if $\mathcal{A}^-(a) = \emptyset$.
- We construct a dummy arc a for each node $n_{p,t}$ with $o(a) = n_{p,t}$ and $d(a) = n_{p,t+T_l}$, if $\delta^-(n_{p,t}, l) = \emptyset$ and $\delta^+(n_{p,t}, l) \neq \emptyset$. To ensure that there is no flow on the dummy arc, we set its arc-related cost $c_a^{\text{fuel}} - r_a^{\text{uti}}$ as $\max_{a' \in \bar{\mathcal{A}}_l} : (c_{a'}^{\text{fuel}} - r_{a'}^{\text{uti}}) + \sum_{k \in \mathcal{K}} c_{l,k}^{\text{opp}} d_{l,k}$. That is, if a cruise leaving home port at time t cannot finish the itinerary within the

planning horizon, its total cost $c_{l,t}^{route}$ would be an extremely large real number.

An illustrative example of the cumulative flow variable is provided below.

Example 5. *Considering a route-based time-expanded network for line l in Figure 3.6, according to the above definition, we have $\mathcal{A}(a_3) = \{a_1, a_2, a_3\}$. Then, we can calculate the cumulative flow occurring at and before arc a_3 by $\chi_{a_3} = x_{a_1} + x_{a_2} + x_{a_3}$. Also, we can present $\mathcal{A}^-(a_3) = \{a_1, a_2\}$ and $\chi_{a_3^-} = \chi_{a_2} = x_{a_1} + x_{a_2}$. For arc a_1, we have $\chi_{a_1} = x_{a_1}$ and $\chi_{a_1^-} = 0$. We need to construct dummy arcs for node $n_{H,3}$, $n_{H,4}$, $n_{H,5}$.*

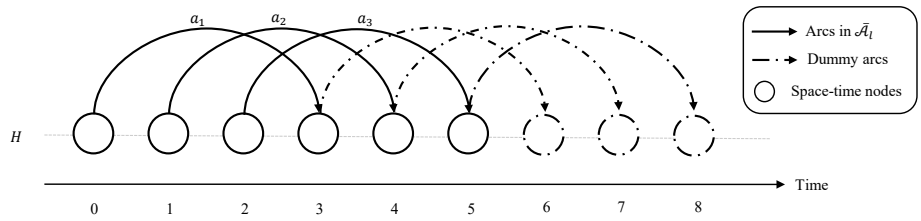


Figure 3.6: An illustrative example of cumulative flow variable for line l

Based on these reformulations, the $\text{SP}_l(\hat{\boldsymbol{u}})$ can be transformed as follows:

$$[\widetilde{\text{SP}}_l(\hat{\boldsymbol{u}})] \quad \min_{\boldsymbol{\chi},\boldsymbol{y}} : \sum_{t\in\mathcal{T}} \sum_{a\in\delta^+(n_{h,t},l)} c_{l,t}^{\text{route}}(\chi_a - \chi_{a^-}) + \sum_{k\in\mathcal{K}} c_{l,k}^{\text{opp}} y_{l,k} \tag{3.8a}$$

$$\text{s.t.} \sum_{a\in\delta^+(n_{h,t},l)} \chi_a - \sum_{a\in\delta^-(n_{h,t},l)} \chi_a \le \hat{u}_{h,l} \qquad \forall h \in \mathcal{H}_l, t \in \mathcal{T}\backslash\{T\}, \tag{3.8b}$$

$$\sum_{a\in\delta^+(n_{h,T},l)} \chi_a - \sum_{a\in\delta^-(n_{h,T},l)} \chi_a = 0 \qquad \forall h \in \mathcal{H}_l, \tag{3.8c}$$

$$\chi_a - \chi_{a^-} \ge 0 \qquad \forall a \in \bar{\mathcal{A}}_l, \tag{3.8d}$$

$$\chi_a - \chi_{a^-} \le 1 \qquad \forall a \in \bar{\mathcal{A}}_l, \tag{3.8e}$$

$$\sum_{a\in\delta^+(n_{h,\overline{\tau}(k)},l)} \chi_a - \sum_{a\in\delta^+(n_{h,\underline{\tau}(k)},l)} \chi_{a^-} + y_{l,k} = d_{l,k} \forall k \in \mathcal{K}, \tag{3.8f}$$

$$\chi_a, y_{l,k} \in \mathbb{N}_0 \qquad \forall k \in \mathcal{K}, a \in \bar{\mathcal{A}}_l. \tag{3.8g}$$

Objective (3.8a) corresponds to objective (3.4a), minimizing the operational cost of line l. Constraints (3.8b)–(3.8c) align with constraints (3.4b)–(3.4c), specifying the flow balance constraint for each homeport node. Constraints (3.8d) and (3.8e) restrict that the incremental flow between a^- and its next arc a is between 0 and 1. Constraints (3.8f) represent the demand requirement in each time period, where $\sum_{a\in\delta^+(n_{h,\overline{\tau}(k)},l)}\chi_a$ is the cumulative flow by the end of the time period k and $\sum_{a\in\delta^+(n_{h,\underline{\tau}(k)},l)}\chi_{a^-}$ is the flow accumulated before time period k. We have Lemma 1, the proof of which is detailed in I.

Lemma 1. *The original subproblem $SP_l(\hat{\boldsymbol{u}})$ is equivalent to the reformulated subproblem $\widetilde{SP}_l(\hat{\boldsymbol{u}})$.*

We further analyze the mathematical properties of the reformulated $\widetilde{\text{SP}}_l(\hat{\boldsymbol{u}})$. First, we state Lemma 2, the proof of which is given in J.

Lemma 2. *The coefficient matrix of $\widetilde{SP}_l(\hat{\boldsymbol{u}})$ is TU.*

Therefore, the integer variables can be relaxed, and $\widetilde{\text{SP}}_l(\hat{\boldsymbol{u}})$ can be written as an LP model as follows:

$$
\begin{aligned}
[\text{LSP}_l(\hat{\boldsymbol{u}})] \quad \min_{\boldsymbol{x},\boldsymbol{y}} : & \sum_{t\in\mathcal{T}}\sum_{a\in\delta^+(n_{h,t},l)} c^{\text{route}}_{l,t}(\chi_a-\chi_{a^-}) + \sum_{k\in\mathcal{K}} c^{\text{opp}}_{l,k} y_{l,k} \\
& \text{s.t. (3.8b)–(3.8f)}, \\
& \chi_a, y_{l,k} \geq 0 \qquad\qquad \forall k\in\mathcal{K}, a\in\bar{\mathcal{A}}_l.
\end{aligned}
$$

Furthermore, we show that the following Lemma 3 holds in K, which implies that we only need to generate Benders optimality cuts.

Lemma 3. *For any feasible RMP solution $\hat{\boldsymbol{u}}$, the subproblem $\widetilde{SP}_l(\hat{\boldsymbol{u}})$ is feasible and bounded.*

3.4.1.2 Master problem

To derive the master problem, we first write out the optimality cut. We associate dual variables $\boldsymbol{\alpha} = (\alpha_{h,t})_{h\in\mathcal{H},t\in\mathcal{T}}$, $\boldsymbol{\beta} = (\beta_h)_{h\in\mathcal{H}}$, $\boldsymbol{\gamma} = (\gamma_a)_{a\in\bar{\mathcal{A}}_l}$, $\boldsymbol{\lambda} = (\lambda_a)_{a\in\bar{\mathcal{A}}_l}$ and $\boldsymbol{\omega} = (\omega_k)_{k\in\mathcal{K}}$ with constraints (3.8b)–(3.8f), respectively. For modeling convenience, we define the indicator function $\mathbb{1}_X(x)$ that takes the value of 1 if $x \in X$, and 0 otherwise. Similar to the definition of a^- in Section 3.4.1.1, we let a^+ be the adjacent arc of a, where $o(a) = o(a^+)$, $d(a) = d(a^+)$, and $t_1(a) = t_1(a^+) - 1$. Then, the dual of $\text{LSP}_l(\hat{\boldsymbol{u}})$ can be presented as below.

$$[\text{DSP}_l(\hat{\boldsymbol{u}})] \quad \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\lambda},\boldsymbol{\omega}} : \sum_{h\in\mathcal{H}_l,t\in\mathcal{T}\setminus\{T\}} \hat{u}_{h,l}\alpha_{h,t} + \sum_{k\in\mathcal{K}} d_{l,k}\omega_k + \sum_{a\in\bar{\mathcal{A}}_l} \lambda_a \tag{3.9a}$$

$$\text{s.t.} \sum_{h\in\mathcal{H}_l}\sum_{t\in\mathcal{T}\setminus\{T\}} (\mathbb{1}_{\delta^+(n_{h,t},l)}(a) - \mathbb{1}_{\delta^-(n_{h,t},l)}(a))\alpha_{h,t} + \sum_{h\in\mathcal{H}_l} (\mathbb{1}_{\delta^+(n_{h,T},l)}(a) - \mathbb{1}_{\delta^-(n_{h,T},l)}(a))\beta_h$$
$$+ \gamma_a + \lambda_a - \mathbb{1}_{\bar{\mathcal{A}}_l}(a^+)(\gamma_{a^+} + \lambda_{a^+}) + \sum_{k\in\mathcal{K}} (\mathbb{1}_{\delta^+(n_{h,\overline{\tau}(k)},l)}(a) - \mathbb{1}_{\delta^+(n_{h,\underline{\tau}(k)},l)}(a^+))\omega_k$$
$$\leq \sum_{t\in\mathcal{T}} (\mathbb{1}_{\delta^+(n_{h,t},l)}(a) - \mathbb{1}_{\delta^+(n_{h,t},l)}(a^+))c^{\text{route}}_{l,t} \qquad \forall a \in \bar{\mathcal{A}}_l, \tag{3.9b}$$

$$\omega_k \leq c^{\text{opp}}_{l,k} \qquad \forall k \in \mathcal{K}, \tag{3.9c}$$

$$\boldsymbol{\alpha}, \boldsymbol{\lambda} \leq \mathbf{0}, \boldsymbol{\gamma} \geq \mathbf{0}. \tag{3.9d}$$

Given any optimal solution $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\omega}})$ to $\text{DSP}_l(\hat{\boldsymbol{u}})$, the Benders optimality cut can be written as:

$$\zeta_l \geq \sum_{h\in\mathcal{H}_l,t\in\mathcal{T}\setminus\{T\}} u_{h,l}\hat{\alpha}_{h,t} + \sum_{k\in\mathcal{K}} d_{l,k}\hat{\omega}_{l,k} + \sum_{a\in\bar{\mathcal{A}}_l} \hat{\lambda}_a, \tag{3.10}$$

where ζ_l is a nonnegative continuous variable representing a lower bound of the operational cost of line l.

Let Ψ_l represent the feasible region of the dual SP of line l. Then, we denote the set of

extreme points of Ψ_l by $\mathcal{E}_{\Psi_l}$. Through the above discussion, the *Benders master problem* can be written as follows:

$$[\text{MP}] \quad \min_{\boldsymbol{z},\boldsymbol{u},\boldsymbol{\zeta}} : \sum_{h\in\mathcal{H}} c_h^{\text{loc}} z_h + \sum_{l\in\mathcal{L}} \zeta_l \tag{3.11a}$$

$$\text{s.t.} \sum_{h\in\mathcal{H}} z_h \leq Z_{\max}, \tag{3.11b}$$

$$\sum_{l\in\mathcal{L}_h} u_{h,l} \leq z_h \qquad \forall h \in \mathcal{H}, \tag{3.11c}$$

$$\zeta_l \geq \sum_{h\in\mathcal{H}_l, t\in\mathcal{T}\setminus\{T\}} u_{h,l}\hat{\alpha}_{h,t} + \sum_{k\in\mathcal{K}} d_{l,k}\hat{\omega}_{l,k} + \sum_{a\in\bar{\mathcal{A}}_l} \hat{\lambda}_a \qquad \forall (\hat{\boldsymbol{\alpha}},\hat{\boldsymbol{\omega}},\hat{\boldsymbol{\lambda}}) \in \mathcal{E}_{\Psi_l}, l \in \mathcal{L}^r, \tag{3.11d}$$

$$\zeta_l \geq \sum_{h\in\mathcal{H}_l, t\in\mathcal{T}\setminus\{T\}} u_{h,l}\hat{\alpha}_{h,t} + \sum_{k\in\mathcal{K}} \sum_{i\in\{0,1\}} d_{l,k}^i \hat{\omega}_{l,k}^i + \sum_{a\in\bar{\mathcal{A}}_l} \hat{\lambda}_a \quad \forall (\hat{\boldsymbol{\alpha}},\hat{\boldsymbol{\omega}},\hat{\boldsymbol{\lambda}}) \in \mathcal{E}_{\Psi_l}, l \in \mathcal{L}^b, \tag{3.11e}$$

$$z_h, u_{h,l} \in \mathbb{N}_0 \qquad \forall h \in \mathcal{H}, l \in \mathcal{L}_h, k \in \mathcal{K}, \tag{3.11f}$$

where constraints (3.11d) and (3.11e) are the Benders optimality cuts.

3.4.2 Accelerating Benders decomposition algorithm

Based on the model decomposition in Section 3.4.1, we can solve the CFD-ISP by the classic BD algorithm. However, the $\text{DSP}_l(\hat{\boldsymbol{u}})$ may have multiple optimal solutions, and excessive valid Benders optimality cuts can be derived with some cuts weaker than others, increasing the number of iterations needed for convergence. Besides, given the thousands of arcs in the time-expanded network, solving the SP can be computationally challenging. Some studies also explore enhanced frameworks for Benders cut generation. For instance, Rahmaniani et al. (2020) propose the Benders dual decomposition (BDD) method, which strengthens the classic Benders decomposition method by reformulating the SP to include

local copies of the master variables. This reformulation enables the generation of stronger cuts and facilitates early convergence to high-quality integer solutions. However, applying BDD to our problem is inefficient, as it requires solving a non-decomposed SP, which is an IP involving all lines and becomes computationally expensive and intractable for large-scale instances, especially given the inherent difficulty of our SP. Furthermore, in our setting, decomposing the SP by line yields more effective and tighter cuts, as confirmed by the findings of Rahmaniani et al. (2017). In addition, Fischetti et al. (2010) show that Benders cut generation can be interpreted as identifying a minimal source of infeasibility in a model with a normalization constraint on the dual variables. However, their framework is primarily designed to unify the generation of feasibility and optimality cuts, which is less effective in our context because our subproblems are always feasible. Therefore, we employ the Magnanti-Wong method, which can generate Pareto-optimal cuts that are more likely to be binding and thus more impactful to strengthen the RMP, for our CFD-ISP (Magnanti and Wong, 1981). Since the Magnanti-Wong method needs to solve two LPs for each SP, we further develop a valid problem-specific bound, called the Magnanti-Wong bound, based on which the simultaneous Magnanti-Wong method can be applied to generate Pareto-optimal cuts by solving one LP for each SP (Perrykkad et al., 2022).

3.4.2.1 Magnanti-Wong method for CFD-ISP

For our problem, we say that a Benders optimality cut defined by dual solution $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\omega}})$ *dominates* another cut defined by dual solution $(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\omega}})$ if for every feasible $\boldsymbol{u}$ the inequality

$$\sum_{h\in\mathcal{H}_l, t\in\mathcal{T}\setminus\{T\}} u_{h,l}\hat{\alpha}_{h,t} + \sum_{k\in\mathcal{K}} d_{l,k}\hat{\omega}_{l,k} + \sum_{a\in\bar{\mathcal{A}}_l} \hat{\lambda}_a \geq \sum_{h\in\mathcal{H}_l, t\in\mathcal{T}\setminus\{T\}} u_{h,l}\bar{\alpha}_{h,t} + \sum_{k\in\mathcal{K}} d_{l,k}\bar{\omega}_{l,k} + \sum_{a\in\bar{\mathcal{A}}_l} \bar{\lambda}_a \tag{3.12}$$

always holds, with the inequality being strictly satisfied for at least one feasible $\boldsymbol{u}$. Furthermore, we refer to the cut defined by $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\omega}})$ as *Pareto-optimal* if it is not dominated by any other valid cut.

The Magnanti-Wong method constructs an LP model, known as the *Magnanti-Wong dual subproblem*, to generate the Pareto-optimal cuts. The main idea is to incorporate a core point, which is in the relative interior of the convex hull of the RMP's feasible region, to the objective coefficients of the original dual SP, while including an additional constraint that fixes the objective to the optimal objective value of the original dual SP. By first solving the original dual SP and then the Magnanti-Wong dual SP, the Pareto-optimal cuts can be generated in each iteration (Theorem 1 in Magnanti and Wong (1981)).

In our model, we obtain a core point denoted by $\boldsymbol{u}^0$, where each component $u^0_{h,l}$ is calculated by the formula $u^0_{h,l} = Z_{\max}/(2 \cdot |\mathcal{H}| \cdot |\mathcal{L}|)$ because it ensures that $\boldsymbol{u}^0 > 0$ and $\sum_{h\in\mathcal{H}} \sum_{l\in\mathcal{L}_h} u^0_{h,l} < Z_{\max}$. We denote $\hat{\eta}_l$ as the optimal objective value of the original dual SP, i.e., $\text{DSP}_l(\hat{\boldsymbol{u}})$. Then, the Magnanti-Wong dual SP can be presented as follows:

$$[\text{MW-DSP}_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\eta}})] \quad \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\lambda},\boldsymbol{\omega}} : \sum_{h\in\mathcal{H}_l, t\in\mathcal{T}\setminus\{T\}} u^0_{h,l}\alpha_{h,t} + \sum_{k\in\mathcal{K}} d_{l,k}\omega_k + \sum_{a\in\bar{\mathcal{A}}_l} \lambda_a \quad (3.13a)$$

$$\text{s.t. (3.9b)–(3.9d)},$$

$$\sum_{h\in\mathcal{H}_l, t\in\mathcal{T}\setminus\{T\}} \hat{u}_{h,l}\alpha_{h,t} + \sum_{k\in\mathcal{K}} d_{l,k}\omega_k + \sum_{a\in\bar{\mathcal{A}}_l} \lambda_a = \hat{\eta}_l. \quad (3.13b)$$

By solving the $\text{MW-DSP}_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\eta}})$, we can obtain a Pareto-optimal cut defined by $(\boldsymbol{\alpha}^*, \boldsymbol{\lambda}^*, \boldsymbol{\omega}^*)$. The pseudocode of the Magnanti-Wong BD is provided in L.

3.4.2.2 Simultaneous Magnanti-Wong method for CFD-ISP

To accelerate the Magnanti-Wong BD, this section applies the simultaneous Magnanti-Wong method and proposes a tailored method to compute the Magnanti-Wong bound for our CFD-ISP.

As described in Section 3.4.2.1, the classic Magnanti-Wong method is equivalent to a *lexicographic optimization problem*, where the $\text{DSP}_l(\hat{\boldsymbol{u}})$ is first solved to obtain the objective value $\hat{\eta}_l$ and then the optimal dual solution $(\boldsymbol{\alpha}^*, \boldsymbol{\lambda}^*, \boldsymbol{\omega}^*)$ is obtained by the Magnanti-Wong dual problem $\text{MW-DSP}_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\eta}})$. According to Sherali and Soyster (1983), for LPs, these two phases can be merged into a single optimization problem by assigning a *weight* to the objective functions. Perrykkad et al. (2022) extend this idea and develop a simultaneous Magnanti-Wong method, where the two-phase Magnanti-Wong method can be combined into a single optimization model, proving a valid lower bound of the weight, i.e., the Magnanti-Wong bound. This bound can be substituted directly into the Magnanti-Wong dual problem to generate Pareto-optimal cuts. Our two-phase Magnanti-Wong method can be transformed into a single optimization problem with Magnanti-Wong bound σ_l, which can be written as follows:

$$
\begin{aligned}
[\text{sMW-DSP}_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}})] \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\lambda},\boldsymbol{\omega}} : & \sum_{h\in\mathcal{H}_l, t\in\mathcal{T}\setminus\{T\}} u^0_{h,l}\alpha_{h,t} + \sum_{k\in\mathcal{K}} d_{l,k}\omega_k + \sum_{a\in\bar{\mathcal{A}}_l} \lambda_a \\
& + \sigma_l \left(\sum_{h\in\mathcal{H}_l, t\in\mathcal{T}\setminus\{T\}} \hat{u}_{h,l}\alpha_{h,t} + \sum_{k\in\mathcal{K}} d_{l,k}\omega_k + \sum_{a\in\bar{\mathcal{A}}_l} \lambda_a \right) \qquad (3.14a) \\
\text{s.t. } & (3.9b)\text{–}(3.9d).
\end{aligned}
$$

Then, we have the lemma below, the proof of which is given in M.

Lemma 4. *Provided a Magnanti-Wong bound σ_l, solving the sMW-DSP$_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}})$ is equivalent to solving the DSP$_l(\hat{\boldsymbol{u}})$ and MW-DSP$_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\eta}})$ sequentially.*

However, Perrykkad et al. (2022) demonstrate that calculating a Magnanti-Wong bound can be as computationally intensive as solving the two phases independently and the bound should be redesigned in different problem structures. We innovatively put forward a tailored Magnanti-Wong bound for our problem. For ease of description, we omit the subscript l in the subsequent discussion, which focuses on one SP. Additionally, we denote

$\boldsymbol{\psi}$ as the decision variable vector $(\boldsymbol{\alpha}, \boldsymbol{\lambda}, \boldsymbol{\omega})$ and $\boldsymbol{b}(\boldsymbol{u})$ as the objective coefficient vector of $\boldsymbol{\psi}$ in the DSP($\boldsymbol{u}$). We also denote the feasible region of $\boldsymbol{\psi}$ subject to the constraints of DSP($\boldsymbol{u}$) as Ψ. Based on the above notation, the DSP($\boldsymbol{u}$) can be re-written as

$$[\text{DSP}(\boldsymbol{u})] \quad \max_{\boldsymbol{\psi}} : \boldsymbol{\psi}^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u})$$
$$\text{s.t. } \boldsymbol{\psi} \in \Psi,$$

and the sMW-DSP($\boldsymbol{u}^0, \hat{\boldsymbol{u}}$) can be re-written as

$$[\text{sMW-DSP}(\boldsymbol{u}^0, \hat{\boldsymbol{u}})] \quad \max_{\boldsymbol{\psi}} : \boldsymbol{\psi}^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0) + \sigma \boldsymbol{\psi}^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}})$$
$$\text{s.t. } \boldsymbol{\psi} \in \Psi.$$

We let UB and LB be an upper bound and a lower bound of the objective value of the DSP($\boldsymbol{u}$) for all feasible $\boldsymbol{u}$. Then, we demonstrate that the UB and LB can be computed in the preprocessing stage of our algorithm, which are independent of $\boldsymbol{u}$, so that can be used over all iterations. Proposition 3 specifies the computational method of UB and LB. Specifically, UB is derived by assuming that no passenger demand is satisfied in any period and that no utility-related revenue is generated. This results in a purely cost-based evaluation that reflects the worst-case scenario in terms of system performance. As any other feasible solution would attempt to satisfy part of the demand and generate the corresponding revenue, its objective value must be lower than this bound, thus ensuring that the computed cost serves as a valid upper bound. In contrast, LB is computed under the optimistic assumption that the entire fleet is fully available at all home ports at all times. This allows for maximum operational flexibility and ensures that all demand can be met in a way that maximizes utility-related revenue. Consequently, the resulting objective value will be at least as high as that of any feasible solution to the original problem, thus providing a valid lower bound.

