

Article

Ballast Water Inspection Planning with Drones Considering Partial Charging

Xin Fan, Hongxing Zheng * and Zhaoyang Wang

College of Transportation Engineering, Dalian Maritime University, Dalian 116026, China;
fan_fancy@dlmu.edu.cn (X.F.); wangzhaoyang@dlmu.edu.cn (Z.W.)

* Correspondence: zhredstar@dlmu.edu.cn

Highlights

What are the main findings?

- The proposed partial charging strategy achieves a significant 25% reduction in total drone energy consumption compared to the threshold charging strategy in large-scale inspection scenarios.
- A hybrid DP-ALNS algorithm produces solutions within 1.06% of the optimal value found by GUROBI for small instances and outperforms the baseline ALNS by an average of 2.08% for large instances.

What are the implications of the main findings?

- By overcoming drone energy limitations through adaptive partial charging, this work provides a practical solution for making it feasible to deploy drones for multi-vessel ballast water inspection.
- The sensitivity analysis on fleet size and drone flight speed offers a decision-support tool for maritime authorities, balancing regulatory compliance with operational costs.

Abstract

In maritime safety regulation, conventional ballast water inspection faces critical time-window violations due to manual sampling delays across dispersed inspection points. While drone-based inspection significantly enhances multi-point accessibility and timeliness, its operational viability is severely constrained by limited battery endurance during multi-vessel missions. To address this challenge, we propose a novel partial charging strategy enabling adaptive energy replenishment. This strategy is formalized in a model incorporating dual energy consumption (propulsion and hovering) and spatiotemporal constraints for sampling operations. An adaptive metaheuristic algorithm combining dynamic programming with large neighborhood search is developed for efficient resolution. Computational experiments demonstrate that partial charging strategy reduces drone energy consumption by 25% compared to the 70%-threshold charging strategy. This study provides maritime authorities with a quantifiable configuration decision tool to optimize drone fleets and flight parameters, balancing regulatory compliance with operational costs.

Keywords: ballast water inspection; drone routing; partial charging strategy; adaptive metaheuristic algorithm; dynamic programming; multi-trip routing



Academic Editor: Diego
González-Aguilera

Received: 2 July 2026

Revised: 29 July 2026

Accepted: 5 August 2026

Published: 10 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The vigorous expansion of global trade has driven rapid growth in the maritime transport industry, marked by continuous increases in vessel numbers and sailing frequency. This growth has led to a significant rise in the use of ballast water. However, the trans-regional discharge of ballast water poses a severe threat to global marine ecosystems. It facilitates the transport and spreads of harmful aquatic organisms and pathogens, leading to biological invasions, disruption of local ecological balances, and marine pollution [1,2]. To address this global challenge, the International Maritime Organization (IMO) established the landmark International Convention for the Control and Management of Ships' Ballast Water and Sediments in 2004 [3]. This convention requires strict standards and procedures to control ballast water treatment and discharge globally. A core requirement for demonstrating compliance is effective ballast water inspection.

Currently, the primary method for ballast water inspection in ports still relies on manually operated patrol boats performing on-site sampling. While useful initially under the convention, this approach has clear limitations in modern high-traffic ports. Key issues include: the limited maneuverability and slow speed of patrol boats, causing long task times; and insufficient patrol boat resources to inspect all the vessels needing checks quickly, especially during peak periods with many vessels. This results in sampling delays, testing backlogs, and potential impacts on port efficiency. Therefore, this traditional labor-intensive approach faces significant problems with efficiency, timeliness, and handling large-scale demands. It struggles to meet growing needs for regulatory compliance and environmental protection. Improving the efficiency and speed of the ballast water sampling and inspection process is now a critical challenge in port management.

Against this backdrop, drone technology has emerged as a promising solution to transform port ballast water inspection processes, leveraging its excellent 3D maneuverability, rapid response capability, and relatively low operating costs [4,5]. Compared to bulky and slow traditional patrol boats, drones can navigate flexibly within congested port environments, quickly reaching designated sampling points on target vessels to perform ballast water sampling. This capability offers the potential to significantly reduce sampling cycle times, enhance overall operational efficiency, and decrease reliance on manual labor. However, successfully integrating drones and scaling up their application for ballast water inspection is challenging. Practical deployment faces several key technical and operational hurdles. The primary constraint is limited flight endurance, which severely restricts single-flight mission duration and range. To enable continuous, efficient, all-weather operations, drones urgently require flexible and efficient charging solutions. Finally, coordinating and intelligently planning tasks for multiple drones within complex port environments to optimize resource use while meeting charging needs presents a highly complex optimization problem [6]. Therefore, to overcome these bottlenecks and unlock the value of drones for efficient ballast water inspection, this study focuses on the ballast water inspection planning with drones considering partial charging (BWI-D) problem. The contributions of this study are three-fold:

1. This work presents the first comprehensive study dedicated to scheduling and planning drone operations for efficient ballast water inspection in port environments. It formally defines the BWI-D problem, establishing a novel research direction at the intersection of maritime environmental monitoring and drone logistics optimization.
2. We develop a mixed-integer programming model for the BWI-D problem. A key innovation of this model is its explicit integration of partial charging operations.
3. To solve the proposed model efficiently, we design an adaptive metaheuristic (AM) algorithm combining dynamic programming (DP) with adaptive large neighborhood search (ALNS). This tailored approach demonstrates superior performance in generat-

ing high-quality, practical solutions efficiently compared to both exact methods and baseline heuristics.