Proposition 3. *We can calculate* $UB = \sum_{k\in\mathcal{K}} d_k c_k^{opp}$ *and* $LB = \min_{\psi\in\Psi} : \sum_{h\in\mathcal{H},t\in\mathcal{T}\setminus\{T\}} Z_{\max}\alpha_{h,t} + \sum_{k\in\mathcal{K}} d_k\omega_k + \sum_{a\in\bar{\mathcal{A}}} \lambda_a$.

Proof. We can calculate the upper bound UB of $\boldsymbol{\psi}^{\mathrm{T}}\boldsymbol{b}(\boldsymbol{u})$ as follows:

$$\begin{aligned}
\boldsymbol{\psi}^{\mathrm{T}}\boldsymbol{b}(\boldsymbol{u}) =& \sum_{h\in\mathcal{H},t\in\mathcal{T}\setminus\{T\}} u_h\alpha_{h,t} + \sum_{k\in\mathcal{K}} d_k\omega_k + \sum_{a\in\bar{\mathcal{A}}}\lambda_a \\
\leq& \sum_{k\in\mathcal{K}} d_k\omega_k && \text{since } \boldsymbol{\alpha},\boldsymbol{\lambda}\leq\mathbf{0} \\
\leq& \sum_{k\in\mathcal{K}} d_k c_k^{\text{opp}} && \text{since } \omega_k\leq c_k^{\text{opp}}, d_k\geq 0, \forall k\in\mathcal{K} \\
=& UB.
\end{aligned}$$

We can calculate the lower bound LB of $\boldsymbol{\psi}^{\mathrm{T}}\boldsymbol{b}(\boldsymbol{u})$ as follows:

$$\begin{aligned}
\boldsymbol{\psi}^{\mathrm{T}}\boldsymbol{b}(\boldsymbol{u}) =& \sum_{h\in\mathcal{H},t\in\mathcal{T}\setminus\{T\}} u_h\alpha_{h,t} + \sum_{k\in\mathcal{K}} d_k\omega_k + \sum_{a\in\bar{\mathcal{A}}}\lambda_a \\
\overset{(i)}{>}& \sum_{h\in\mathcal{H},t\in\mathcal{T}\setminus\{T\}} Z_{\max}\alpha_{h,t} + \sum_{k\in\mathcal{K}} d_k\omega_k + \sum_{a\in\bar{\mathcal{A}}}\lambda_a \\
\geq& \min_{\psi\in\Psi} : \sum_{h\in\mathcal{H},t\in\mathcal{T}\setminus\{T\}} Z_{\max}\alpha_{h,t} + \sum_{k\in\mathcal{K}} d_k\omega_k + \sum_{a\in\bar{\mathcal{A}}}\lambda_a \\
=& LB.
\end{aligned}$$

Inequality (i) holds since we have $\boldsymbol{\alpha}\leq\mathbf{0}, 0\leq u_h\leq Z_{\max}, \forall h\in\mathcal{H}, \exists u_h < Z_{\max}$. Notably, the optimization problem $\min_{\psi\in\Psi} : \sum_{h\in\mathcal{H},t\in\mathcal{T}\setminus\{T\}} Z_{\max}\alpha_{h,t} + \sum_{k\in\mathcal{K}} d_k\omega_k + \sum_{a\in\bar{\mathcal{A}}}\lambda_a$ is bounded since its primal problem is feasible. □

Then, the Magnanti-Wong bound can be defined using UB and LB as stated in Lemma 5.

Lemma 5. $\sigma = UB - LB$ *is a Magnanti-Wong bound of the sMW-DSP*$_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}})$.

Proof. We complete the proof in two steps:

(i) Given $\sigma = UB - LB$, an optimal solution to sMW-DSP($\boldsymbol{u}^0, \hat{\boldsymbol{u}}$) must be an optimal solution to DSP($\hat{\boldsymbol{u}}$).

Since the SP($\boldsymbol{u}$) is an LP and is always feasible and bounded given any feasible RMP solution $\boldsymbol{u}$, the optimal solutions to sMW-DSP($\boldsymbol{u}^0, \hat{\boldsymbol{u}}$) are on the extreme points of the feasible region, denoted by $\mathcal{E}_\Psi$. We denote the set of optimal extreme points of sMW-DSP($\boldsymbol{u}^0, \hat{\boldsymbol{u}}$) by $\mathring{\mathcal{E}}_\Psi := \arg\max_{\boldsymbol{\psi} \in \mathcal{E}_\Psi} [(\boldsymbol{\psi})^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0) + \sigma (\boldsymbol{\psi})^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}})]$. Then, for any optimal solution $\boldsymbol{\psi}^i \in \mathring{\mathcal{E}}_\Psi$ and non-optimal solution $\boldsymbol{\psi}^j \in \mathcal{E}_\Psi \backslash \mathring{\mathcal{E}}_\Psi$, we have

$$
\begin{aligned}
&(\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0) + \sigma (\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) > (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0) + \sigma (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) \\
\Rightarrow &\left((\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) - (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}})\right) \sigma > (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0) - (\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0).
\end{aligned}
$$

Since the objective coefficients of $\widetilde{\mathrm{SP}}(\boldsymbol{u})$ are all integers and the constraint matrix of $\widetilde{\mathrm{SP}}(\boldsymbol{u})$ is TU, the extreme points of Ψ are integer vertices. Considering that the vector $\boldsymbol{b}(\hat{\boldsymbol{u}})$ is also an integer vector, we have $(\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) - (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) \geq 1$, or $(\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) - (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) \leq -1$, or $(\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) = (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}})$.

Assume that $(\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) - (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) \leq -1$, we have

$$
\begin{aligned}
&\left((\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) - (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}})\right) \sigma > (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0) - (\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0) && \\
\Rightarrow &\left((\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) - (\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}})\right) \sigma < (\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0) - (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0) && \text{multiply both sides by } -1 \\
\Rightarrow &\sigma < \frac{(\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0) - (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0)}{(\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) - (\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}})} && \text{since } (\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) < (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) \\
\Rightarrow &\sigma < (\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0) - (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0) && \text{according to the assumption} \\
\Rightarrow &UB - LB < (\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0) - (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0). && \text{since } \sigma = UB - LB
\end{aligned}
$$

Since this conclusion contradicts $UB \geq (\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0), LB < (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\boldsymbol{u}^0)$, the original assumption $(\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) - (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) \leq -1$ is rejected. Therefore, if $\boldsymbol{\psi}^i \in \mathring{\mathcal{E}}_\Psi$ and $\boldsymbol{\psi}^j \in \mathcal{E}_\Psi \backslash \mathring{\mathcal{E}}_\Psi$, we must have $(\boldsymbol{\psi}^i)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}}) \geq (\boldsymbol{\psi}^j)^{\mathrm{T}} \boldsymbol{b}(\hat{\boldsymbol{u}})$.

Following the same logic, we can prove that for any optimal solution $\boldsymbol{\psi}^i \in \mathring{\mathcal{E}}_\Psi$ and another optimal solution $\boldsymbol{\psi}^{i'} \in \mathring{\mathcal{E}}_\Psi$, we must have $(\boldsymbol{\psi}^i)^{\mathrm{T}}\boldsymbol{b}(\hat{\boldsymbol{u}}) = (\boldsymbol{\psi}^{i'})^{\mathrm{T}}\boldsymbol{b}(\hat{\boldsymbol{u}})$. Thus, an optimal solution $\boldsymbol{\psi}^i$ to sMW-DSP($\boldsymbol{u}^0, \hat{\boldsymbol{u}}$) must be an optimal solution to DSP($\hat{\boldsymbol{u}}$).

(ii) Given $\sigma = UB - LB$, solving the sMW-DSP($\boldsymbol{u}^0, \hat{\boldsymbol{u}}$) is equivalent to the lexicographic optimization problem, where the DSP($\hat{\boldsymbol{u}}$) is first solved to obtain the objective value $\hat{\eta}$ and then the Magnanti-Wong dual problem MW-DSP($\boldsymbol{u}^0, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\eta}}$) is solved.

According to (i), an optimal solution $\boldsymbol{\psi}^i \in \mathring{\mathcal{E}}_\Psi$ to sMW-DSP($\boldsymbol{u}^0, \hat{\boldsymbol{u}}$) is also an optimal solution to DSP($\hat{\boldsymbol{u}}$). Therefore, the set of optimal solutions to sMW-DSP($\boldsymbol{u}^0, \hat{\boldsymbol{u}}$) can be defined by $\mathcal{E}^*_\Psi := \arg\max_{\boldsymbol{\psi} \in \mathring{\mathcal{E}}_\Psi} (\boldsymbol{\psi})^{\mathrm{T}}\boldsymbol{b}(\boldsymbol{u}^0)$, i.e., solving the sMW-DSP($\boldsymbol{u}^0, \hat{\boldsymbol{u}}$) is equivalent to solving the DSP($\boldsymbol{u}^0$) among the optimal solutions of DSP($\hat{\boldsymbol{u}}$). □

Based on this Magnanti-Wong bound, the pseudocode of the simultaneous Magnanti-Wong BD is specified in N. By incorporating the simultaneous Magnanti-Wong BD, only one LP needs to be solved in each iteration to generate a Pareto-optimal cut, which is expected to reduce the computation time significantly.

Note that Lemma 5 requires that UB and LB be *globally* valid bounds on the objective value of DSP($\boldsymbol{u}$) for *all feasible* $\boldsymbol{u}$. Although the bounds of the incumbent SP DSP($\hat{\boldsymbol{u}}$) can be updated iteratively using values computed at the current master solution $\hat{\boldsymbol{u}}$, such bounds are only valid locally. As a result, the corresponding weight σ may no longer satisfy the global validity condition. This would undermine the theoretical equivalence between our reformulated single SP approach sMW-DSP($\boldsymbol{u}^0, \hat{\boldsymbol{u}}$) and the classic sequential Magnanti-Wong method, and could compromise the Pareto-optimality of the generated cuts.

3.5 Computational experiments

In this section, we first use randomly generated instances of different scales to assess the proposed algorithms and SP models. We then take the real data in Goldjoy Holidays

company in the Asia-Pacific area (Goldjoy Holidays, 2024) as a case study to evaluate the integrated model of CFD-ISP. Extensive sensitivity analyses on crucial parameters are also conducted to provide decision-makers with actionable insights on cruise management. The algorithms were coded in Java using CPLEX 12.7 as the solver. All experiments are conducted on a desktop running Windows 10 equipped with an Intel(R) Core(TM) i7-8700 CPU and 64 GB of RAM. All the computing times in this section are measured in seconds and the time limit is set as 3,600 seconds.

3.5.1 Comparison of solving methods

This section validates and compares our proposed algorithms and SP models. The simulation instances are generated referring to real data, with each parameter adhering to the data ranges obtained from Goldjoy Holidays (2024), which are detailed in O. Each instance is denoted by $|\mathcal{L}|$-i, where $|\mathcal{L}|$ represents the number of cruise lines, and the suffix i indicates the index of the instance within that particular scale. We generate instances for different numbers of cruise lines, specifically $\{30, 50, 80, 100\}$, producing five instances at each scale. For description convenience, the specific notation of different solving methods are presented in Table 3.1. While the first three methods are established in previous literature, the latter four methods incorporate our proposed algorithm and/or our proposed (SP) model.

The average performances of all solving methods on each scale of instances are shown in Table 3.2, with the detailed results given in Q. All costs in the result tables are measured in 10,000 dollars. Columns "#Line", "#Iter", "#Cut" represent the number of lines, the average number of iterations, and the average number of cuts generated, respectively. The average computation time for solving an instance and the average computation time for solving an SP in seconds are shown in Columns "Total CPU(s)" and "Per SP CPU(s)", respectively. The number in Column "Diff(%)" indicates the percentage difference in CPU

Table 3.1: Notation of different solving methods

Name	Algorithm	(SP) Model	Type
CPLEX-a	CPLEX	arc-based model	Existing
CPLEX-r	CPLEX	**route-based model**	Proposed
t-BD-a	traditional BD	arc-based model[1]	Existing
MW-BD-a	Magnanti-Wong BD	arc-based model	Existing
sMW-BD-a	**simultaneous Magnanti-Wong BD**	arc-based model	Proposed
t-BD-r	traditional BD	**route-based model**[2]	Proposed
MW-BD-r	Magnanti-Wong BD	**route-based model**	Proposed
sMW-BD-r	**simultaneous Magnanti-Wong BD**	**route-based model**	Proposed

[1] : The arc-based model after introducing the cumulative flow variable is provided in P, where the integer variables are relaxed due to the TU property that can be proven similarly to Lemma 2.

[2] : The route-based model $\mathrm{LSP}_l(\hat{\boldsymbol{u}})$ is presented in Section 3.4.1.1.

time between various solving methods and the t-BD-a method for each instance scale. The Column "Gap(%)" is the average optimality gap of instances in each scale. We also provide the average computation time of each solving method on different scales of instances in Figure 3.7.

From the perspective of the *solution quality*, across all test instances, the decomposition algorithms consistently close the gap (0.00%), indicating they all find optimal solutions. In contrast, CPLEX-a fails to close the gap within the allotted time (3,600 seconds) for the smallest instances of 30 cruise lines and encounters memory overflow errors when processing larger-scale instances due to the extensive number of variables and constraints of the integrated model. Besides, all algorithms experience longer computation time as the scale of instances grows larger, reflecting the increased computational complexity of a more sophisticated cruise network.

From the perspective of *solving efficiency*, while all algorithms experience increased CPU time as the scale of the instance grows, the sMW-BD algorithm consistently achieves the shortest computation time across various instance sizes and different SP models. As shown in Figure 3.7, this is particularly evident in algorithms employing arc-based SPs, where the sMW-BD algorithm significantly reduces CPU time, achieving a reduction of 19.79% to 36.09% compared to the t-BD algorithm. In contrast, the MW-BD algorithm

Table 3.2: Average performances of different solving methods

#Line	Solving method	Gap(%)	Total CPU(s)	Diff(%)	Per SP CPU(s)	#Iter	#Cut
30	CPLEX-a	625.94	3,600.00	/	/	/	/
	CPLEX-r	0.00	221.49	−23.01	/	/	/
	t-BD-a	0.00	287.67	base	1.44	39	158
	MW-BD-a	0.00	281.88	−2.01	1.60	34	120
	sMW-BD-a	0.00	183.86	−36.09	1.05	34	112
	t-BD-r	0.00	31.17	−89.17	0.14	39	153
	MW-BD-r	0.00	35.89	−87.52	0.18	34	112
	sMW-BD-r	0.00	28.02	−90.26	0.14	33	111
50	CPLEX-a[1]	/	/	/	/	/	/
	CPLEX-r	0.00	443.05	−24.58	/	/	/
	t-BD-a	0.00	587.43	base	1.92	60	258
	MW-BD-a	0.00	562.32	−4.28	2.09	53	218
	sMW-BD-a	0.00	406.95	−30.72	1.52	52	184
	t-BD-r	0.00	63.91	−89.12	0.20	59	249
	MW-BD-r	0.00	66.34	−88.71	0.22	53	187
	sMW-BD-r	0.00	61.04	−89.61	0.20	53	184
80	CPLEX-a	/	/	/	/	/	/
	CPLEX-r	0.00	1,213.95	−5.27	/	/	/
	t-BD-a	0.00	1,281.54	base	2.74	92	424
	MW-BD-a	0.00	1,251.32	−2.36	2.90	84	391
	sMW-BD-a	0.00	960.66	−25.04	2.23	84	307
	t-BD-r	0.00	159.02	−87.59	0.32	91	418
	MW-BD-r	0.00	170.65	−86.68	0.36	84	327
	sMW-BD-r	0.00	157.45	−87.71	0.34	84	308
100	CPLEX-a	/	/	/	/	/	/
	CPLEX-r	0.00	2,178.54	30.20	/	/	/
	t-BD-a	0.00	1,673.26	base	2.93	113	511
	MW-BD-a	0.00	1,514.41	−9.49	2.79	107	551
	sMW-BD-a	0.00	1,342.21	−19.79	2.50	106	377
	t-BD-r	0.00	236.10	−85.89	0.38	112	504
	MW-BD-r	0.00	240.85	−85.61	0.41	106	377
	sMW-BD-r	0.00	229.75	−86.27	0.37	109	407

[1] : Note that the results for CPLEX are not presented for instances involving 50, 80, and 100 cruise lines, as it encountered an out-of-memory error.

shows a more modest improvement in arc-based SPs, with less than 10% reduction in computation time compared to the t-BD algorithm. The limited improvement of the MW-BD algorithm is attributed to the fact that although the algorithm can generate Pareto-optimal cuts in each iteration, the time required to solve the SPs during a single iteration increases. When using the route-based SP model, while the sMW-BD method still consistently solves most instances more quickly, the computation time of the three algorithms is relatively

comparable. This is attributed to the notably brief duration required to solve the route-based SP model, typically under one second. Consequently, the primary bottleneck in this scenario shifts towards solving the RMP rather than the SPs. Besides, CPLEX-r consistently outperforms its arc-based counterpart (CPLEX-a) in all instances where comparison is possible. This improvement demonstrates the computational advantage of route-based reformulation, which provides a more compact model structure that is easier for commercial solvers to exploit. While CPLEX-r demonstrates fast solution times for small and medium-sized instances, outperforming the t-BD-a and MW-BD-a methods, its scalability becomes a notable limitation as the problem size increases. Specifically, for instances with 100 cruise lines, the total solving time of CPLEX-r (2,178.54 seconds) exceeds that of all BD methods. This sharp increase in computational time is primarily due to the NP-hard nature of the direct MIP formulation. In contrast, decomposition-based methods benefit from exploiting the problem structure and dividing the solving process into smaller, more manageable SPs.

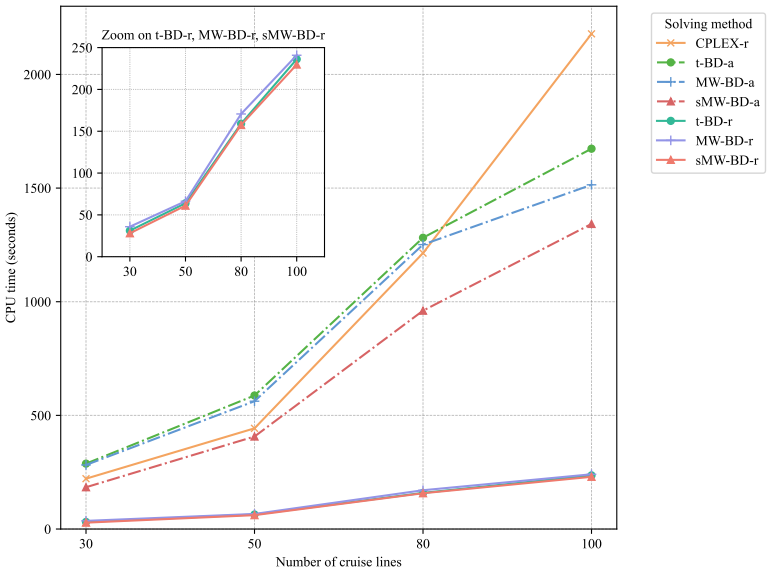


Figure 3.7: The computation time comparison of different solving methods

From the perspective of *iteration times*, the results also reveal that both the MW-BD

and sMW-BD algorithms require fewer iterations overall than t-BD, particularly as the instance size increases. This efficiency is attributed to the two algorithms' ability to generate Pareto-optimal cuts. Besides, the number of cuts is larger than the number of iterations. This is because our algorithms can solve multiple SPs within a single iteration, thereby generating several cuts simultaneously in one iteration, which also contributes to the solving efficiency.

From the perspective of *SP models*, as shown in Figure 3.7, all three algorithms demonstrate substantial enhancements when employing the route-based SP model compared to the arc-based SP model, with improvements reaching more than 9 times. Moreover, the average computation time in each iteration is reduced tenfold. This marked efficiency is due to our newly proposed route-based model, which effectively addresses the previously identified main bottlenecks: the extensive number of time-expanded arcs and nodes.

In conclusion, our proposed sMW-BD algorithm effectively addresses the issue of increasing SP computation time criticized in the traditional MW-BD method, while ensuring the generation of Pareto-optimal cuts. This significantly enhances both the solving efficiency and solution quality across various instance scales, particularly when the SP is challenging to solve. Additionally, our route-based SP model yields substantial improvements in computational efficiency, highlighting the potential for problem-specific reformulations in similar time-expanded networks, such as railway systems characterized by predefined traveling sequences and durations.

3.5.2 Comparison with other variants of Benders decomposition algorithms

In this section, we compare our simultaneous Magnanti-Wong BD framework with three BD variants.

3.5.2.1 Benders decomposition with pre-generated cuts

As each round-trip line involves only one RMP variable $u_{h,l}$, we adopt a modified BD approach by precomputing the SP values for fixed values of $u_{h,l}$. For each round-trip line, we enumerate all possible values of $u_{h,l}$, solve the corresponding SPs offline, and directly embed the resulting cuts in the RMP. This eliminates the need to generate Benders cuts for the round-trip lines during the solving process. To evaluate the impact of incorporating pre-generated round-trip SP values into the RMP in an enumerative manner, we conduct a computational study across all instance scales. Table 3.3 summarizes the average performance of the six BD methods listed in Table 3.1 in this setting. For each method, we report the number of optimally solved instances (#Opt), the total CPU time in seconds (Total CPU(s)), the percentage difference in CPU time compared with the baseline methods t-BD-a (Diff(%)), the CPU time spent per SP (Per SP CPU(s)), the number of Benders iterations (#Iter), and the number of generated Benders cuts (#Cut).

Overall, the results show that presolving round-trip SPs has a notable effect on the decomposition process. As expected, the total number of Benders iterations and generated cuts is consistently lower across all methods compared with the results without pre-generated cuts (see Table 3.2). This confirms that embedding the value functions for the round-trip lines strengthens the master formulation and accelerates convergence.

However, this improvement in convergence behavior does not necessarily translate into reduced overall computational time. In fact, the total CPU times are generally higher than those observed in the original decomposition framework. For instance, in the largest instance group with 100 lines, all methods incur a CPU time increase compared with their non-pre-generated counterparts. In addition, in several large-scale instances with 80 and 100 lines, the enumeration of round-trip cuts results in out-of-memory errors. As a result, only a few instances are solved to optimality, while the remaining cases yield no output due to computational failure. This prohibits the method from successfully completing

Table 3.3: Average performances of different solving methods with pre-generated cuts

#Line	Solving method	#Opt	Total CPU(s)	Diff(%)	Per SP CPU(s)	#Iter	#Cut
30	t-BD-a	5	486.58	base	1.95	24	106
	MW-BD-a	5	475.47	−2.28	2.22	20	78
	sMW-BD-a	5	362.91	−25.42	1.19	20	72
	t-BD-r	5	58.17	−88.05	0.18	23	103
	MW-BD-r	5	57.16	−88.25	0.22	20	75
	sMW-BD-r	5	51.80	−89.35	0.15	20	73
50	t-BD-a	5	667.11	base	2.24	59	257
	MW-BD-a	5	625.27	−6.27	2.54	44	188
	sMW-BD-a	5	389.65	−41.59	1.49	52	184
	t-BD-r	5	66.61	−90.02	0.22	57	255
	MW-BD-r	5	66.48	−90.03	0.25	52	189
	sMW-BD-r	5	54.51	−91.83	0.20	51	184
80	t-BD-a	4	1,888.12	base	3.85	91	434
	MW-BD-a	5	1,763.37	−6.61	4.75	78	384
	sMW-BD-a	5	961.64	−49.07	2.27	84	310
	t-BD-r	5	179.23	−90.51	0.38	89	431
	MW-BD-r	5	211.26	−88.81	0.43	77	316
	sMW-BD-r	5	155.19	−91.78	0.35	83	310
100	t-BD-a	1	1,922.47	base	16.29	117	524
	MW-BD-a	1	2,034.95	5.85	15.93	126	552
	sMW-BD-a	1	1,098.85	−42.84	8.27	128	418
	t-BD-r	1	312.59	−83.74	2.50	114	524
	MW-BD-r	1	367.92	−80.86	2.65	127	414
	sMW-BD-r	1	307.27	−84.02	2.26	126	413

the solving process, limiting its practical viability for real-world large-scale applications. The main reason is the additional computational burden introduced by precomputing and storing all possible SP solutions for round-trip lines, which significantly increases the size and complexity of the RMP.

Moreover, the sMW-BD-r method continues to demonstrate superior performance across all instance sizes. It maintains the lowest total CPU time and the fewest cuts. This result further reinforces the effectiveness of the simultaneous Magnanti-Wong BD framework combined with route-based reformulation.

3.5.2.2 Branch-and-Benders-cut framework

The Branch-and-Benders-cut (BBC) algorithm that integrates the principles of BD with a branch-and-cut framework, first proposed by Geoffrion and Graves (1974), is highly appreciated when the RMP including discrete variables is difficult to solve (Rahmaniani et al., 2017). In this section, we implement the BBC framework using CPLEX's user-written lazy constraint callback and compare the results of BBC-based BD algorithms with BD algorithms where the RMP is solved directly to optimality. The results are presented in Table 3.4 with performance metrics similar to those in Table 3.3. The column "#Node" represents the average number of nodes in the branch-and-bound tree.

Table 3.4: Average performances of different solving methods in a BBC framework

#Line	Solving method	#Opt	Gap(%)	Total CPU(s)	Diff(%)	Per SP CPU(s)	#Cut	#Node
30	t-BD-a	5	0.00	209.20	base	0.55	184	438,050
	MW-BD-a	5	0.00	199.59	−0.05	0.51	137	10,417
	sMW-BD-a	5	0.00	177.94	−0.15	0.48	120	27,415
	t-BD-r	5	0.00	30.72	−0.85	0.33	182	78,110
	MW-BD-r	5	0.00	29.41	−0.87	0.08	119	25,733
	sMW-BD-r	5	0.00	27.63	−0.86	0.07	120	31,803
50	t-BD-a	3	0.25	2278.07	base	0.83	297	31,845,226
	MW-BD-a	5	0.00	788.44	−0.65	0.91	236	6,113,983
	sMW-BD-a	5	0.00	688.09	−0.70	0.88	198	4,178,604
	t-BD-r	5	0.00	365.69	−0.84	0.17	361	2,782,084
	MW-BD-r	5	0.00	315.75	−0.87	0.18	200	4,597,827
	sMW-BD-r	5	0.00	286.38	−0.86	0.18	201	5,024,406
80	t-BD-a	0	/[1]	/	/	/	/	/
	MW-BD-a	1	3.37	3600.00	/	2.24	420	51,151,880
	sMW-BD-a	1	3.57	3600.00	/	2.17	337	58,387,371
	t-BD-r	0	/	/	/	/	/	/
	MW-BD-r	1	3.53	3600.00	/	1.54	348	55,891,976
	sMW-BD-r	1	3.27	3600.00	/	0.62	342	52,979,246

[1] : Instances not solved to optimality may have either reached the computational time limit or encountered an out-of-memory error.

It can be observed that while all variants achieve optimality on moderate-sized instances, performance deteriorates significantly as the problem size increases. Notably, in instances with 80 lines, most methods fail to solve the problems to optimality, either because they reach the time limit or because they encounter out-of-memory errors. In

contrast, the traditional framework, as shown in Table 3.2, where the RMP is solved to optimality at each iteration and the subproblems are then solved based on the current master solution, demonstrates superior performance by solving all instances to optimality in shorter computational times.

The inferior performance of the BBC framework in our context can be attributed to several key factors. First, the RMP in our formulation is relatively simple and can be solved very efficiently at each iteration. As a result, the primary advantage of the BBC framework, namely reducing the number of RMP solving times by integrating cut generation into the branch-and-bound tree, offers limited benefit in our case. Second, managing a large BB tree incurs substantial memory and computational overhead, which outweighs the potential gains from early cut generation. For example, even in instances with 50 lines, all methods generate more than 2 million nodes, with some exceeding 5 million. Furthermore, for instances with 80 lines, all BD methods encounter out-of-memory errors. Third, using CPLEX's callback mechanism to implement Benders cuts disables several of the solver's internal acceleration features (such as pre-solved routines, dynamic search strategies, and parallelism), leading to further degradation of overall performance (IBM Corporation, 2021).

3.5.2.3 Minimal infeasible subsystem Benders cut

The minimal infeasible subsystem (MIS) cut, which is another stream that enhances the cut selection scheme, compares favorably with the classic BD cut (Fischetti et al., 2010). To further evaluate the performance of our adopted simultaneous BD with route-based reformulation (sMW-BD-r), we conduct a comparative study against the MIS BD with route-based reformulation (MIS-BD-r) method. Detailed descriptions of MIS cuts and the corresponding SP for our CFD-ISP are detailed in R. The experiment is also performed on instances with 30 and 50 lines. The results are presented in Table 3.5 with performance metrics similar to those in Table 3.3.

Table 3.5: Performances of sMW-BD-r and MIS-BD-r on instances with 30 and 50 lines

Instance	Solving method	Gap%	Total CPU(s)	Diff(%)	Per SP CPU(s)	#Iter	#Cut
30-1	sMW-BD-r	0.00	26.01	base	0.66	32	105
	MIS-BD-r	0.00	78.72	202.66	7.86	10	123
30-2	sMW-BD-r	0.00	28.09	base	0.69	33	106
	MIS-BD-r	0.00	69.05	145.80	7.66	9	119
30-3	sMW-BD-r	0.00	27.44	base	0.72	34	112
	MIS-BD-r	0.00	72.22	163.19	4.81	15	107
30-4	sMW-BD-r	0.00	27.17	base	0.69	33	114
	MIS-BD-r	0.00	78.25	187.99	4.56	4	51
30-5	sMW-BD-r	0.00	31.42	base	0.73	35	119
	MIS-BD-r	0.00	34.79	10.74	3.86	9	67
50-1	sMW-BD-r	0.00	60.47	base	0.99	54	191
	MIS-BD-r	0.00	112.56	86.14	7.50	15	256
50-2	sMW-BD-r	0.00	63.43	base	1.01	54	179
	MIS-BD-r	0.00	84.24	32.81	16.85	5	82
50-3	sMW-BD-r	0.00	54.40	base	0.93	54	170
	MIS-BD-r	0.00	125.76	131.18	11.43	11	118
50-4	sMW-BD-r	0.00	60.86	base	1.14	48	190
	MIS-BD-r	0.00	143.90	136.44	11.99	12	132
50-5	sMW-BD-r	0.00	66.03	base	1.02	57	191
	MIS-BD-r	0.00	138.77	110.16	10.67	13	216

In all instances, the sMW-BD-r method consistently outperforms the MIS-BD-r method in terms of total CPU time. On average, sMW-BD-r achieves more than 100% time savings compared with MIS-BD-r. For example, in instance 30-1, sMW-BD-r solves the problem in 26.01 seconds, while MIS-BD-r takes 78.72 seconds, which is more than three times as long as sMW-BD-r. Similar trends are observed in instances with 30 and 50 lines. A key observation lies in the average SP solving time. The MIS-BD-r method exhibits significantly higher per SP CPU times, often exceeding 7–10 seconds, while sMW-BD-r maintains SP solving times below 1.2 seconds in all instances. This large discrepancy highlights a critical numerical issue in the MIS-BD-r formulation.

The inferior performance of MIS-BD-r can be attributed to two main factors. First, although MIS-BD-r is designed to generate feasibility and optimality cuts in a unified framework, this advantage is not realized in our setting. As the SPs in our model are al-

ways feasible by construction, the benefit of feasibility cuts becomes redundant. More importantly, the MIS-BD-r SP method includes a normalization constraint (R.14d) that forces the dual ray to have unit norm. Although this constraint is necessary for the validity of MIS cuts, it introduces severe numerical instability in our context due to the large magnitude of the dual variables. As a result, commercial solvers encounter significant numerical difficulties, manifested by substantially increased SP solving times.

3.5.3 Case study

In this section, we utilize the 46 cruise lines in the Pacific Ocean area derived from Goldjoy Holidays company as a case study. As in the simulation test, the total number of available cruises is set as $3 \times 46 = 138$. We evaluate our CFD-ISP model alongside two policy-based models, where the cruise deployment decision and the itinerary scheduling decisions are made sequentially, in terms of the impact of cost parameters. Specifically, we implement two cruise deployment policies currently utilized by cruise companies: the minimum cost policy (P1) and the maximum demand policy (P2). In policy P1, we randomly deploy all available cruises to the home ports with minimized location costs; whereas, in policy P2, we randomly allocate the cruises to the home ports experiencing the highest demand during the planning period. To ensure a fair comparison, we enforce that all cruises are deployed. For itinerary scheduling decisions in P1 and P2, we employ an optimization model that aims to minimize the total operational costs.

By varying the cost parameters as presented in Table 3.6, we compare the decisions made by the CFD-ISP model, P1, and P2 in terms of each cost component. Significantly, the *benchmark* scenario, characterized by medium c^{loc}, low c^{fuel}, medium c^{opp}, and low r^{uti}, most closely mirrors real-world conditions and aligns with the settings in our simulation tests. The specific results of different models under various cost settings are presented in Table 3.7, where the costs are measured in million dollars. For notation convenience, we

denote "Obj" as the objective value, including the cruise deployment cost ("CDC") and overall operational cost ("OC"), and we denote the number of trips that cruises take during the planning horizon as "#Trips". The last two columns show the percentage difference between the total cost of P1 (P2) and that of the CFD-ISP model.

Table 3.6: Cost settings

Cost parameter	Level	Value
Cruise deployment cost (c^{loc})	High	selected within $\{20, 40, 60, 80\}$ ($\times$1,000,000 dollars)
	Medium	selected within $\{10, 20, 30, 40\}$ ($\times$1,000,000 dollars)
	Low	selected within $\{2, 4, 6, 8\}$ ($\times$1,000,000 dollars)
Fuel cost (c^{fuel})	High	30,000/per hour (dollar)
	Medium	20,000/per hour (dollar)
	Low	10,000/per hour (dollar)
Opportunity cost (c^{opp})	High	2,000 $\times$ single economy class cabin fare[1](dollar)
	Medium	1,000 $\times$ single economy class cabin fare (dollar)
	Low	200 $\times$ single economy class cabin fare (dollar)
Utility revenue (r^{uti})	High	10,000 $\times$ utility ($\times$10,000 dollars)
	Medium	5,000 $\times$ utility ($\times$10,000 dollars)
	Low	1,000 $\times$ utility ($\times$10,000 dollars)

[1] : This is the price of the single economy class cabin of each cruise line gained from the Goldjoy Holidays company.

Table 3.7: Integrated and policy-based model comparison under different cost-related parameters

	CFD-ISP model				P1				P2				Diff(%)	
Factor	Obj[2]	CDC	OC	#Trips	Obj	CDC	OC	#Trips	Obj	CDC	OC	#Trips	P1	P2
Cruise deployment cost[1] (c^{loc})														
High	328.0	490.0	−162.0	835	1,357.9	276.0	1,081.9	40	1,223.6	828.0	1,211.2	70	75.9	73.2
Medium	73.5	267.0	−193.5	849	1,338.3	138.0	1,200.3	70	1,223.6	414.0	1,211.2	70	94.5	94.0
Low	−152.6	60.6	−213.2	881	1,257.0	27.6	1,229.4	80	1,223.6	82.8	1,211.2	70	112.1	112.5
Fuel cost (c^{fuel})														
High	1,103.3	236.0	867.3	633	1,386.4	138.0	1,248.4	0	1,260.8	414.0	1,248.4	0	20.4	12.5
Medium	602.4	261.0	341.4	769	1,336.0	138.0	1,198.0	80	1,260.8	414.0	1,248.4	0	54.9	52.2
Low	73.5	267.0	−193.5	849	1,338.3	138.0	1,200.3	70	1,223.6	414.0	1,211.2	70	94.5	94.0
Opportunity cost (c^{opp})														
High	914.0	267.0	647.0	871	2,533.9	138.0	2,395.9	60	2,460.2	414.0	2,447.7	70	63.9	62.9
Medium	73.5	267.0	−193.5	849	1,338.3	138.0	1,200.3	70	1,223.6	414.0	1,211.2	70	94.5	94.0
Low	−633.1	271.0	−904.1	797	344.7	138.0	206.7	70	234.3	414.0	221.9	66	283.7	370.2
Utility revenue (r^{uti})														
High	−14,638.9	303.0	−14,941.9	801	334.5	138.0	196.5	80	563.6	414.0	551.1	70	4,476.5	2,697.6
Medium	−6,454.6	299.0	−6,753.6	801	871.0	138.0	733.0	80	930.2	414.0	917.8	70	841.0	793.9
Low	73.5	267.0	−193.5	849	1,338.3	138.0	1,200.3	70	1,223.6	414.0	1,211.2	70	94.5	94.0

[1] : All costs are measured in million dollars in the table.
[2] : Negative objective value means that the utility revenue is larger than the costs; while positive objective value means that the sum of cruise deployment cost, opportunity cost, and fuel cost is larger than the utility revenue.

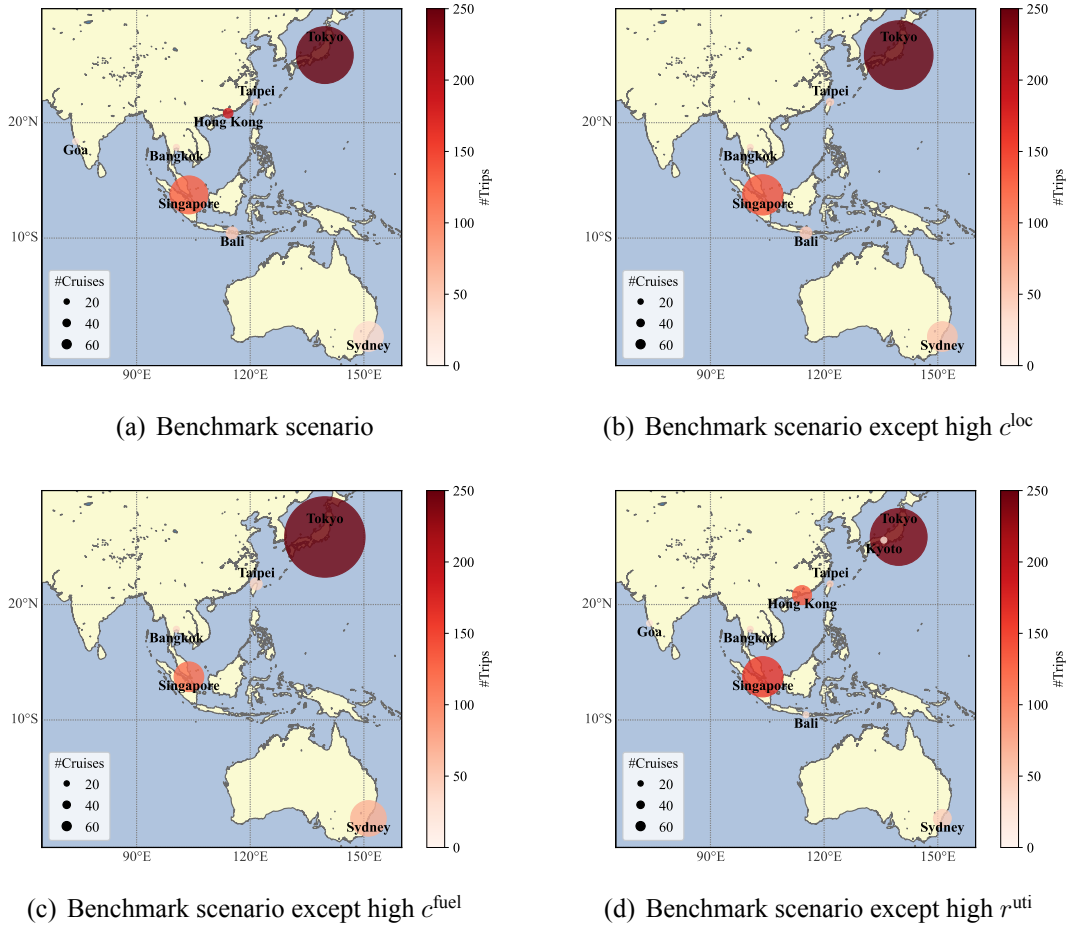
3.5.3.1 Sensitivity analysis of cost settings

Focusing on the results of our CFD-ISP model, we can observe that the overall cost fluctuates across various levels of parameters. Notably, when the location cost, denoted as c^{loc}, decreases, the total cost also diminishes from 328 million dollars to a negative value of -152.6 million dollars, which means the utility revenue is larger than the total cost. This trend is primarily driven by the reduction in cruise deployment cost as well as a modest decrease in operational cost. Conversely, increases in either fuel cost or opportunity cost lead to a corresponding rise in the total cost, primarily due to escalated operational expenses, while the cruise deployment cost remains comparatively stable. Additionally, when utility profits increase, total cost turns negative, indicating more profitability.

Regarding the number of trips during the planning horizon, these appear to be closely linked to operational-related parameters. An increase in fuel cost results in fewer trips being taken; conversely, higher opportunity cost prompts more trips to mitigate the impact of high unmet demand. While these results generally align with expectations, the number of trips decreases as the utility revenue parameter increases, which contradicts general intuition. This could be attributed to cruises opting for longer, more utility-rich routes, thereby maintaining or reducing the number of trips while decreasing the operational costs.

In Figure 3.8, we display the integrated model's cruise deployment and itinerary scheduling results on a geographic map, whose numerical results are detailed in S. The size of the circle presents the number of cruises deployed in this port, while the color of the circle indicates the number of trips that cruises take from this port during the planning horizon, which are shown in the sidebar.

As shown in Figure 3.8(a), the benchmark scenario features cruises deployed across eight home ports, with Tokyo serving as the home port hosting the largest number of cruises. Conversely, Figure 3.8(b) demonstrates that when the cost associated with cruise placement increases, the number of ports used for cruise deployment is reduced to six,



(a) Benchmark scenario

(b) Benchmark scenario except high c^{loc}

(c) Benchmark scenario except high c^{fuel}

(d) Benchmark scenario except high r^{uti}

Figure 3.8: Results comparison on the geographic map with different cost settings

notably excluding higher-cost ports such as Hong Kong. Further centralization is evident in Figure 3.8(c), where, in response to rising cruise fuel costs, deployment is restricted to just five ports. This strategic adjustment significantly affects more remote ports such as Goa, which are excluded from cruise placements. As a result, cruise lines associated with these ports also see a reduction in operations. Figure 3.8(d) shows that when utility revenue is prioritized, cruise placement expands to include an additional home port, Kyoto, alongside the original eight ports. This strategy likely arises as a strategic approach to maximize utility profits by enhancing flexibility in cruise line selection and timing of stops.

In conclusion, these decision adaptations highlight our CFD-ISP model's capability

to effectively respond to variations in input parameters, thereby minimizing overall expenses through the optimized deployment of cruises at the tactical level. Furthermore, the network design reveals that in times of market instability, the optimized results favor a more centralized placement of cruises. This indicates that concentrating investments in a smaller area can yield higher profits under such conditions. Additionally, when the utility is prioritized, positioning cruises in multiple home ports enhances the flexibility of the itinerary schedules, which in turn increases overall profits.

3.5.3.2 Comparison of the integrated and policy-based models

As demonstrated in Table 3.7, our CFD-ISP model consistently outperforms the other two policy-based sequential models, i.e., P1 and P2, by achieving lower total cost across various cost settings. The degree of improvement ranges from 12.5% to over 4,000%. Notably, the superiority of our CFD-ISP model becomes particularly evident when there are fluctuations in utility revenue, underscoring the model's effectiveness in generating optimized itinerary schedules that enhance utility revenue. This increase in utility revenue is indicative of higher customer satisfaction and improved service quality.

Furthermore, the CFD-ISP model consistently facilitates a larger number of trips compared to models P1 and P2. The significantly small number of trips derived from sequential decisions in two levels may be attributed to inefficient cruise placements, leading to reduced profits or even losses that discourage trip execution. This issue is more evident when the fuel cost increases, as sometimes no trip is taken by P1 and P2. Besides, the small number of trips also results in the elevated opportunity cost associated with sequential decisions.

We further analyze the total cost of different models under varying cost parameters, as depicted in the box plot, Figure 3.9. It can be observed that the median total cost of our CFD-ISP model is consistently lower than that of the other two models. In approximately half of the scenarios, our model achieves negative total costs, indicating more profit gain

through utility revenue. Additionally, the outliers in the CFD-ISP model suggest that total profits are highly sensitive to the settings of utility-related parameters. In contrast, the outliers of the other two policy-based sequential models mainly correspond to the settings of opportunity cost, specifically the opportunity cost coefficient, highlighting that unmet demand is the predominant factor influencing the total cost of these models.

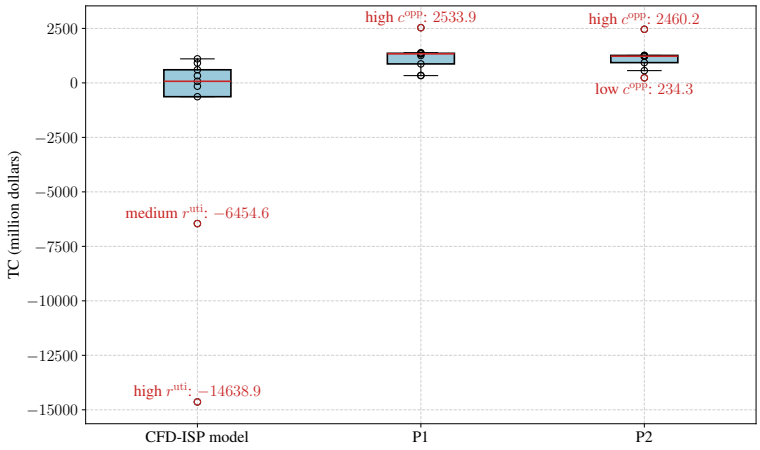


Figure 3.9: Total costs of different models under different cost settings

In summary, the model comparison underscores the critical importance of integrating itinerary scheduling within the cruise deployment phase. This integrated approach is particularly valuable in scenarios characterized by high cost and high utility revenue, suggesting a high-risk, high-return market environment.

3.5.3.3 Comparison of the integrated model and the greedy strategy

In this section, we further compare our integrated model with a greedy heuristic, which iteratively assigns ships to line–port combinations to maximize system revenue. The algorithm initializes with no assigned ships, i.e., $\boldsymbol{u} = 0$, and iteratively deploys one cruise at a time. At each iteration, the algorithm evaluates all feasible line–port combinations by temporarily increasing $u_{h,l}$ by one and solving the associated itinerary scheduling problem to estimate the resulting system revenue. The combination (h^*, l^*) that yields the high-

est marginal increase in revenue is selected and the corresponding deployment decision is updated. The process continues until all cruises are deployed. The pseudocode is given in T. Similar to Table 3.7, Table 3.8 reports the performance of the integrated CFD-ISP model and a greedy baseline strategy as the key cost-related parameters vary.

Table 3.8: Integrated model and greedy strategy comparison under different cost-related parameters

	CFD-ISP model				Greedy				Diff(%)
Factor	Obj[2]	CDC	OC	#Trips	Obj	CDC	OC	#Trips	
Cruise deployment cost[1] (c^{loc})									
High	328.0	490.0	−162.0	835	1,085.7	60.0	1,025.7	140	69.8
Medium	73.5	267.0	−193.5	849	1,051.8	35.0	1,016.8	130	93.0
Low	−152.6	60.6	−213.2	881	1,023.4	8.2	1,015.2	136	114.9
Fuel cost (c^{fuel})									
High	1,103.3	236.0	867.3	633	1,201.2	32.0	1,169.2	104	8.2
Medium	602.4	261.0	341.4	769	1,130.4	35.0	1,095.4	160	46.7
Low	73.5	267.0	−193.5	849	1,051.8	35.0	1,016.8	130	93.0
Opportunity cost (c^{opp})									
High	914.0	267.0	647.0	871	2,200.8	35.0	2,165.8	180	58.5
Medium	73.5	267.0	−193.5	849	1,051.8	35.0	1,016.8	130	93.0
Low	−633.1	271.0	−904.1	797	126.0	31.0	95.01	134	602.4
Utility revenue (r^{uti})									
High	−14,638.9	303.0	−14,941.9	801	−1,011.9	60.0	−1,071.9	140	1,346.7
Medium	−6,454.6	299.0	−6,753.6	801	167.1	60.0	107.1	140	3,962.1
Low	73.5	267.0	−193.5	849	1,051.8	35.0	1,016.8	130	93.0

[1] : All costs are measured in million dollars in the table.
[2] : Negative objective value means that the utility revenue is larger than the costs; while positive objective value means that the sum of cruise deployment cost, opportunity cost, and fuel cost is larger than the utility revenue.

Across all parameter settings, the CFD-ISP model consistently achieves significantly lower total costs than the greedy strategy. Under high cruise deployment cost settings, the CFD-ISP model yields an objective value of USD328.0 million, compared with USD1,085.7 million under the greedy strategy, a reduction of approximately 69.8%. Similar trends are observed in the medium and low cost scenarios, where the CFD-ISP model achieves 93.0% and 114.9% lower total costs, respectively. This indicates the integrated model's ability to make globally coordinated ship deployment and itinerary scheduling decisions.

In the case of fuel cost variations, the CFD-ISP model outperforms the greedy ap-

proach with lower objective values of 8.2% to 93.0%. The difference is particularly marked when opportunity costs vary. In the high opportunity cost case, the CFD-ISP model reduces total costs by 58.5% relative to the greedy strategy. Remarkably, under low opportunity costs, the CFD-ISP model achieves a negative objective value of USD633.1 million, suggesting a net gain in utility revenue, while the greedy strategy results in a positive cost of USD126 million. Most importantly, changes in utility revenue have the greatest impact on model performance. Under high utility revenue, the CFD-ISP model achieves a significantly lower negative objective value, highlighting its superior ability to capture and exploit revenue potential. Similar trends, with differences exceeding 1,000% in relative terms, are observed under medium utility revenue. The number of trips further reflects the operational efficiency of the two strategies. In all scenarios, although the greedy approach yields more trips than the industry-standard policies P1 and P2 (as shown in Table 3.7), its number of trips remains substantially lower than that achieved by the CFD-ISP model.

Overall, the CFD-ISP model demonstrates its clear superiority over the greedy strategy both in terms of cost-effectiveness and operational capacity. The integrated optimization framework enables the model to make globally optimal deployment decisions that balance costs and utility, while the greedy strategy, lacking coordination and foresight, leads to suboptimal outcomes. These findings reinforce the value of solving the CFD-ISP problem in an integrated manner, especially when cost structures and revenue potentials vary across the network.

3.5.3.4 Managerial implications

Through the case study based on real data from the cruise company, the managerial implications of our work can be drawn as follows: (i) *Integration of itinerary scheduling with cruise deployment*: Our comparison of models reveals that integrating itinerary scheduling within the cruise deployment phase is crucial, especially in high-risk, high-return mar-

ket environments. This holistic approach ensures that the schedule aligns with strategic deployment goals, enhancing market responsiveness and profitability. (ii) *Tactical-level responsiveness to operational dynamics*: Our sensitivity analysis shows that adapting tactical-level decisions dynamically in response to operational-level fluctuations can significantly reduce overall expenses, which allows the company to remain competitive and efficient in a volatile market. (iii) *Focused investment*: Interestingly, our results indicate that concentrating investments in a geographically smaller area can lead to higher profits, particularly in unstable market conditions. This strategy enables more directed resource allocation and potentially higher returns on investment. (iv) *Decentralization of cruise deployment*: The decentralization of cruise deployment can increase the flexibility of itinerary schedules when utility is prioritized. This enhanced flexibility not only adapts more effectively to passenger preferences but also boosts overall profitability.

3.6 Conclusions

Driven by the growing popularity of cruise tourism, cruise companies are keen to explore this burgeoning market to boost their revenues. Nevertheless, to take full advantage of this increasing demand, managers need to integrate cruise deployment and itinerary scheduling decisions that not only reduce overall costs but also enhance passenger satisfaction. To address this issue, this paper formulates an IP model for the CFD-ISP, which, however, is limited to small-scale instances when solved using commercial solvers.

From the perspective of the algorithm, this paper develops an advanced BD algorithm that incorporates several tailored accelerating approaches: (i) applying cumulative-flow variables, (ii) proving the TU property of the SP, (iii) proposing a route-based reformulation of the time-expanded network, and (iv) implementing the simultaneous Magnanti-Wong method by introducing a problem specific Magnanti-Wong bound. Theoretically, the key advancement lies in that the integer variables of the reformulated SP can be relaxed

to generate Benders cuts and the Magnanti-Wong bound can be pre-solved to guarantee a Pareto-optimal cut by solving a single LP model. Computationally, experiment results demonstrate that our simultaneous Magnanti-Wong algorithm surpasses both CPLEX and other traditional BD algorithms in terms of computational efficiency and effectiveness by obtaining the optimal solutions for instances with up to 100 cruise lines in minutes. Moreover, our proposed route-based SP can streamline the time-expanded network, achieving an average CPU time that is 9 times faster for large-scale instances, which can also be applied to similar problems such as rolling stock and aircraft schedules. We note that while our model and algorithm integrate tactical and operational decisions, omitting strategic decisions as they are often long-term and predetermined, they can easily be adapted to include strategic decisions by treating the total number of available cruises as a decision variable.

Armed with this algorithm, we conduct a case study to emphasize the necessity of integration and to obtain critical management insights. We benchmark our CFD-ISP model against two policy-based sequential models where cruise deployment and itinerary scheduling decisions are made independently. The results reveal that our CFD-ISP model can significantly enhance performance, yielding over a 90% increase in profits under benchmark parameters. Sensitivity analysis further confirms that our integrated model consistently outperforms the policy-based models by maintaining lower costs across various cost settings, especially when the utility factor is emphasized. From a practical perspective, several management insights are drawn.

Chapter 4

Concluding remarks

4.1 Conclusions

This thesis develops mathematical models for complex maritime logistics problems, including the DSP-SSD and the integrated CFD-ISP, by leveraging time discretization and time-expanded network structures. To solve these combinatorial challenges, we propose tailored large-scale decomposition algorithms, such as BPC and BD, that incorporate advanced acceleration techniques to achieve scalability and obtain high-quality optimal solutions for real-world instances.

Chapter 2 addresses the DSP-SSD, a unique variant of vehicle routing characterized by moving targets (vessels) and the necessity of planning multiple drone trips under strict capacity and battery constraints. The problem is modeled using an MIP formulation based on a time-expanded network created via time discretization. To achieve practical scalability, the authors propose a tailored BPC algorithm. This exact approach incorporates specialized cutting planes and a unique labeling algorithm to consistently outperform commercial solvers in terms of both solution quality and computational speed for complex instances.

Chapter 3 developed an integrated model and provides a scalable algorithmic solution for the CFD-ISP. The model couples medium-term tactical deployment with short-term op-

erational scheduling to minimize costs while optimizing passenger utility. We establish the route-based reformulation of the time-expanded network and prove the subproblem's TU property, simplifying the model significantly. The problem is solved using an enhanced BD approach, which incorporates the tailored simultaneous Magnanti-Wong method to efficiently generate strong Pareto-optimal cuts in each iteration. This integrated methodology is shown to yield significantly lower costs (over 90% reduction in benchmarks) compared to traditional sequential policy models.

Collectively, these studies underscore the great potential of the application of large-scale decomposition algorithms in modern maritime logistics. The proposed model and exact algorithms not only significantly enhance operational efficiency but also promote more sustainable industry practices by optimizing resource allocation and improving decision quality. Given the continuously increasing complexity of integrated and dynamic maritime operations, the specific algorithms developed in this research establish a vital foundation for future innovations in network design and scalable resource management within the sector. At the same time, our models and algorithms can also serve as a foundational basis for future research in related domains, inspiring further extensions or the development of practical heuristics and managerial rules tailored for real-world industry applications.

4.2 Future research directions

The first study introduces the drone scheduling problem in the shore-to-ship delivery and establishes a BPC algorithm. This research could be further extended in several aspects. Methodologically, more advanced methods could be adopted to find a better UB at the early stages of the BPC algorithm (Desaulniers et al., 2006). Anti-degradation techniques such as the dual stabilization method could be considered (Neame, 2000) to avoid the discrete time-expanded network inducing many solutions with the same objective value.

Additionally, statistical hypothesis tests (e.g., paired t-tests or Wilcoxon signed-rank tests) can be employed to rigorously compare the performance of alternative algorithms under various scenarios, providing more robust empirical validation. Practically, to promote the application of shore-to-ship delivery, more realistic operational factors could be considered in the model. One promising direction is to investigate the synergy between drones, ships, and tender boats, enabling hybrid delivery strategies that leverage the strengths of multiple transport modes. For instance, drones and tender boats could be deployed simultaneously to increase delivery capacity within a given time window. Tender boats could also serve as mobile drone launch platforms, effectively expanding the service range of drones beyond their standalone battery, i.e., limited flight radius. Moreover, uncertain elements, such as weather conditions (Cheng et al., 2024), sea state, and battery power consumption, can be incorporated into the planning process through stochastic or robust optimization frameworks. These extensions would significantly improve the resilience and reliability of delivery operations in real-world maritime environments. Finally, from a managerial perspective, it is valuable to analyze how the integration of complementary resources, i.e., drones and tender boats, can contribute to cost savings and operational efficiency. Such insights can support decision-making for shipping operators and port authorities considering the deployment of unmanned aerial delivery systems.

The second study provides a comprehensive solving scheme for the CFD-ISP. This research can be extended in two major directions. First, from a modeling perspective, considering the primary bottleneck in solving efficiency is the extensive scale of the time-expanded network, employing new solution methods that utilize dynamic time discretization could effectively manage a large number of time points (Boland et al., 2017). Second, from a practical standpoint, the proposed model can be extended by incorporating fleet sizing decisions, enabling more accurate and long-term strategic planning for cruise operations. Furthermore, the modeling framework and solution methodology have the potential to be adapted to other integrated planning problems, such as rolling stock optimization in

railway systems (Yin et al., 2023a) and aircraft scheduling in airline operations (Drabas and Wu, 2013), thereby broadening the applicability of the approach across transportation domains.

Appendices

A Calculating the traveling time

The flying time of each drone from ship v at time t to another ship v', denoted by $\tau_{v,v'}(t)$, can be calculated by

$$\left[S^{\mathcal{D}}\tau_{v,v'}(t)\right]^2 = \left[\alpha_v(t) - \alpha_{v'}\left(t + \tau_{v,v'}(t)\right)\right]^2 + \left[\beta_v(t) - \beta_{v'}\left(t + \tau_{v,v'}(t)\right)\right]^2. \quad \text{(A.1)}$$

Similarly, the flying time of a drone from ship v at time t to the drone station, denoted by $\tau_{v,0}(t)$, can be calculated by

$$\left[S^{\mathcal{D}}\tau_{v,0}(t)\right]^2 = \left[\alpha_v\left(t\right) - 0\right]^2 + \left[\beta_v\left(t\right) - 0\right]^2. \quad \text{(A.2)}$$

The flying time from the drone station at time t to ship v, denoted by $\tau_{0,v}(t)$, can be computed by

$$\left[S^{\mathcal{D}}\tau_{0,v}(t)\right]^2 = \left[0 - \alpha_v\left(t + \tau_{0,v}(t)\right)\right]^2 + \left[0 - \beta_v\left(t + \tau_{0,v}(t)\right)\right]^2. \quad \text{(A.3)}$$

The valid ranges of t in the above equations (A.1)–(A.3) are provided as follows.

Notably, we abbreviate $\tau_{v,v'}(t)$, $\tau_{v,0}(t)$ and $\tau_{0,v}(t)$ as τ in the following proof.

(i) For equation (A.2), the range of t is $[t_{ve}, t_{vl}]$.

(ii) For equation (A.3), a drone leaves the station at time t and arrives at the ship v

at time $t+\tau$, which should be in the vessel's time window, i.e., $[t_{ve}, t_{vl}]$. Therefore, we denote the range of t as $[t_{\text{min}}, t_{\text{max}}]$, where t_{min} is calculated by

$$\begin{aligned}
&t_{\text{min}} + \tau = t_{ve}, \\
&\tau \geq 0, \\
&\left[S^{\mathcal{D}}\tau\right]^2 = \left[0 - \alpha_v\left(t_{ve}\right)\right]^2 + \left[0 - \beta_v\left(t_{ve}\right)\right]^2,
\end{aligned}$$

and t_{max} is calculated by

$$\begin{aligned}
&t_{\text{max}} + \tau = t_{vl}, \\
&\tau \geq 0, \\
&\left[S^{\mathcal{D}}\tau\right]^2 = \left[0 - \alpha_v\left(t_{vl}\right)\right]^2 + \left[0 - \beta_v\left(t_{vl}\right)\right]^2.
\end{aligned}$$

(iii) For equation (A.1), firstly, we have $t \in [t_{ve}, t_{vl}]$. Besides, the drone leaving vessel v at time t should arrive at vessel v' within its time window $[t_{v'e}, t_{v'l}]$. Therefore, we have

$$\begin{aligned}
\hat{\mathcal{T}} = \{&t \in [t_{ve}, t_{vl}] \mid \\
&t_{v'e} \leq \tau + t \leq t_{v'l}, \left[S^{\mathcal{D}}\tau\right]^2 = \left[\alpha_v(t) - \alpha_{v'}\left(t+\tau\right)\right]^2 + \left[\beta_v(t) - \beta_{v'}\left(t+\tau\right)\right]^2, \tau \geq 0\}.
\end{aligned}$$

In this case, the valid t is in $\hat{\mathcal{T}}$.

B Existence of delivery time

Proposition 4. *Given $S^{\mathcal{D}} > S^{\mathcal{V}}$, there always exists a unique and positive root of $\tau_{\cdot,\cdot}(t)$ for equations* (A.1)*,* (A.2) *and* (A.3)*.*

The proof is given as follows. Notably, we abbreviate $\tau_{v,v'}(t)$, $\tau_{v,0}(t)$ and $\tau_{0,v}(t)$ as τ in the following proof.

(i) For equation (A.2), the right-hand side is a positive value. Therefore, we have

$$\tau = \pm\sqrt{\frac{\left[\alpha_v\left(t\right)-0\right]^2+\left[\beta_v\left(t\right)-0\right]^2}{S^{\mathcal{D}}}}.$$

Then, there always exists one unique positive root of $\tau = \sqrt{\frac{[\alpha_v(t)]^2+[\beta_v(t)]^2}{S^{\mathcal{D}}}}$.

(ii) For equations (A.1) and (A.3), we first prove that there is at least one positive root of τ. We assume that the drone leaves the current station (or ship) at time t_1 and arrives at the next ship v at t_2 and $t_2 = t_1 + \tau$. The original location of the drone is (α_0, β_0), and its distance to the next ship v at time t_1 is denoted as l_0. Then, we define $g(\tau) = [S^{\mathcal{D}}\tau]^2 - [\alpha_v(t_2) - \alpha_0]^2 - [\beta_v(t_2) - \beta_0]^2$, which is a function of τ.

When $\tau = 0$, we have

$$g(0) = [S^{\mathcal{D}} \cdot 0]^2 - l_0^2 < 0.$$

When $\tau > 0$, we have

$$\begin{aligned}
g(\tau) &= [S^{\mathcal{D}}\tau]^2 - [\alpha_v(t_2) - \alpha_0]^2 - [\beta_v(t_2) - \beta_0]^2 \\
&= [S^{\mathcal{D}}\tau]^2 - [\alpha_v(t_1+\tau) - \alpha_0]^2 - [\beta_v(t_1+\tau) - \beta_0]^2 \\
&\geq [S^{\mathcal{D}}\tau]^2 - [l_0 + S^{\mathcal{V}} \cdot \tau]^2 && \text{(By triangle inequality)} \\
&= ([S^{\mathcal{D}}]^2 - [S^{\mathcal{V}}]^2)\tau^2 - (l_0)^2 - 2l_0 S^{\mathcal{V}}\tau. && \text{(B.4)}
\end{aligned}$$

The quadratic formula (B.4) is equal to 0 when

$$\begin{aligned}
\tau =& \frac{2l_0 S^{\mathcal{V}} \pm \sqrt{4(l_0)^2[S^{\mathcal{V}}]^2 + 4(l_0)^2([S^{\mathcal{D}}]^2 - [S^{\mathcal{V}}]^2)}}{2([S^{\mathcal{D}}]^2 - [S^{\mathcal{V}}]^2)} \\
=& \frac{2l_0 S^{\mathcal{V}} \pm 2l_0 S^{\mathcal{D}}}{2([S^{\mathcal{D}}]^2 - [S^{\mathcal{V}}]^2)} \\
=& \frac{l_0(S^{\mathcal{V}} \pm S^{\mathcal{D}})}{[S^{\mathcal{D}}]^2 - [S^{\mathcal{V}}]^2} && \text{(since } S^{\mathcal{D}} > S^{\mathcal{V}} \text{ and } \tau > 0)
\end{aligned}$$

$$=\frac{l_0}{S^D - S^V}.$$

Therefore, we have $\tau > \frac{l_0}{S^D - S^V}$ such that the quadratic formula (B.4) is greater than 0. Therefore, when τ grows large enough, we have $g(\tau) > 0$. Together with $g(0) < 0$ and $g(\tau)$ is continuous, we have at least one positive root of τ.

Next, we prove that the root is unique by contradiction. As shown in Figure B.1, assume that there are two positive roots of τ, i.e., the drone can leave its current location and arrive at the next ship after τ_1 or τ_2, where $\tau_1 < \tau_2$. The distance between the current location of the drone and the location where the drone arrives at the next ship is l_1 and l_2, respectively. The distance between the locations of the ship after τ_1 and τ_2 is l_3.

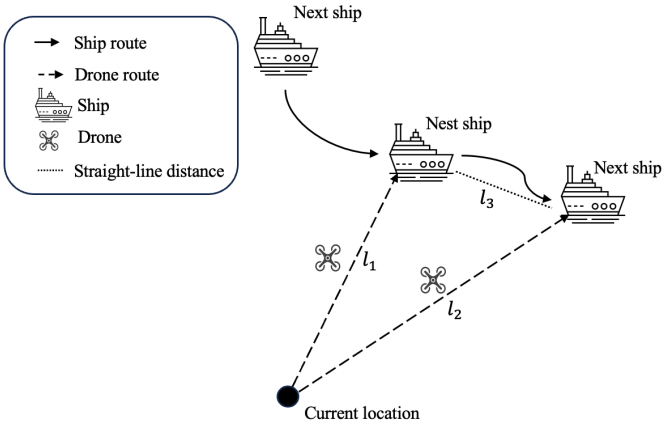


Figure B.1: The situation that the drone can arrive at a ship at two different time points

Then, we have

$$\begin{aligned} l_2 - l_1 &= S^{\mathcal{D}}(\tau_2 - \tau_1) \\ &\leq l_3 && \text{(By triangle inequality)} \\ &\leq S^{\mathcal{V}}(\tau_2 - \tau_1) && \text{(The ship's sailing route cannot be shorter than the straight line).} \end{aligned}$$

The result implies $S^{\mathcal{D}}(\tau_2 - \tau_1) \leq S^{\mathcal{V}}(\tau_2 - \tau_1)$, i.e., $S^{\mathcal{D}} \leq S^{\mathcal{V}}$, which contradicts our

assumption $S^{\mathcal{V}} < S^{\mathcal{D}}$. Therefore, there can not be more than one positive root of τ.

Conclusively, given $S^{\mathcal{D}} > S^{\mathcal{V}}$, there always exists one unique and positive root of $\tau_{,\cdot}(t)$ for equations (A.1), (A.2) and (A.3).

C Proof of Proposition 1

Constraints (2.1b) and (2.1n) restrict that the number of available drones in the station at each time point should be larger than 0 and smaller than $|\mathcal{D}|$, that is, the number of working drones at each time point should be smaller than $|\mathcal{D}|$ and larger than 0, which is exactly the meaning of constraints (2.3c). This completes the proof.

D Dynamic programming method for 2-path inequalities

A tailored backward DP formulation is introduced to check whether a subset of vessels $\mathcal{V}^{tp}$ can be served by one drone. We perform the DP for each station-time node $i_0 \in \mathcal{N}^0$, and the procedure stops once a feasible solution that visits all vessels can be found. Firstly, we define the state $(\widetilde{\mathcal{V}}, j, \widetilde{q})$ as the family of all feasible paths, which starts in vessel u_j at time t_j, visits an unordered set of vessels $\widetilde{\mathcal{V}} \subseteq \mathcal{V}^{tp}$, and arrives at the station-time node i_0. $\widetilde{q}$ represents the minimum battery consumption of a path starting at node j, passing through every vessel of $\widetilde{\mathcal{V}}$ exactly once, ending at node i_0. Let $\mathscr{S}_k$ be the set of states that visits k vessels.

Step 1: Initialization. Set $\mathscr{S}_0 = \{(\emptyset, i_0, 0)\}$ and $\mathscr{S}_k = \emptyset$, for $k = 2, \cdots, |\mathcal{V}^{tp}|$. Let $\hat{k} = 1$.

Step 2: Expansion of every state in set $\mathscr{S}_{\hat{k}-1}$. For each $(\widetilde{\mathcal{V}}, j, \widetilde{q}) \in \mathscr{S}_{\hat{k}-1}$, repeat the Step 3.

Step 3: Generate reachable states of $\mathscr{S}_{\hat{k}}$ from $(\widetilde{\mathcal{V}}, j, \widetilde{q})$. For each $a_{i,j} \in \mathcal{A}$ such that $u_i \in \mathcal{V}^{tp} \backslash \widetilde{\mathcal{V}}$, we generate $(\widetilde{\mathcal{V}}', i, \widetilde{q}')$, where $\widetilde{\mathcal{V}}' = \widetilde{\mathcal{V}} \cup \{u_i\}$ and $\widetilde{q}' = \widetilde{q} + \theta \left(\sum_{v \in \widetilde{\mathcal{V}}'} w_v + W^{\mathcal{D}} \right) (t_i - t_j)$.

The state is rejected if any of the following conditions folds: (i) feasibility check: $\widetilde{q}' \geq Q^{\mathcal{D}}$; (ii) dominance check: $\mathscr{S}_{\hat{k}}$ contains a state $(\widetilde{\mathcal{V}}', i, \widetilde{q}'')$ such that $\widetilde{q}' > \widetilde{q}''$. If the state is not rejected, let $\mathscr{S}_{\hat{k}} = \mathscr{S}_{\hat{k}} \cup \{(\widetilde{\mathcal{V}}', i, \widetilde{q}')\}$ and remove any state in $\mathscr{S}_{\hat{k}}$ that is dominated by $(\widetilde{\mathcal{V}}', i, \widetilde{q}')$.

Step 4: Update. Set $\hat{k} \leftarrow \hat{k} + 1$; if $\hat{k} \leq |\widetilde{\mathcal{V}}|$ go to Step 2, otherwise the DP is finished.

If we have any state $(\widetilde{\mathcal{V}}, j, \widetilde{q}) \in \mathscr{S}_{|\widetilde{\mathcal{V}}|}$ such that $\widetilde{q} + \theta t \left(\sum_{v \in \widetilde{\mathcal{V}}} w_v + W^{\mathcal{D}}\right)(t_i - t_j - \tau_0) \leq Q^{\mathcal{D}}$, this subset of vessels can be visited by one drone; otherwise, perform the DP for another unexplored station-time node until all station-time nodes have been tested.

E Proof of Proposition 2

It suffices to show that if (i) to (vii) hold, then $\forall j \in \mathcal{N}^{\mathcal{V}} : u_j \in \bar{\mathcal{V}}_{p_2}$ such that L_{p_2} can be extended to a feasible $L_{p_2'}$, we have that L_{p_1} can also be extended to a feasible $L_{p_1'}$ by visiting j and (i) to (v) still hold for $L_{p_1'}$ and $L_{p_2'}$.

Step 1: We show that the resource vector of $L_{p_1'}$ is less than $L_{p_2'}$. Firstly, according to (F.5), condition (i), (ii), and (iv) hold trivially.

Note that when we extend L_{p_1} to $L_{p_1'}$, $i^{p_1} = i^{p_2}$ cannot be a station-time node. Then, the battery energy consumption of p_1' and p_2' are $q^{p_1'} = q^{p_1} + \theta(t_{i^{p_1}} - t_j)(y^{p_1'} + W^D)$ and $q^{p_2'} = q^{p_2} + \theta(t_{i^{p_2}} - t_j)(y^{p_2'} + W^D)$, respectively. According to condition (v), we have $t_{i^{p_1}} = t_{i^{p_2}}$. Therefore, we can prove that

$$
\begin{aligned}
q^{p_1'} &= q^{p_1} + \theta(t_{i^{p_1}} - t_j)(y^{p_1'} + W^D) \\
&\leq q^{p_2} + \theta(t_{i^{p_1}} - t_j)(y^{p_1'} + W^D) && \text{(By initial condition (ii) } q^{p_1} \leq q^{p_2}) \\
&= q^{p_2} + \theta(t_{i^{p_2}} - t_j)(y^{p_1'} + W^D) && \text{(Since } t_{i^{p_1}} = t_{i^{p_2}}) \\
&\leq q^{p_2} + \theta(t_{i^{p_2}} - t_j)(y^{p_2'} + W^D) && \text{(Since } y^{p_1'} \leq y^{p_2'}) \\
&= q^{p_2'},
\end{aligned}
$$

which completes condition (iii).

In addition, recalling the definition of the partial reduced cost, we have

$$\begin{aligned} b_{p_1'} &= b_{p_1} + b_{j,i^{p_1}} && \text{(By definition)} \\ &= b_{p_1} + b_{j,i^{p_2}} && \text{(By initial condition (iv) } i^{p_1} = i^{p_2}) \\ &\leq b_{p_2} + b_{j,i^{p_2}} && \text{(By initial condition (v) } b_{p_1} \leq b_{p_2}) \\ &= b_{p_2'}. && \text{(By definition)} \end{aligned}$$

Therefore, condition (vi) holds.

In view of the condition (vii), since $(\mathcal{V}^{sr} \cup \mathcal{V}_{p_1}) \subseteq (\mathcal{V}^{sr} \cup \mathcal{V}_{p_2}), \forall sr \in \mathcal{SR}$, we have $(\mathcal{V}^{sr} \cup \mathcal{V}_{p_1'}) \subseteq (\mathcal{V}^{sr} \cup \mathcal{V}_{p_2'}), \forall sr \in \mathcal{SR}$.

Step 2: We show that the full reduced cost of $L_{p_1'}$ is less than $L_{p_2'}$. Since $\mathcal{V}_{p_1'} \subseteq \mathcal{V}_{p_2'}$, we have

$$\mathcal{K}_{p_1'} = \{k \in \mathcal{K} : |\mathcal{V}_{p_1'} \cap \mathcal{V}^k| \geq 2\} \subseteq \{k \in \mathcal{K} : |\mathcal{V}_{p_2'} \cap \mathcal{V}^k| \geq 2\} = \mathcal{K}_{p_2'}.$$

Thus, $\forall k \in \mathcal{K}$, we have $0 \leq \mathbb{1}_{\mathcal{K}_{p_1'}}(k) \leq \mathbb{1}_{\mathcal{K}_{p_2'}}(k)$. Furthermore, since $\mu_k \leq 0$, we have

$$-\sum_{k \in \mathcal{K}} \mathbb{1}_{\mathcal{K}_{p_1'}}(k)\mu_k \leq -\sum_{k \in \mathcal{K}} \mathbb{1}_{\mathcal{K}_{p_2'}}(k)\mu_k.$$

Considering initial condition (vi) $b_{p_1'} \leq b_{p_2'}$, we can prove that

$$\bar{c}_{p_1'} = b_{p_1'} - \sum_{k \in \mathcal{K}} \mathbb{1}_{\mathcal{K}_{p_1'}}(k)\mu_k \leq b_{p_2'} - \sum_{k \in \mathcal{K}} \mathbb{1}_{\mathcal{K}_{p_2'}}(k)\mu_k = \bar{c}_{p_2'}.$$

Step 3: We then prove the feasibility of $L_{p_1'}$ by showing that constraints (2.13) *hold for $L_{p_1'}$.* Since conditions (ii) and (iii) hold for $L_{p_1'}$ and $L_{p_2'}$, i.e., $y^{p_1'} \leq y^{p_2'}$ and $q^{p_1'} \leq q^{p_2'}$.

Therefore, considering the feasibility of $L_{p'_2}$, we have

$$y^{p'_1} \leq y^{p'_2} \leq C^{\mathrm{D}}, q^{p'_1} \leq q^{p'_2} \leq Q^{\mathrm{D}}.$$

F Algorithms

F.1 Feasibility pump pseudocode

Algorithm F.1: Feasibility pump algorithm.

Input: The number of variables to be flipped κ, maximum iteration number $\hat{\eta}$.
Output: Feasible integer solution $\boldsymbol{\chi}^*$.

Compute $\boldsymbol{\chi}^* \in \arg\min\{\sum_{p\in\mathcal{P}'} c_p\chi_p$: s.t.(2.4b) − (2.4d), (2.10)$\}$
if $\boldsymbol{\chi}^*$ *is integer* **then**
  return $\boldsymbol{\chi}^*$
else
  Let $\tilde{\boldsymbol{\chi}} := [\boldsymbol{\chi}^*]$, $\eta = 0$ `//` $[x]$ `means round` x `to the nearest integer.`
  while $\eta < \hat{\eta}$ **do**
    Let $\eta = \eta + 1$
    Compute $\boldsymbol{\chi}^* \in \arg\min\{\sum_{p\in\mathcal{P}':\tilde{\chi}_p=0} \chi_p + \sum_{p\in\mathcal{P}':\tilde{\chi}_p=1}(1-\chi_p)$: s.t.(2.4b) − (2.4d), (2.10)$\}$
    if $\boldsymbol{\chi}^*$ *is integer* **then**
      return $\boldsymbol{\chi}^*$
    end
    if $\exists p \in \mathcal{P}' : \tilde{\chi}_p \neq \left[\chi^*_p\right]$ **then**
      $\tilde{\boldsymbol{\chi}} := [\boldsymbol{\chi}^*]$
    else
      Flip the $\tilde{\kappa}$ entries $\tilde{\chi}_p$ $(p \in \mathcal{P}')$ with highest $\left|\chi^*_p - \tilde{\chi}_p\right|$, where $\tilde{\kappa}$ is a random integer between $[\kappa/2, 3\kappa/2]$
      `/* Flip is to prevent stalling issues by taking the reverse rounding of some variables when all rounded variables are the same as the last iteration. */`
    end
  end
end

F.2 Labeling algorithm pseudocode

Algorithm F.2: Backward labeling algorithm to generate drone path.

Input: The time-expanded network and the reduced cost of each arc. Parameters Num_L, Num_{L^*} and $flag$.

Output: The generated columns and revised $flag$.

$List_{L^*} = \emptyset$ and $List_L = \emptyset$
foreach $i_0 \in \mathcal{N}^0$ **do**
  Initialization: a single label $L_p = (\mathcal{V}_p, y^p, q^p, \mathcal{T}_p, i^p, b_p, \bar{c}_p)$ is generated, where $\mathcal{V}_p = \emptyset$, $q^p = 0$, $y^p = 0$, $\mathcal{T}_p = \emptyset$, $i^p = i_0$, $b_p = 0$, $\bar{c}_p = 0$, $List_L$.add(L_p)
  if $List_L$.*size*$> Num_L$ & *flag* **then**
    Sort $List_L$ by the increasing order of the reduced costs of the labels
    Delete the last $List_L$.size$-Num_L$ labels with the largest reduced costs
  end
  while $List_L \neq \emptyset$ **do**
    Get label $L_p = (\mathcal{V}_p, y^p, q^p, \mathcal{T}_p, i^p, b_p, \bar{c}_p)$ from $List_L$
    foreach $a_{j,i^p} \in \mathcal{A}$ **do** `// There are arcs whose target node is` i^p
      Extension: extend L_p towards node j such that $a_{j,i^p} \in \mathcal{A}$ and $u_j \notin \mathcal{V}_p$; generate label $L_{p'} = (\mathcal{V}_{p'}, y^{p'}, q^{p'}, \mathcal{T}_{p'}, i^{p'}, b_{p'}, \bar{c}_{p'})$ if it is feasible according to (2.13), where

$$\begin{cases} \mathcal{V}_{p'} = \begin{cases} \mathcal{V}_p & \text{if } j \in \mathcal{N}^0 \\ \mathcal{V}_p \cup \{u_j\} & \text{if } j \in \mathcal{N}^{\mathcal{V}} \end{cases} \\ y^{p'} = y^p + w_{u_j} \\ q^{p'} = \begin{cases} q^p + \theta(y^p + W^{\mathcal{D}})(t_{i^p} - t_j - \tau_0) & \text{if } i^p \in \mathcal{N}^0 \\ q^p + \theta(y^p + W^{\mathcal{D}})(t_{i^p} - t_j) & \text{if } i^p \in \mathcal{N}^{\mathcal{V}} \end{cases} \\ \mathcal{T}_{p'} = \mathcal{T}_p \cup \{t \mid t = k\Delta, t_j \leq t < t_{i^p}, k \in \mathbb{N}_0\} \\ i^{p'} = j \\ b_{p'} = b_p + b_{j,i^p} \\ \text{compute } \bar{c}_{p'} \text{ by (2.11)} \end{cases} \tag{F.5}$$

      if $i^{p'} \in \mathcal{N}^0$ **then** `// A complete feasible path is generated`
        $List_{L^*}$.add($L_{p'}$)
      else
        $List_L$.add($L_{p'}$)
        Dominance rule: apply Proposition 2 and delete the dominated labels from $List_L$ or $List_{L^*}$
      end
    end
    Delete L_p from $List_L$
  end
end
Sort $List_L^*$ by the increasing order of the labels' reduced costs, set $flag$ = false
if *the first label's reduced cost of* $List_L^*$ *is negative* **then**
  Select the first Num_{L^*} paths with the most negative reduced costs from $List_{L^*}$ and set $flag$ = true
end
Return the columns corresponding to paths and flag.

F.3 Branch-and-price-and-cut pseudocode

Algorithm F.3: Branch-and-price-and-cut algorithm for drone scheduling problem.

Input: The constructed expanded network and a convergence gap ϵ.
Output: Optimal integer solution $\boldsymbol{\chi}^*$.

```
Create a B&B tree and insert the root node into the tree
Set LB= −∞, UB= +∞, BestSol← ∅
while UB−LB ≥ ϵ do
    Select a node from the tree by best-bound-first strategy
    if the current node is the root node then
        Add the SRIs
        Solve the RMP by CG and generate the RCIs and TPIs iteratively
    else
        Solve the RMP by CG with existing SRIs, RCIs and TPIs
    end
    if the RMP is infeasible then
        Mark the current node as explored
    else
        Get an optimal solution χ̄ with objective value Ψ
        Set LB = min{the local objective of all leaf nodes}
        if χ̄ is integral then
            if Ψ <UB then Set BestSol← χ̄ and UB = Ψ
            Mark the current node as explored
        else
            if the current node is the root node then
                Perform the FP algorithm or solve the IP form of RMP to get the
                  solution χ̂ with objective value Ψ̂
                if χ̂ is integeral then Set BestSol← χ̂, UB= Ψ̂
            end
            if Ψ <UB then
                Branch on χ̄ on the arc
                Set the estimated bound for each child node as Ψ
                Insert two resulting child nodes into the tree
            end
            Mark the current node as explored
        end
    end
end
Return the BestSol and UB
```

G Summary of critical notations

Table G.1: Summary of sets, parameters, and decision variables

Notation	Description
$\mathcal{L}$	Set of cruise lines, indexed by l. It includes the set of bi-directional $\mathcal{L}^b$ and round-trip itineraries $\mathcal{L}^r$.
$\mathcal{H}$	Set of home ports, indexed by h. The set of home port(s) for a cruise line l is denoted by $\mathcal{H}_l$.
$\mathcal{S}$	Set of station ports, indexed by s. The set of station ports for a cruise line l is denoted by $\mathcal{S}_l$.
$\mathcal{P}$	Set of ports, indexed by p. It includes the set of home ports $\mathcal{H}$ and station ports $\mathcal{S}$. The set of ports for a cruise line l is denoted by $\mathcal{P}_l = \mathcal{H}_l \cup \mathcal{S}_l$.
$\mathcal{K}$	The set of time periods.
$\mathcal{T}$	The set of timestamps in the time-expanded network after discretization. The set of timestamps in the time-expanded network before time t is denoted by $\mathcal{T}_t$.
$\widetilde{\mathcal{T}}_k$	The set of timestamps contained in a time period k.
$\mathcal{N}$	The set of nodes in the time-expanded network.
$\mathcal{A}$	The set of arcs in the time-expanded network. The arc set of cruise line l is denoted by $\mathcal{A}_l$.
$\mathcal{A}_{l,k}$	The set of outflow arcs of a home port h of a round cruise line $l \in \mathcal{L}^r$.
$\mathcal{A}^i_{l,k}$	The set of outflow arcs of a home port h of a bi-directional line $l \in \mathcal{L}^b$ in directional $i \in \{0,1\}$.
$t^{\text{sail}}_{p,q}$	The sailing time from port $p \in \mathcal{P}_l$ to $q \in \mathcal{P}_l$.
$t^{\text{opp}}_{s,l}$	The touring time at each station port $s \in \mathcal{S}_l$ of cruise line $l \in \mathcal{L}$.
$t^{\text{layover}}_{h,l}$	The layover time at each home port $h \in \mathcal{H}_l$ of cruise line $l \in \mathcal{L}$.
$d_{l,k}$	The required number of cruises for each period $k \in \mathcal{K}$ of a round-trip $l \in \mathcal{L}^r$.
$d^i_{l,k}$	The required number of cruises for each period $k \in \mathcal{K}$ of a bi-directional line $l \in \mathcal{L}^b$ in direction $i \in \{0,1\}$.
z_h	Nonnegative integer, indicating the number of cruises allocated at home port h.
$u_{h,l}$	Nonnegative integer, indicating the number of cruises allocated at home port h and assigned to line l.
x_a	Binary, taking 1 if arc a is included in the cruises' itineraries and 0, otherwise.
$y_{l,k}$	Nonnegative integer, indicating the number of unsatisfied demands of a round-trip itinerary $l \in \mathcal{L}^r$ at time period k.
$y^i_{l,k}$	Nonnegative integer, indicating the number of unsatisfied demands of a bi-directional itinerary $l \in \mathcal{L}^b$ at time period k in direction $i \in \{0,1\}$.

H Branching on the number of drones

For each station-time node $i \in \mathcal{N}^0$, we let d_i be the number of drones flying out of node i, which can be calculated by $\sum_{p\in\mathcal{P}'} \sum_{a_{i,j}\in\mathcal{A}:j\in\mathcal{N}^{\mathcal{V}}} \mathbb{1}_{\mathcal{A}_p}(a_{i,j})\chi_p^*$. If d_i is factional, we impose a constraint $\sum_{p\in\mathcal{P}'} \sum_{a_{i,j}\in\mathcal{A}:j\in\mathcal{N}^{\mathcal{V}}} \mathbb{1}_{\mathcal{A}_p}(a_{i,j})\chi_p \geq \lceil d_i \rceil$ to the RMP of one child node and force $\sum_{p\in\mathcal{P}'} \sum_{a_{i,j}\in\mathcal{A}:j\in\mathcal{N}^{\mathcal{V}}} \mathbb{1}_{\mathcal{A}_p}(a_{i,j})\chi_p \leq \lfloor d_i \rfloor$ to the RMP of the other child node. We choose to branch on the earlier time point when there are multiple fractional d_i. Notably, the dual prices associated with each drone-number branching constraint should also be included when calculating the reduced cost of each path.

I Proof of Lemma 1

We prove that $\text{SP}_l(\hat{\boldsymbol{u}})$ and $\widetilde{\text{SP}}_l(\hat{\boldsymbol{u}})$ are equivalent by showing that any feasible solution to $\text{SP}_l(\hat{\boldsymbol{u}})$ is also feasible to $\widetilde{\text{SP}}_l(\hat{\boldsymbol{u}})$ with the objective value remaining the same, and the converse is also true.

(i) Assume that $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is a feasible solution to $\text{SP}_l(\hat{\boldsymbol{u}})$. Firstly, the binary restriction of x_a can be achieved by constraints (8d)–(8e) combined with integer restrictions of χ_a.

According to the definition, we have $\sum_{\tau\in\mathcal{T}_t}(\sum_{a\in\delta^+(n_{h,\tau},l)} x_a^* - \sum_{a\in\delta^-(n_{h,\tau},l)} x_a^*) = \sum_{a\in\delta^+(n_{h,t},l)} \chi_a - \sum_{a\in\delta^-(n_{h,t},l)} \chi_a$. Therefore, constraints (8b) and constraints (8c) hold trivially. Similarly, by definition, we have $\sum_{a\in\mathcal{A}_{l,k}} x_a^* = \sum_{a\in\delta^+(n_{h,\overline{\tau}(k)},l)} \sum_{a'\in\mathcal{A}(a)} x_{a'}^* - \sum_{a\in\delta^+(n_{h,\underline{\tau}(k)},l)} \sum_{a'\in\mathcal{A}^-(a)} x_{a'}^* = \sum_{a\in\delta^+(n_{h,\overline{\tau}(k)},l)} \chi_a - \sum_{a\in\delta^+(n_{h,\underline{\tau}(k)},l)} \chi_{a^-}$. Thus, constraints (8f) are met.

Moreover, according to $x_a^* = \chi_a - \chi_{a^-}$ for $a \in \delta^+(n_{h,t}, l)$ and constraints (4d), we have $x_{a'}^* = \chi_a - \chi_{a^-}, \forall a' \in \mathcal{A}_l(h,t), a \in \delta^+(n_{h,t}, l)$. Combining with $c_{l,t}^{\text{route}} = \sum_{a\in\mathcal{A}_l(h,t)}(c_a^{\text{fuel}} - r_a^{\text{uti}})$, the objective values are equivalent.

(ii) Conversely, assume that $(\boldsymbol{\chi}^*, \boldsymbol{y}^*)$ is a feasible solution to $\widetilde{\text{SP}}_l(\hat{\boldsymbol{u}})$. Based on the definition and proof in (i), constraints (4b)–(4c) and (4e)–(4g) are straightforward. By

defining $x_{a'} = \chi^*_a - \chi^*_{a^-}, \forall a' \in \mathcal{A}_l(h,t), a \in \delta^+(n_{h,t}, l)$, constraints (4d) are satisfied.

The proof is completed.

J Proof of Lemma 2

For ease of explanation, we denote A as the coefficient matrix of constraints (8b)–(8f), which includes matrix A_1 associating to the coefficients of $\boldsymbol{\chi}$, matrix $\mathbf{0}$ associating to the coefficients of $\boldsymbol{y}$ in constraints (8b)–(8e), and matrix $\boldsymbol{I}$ associating to the coefficients of $\boldsymbol{y}$ in constraints (8f). The structure of A can be presented as follows:

$$A = \begin{array}{c} \begin{matrix} \boldsymbol{\chi} & \boldsymbol{y} \end{matrix} \\ \begin{bmatrix} & \mathbf{0} \\ A_1 & \\ & \boldsymbol{I} \end{bmatrix} \end{array}$$

Firstly, according to Hoffman and Kruskal (2010), a sufficient condition for a matrix M to be TU is: row indices of matrix M can be partitioned into two sets such that the following four conditions are all satisfied: (i) each entry m_{ij} of M satisfies $m_{ij} \in \{0, 1, -1\}$; (ii) each column of M contains at most two non-zero entries; (iii) if a column has two entries of the same sign, their row indices are in different sets; and (iv) if a column has two entries of different signs, their row indices are in the same set.

For our matrix A_1, we consider its transpose matrix A_1^T. Considering the coefficients structure of matrix A_1, it contains at most one 1 and one -1 in each row, which implies that there are no more than two nonzero entries per column in the matrix A_1^T. Then, conditions (i) and (ii) are satisfied immediately. Therefore, we can partition all rows of A_1^T into a single group such that the sum of the columns results in a vector with entries of $0, 1$, and -1, meeting the criteria (iii) and (iv). Thus, we conclude that matrix A_1^T is TU.

Following the properties of a TU matrix (Fulkerson and Gross, 1965), we have (i) the transpose matrix of TU is also TU; and (ii) the resulting matrix after appending a unit

vector $\boldsymbol{e}$ to a TU, is also TU. Therefore, after appending the coefficient matrix associated with $\boldsymbol{y}$, the original coefficient matrix A is still TU.

K Proof of Lemma 3

(i) Feasibility: For each $\widetilde{\text{SP}}_l(\hat{\boldsymbol{u}})$, consider a valid solution $\hat{\boldsymbol{u}} \geq 0$, there always exists a solution $\chi_a = 0, \forall a \in \bar{\mathcal{A}}_l$ and $y_{l,k} = d_{l,k}, \forall k \in \mathcal{K}$ such that the model is feasible.

(ii) Boundedness: Since $0 \leq \chi_a - \chi_{a^-} \leq 1$, we can calculate a lower bound of the first term of the objective, i.e., $\sum_{t\in\mathcal{T}} \sum_{a\in\delta^+(n_{h,t},l)} c^{\text{route}}_{l,t}(\chi_a - \chi_{a^-})$, as $\sum_{t\in\{t'\in\mathcal{T}|c^{\text{route}}_{l,t'}<0\}} c^{\text{route}}_{l,t}$. We can also calculate an upper bound for the first term as $\sum_{t\in\{t'\in\mathcal{T}|c^{\text{route}}_{l,t'}>0\}} c^{\text{route}}_{l,t}$. Besides, the second term $\sum_{k\in\mathcal{K}} c^{\text{opp}}_{l,k} y_{l,k}$ lies in $[0, \sum_{k\in\mathcal{K}} c^{\text{opp}}_{l,k} d_{l,k}]$. Therefore, $\sum_{t\in\{t'\in\mathcal{T}|c^{\text{route}}_{l,t'}<0\}} c^{\text{route}}_{l,t}$ and $\sum_{t\in\{t'\in\mathcal{T}|c^{\text{route}}_{l,t'}>0\}} c^{\text{route}}_{l,t} + \sum_{k\in\mathcal{K}} c^{\text{opp}}_{l,k} d_{l,k}$ is a lower and upper bound of the objective, respectively. In summary, $\widetilde{\text{SP}}_l(\hat{\boldsymbol{u}})$ is bounded.

L Magnanti-Wong Benders cut generation algorithm

Algorithm L.4: Magnanti-Wong Benders cut generation.

Input: RMP feasible solution $(\hat{\boldsymbol{u}}, \hat{\boldsymbol{\zeta}})$.
Output: Benders cuts.

```
for l ∈ L do
    Solve DSP_l(û) and obtain optimal objective value η̂_l
    if η̂_l > ζ̂_l then
        Solve MW-DSP_l(u^0, û, η̂) and obtain optimal solution (α*, λ*, ω*)
        Add the Benders optimality cut (10) based on (α*, λ*, ω*) to RMP
    end
end
```

M Proof of Lemma 4

We prove the statement from the perspective of primal problems. We first write out the Magnanti-Wong primal problem MW-PSP$_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\eta}})$ as follows:

$$\min_{\boldsymbol{\chi},\boldsymbol{y},\sigma_l} : \sum_{t\in\mathcal{T}} \sum_{a\in\delta^+(n_{h,t},l)} c^{\text{route}}_{l,t}(\chi_a - \chi_{a^-}) + \sum_{k\in\mathcal{K}} c^{\text{opp}}_{l,k} y_{l,k} - \quad \hat{\eta}_l\sigma_l \tag{M.6}$$

$$\text{s.t.} \sum_{a\in\delta^+(n_{h,t},l)} \chi_a - \sum_{a\in\delta^-(n_{h,t},l)} \chi_a \le u^0_{h,l} + \hat{u}_{h,l}\sigma_l \qquad \forall h\in\mathcal{H}, t\in\mathcal{T}\backslash\{T\}, \tag{M.7}$$

$$\chi_a - \chi_{a^-} \le 1 + \sigma_l \qquad \forall a\in\bar{\mathcal{A}}_l, \tag{M.8}$$

$$\sum_{a\in\delta^+(n_{h,\overline{\tau}(k)},l)} \chi_a - \sum_{a\in\delta^+(n_{h,\underline{\tau}(k)},l)} \chi_{a^-} + y_{l,k} = d_{l,k} + d_{l,k}\sigma_l \forall k\in\mathcal{K}, \tag{M.9}$$

(8c) − (8d),

$$\chi_a, y_{l,k} \ge 0 \qquad \forall k\in\mathcal{K}, a\in\bar{\mathcal{A}}_l, \tag{M.10}$$

$$\sigma_l \in \mathbb{R}, \tag{M.11}$$

where σ_l is the dual of constraint (13b).

We next show that given an optimal solution to SP LSP$_l(\hat{\boldsymbol{u}})$, i.e., $(\hat{\boldsymbol{\chi}}, \hat{\boldsymbol{y}})$, with optimal objective $\hat{\eta}_l$, and an optimal solution to the Magnanti-Wong primal problem MW-PSP$_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\eta}})$, i.e., $(\boldsymbol{\chi}^*, \boldsymbol{y}^*, \sigma_l^*)$, the solution defined by $\boldsymbol{\chi} = \boldsymbol{\chi}^* + \nu\hat{\boldsymbol{\chi}}, \boldsymbol{y} = \boldsymbol{y}^* + \nu\hat{\boldsymbol{y}}, \sigma_l = \sigma_l^* + \nu$, where $\nu \ge 0$, is also an optimal solution to MW-PSP$_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\eta}})$.

(i) Feasibility: Since all variables are nonnegative variables, their combinations with $\nu \ge 0$ are also nonnegative, which adhere to constraints (M.10)–(M.11). Besides, as the right-hand side of constraints (8c)–(8d) are the same in LSP$_l(\hat{\boldsymbol{u}})$ and MW-PSP$_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\eta}})$, these constraints also hold trivally. For constraints (M.7), we have

$$\sum_{a\in\delta^+(n_{h,t},l)} (\chi^*_a + \nu\hat{\chi}_a) - \sum_{a\in\delta^-(n_{h,t},l)} (\chi^*_a + \nu\hat{\chi}_a)$$

$$
\begin{aligned}
&= \sum_{a\in\delta^+(n_{h,t},l)} \chi^*_a - \sum_{a\in\delta^-(n_{h,t},l)} \chi^*_a + \sum_{a\in\delta^+(n_{h,t},l)} \nu\hat{\chi}_a - \sum_{a\in\delta^-(n_{h,t},l)} \nu\hat{\chi}_a \\
&\le u^0_{h,l} + \hat{u}_{h,l}\sigma^*_l + \nu\hat{u}_{h,l} \\
&= u^0_{h,l} + \hat{u}_{h,l}(\sigma^*_l + \nu).
\end{aligned}
$$

Similarly, constraints (M.8)–(M.9) are satisfied. Therefore, the new solution is feasible.

(ii) Optimality: for the new objective, we have

$$
\begin{aligned}
&\sum_{t\in\mathcal{T}} \sum_{a\in\delta^+(n_{h,t},l)} c^{\text{route}}_{l,t}(\chi^*_a - \chi^*_{a^-} + \nu(\hat{\chi}_a - \hat{\chi}_{a^-})) + \sum_{k\in\mathcal{K}} c^{\text{opp}}_{l,k}(y^*_{l,k} + \nu\hat{y}_{l,k}) - \hat{\eta}_l(\sigma^*_l + \nu) \\
=&\sum_{t\in\mathcal{T}} \sum_{a\in\delta^+(n_{h,t},l)} c^{\text{route}}_{l,t}(\chi^*_a - \chi^*_{a^-}) + \sum_{k\in\mathcal{K}} c^{\text{opp}}_{l,k} y^*_{l,k} - \hat{\eta}_l\sigma^*_l \\
&+ \nu\left(\sum_{t\in\mathcal{T}} \sum_{a\in\delta^+(n_{h,t},l)} c^{\text{route}}_{l,t}(\hat{\chi}_a - \hat{\chi}_{a^-}) + \sum_{k\in\mathcal{K}} c^{\text{opp}}_{l,k}\hat{y}_{l,k} - \hat{\eta}_l\right) \\
=&\sum_{t\in\mathcal{T}} \sum_{a\in\delta^+(n_{h,t},l)} c^{\text{route}}_{l,t}(\chi^*_a - \chi^*_{a^-}) + \sum_{k\in\mathcal{K}} c^{\text{opp}}_{l,k} y^*_{l,k} - \hat{\eta}_l\sigma^*_l + \nu(\hat{\eta}_l - \hat{\eta}_l) \\
=&\sum_{t\in\mathcal{T}} \sum_{a\in\delta^+(n_{h,t},l)} c^{\text{route}}_{l,t}(\chi^*_a - \chi^*_{a^-}) + \sum_{k\in\mathcal{K}} c^{\text{opp}}_{l,k} y^*_{l,k} - \hat{\eta}_l\sigma^*_l,
\end{aligned}
$$

which equals the optimal objective value of MW-PSP$_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\eta}})$.Therefore, given a σ_l larger than σ^*_l, there always exists an optimal solution to the MW-PSP$_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\eta}})$. Then, we can fix σ_l and solve the model only with variables $\boldsymbol{\chi}$ and $\boldsymbol{y}$, that is, we can drop the term $-\hat{\eta}_l\sigma_l$ and there is no need to solve the original LSP$_l(\hat{\boldsymbol{u}})$. Besides, the dual of MW-PSP$_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\eta}})$ after removing the term $-\hat{\eta}_l\sigma_l$ in the objective leads to sMW-DSP$_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}})$.

Conclusively, provided a sufficient large σ_l, solving the sMW-DSP$_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}})$ is equivalent to solving the DSP$_l(\hat{\boldsymbol{u}})$ and MW-DSP$_l(\boldsymbol{u}^0, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\eta}})$ sequentially.

N Simultaneous Magnanti-Wong Benders cut generation algorithm

Algorithm N.5: Simultaneous Magnanti-Wong Benders cut generation.

Input: RMP feasible solution $(\hat{\boldsymbol{u}}, \hat{\boldsymbol{\zeta}})$, UB_l and LB_l.
Output: Benders cuts.

```
for l ∈ 𝓛 do
    Set σ = UB_l − LB_l
    Solve sMW-DSP_l(u^0, û) and obtain optimal solution (α*, λ*, ω*) with
      objective value η̂_l
    if η̂_l > ζ̂_l then
        Add the Benders optimality cut (10) based on (α*, λ*, ω*) to RMP
    end
end
```

O Simulation instance generation

The parameters for each instance are determined according to the following procedures:

• *Cruise-related parameters.* Each line is first assigned a type, either bi-directional or round-trip, with a probability of 0.5 for each type. The number of station ports for each line, i.e., $|\mathcal{S}_l|$ for round-trip line or $|\mathcal{S}_l^0| = |\mathcal{S}_l^1|$ for bi-directional line, is selected uniformly at random from $\{5, 6, \cdots, 9\}$. The demand for each time period $k \in \mathcal{K}$ is selected uniformly at random from $\{0, 1, 2, 3\}$. The total number of available cruises, i.e., $Z_{\max}$, is set as $3|\mathcal{L}|$.

• *Cost-related parameters.* The fixed cruise deployment cost for each home port is selected uniformly at random from $\{10, 20, 30, 40\}$ million dollars. The hourly fuel cost is set as 10,000 dollars/hour according to the daily fuel consumption of a cruise (Marine Insight, 2024) and the fuel cost (Ship & Bunker, 2024). We establish the time utility function based on the model presented in Wang et al. (2017). The utility is multiplied

by a constant of 10,000,000, considering that the number of people on a medium-sized cruise ship is approximately 1,000 and each unit of utility corresponds to a revenue gain of 10,000 dollars. Additionally, we assume a 0.1 probability that a port city is suitable for evening excursions, in which case the time utility function is shifted by 12 hours from the original utility function. The unit opportunity cost of one cruise line is set to $5\times|\mathcal{S}_l|$ million dollars as usually the longer the cruise line the higher the price.

• *Time-related parameters.* The sailing time between two successive ports is generated based on a uniform distribution ranging from 10 to 100 hours. The traveling time at each station port is generated following a uniform distribution ranging from 5 to 50 hours. The layover time of each home port is set as 48 hours. The planning horizon is set as 180 days and the time discretization parameter is set as 1 hour. Each time period $k \in \mathcal{K}$ is set to 9 days.

P Arc-based model

$$\min_{\boldsymbol{\chi},\boldsymbol{y}} : \sum_{a\in\bar{\mathcal{A}}_l}(c_a^{\text{fuel}} - r_a^{\text{uti}})(\chi_a - \chi_{a^-}) + \sum_{k\in\mathcal{K}} c_{l,k}^{\text{opp}} y_{l,k} \tag{P.12a}$$

$$\text{s.t.} \sum_{a\in\delta^+(n_{h,t},l)} \chi_a - \sum_{a\in\delta^-(n_{h,t},l)} \chi_a \le \hat{u}_{h,l} \qquad \forall h\in\mathcal{H}, t\in\mathcal{T}\backslash\{T\}, \tag{P.12b}$$

$$\sum_{a\in\delta^+(n_{h,T},l)} \chi_a - \sum_{a\in\delta^-(n_{h,T},l)} \chi_a = 0 \qquad \forall h\in\mathcal{H}, \tag{P.12c}$$

$$\sum_{a\in\delta^+(n_{s,t},l)} \chi_a - \sum_{a\in\delta^-(n_{s,t},l)} \chi_a = 0 \qquad \forall s\in\mathcal{S}, t\in\mathcal{T}, \tag{P.12d}$$

$$\chi_a - \chi_{a^-} \ge 0 \qquad \forall a\in\bar{\mathcal{A}}_l, \tag{P.12e}$$

$$\chi_a - \chi_{a^-} \le 1 \qquad \forall a\in\bar{\mathcal{A}}_l, \tag{P.12f}$$

$$\sum_{a\in\delta^+(n_{h,\overline{\tau}(k)},l)} \chi_a - \sum_{a\in\delta^+(n_{h,\underline{\tau}(k)},l)} \chi_{a^-} + y_{l,k} = d_{l,k} \qquad \forall k\in\mathcal{K}, \tag{P.12g}$$

$$\chi_a, y_{l,k} \geq 0 \qquad \forall k \in \mathcal{K}, a \in \bar{\mathcal{A}}_l. \qquad \text{(P.12h)}$$

Q Detailed results on simulation instances

The detailed results on simulation instances are given in Table Q.1–Q.4. All costs in the table below are measured in 10,000 dollars.

Table Q.1: Detailed results of different solving methods on each instance I

Instance	Solving method	UB	LB	Gap%	Total CPU(s)	Per SP CPU(s)	#Iter	#Cut
30-1	CPLEX-a	2,916,500	$-28,459$	1075.80	4426.92	/	/	/
	CPLEX-r	958,229	958,229	0.00	142.65	/	/	/
	t-BD-a	958,229	958,229	0.00	254.64	7.55	33	111
	MW-BD-a	958,229	958,229	0.00	284.99	7.21	39	157
	sMW-BD-a	958,229	958,229	0.00	166.46	5.09	32	105
	t-BD-r	958,229	958,229	0.00	31.28	0.78	33	105
	MW-BD-r	958,229	958,229	0.00	27.53	0.67	37	142
	sMW-BD-r	958,229	958,229	0.00	26.01	0.66	32	105
30-2	CPLEX-a	779,573	746,679	4.20	4609.87	/	/	/
	CPLEX-r	767,103	767,103	0.00	208.36	/	/	/
	t-BD-a	767,103	767,103	0.00	290.43	7.94	36	111
	MW-BD-a	767,103	767,103	0.00	301.39	7.26	41	159
	sMW-BD-a	767,103	767,103	0.00	199.75	5.61	35	109
	t-BD-r	767,103	767,103	0.00	38.31	0.99	35	110
	MW-BD-r	767,103	767,103	0.00	31.55	0.69	40	153
	sMW-BD-r	767,103	767,103	0.00	28.09	0.69	33	106
30-3	CPLEX-a	2,658,000	$-28,459$	1089.10	5313.91	/	/	/
	CPLEX-r	864,246	864,246	0.00	228.42	/	/	/
	t-BD-a	864,246	864,246	0.00	298.24	8.67	34	134
	MW-BD-a	864,246	864,246	0.00	286.79	7.65	37	155
	sMW-BD-a	864,246	864,246	0.00	184.94	5.32	34	112
	t-BD-r	864,246	864,246	0.00	32.69	0.87	34	114
	MW-BD-r	864,246	864,246	0.00	31.94	0.72	39	152
	sMW-BD-r	864,246	864,246	0.00	27.44	0.72	34	112
30-4	CPLEX-a	740,110	682,989	0.08	4560.30	/	/	/
	CPLEX-r	691,084	691,084	0.00	228.86	/	/	/
	t-BD-a	691,084	691,084	0.00	259.86	7.30	35	120
	MW-BD-a	691,084	691,084	0.00	268.59	6.45	41	157
	sMW-BD-a	691,084	691,084	0.00	164.94	4.85	33	115
	t-BD-r	691,084	691,084	0.00	40.66	1.11	33	114
	MW-BD-r	691,084	691,084	0.00	31.92	0.69	41	160
	sMW-BD-r	691,084	691,084	0.00	27.17	0.69	33	114

Table Q.2: Detailed results of different solving methods on each instance II

Instance	Solving method	UB	LB	Gap%	Total CPU(s)	Per SP CPU(s)	#Iter	#Cut
30-5	CPLEX-a	3,000,500	1,059,046	952.90	5306.40	/	/	/
	CPLEX-r	1,134,948	1,134,948	0.00	299.14	/	/	/
	t-BD-a	1,134,948	1,134,948	0.00	306.24	8.56	35	125
	MW-BD-a	1,134,948	1,134,948	0.00	296.60	7.50	39	163
	sMW-BD-a	1,134,948	1,134,948	0.00	203.20	5.45	36	119
	t-BD-r	1,134,948	1,134,948	0.00	36.51	0.85	35	120
	MW-BD-r	1,134,948	1,134,948	0.00	32.90	0.72	39	161
	sMW-BD-r	1,134,948	1,134,948	0.00	31.42	0.73	35	119
50-1	CPLEX-r	1,409,465	1,409,465	0.00	390.56	/	/	/
	t-BD-a	1,409,465	1,409,465	0.00	568.85	10.42	54	209
	MW-BD-a	1,409,465	1,409,465	0.00	594.25	9.50	62	261
	sMW-BD-a	1,409,465	1,409,465	0.00	404.70	7.51	53	191
	t-BD-r	1,409,465	1,409,465	0.00	63.89	1.07	54	192
	MW-BD-r	1,409,465	1,409,465	0.00	63.19	0.95	61	245
	sMW-BD-r	1,409,465	1,409,465	0.00	60.47	0.99	54	191
50-2	CPLEX-r	1,454,338	1,454,338	0.00	516.35	/	/	/
	t-BD-a	1,454,338	1,454,338	0.00	558.89	10.80	51	199
	MW-BD-a	1,454,338	1,454,338	0.00	605.38	10.01	60	259
	sMW-BD-a	1,454,338	1,454,338	0.00	411.23	7.70	52	177
	t-BD-r	1,454,338	1,454,338	0.00	66.06	1.11	52	177
	MW-BD-r	1,454,338	1,454,338	0.00	64.08	0.99	59	249
	sMW-BD-r	1,454,338	1,454,338	0.00	63.43	1.01	54	179
50-3	CPLEX-r	1,061,253	1,061,253	0.00	375.92	/	/	/
	t-BD-a	1,061,253	1,061,253	0.00	523.61	9.41	55	224
	MW-BD-a	1,061,253	1,061,253	0.00	545.40	8.84	61	246
	sMW-BD-a	1,061,253	1,061,253	0.00	376.16	7.01	53	170
	t-BD-r	1,061,253	1,061,253	0.00	58.23	0.99	54	174
	MW-BD-r	1,061,253	1,061,253	0.00	56.04	0.87	59	226
	sMW-BD-r	1,061,253	1,061,253	0.00	54.40	0.93	54	170
50-4	CPLEX-r	1,406,902	1,406,902	0.00	469.78	/	/	/
	t-BD-a	1,406,902	1,406,902	0.00	558.80	11.35	49	210
	MW-BD-a	1,406,902	1,406,902	0.00	599.46	10.04	59	264
	sMW-BD-a	1,406,902	1,406,902	0.00	402.30	8.27	48	189
	t-BD-r	1,406,902	1,406,902	0.00	69.25	1.26	49	192
	MW-BD-r	1,406,902	1,406,902	0.00	68.95	1.09	59	269
	sMW-BD-r	1,406,902	1,406,902	0.00	60.86	1.14	48	190
50-5	CPLEX-r	1,317,487	1,317,487	0.00	462.61	/	/	/
	t-BD-a	1,317,487	1,317,487	0.00	601.45	10.38	57	250
	MW-BD-a	1,317,487	1,317,487	0.00	592.67	9.62	61	263
	sMW-BD-a	1,317,487	1,317,487	0.00	440.38	7.56	57	194
	t-BD-r	1,317,487	1,317,487	0.00	74.28	1.12	58	202
	MW-BD-r	1,317,487	1,317,487	0.00	67.30	1.02	60	258
	sMW-BD-r	1,317,487	1,317,487	0.00	66.03	1.02	57	191

Table Q.3: Detailed results of different solving methods on each instance III

Instance	Solving method	UB	LB	Gap%	Total CPU(s)	Per SP CPU(s)	#Iter	#Cut
80-1	CPLEX-r	476,774	476,774	0.00	2,472.60	/	/	/
	t-BD-a	476,774	476,774	0.00	675.30	6.83	96	406
	MW-BD-a	476,774	476,774	0.00	580.17	6.33	90	376
	sMW-BD-a	476,774	476,774	0.00	535.60	5.31	96	319
	t-BD-r	476,774	476,774	0.00	183.65	1.71	95	324
	MW-BD-r	476,774	476,774	0.00	150.21	1.51	90	377
	sMW-BD-r	476,774	476,774	0.00	182.69	1.68	96	318
80-2	CPLEX-r	2,592,712	2,592,712	0.00	881.04	/	/	/
	t-BD-a	2,592,712	2,592,712	0.00	1,393.19	16.32	84	383
	MW-BD-a	2,592,712	2,592,712	0.00	1,394.31	15.22	91	429
	sMW-BD-a	2,592,712	2,592,712	0.00	1,094.07	12.66	85	313
	t-BD-r	2,592,712	2,592,712	0.00	160.07	1.70	85	344
	MW-BD-r	2,592,712	2,592,712	0.00	153.19	1.57	90	428
	sMW-BD-r	2,592,712	2,592,712	0.00	159.26	1.67	85	316
80-3	CPLEX-r	2,555,774	2,555,774	0.00	774.38	/	/	/
	t-BD-a	2,555,774	2,555,774	0.00	1,384.78	17.06	80	357
	MW-BD-a	2,555,774	2,555,774	0.00	1,486.20	16.04	92	432
	sMW-BD-a	2,555,774	2,555,774	0.00	1,070.59	13.26	80	301
	t-BD-r	2,555,774	2,555,774	0.00	145.66	1.71	79	301
	MW-BD-r	2,555,774	2,555,774	0.00	160.11	1.60	93	422
	sMW-BD-r	2,555,774	2,555,774	0.00	144.58	1.70	79	301
80-4	CPLEX-r	2,490,002	2,490,002	0.00	1,088.33	/	/	/
	t-BD-a	2,490,002	2,490,002	0.00	1,411.77	17.00	82	425
	MW-BD-a	2,490,002	2,490,002	0.00	1,467.53	15.43	94	442
	sMW-BD-a	2,490,002	2,490,002	0.00	1,042.78	12.81	80	302
	t-BD-r	2,492,727	2,492,727	0.00	184.83	1.78	81	358
	MW-BD-r	2,490,002	2,490,002	0.00	168.03	1.58	94	431
	sMW-BD-r	2,490,002	2,490,002	0.00	148.79	1.68	79	305
80-5	CPLEX-r	2,774,685	2,774,685	0.00	853.42	/	/	/
	t-BD-a	2,774,685	2,774,685	0.00	1,391.55	16.73	82	388
	MW-BD-a	2,774,685	2,774,685	0.00	1,479.46	15.22	96	442
	sMW-BD-a	2,774,685	2,774,685	0.00	1,060.25	12.73	82	303
	t-BD-r	2,774,685	2,774,685	0.00	179.04	1.99	82	309
	MW-BD-r	2,774,685	2,774,685	0.00	163.55	1.61	92	435
	sMW-BD-r	2,774,685	2,774,685	0.00	151.92	1.71	81	304
100-1	CPLEX-r	618,540	618,540	0.00	2,200.52	/	/	/
	t-BD-a	618,540	618,540	0.00	1,189.86	9.40	122	517
	MW-BD-a	618,540	618,540	0.00	899.37	7.88	113	477
	sMW-BD-a	618,540	618,540	0.00	914.68	7.13	122	407
	t-BD-r	618,540	618,540	0.00	299.09	2.04	122	408
	MW-BD-r	618,540	618,540	0.00	272.66	1.92	116	485
	sMW-BD-r	618,540	618,540	0.00	309.53	2.08	122	425

Table Q.4: Detailed results of different solving methods on each instance IV

Instance	Solving method	UB	LB	Gap%	Total CPU(s)	Per SP CPU(s)	#Iter	#Cut
	CPLEX-r	2,749,930	2,749,930	0.00	2,331.50	/	/	/
	t-BD-a	2,749,930	2,749,930	0.00	1,010.78	9.44	103	475
	MW-BD-a	2,749,930	2,749,930	0.00	1,960.79	17.05	114	527
100-2	sMW-BD-a	2,749,930	2,749,930	0.00	1,477.71	14.35	102	383
	t-BD-r	2,749,930	2,749,930	0.00	222.30	2.05	102	383
	MW-BD-r	2,749,930	2,749,930	0.00	215.48	1.80	112	517
	sMW-BD-r	2,749,930	2,749,930	0.00	223.39	2.05	103	424
	CPLEX-r	2,665,245	2,665,245	0.00	2,109.65	/	/	/
	t-BD-a	2,665,245	2,665,245	0.00	1,774.21	16.64	106	562
	MW-BD-a	2,665,245	2,665,245	0.00	1,763.24	15.67	112	504
100-3	sMW-BD-a	2,665,245	2,665,245	0.00	1,429.67	13.75	103	374
	t-BD-r	2,665,245	2,665,245	0.00	206.74	1.92	104	373
	MW-BD-r	2,665,245	2,665,245	0.00	206.47	1.75	112	513
	sMW-BD-r	2,665,245	2,665,245	0.00	200.98	1.85	103	376
	CPLEX-r	3,068,185	3,068,185	0.00	2,271.02	/	/	/
	t-BD-a	3,068,185	3,068,185	0.00	1,968.29	18.99	103	558
	MW-BD-a	3,068,185	3,068,185	0.00	2,111.83	17.97	117	566
100-4	sMW-BD-a	3,068,185	3,068,185	0.00	1,612.21	15.58	103	377
	t-BD-r	3,068,185	3,068,185	0.00	282.89	2.52	104	381
	MW-BD-r	3,068,185	3,068,185	0.00	295.93	2.45	113	541
	sMW-BD-r	3,068,185	3,068,185	0.00	181.36	1.37	105	384
	CPLEX-r	2,326,163	2,326,163	0.00	1,980.01	/	/	/
	t-BD-a	2,326,163	2,326,163	0.00	1,628.90	15.97	101	646
	MW-BD-a	2,326,163	2,326,163	0.00	1,631.11	14.61	111	481
100-5	sMW-BD-a	2,326,163	2,326,163	0.00	1,276.76	12.65	100	346
	t-BD-r	2,326,163	2,326,163	0.00	193.24	1.82	100	344
	MW-BD-r	2,326,163	2,326,163	0.00	189.95	1.65	110	468
	sMW-BD-r	2,326,163	2,326,163	0.00	233.50	1.92	116	426

R Minimal infeasible subsystems cuts

According to Fischetti et al. (2010), the Benders' cuts generation can be regarded as detecting a minimal source of infeasibility of the following model [MIS-SP$_l(\hat{\boldsymbol{u}})$], so as to detect a small set of constraints in the slave that suffices to cut the master solution.

$$[\text{MIS-SP}_l(\hat{\boldsymbol{u}})] \quad \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\lambda},\boldsymbol{\omega}} : \sum_{h\in\mathcal{H}_l, t\in\mathcal{T}\setminus\{T\}} \hat{u}_{h,l}\alpha_{h,t} + \sum_{k\in\mathcal{K}} d_{l,k}\omega_k + \sum_{a\in\bar{\mathcal{A}}_l} \lambda_a - \hat{\zeta}_l\pi \tag{R.13a}$$

$$\text{s.t.} \sum_{h\in\mathcal{H}_l}\sum_{t\in\mathcal{T}\setminus\{T\}} (\mathbb{1}_{\delta^+(n_{h,t},l)}(a) - \mathbb{1}_{\delta^-(n_{h,t},l)}(a))\alpha_{h,t} + \sum_{h\in\mathcal{H}_l} (\mathbb{1}_{\delta^+(n_{h,T},l)}(a) - \mathbb{1}_{\delta^-(n_{h,T},l)}(a))\beta_h$$

$$+ \gamma_a + \lambda_a - \mathbb{1}_{\bar{\mathcal{A}}_l}(a^+)(\gamma_{a^+} + \lambda_{a^+}) + \sum_{k\in\mathcal{K}} (\mathbb{1}_{\delta^+(n_{h,\overline{\tau}(k)},l)}(a) - \mathbb{1}_{\delta^+(n_{h,\underline{\tau}(k)},l)}(a^+))\omega_k$$

$$\leq \sum_{t\in\mathcal{T}} (\mathbb{1}_{\delta^+(n_{h,t},l)}(a) - \mathbb{1}_{\delta^+(n_{h,t},l)}(a^+)) c^{\text{route}}_{l,t}\pi \qquad \forall a \in \bar{\mathcal{A}}_l, \tag{R.13b}$$

$$\omega_k \leq c^{\text{opp}}_{l,k}\pi \qquad \forall k \in \mathcal{K}, \tag{R.13c}$$

$$- \sum_{h\in\mathcal{H}_l, t\in\mathcal{T}\setminus\{T\}} v^1_{h,t}\alpha_{h,t} + v^0\pi = 1,$$

$$\boldsymbol{\alpha}, \boldsymbol{\lambda} \leq \mathbf{0}, \boldsymbol{\gamma}, \boldsymbol{v} \geq \mathbf{0}, \pi \geq 0. \tag{R.13d}$$

The classic Benders' decomposition algorithm arises as a special case by setting $v^0 = 1$ and $v^1_{h,t} = 0$, for all $h \in \mathcal{H}_l, t \in \mathcal{T}\setminus\{T\}$. Fischetti et al. (2010) point out that a more clever choice is however to set $v^0 = 1$ and $v^1_{h,t} = 1$, for all $h \in \mathcal{H}_l, t \in \mathcal{T}\setminus\{T\}$. Then, the SP can be written as follows:

$$[\text{MIS-SP}'_l(\hat{\boldsymbol{u}})] \quad \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\lambda},\boldsymbol{\omega}} : \sum_{h\in\mathcal{H}_l, t\in\mathcal{T}\setminus\{T\}} \hat{u}_{h,l}\alpha_{h,t} + \sum_{k\in\mathcal{K}} d_{l,k}\omega_k + \sum_{a\in\bar{\mathcal{A}}_l} \lambda_a - \hat{\zeta}_l\pi \tag{R.14a}$$

$$\text{s.t.} \sum_{h\in\mathcal{H}_l} \sum_{t\in\mathcal{T}\setminus\{T\}} (\mathbb{1}_{\delta^+(n_{h,t},l)}(a) - \mathbb{1}_{\delta^-(n_{h,t},l)}(a))\alpha_{h,t} + \gamma_a + \lambda_a - \mathbb{1}_{\bar{\mathcal{A}}_l}(a^+)(\gamma_{a^+} + \lambda_{a^+})$$

$$+ \sum_{h\in\mathcal{H}_l} (\mathbb{1}_{\delta^+(n_{h,T},l)}(a) - \mathbb{1}_{\delta^-(n_{h,T},l)}(a))\beta_h + \sum_{k\in\mathcal{K}} (\mathbb{1}_{\delta^+(n_{h,\overline{\tau}(k)},l)}(a) - \mathbb{1}_{\delta^+(n_{h,\underline{\tau}(k)},l)}(a^+))\omega_k$$

$$\leq \sum_{t\in\mathcal{T}} (\mathbb{1}_{\delta^+(n_{h,t},l)}(a) - \mathbb{1}_{\delta^+(n_{h,t},l)}(a^+)) c^{\text{route}}_{l,t}\pi \qquad \forall a \in \bar{\mathcal{A}}_l, \tag{R.14b}$$

$$\omega_k \leq c^{\text{opp}}_{l,k}\pi \qquad \forall k \in \mathcal{K}, \tag{R.14c}$$

$$- \sum_{h\in\mathcal{H}_l, t\in\mathcal{T}\setminus\{T\}} \alpha_{h,t} + \pi = 1, \tag{R.14d}$$

$$\boldsymbol{\alpha}, \boldsymbol{\lambda} \leq \mathbf{0}, \boldsymbol{\gamma} \geq \mathbf{0}, \pi \geq 0. \tag{R.14e}$$

Then, the corresponding MIS Benders' cuts associated with a given solution of $[\text{MIS-SP}'_l(\hat{\boldsymbol{u}})]$

reads

$$\zeta_l \hat{\pi} \geq \sum_{h \in \mathcal{H}_l, t \in \mathcal{T} \setminus \{T\}} u_{h,l} \hat{\alpha}_{h,t} + \sum_{k \in \mathcal{K}} d_{l,k} \hat{\omega}_k + \sum_{a \in \bar{\mathcal{A}}_l} \hat{\lambda}_a. \tag{R.15}$$

S Detailed numerical results on Figure 3.8

The detailed numerical results in Figure 8 are given in Table S.1–S.4.

Table S.1: Opened cruise lines and deployed number of cruises under benchmark scenario

Opened cruise line name	#Trips	Home port 1	#Cruises	Home port 2	#Cruises
10 Nights Grand Voyage	90	Tokyo	5	Hong Kong	/
11 Nights Hong Kong To Tokyo	80	Hong Kong	/	Tokyo	5
16 Nights World Cruise	80	Hong Kong	3	Tokyo	5
18 Nights World Cruise	50	Bali	5	Hong Kong	/
21 Nights Grand Voyage	90	Singapore	5	Taipei	5
22 Nights Grand Voyage	80	Tokyo	5	Bangkok	5
27 Nights Grand Voyage	42	Singapore	7	Incheon	/
29 Nights Grand Voyage	30	Singapore	5	Kyoto	/
32 Nights Grand Voyage	50	Tokyo	5	Bali	5
33 Nights Grand Voyage	50	Singapore	5	Tokyo	5
36 Nights Grand Voyage	50	Goa	5	Hong Kong	5
41 Nights Grand Voyage	12	Tokyo	3	Phuket	/
42 Nights Grand Voyage	20	Hong Kong	/	Singapore	5
60 Nights Grand Voyage	50	Tokyo	5	Sydney	15
63 Nights Grand Voyage	60	Tokyo	10	Sydney	10
57 Nights Grand Voyage[1]	15	Singapore	5	/	/

1: This cruise line with only one home port is a round-trip cruise line.

Table S.2: Opened cruise lines and deployed number of cruises with high r^{uti}

Opened cruise line name	#Trips	Home port 1	#Cruises	Home port 2	#Cruises
10 Nights Grand Voyage	90	Tokyo	4	Hong Kong	1
16 Nights World Cruise	82	Hong Kong	4	Tokyo	4
18 Nights World Cruise	10	Bali	/	Hong Kong	1
21 Nights Grand Voyage	90	Singapore	5	Taipei	5
22 Nights Grand Voyage	80	Tokyo	5	Bangkok	5
57 Nights Grand Voyage	15	Singapore	5	/	/
27 Nights Grand Voyage	54	Singapore	9	Incheon	/
29 Nights Grand Voyage	60	Singapore	5	Kyoto	5
32 Nights Grand Voyage	50	Tokyo	5	Bali	5
33 Nights Grand Voyage	50	Singapore	5	Tokyo	5
36 Nights Grand Voyage	50	Goa	5	Hong Kong	5
57 Nights Grand Voyage	15	Singapore	5	/	/
41 Nights Grand Voyage	40	Tokyo	10	Phuket	/
42 Nights Grand Voyage	40	Hong Kong	5	Singapore	5
60 Nights Grand Voyage	30	Tokyo	5	Sydney	5
63 Nights Grand Voyage	60	Tokyo	10	Sydney	10
57 Nights Grand Voyage	15	Singapore	5	/	/

Table S.3: Opened cruise lines and deployed number of cruises with high c^{loc}

Opened cruise line name	#Trips	Home port 1	#Cruises	Home port 2	#Cruises
10 Nights Grand Voyage	90	Tokyo	5	Hong Kong	/
11 Nights Hong Kong To Tokyo	80	Hong Kong	/	Tokyo	5
16 Nights World Cruise	74	Hong Kong	/	Tokyo	8
18 Nights World Cruise	50	Bali	5	Hong Kong	/
21 Nights Grand Voyage	88	Singapore	4	Taipei	6
22 Nights Grand Voyage	80	Tokyo	5	Bangkok	5
27 Nights Grand Voyage	54	Singapore	9	Incheon	/
29 Nights Grand Voyage	30	Singapore	5	Kyoto	/
32 Nights Grand Voyage	50	Tokyo	5	Bali	5
33 Nights Grand Voyage	54	Singapore	6	Tokyo	5
41 Nights Grand Voyage	40	Tokyo	10	Phuket	/
42 Nights Grand Voyage	20	Hong Kong	/	Singapore	5
60 Nights Grand Voyage	50	Tokyo	5	Sydney	15
63 Nights Grand Voyage	60	Tokyo	10	Sydney	10
57 Nights Grand Voyage	15	Singapore	5	/	/

Table S.4: Opened cruise lines and deployed number of cruises with high c^{fuel}

Opened cruise line name	#Trips	Home port 1	#Cruises	Home port 2	#Cruises
10 Nights Grand Voyage	90	Tokyo	5	Hong Kong	/
16 Nights World Cruise	74	Hong Kong	/	Tokyo	8
21 Nights Grand Voyage	80	Singapore	/	Taipei	10
22 Nights Grand Voyage	80	Tokyo	5	Bangkok	5
29 Nights Grand Voyage	6	Singapore	1	Kyoto	/
33 Nights Grand Voyage	70	Singapore	5	Tokyo	10
41 Nights Grand Voyage	40	Tokyo	10	Phuket	/
42 Nights Grand Voyage	20	Hong Kong	/	Singapore	5
60 Nights Grand Voyage	70	Tokyo	15	Sydney	15
63 Nights Grand Voyage	70	Tokyo	15	Sydney	15
57 Nights Grand Voyage	33	Singapore	14	/	/

T Greedy heuristic

Algorithm T.6: Greedy heuristic for CFD-ISP.

Input: The number of cruises $Z_{\max}$.
Output : Cruise deployment strategy $\boldsymbol{u}^*$ and itinerary plan.

$\boldsymbol{u} \leftarrow \mathbf{0}$, $z \leftarrow 0$
while $z < Z_{\max}$ **do**
  Set best revenue $rev^* \leftarrow -\infty$
  Set best position $(h^*, l^*) \leftarrow$ None
  foreach $l \in \mathcal{L}$ **do**
    foreach $h \in \mathcal{H}_l$ **do**
      Compute the increased revenue, denoted by rev, by solving the itinerary scheduling SP with the number of cruises at line l and port h as $u_{h,l} + 1$
      if *rev* $>$ *best-revenue* **then**
        $rev^* \leftarrow rev$
        $(h^*, l^*) \leftarrow (h, l)$
      end
    end
    $u_{h^*,l^*} \leftarrow u_{h^*,l^*} + 1$
    $z \leftarrow z + 1$
  end
end
Compute the itinerary scheduling SPs with the cruise deployment result $\boldsymbol{u}$

References

Airbus Helicopters, 2019. Shore-to-ship trials with Skyways parcel delivery drone. https://www.youtube.com/watch?v=wz01SnFHa-w.

Asiimwe, R., Anvar, A., 2012. Automation of the maritime uav command, control, navigation operations, simulated in real-time using kinect sensor: A feasibility study. International Journal of Computer and Systems Engineering 6 (12), 2606–2610.

Augerat, P., 1995. Polyhedral approach of the vehicle routing problem. Ph.D. thesis, Université Joseph Fourier, available at HAL.

Azi, N., Gendreau, M., Potvin, J.-Y., 2007. An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. European Journal of Operational Research 178 (3), 755–766.

Azi, N., Gendreau, M., Potvin, J.-Y., 2010. An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. European Journal of Operational Research 202 (3), 756–763.

Bai, X., Zhang, X., Li, K. X., Zhou, Y., Yuen, K. F., 2021. Research topics and trends in the maritime transport: A structural topic model. Transport Policy 102, 11–24.

Betti Sorbelli, F., Corò, F., Das, S. K., Palazzetti, L., Pinotti, C. M., 2022. Greedy algorithms for scheduling package delivery with multiple drones. In: Proceedings of the 23rd International Conference on Distributed Computing and Networking. pp. 31–39.

Biehn, N., 2006. A cruise ship is not a floating hotel. Journal of Revenue and Pricing Management 5, 135–142.

Birolini, S., Antunes, A. P., Cattaneo, M., Malighetti, P., Paleari, S., 2021. Integrated flight scheduling and fleet assignment with improved supply-demand interactions. Transportation Research Part B: Methodological 149, 162–180.

Boland, N., Hewitt, M., Marshall, L., Savelsbergh, M., 2017. The continuous-time service network design problem. Operations Research 65 (5), 1303–1321.

Chai, S., Yin, J., D'Ariano, A., Liu, R., Yang, L., Tang, T., 2024. A branch-and-cut algorithm for scheduling train platoons in urban rail networks. Transportation Research Part B: Methodological 181, 102891.

Charles Alcock, 2024. Shore-to-Ship drone deliveries could cut costs and carbon. Accessed April 8, 2024, https://www.ainonline.com/aviation-news/business-aviation/2022-02-16/shore-ship-drone-deliveries-could-cut-costs-and-carbon.

Chen, S., Li, Y., Zhou, W., 2019. Joint decisions for blood collection and platelet inventory control. Production and Operations Management 28 (7), 1674–1691.

Cheng, C., Adulyasak, Y., Rousseau, L.-M., 2020. Drone routing with energy function: Formulation and exact algorithm. Transportation Research Part B: Methodological 139, 364–387.

Cheng, C., Adulyasak, Y., Rousseau, L.-M., 2024. Robust drone delivery with weather information. Manufacturing & Service Operations Management, accepted.

Costa, L., Contardo, C., Desaulniers, G., 2019. Exact branch-price-and-cut algorithms for vehicle routing. Transportation Science 53 (4), 946–985.

Crainic, T. G., Hewitt, M., 2021. Service network design. Network Design with Applications to Transportation and Logistics, 347–382.

Crainic, T. G., Hewitt, M., Toulouse, M., Vu, D. M., 2016. Service network design with resource constraints. Transportation Science 50 (4), 1380–1393.

Crainic, T. G., Hewitt, M., Toulouse, M., Vu, D. M., 2018. Scheduled service network design with resource acquisition and management. EURO Journal on Transportation and Logistics 7, 277–309.

Dall'Olio, G., Kolisch, R., 2023. Formation and routing of worker teams for airport ground handling operations: A branch-and-price-and-check approach. Transportation Science 57 (5), 1231–1251.

Davis, A. R., Broad, A., Gullett, W., Reveley, J., Steele, C., Schofield, C., 2016. Anchors away? the impacts of anchor scour by ocean-going vessels and potential response options. Marine Policy 73, 1–7.

Desaulniers, G., Desrosiers, J., Solomon, M. M., 2006. Column generation. Vol. 5. New York: Springer Science & Business Media.

Desrochers, M., Desrosiers, J., Solomon, M., 1992. A new optimization algorithm for the vehicle routing problem with time windows. Operations Research 40 (2), 342–354.

Di Pillo, G., Fabiano, M., Lucidi, S., Roma, M., 2021. Cruise itineraries optimal scheduling. Optimization Letters 15 (5), 1665–1689.

Ding, D., Chou, M. C., 2015. Stowage planning for container ships: A heuristic algorithm to reduce the number of shifts. European Journal of Operational Research 246 (1), 242–249.

DJI, 2024. DJI FlyCart 30. Accessed April 8, 2024, `https://www.dji.com/hk/flycart-30/specs`.

Dorling, K., Heinrichs, J., Messier, G. G., Magierowski, S., 2017. Vehicle routing problems for drone delivery. IEEE Transactions on Systems, Man, and Cybernetics: Systems 47 (1), 70–85.

Drabas, T., Wu, C.-L., 2013. Modelling air carrier choices with a segment specific cross nested logit model. Journal of Air Transport Management 32, 8–16.

Dror, M., 1994. Note on the complexity of the shortest path models for column generation in vrptw. Operations Research 42 (5), 977–978.

Dumas, Y., Desrosiers, J., Gelinas, E., Solomon, M. M., 1995. An optimal algorithm for the traveling salesman problem with time windows. Operations Research 43 (2), 367–371.

Farahani, R. Z., Miandoabchi, E., Szeto, W. Y., Rashidi, H., 2013. A review of urban transportation network design problems. European Journal of Operational Research 229 (2), 281–302.

Feillet, D., Dejax, P., Gendreau, M., Gueguen, C., 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks: An International Journal 44 (3), 216–229.

Fischetti, M., Glover, F., Lodi, A., 2005. The feasibility pump. Mathematical Programming 104, 91–104.

Fischetti, M., Salvagnin, D., Zanette, A., 2010. A note on the selection of Benders' cuts. Mathematical Programming 124, 175–182.

Fontaine, P., Crainic, T. G., Jabali, O., Rei, W., 2021. Scheduled service network design with resource management for two-tier multimodal city logistics. European Journal of Operational Research 294 (2), 558–570.

Forbes, 2024. Cruise industry continues strong growth post-COVID. Accessed July 11, 2024, https://www.forbes.com/sites/judykoutsky/2024/02/22/cruise-industry-continues-to-grow-strong-post-covid/https://www.forbes.com/.

Fowler, T. G., Sørgård, E., 2000. Modeling ship transportation risk. Risk Analysis 20 (2), 225–244.

Fransoo, J. C., Lee, C.-Y., 2013. The critical role of ocean container transport in global supply chain performance. Production and Operations Management 22 (2), 253–268.

Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M. P. d., Reis, M., Uchoa, E., Werneck, R. F., 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical Programming 106, 491–511.

Fulkerson, D., Gross, O., 1965. Incidence matrices and interval graphs. Pacific Journal of Mathematics 15 (3), 835–855.

Gambella, C., Naoum-Sawaya, J., Ghaddar, B., 2018. The vehicle routing problem with floating targets: Formulation and solution approaches. INFORMS Journal on Computing 30 (3), 554–569.

Garey, M. R., Johnson, D. S., 1983. Computers and intractability: A guide to the theory of NP-completeness. The Journal of Symbolic Logic 48 (2), 498–500.

Garg, V., Niranjan, S., Prybutok, V., Pohlen, T., Gligor, D., 2023. Drones in last-mile delivery: A systematic review on efficiency, accessibility, and sustainability. Transportation Research Part D: Transport and Environment 123, 103831.

Geoffrion, A. M., Graves, G. W., 1974. Multicommodity distribution system design by Benders decomposition. Management Science 20 (5), 822–844.

Golari, M., Fan, N., Jin, T., 2017. Multistage stochastic optimization for production-inventory planning with intermittent renewable energy. Production and Operations Management 26 (3), 409–425.

Goldjoy Holidays, 2024. Cruises in Southeast Asia. Accessed September 11, 2024, https://goldjoy.com/.

Gui, L., Russo, A. P., 2011. Cruise ports: A strategic nexus between regions and global lines—evidence from the Mediterranean. Maritime Policy & Management 38 (2), 129–150.

Guihaire, V., Hao, J.-K., 2008. Transit network design and scheduling: A global review. Transportation Research Part A: Policy and Practice 42 (10), 1251–1273.

Ha, Q. M., Deville, Y., Pham, Q. D., Hà, M. H., 2018. On the min-cost traveling salesman problem with drone. Transportation Research Part C: Emerging Technologies 86, 597–621.

Hernandez, F., Feillet, D., Giroudeau, R., Naud, O., 2016. Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. European Journal of Operational Research 249 (2), 551–559.

Hewitt, M., Crainic, T. G., Nowak, M., Rei, W., 2019. Scheduled service network design with resource acquisition and management under uncertainty. Transportation Research Part B: Methodological 128, 324–343.

Hoffman, A. J., Kruskal, J. B., 2010. Integral boundary points of convex polyhedra. 50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art, 49–76.

Huang, N., Li, J., Zhu, W., Qin, H., 2021. The multi-trip vehicle routing problem with

time windows and unloading queue at depot. Transportation Research Part E: Logistics and Transportation Review 152, 102370.

Huang, N., Qin, H., Du, Y., Wang, L., 2024a. An exact algorithm for the multi-trip vehicle routing problem with time windows and multi-skilled manpower. European Journal of Operational Research 319 (1), 31–49.

Huang, N., Qin, H., Xu, G., Wan, F., 2024b. An enhanced exact algorithm for the multi-trip vehicle routing problem with time windows and capacitated unloading station. Computers & Operations Research 168, 106688.

IBM Corporation, 2021. IBM ILOG CPLEX Optimization Studio CPLEX User's Manual. IBM, https://www.ibm.com/docs/zh/icos/22.1.1?topic=optimizers-users-manual-cplex.

Jeong, H. Y., Song, B. D., Lee, S., 2019. Truck-drone hybrid delivery routing: Payload-energy dependency and no-fly zones. International Journal of Production Economics 214, 220–233.

Jepsen, M., Petersen, B., Spoorendonk, S., Pisinger, D., 2008. Subset-row inequalities applied to the vehicle-routing problem with time windows. Operations Research 56 (2), 497–511.

Kohl, N., Desrosiers, J., Madsen, O. B., Solomon, M. M., Soumis, F., 1999. 2-path cuts for the vehicle routing problem with time windows. Transportation Science 33 (1), 101–116.

Kumar, R. P., Devi, V. A., 2023. Air drone as an alternative to supply boats in sea ports. Marine Engineers Review (India) 2, 19–24.

Land, A., Doig, A., 1960. An automatic method of solving discrete programming problems. Econometrica 28 (3), 497–520.

Legros, B., Bouchery, Y., Fransoo, J., 2019. A time-based policy for empty container management by consignees. Production and Operations Management 28 (6), 1503–1527.

Li, Q., Üster, H., Zhang, Z.-H., 2023a. A bilevel model for robust network design and biomass pricing under farmers' risk attitudes and supply uncertainty. Transportation Science 57 (5), 1296–1320.

Li, S., Zhu, X., Shang, P., Li, T., Liu, W., 2023b. Optimizing a shared freight and passenger high-speed railway system: A multi-commodity flow formulation with Benders decomposition solution approach. Transportation Research Part B: Methodological 172, 1–31.

Liu, B., Wang, Y., Li, Z.-C., Zheng, J., 2023. An exact method for vessel emission monitoring with a ship-deployed heterogeneous fleet of drones. Transportation Research Part C: Emerging Technologies 153, 104198.

Liu, N., Truong, V.-A., Wang, X., Anderson, B. R., 2019. Integrated scheduling and capacity planning with considerations for patients' length-of-stays. Production and Operations Management 28 (7), 1735–1756.

Liu, S., Qin, S., Zhang, R., 2018. A branch-and-price algorithm for the multi-trip multi-repairman problem with time windows. Transportation Research Part E: Logistics and Transportation Review 116, 25–41.

Macedo, R., Alves, C., de Carvalho, J. V., Clautiaux, F., Hanafi, S., 2011. Solving the vehicle routing problem with time windows and multiple routes exactly using a pseudo-polynomial model. European Journal of Operational Research 214 (3), 536–545.

Macrina, G., Pugliese, L. D. P., Guerriero, F., Laporte, G., 2020. Drone-aided routing: A literature review. Transportation Research Part C: Emerging Technologies 120, 102762.

Maddah, B., Moussawi-Haidar, L., El-Taha, M., Rida, H., 2010. Dynamic cruise ship revenue management. European Journal of Operational Research 207 (1), 445–455.

Magnanti, T. L., Wong, R. T., 1981. Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. Operations Research 29 (3), 464–484.

Mahmoudi, M., Chen, J., Shi, T., Zhang, Y., Zhou, X., 2019. A cumulative service state representation for the pickup and delivery problem with transfers. Transportation Research Part B: Methodological 129, 351–380.

Marine Insight, 2024. How much fuel does a cruise ship consume? Accessed September 11, 2024, https://www.marineinsight.com/videos/video-how-much-fuel-does-a-cruise-ship-consume/.

Meng, S., Guo, X., Li, D., Liu, G., 2023. The multi-visit drone routing problem for pickup and delivery services. Transportation Research Part E: Logistics and Transportation Review 169, 102990.

Mingozzi, A., Roberti, R., Toth, P., 2013. An exact algorithm for the multitrip vehicle routing problem. INFORMS Journal on Computing 25 (2), 193–207.

Muhammad, B., Gregersen, A., 2022. Maritime drone services ecosystem-potentials and challenges. In: 2022 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom). IEEE, pp. 6–13.

Murray, C. C., Chu, A. G., 2015. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transportation Research Part C: Emerging Technologies 54, 86–109.

Murray, C. C., Raj, R., 2020. The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. Transportation Research Part C: Emerging Technologies 110, 368–398.

Neame, P. J., 2000. Nonsmooth dual methods in integer programming. Ph.D. thesis, University of Melbourne, Department of Mathematics and Statistics.

Nguyen, H. P., Hoang, A. T., Nizetic, S., Nguyen, X. P., Le, A. T., Luong, C. N., Chu, V. D., Pham, V. V., 2021. The electric propulsion system as a green solution for management strategy of co2 emission in ocean shipping: A comprehensive review. International Transactions on Electrical Energy Systems 31 (11), e12580.

Ozbaygin, G., Karasan, O. E., Savelsbergh, M., Yaman, H., 2017. A branch-and-price algorithm for the vehicle routing problem with roaming delivery locations. Transportation Research Part B: Methodological 100, 115–137.

Özer, Ö., Uncu, O., 2013. Competing on time: An integrated framework to optimize dynamic time-to-market and production decisions. Production and Operations Management 22 (3), 473–488.

Papadakos, N., 2008. Practical enhancements to the magnanti–wong method. Operations Research Letters 36 (4), 444–449.

Papathanassis, A., 2017. Cruise tourism management: State of the art. Tourism Review 72 (1), 104–119.

Papathanassis, A., Breitner, M. H., Schoen, C., 2011. Cruise Management: Information and Decision Support Systems. Springer Science & Business Media, New York.

Paradiso, R., Roberti, R., Laganá, D., Dullaert, W., 2020. An exact solution framework for multitrip vehicle-routing problems with time windows. Operations Research 68 (1), 180–198.

Pereira, F. L., 2021. Optimal control problems in drone operations for disaster search and rescue. Procedia Computer Science 186, 78–86.

Perrykkad, A., Ernst, A. T., Krishnamoorthy, M., 2022. A simultaneous Magnanti-Wong method to accelerate Benders decomposition for the metropolitan container transportation problem. Operations Research 70 (3), 1531–1559.

Poikonen, S., Wang, X., Golden, B., 2017. The vehicle routing problem with drones: Extended models and connections. Networks 70 (1), 34–43.

Pu, F., Yin, J., Wang, Y., Su, S., Yang, L., Tang, T., 2022. Rolling stock allocation and timetabling for urban rail transit network with multiple depots. Transportation Research Record 2676 (11), 422–435.

Puri, V., Nayyar, A., Raja, L., 2017. Agriculture drones: A modern breakthrough in precision agriculture. Journal of Statistics and Management Systems 20 (4), 507–518.

Rahmaniani, R., Ahmed, S., Crainic, T. G., Gendreau, M., Rei, W., 2020. The Benders dual decomposition method. Operations Research 68 (3), 878–895.

Rahmaniani, R., Crainic, T. G., Gendreau, M., Rei, W., 2017. The Benders decomposition algorithm: A literature review. European Journal of Operational Research 259 (3), 801–817.

Rahmaniani, R., Crainic, T. G., Gendreau, M., Rei, W., 2018. Accelerating the Benders decomposition method: Application to stochastic network design problems. SIAM Journal on Optimization 28 (1), 875–903.

Rejeb, A., Abdollahi, A., Rejeb, K., Treiblmaier, H., 2022. Drones in agriculture: A review and bibliometric analysis. Computers and Electronics in Agriculture 198, 107017.

Righini, G., Salani, M., 2006. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Discrete Optimization 3 (3), 255–273.

Roberti, R., Ruthmair, M., 2021. Exact methods for the traveling salesman problem with drone. Transportation Science 55 (2), 315–335.

Rodrigue, J.-P., Notteboom, T., 2013. The geography of cruises: Itineraries, not destinations. Applied Geography 38, 31–42.

Santos, T. A., Martins, P., Guedes Soares, C., 2021. Cruise shipping in the Atlantic coast of the Iberian Peninsula. Maritime Policy & Management 48 (1), 129–145.

Service, S. D., 2023. The power of ship-to-shore drone delivery in maritime supply chains. Accessed July 15, 2024, https://skyportsdroneservices.com/2023/01/the-power-of-ship-to-shore-drone-delivery-in-maritime-supply-chains/.

Shen, L., Wang, Y., Liu, K., Yang, Z., Shi, X., Yang, X., Jing, K., 2020. Synergistic path planning of multi-uavs for air pollution detection of ships in ports. Transportation Research Part E: Logistics and Transportation Review 144, 102128.

Shen, S., You, M., Ma, Y., 2017. Single-commodity stochastic network design under demand and topological uncertainties with insufficient data. Naval Research Logistics 64 (2), 154–173.

Sherali, H. D., Lunday, B. J., 2013. On generating maximal nondominated Benders cuts. Annals of Operations Research 210, 57–72.

Sherali, H. D., Soyster, A. L., 1983. Preemptive and nonpreemptive multi-objective programming: Relationship and counterexamples. Journal of Optimization Theory and Applications 39, 173–186.

Shi, T., Zhou, X., 2015. A mixed integer programming model for optimizing multi-level operations process in railroad yards. Transportation Research Part B: Methodological 80, 19–39.

Ship & Bunker, 2024. Shipping news and bunker price indications: Bunker prices. Accessed September 11, 2024, https://shipandbunker.com/.

Soriani, S., Bertazzon, S., Cesare, F. D., Rech, G., 2009. Cruising in the Mediterranean: Structural aspects and evolutionary trends. Maritime Policy & Management 36 (3), 235–251.

Squires, A. M., 1986. The tender ship: governmental management of technological change. New York: Springer Science & Business Media.

Stieber, A., Fügenschuh, A., Epp, M., Knapp, M., Rothe, H., 2015. The multiple traveling salesmen problem with moving targets. Optimization Letters 9 (8), 1569–1583.

Sudermann-Merx, N., Rebennack, S., Timpe, C., 2021. Crossing minimal edge-constrained layout planning using Benders decomposition. Production and Operations Management 30 (10), 3429–3447.

Sulligoi, G., Bosich, D., Pelaschiar, R., Lipardi, G., Tosato, F., 2015. Shore-to-ship power. Proceedings of the IEEE 103 (12), 2381–2400.

Thompson, M., Davies, M., 2021. Maritime uses of drones. In: Drone Law and Policy. Routledge, pp. 79–113.

Tilk, C., Bianchessi, N., Drexl, M., Irnich, S., Meisel, F., 2018. Branch-and-price-and-cut for the active-introduction topassive vehicle-routing problem. Transportation Science 52 (2), 300–319.

Toth, P., Vigo, D., 2014. Vehicle Routing: Problems, Methods, and Applications. SIAM, Philadelphia, PA.

Vanderbeck, F., Wolsey, L. A., 1996. An exact algorithm for ip column generation. Operations Research Letters 19 (4), 151–159.

Véronneau, S., Roy, J., 2009. Global service supply chains: An empirical study of current practices and challenges of a cruise line corporation. Tourism Management 30 (1), 128–139.

Wang, K., Wang, S., Zhen, L., Qu, X., 2016. Cruise shipping review: Operations planning and research opportunities. Maritime Business Review 1 (2), 133–148.

Wang, S., Wang, K., Zhen, L., Qu, X., 2017. Cruise itinerary schedule design. IISE Transactions 49 (6), 622–641.

Wang, S., Zhen, L., Zhuge, D., 2018a. Dynamic programming algorithms for selection of waste disposal ports in cruise shipping. Transportation Research Part B: Methodological 108, 235–248.

Wang, Y., D'Ariano, A., Yin, J., Meng, L., Tang, T., Ning, B., 2018b. Passenger demand oriented train scheduling and rolling stock circulation planning for an urban rail transit line. Transportation Research Part B: Methodological 118, 193–227.

Wang, Z., Sheu, J.-B., 2019. Vehicle routing problem with drones. Transportation Research Part B: Methodological 122, 350–364.

Wu, Y., Wang, S., Zhen, L., Laporte, G., Tan, Z., Wang, K., 2023. How to operate ship fleets under uncertainty. Production and Operations Management 32 (10), 3043–3061.

Xia, J., Wang, K., Wang, S., 2019. Drone scheduling to monitor vessels in emission control areas. Transportation Research Part B: Methodological 119, 174–196.

Yan, R., Wang, S., Zhen, L., Laporte, G., 2021. Emerging approaches applied to maritime transport research: Past and future. Communications in Transportation Research 1, 100011.

Yang, Y., 2023. An exact price-cut-and-enumerate method for the capacitated multitrip vehicle routing problem with time windows. Transportation Science 57 (1), 230–251.

Yang, Y., 2025. Deluxing: Deep lagrangian underestimate fixing for column-generation-based exact methods. Operations Research 73 (3), 1184–1207.

Yin, J., Pu, F., Yang, L., D'Ariano, A., Wang, Z., 2023a. Integrated optimization of rolling stock allocation and train timetables for urban rail transit networks: A Benders decomposition approach. Transportation Research Part B: Methodological 176, 102815.

Yin, J., Wang, M., D'Ariano, A., Zhang, J., Yang, L., 2023b. Synchronization of train timetables in an urban rail network: A bi-objective optimization approach. Transportation Research Part E: Logistics and Transportation Review 174, 103142.

Zhang, W., Jacquillat, A., Wang, K., Wang, S., 2023. Routing optimization with vehicle–customer coordination. Management Science 69 (11), 6876–6897.

Zhen, L., He, X., Wang, S., Wu, J., Liu, K., 2023. Vehicle routing for customized on-demand bus services. IISE Transactions 55 (12), 1277–1294.

Zhen, L., Li, M., Hu, Z., Lv, W., Zhao, X., 2018. The effects of emission control area regulations on cruise shipping. Transportation Research Part D: Transport and Environment 62, 47–63.