The remainder of this paper is organized as follows. Section 2 reviews related work relevant to the BWI-D problem. Section 3 formally defines the problem and develops a rigorous MIP model. Section 4 presents the tailored AM algorithm designed to solve the problem efficiently. Section 5 details the numerical experiments and analyzes the results. Finally, Section 6 concludes the paper and discusses potential future research directions.

2. Literature Review

Ballast water discharge is a major contributor to invasive species introduction and marine pollution. It poses a serious threat to port environments and nearby marine ecosystems. Invasive species transported in ballast water can disrupt biological competition, damage native ecological structures, and threaten marine biodiversity and fisheries sustainability [7,8]. Additionally, ballast water often carries harmful pathogens and chemical pollutants, leading to water contamination and degraded environmental quality in port areas [9,10]. These significant ecological and environmental risks make effective ballast water management an urgent global priority. The full implementation of the IMO ballast water management convention has set higher standards and stricter requirements for global port regulation [11,12]. However, traditional inspection methods relying on sampling boats for manual onboard sampling face critical challenges. Studies consistently report high labor and time costs, operational inefficiency, and inadequate coverage [13,14]. This is especially problematic during peak vessel traffic periods. The conventional approach struggles to meet the convention's regulatory requirements or support efficient port operations due to insufficient timeliness and comprehensiveness. Therefore, developing more efficient inspection technologies with expanded coverage is essential. These new approaches must overcome the limitations of traditional methods and improve overall regulatory effectiveness. This pursuit is now a key research focus and critical development direction in ballast water management.

Drones show increasing potential in environmental inspection due to their high mobility, real-time capability, and relatively low operational costs. Research demonstrates their significant advantages in tasks such as air quality assessment [15], mapping and management [16], and agricultural monitoring [17]. Compared to traditional manual methods, drones respond rapidly and reach target areas efficiently. This makes them particularly suitable for time-sensitive inspection scenarios. In marine environmental inspection, drone applications have successfully expanded to include marine debris tracking [18], oil spill inspection [19], and water sampling [20]. Notably, Horricks et al. found that drone-based water sampling can effectively supplement or even replace traditional manned sampling boats [20]. This provides a crucial theoretical and practical foundation for applying drones to ballast water sampling at sea, confirming their feasibility for standardized maritime sampling tasks. However, despite progress in nearshore water sampling, research on drone applications specifically for port ballast water inspection remains limited. This gap becomes most apparent in contexts involving the need to coordinate inspection across multiple vessels, and the resulting challenges in efficient scheduling and energy management.

The drone routing problem shares similarities with the classic vehicle routing problem. However, its core challenge lies in flight endurance constraints, which make charging strategies critical for mission planning. With recent advancements in new energy technologies, significant research has focused on incorporating power constraints into drone scheduling, primarily addressing three aspects: energy consumption modeling, charging infrastructure layout, and charging strategies. Regarding energy consumption modeling, drone energy consumption is inherently nonlinear and influenced by multiple factors such

as flight speed, payload, and wind speed [21]. To enhance model solvability, Alyassi et al. demonstrated through empirical analysis that energy consumption can be reasonably simplified to a linear relationship with flight duration under specific operating conditions [22]. In addition, Wang et al. incorporated the energy consumption of hovering states, laying the groundwork for scheduling problems involving complex time windows such as waiting for observation [23]. In terms of charging infrastructure layout, some studies optimize the spatial placement of charging or battery swapping stations [24]. This aims to reduce the distance drones need to travel for charging, thereby extending effective mission time and lowering operational costs. Concerning charging strategies, mainstream approaches can be categorized as full charging and partial charging strategies, with the contrast visually demonstrated in Figure 1. Under the full charging strategy, drones must either swap for a fully charged battery or charge to full capacity before leaving a station. The charging time is typically modeled as a fixed constant [25]. However, this strategy implicitly assumes that charging stations possess unlimited battery reserves or instantaneous charging capability, which is often impractical in real-world port settings with limited resources. To overcome this limitation, Wang et al. allowed drones to flexibly determine charging duration based on subsequent mission requirements [23]. Notably, despite adopting both a linear energy model and partial charging, their solution employed a two-stage “route first, charge later” approach—first fixing flight paths, then optimizing charging durations. Crucially, their computation still mandated full battery levels before drone departure, creating a fundamental contradiction with the partial charging concept.

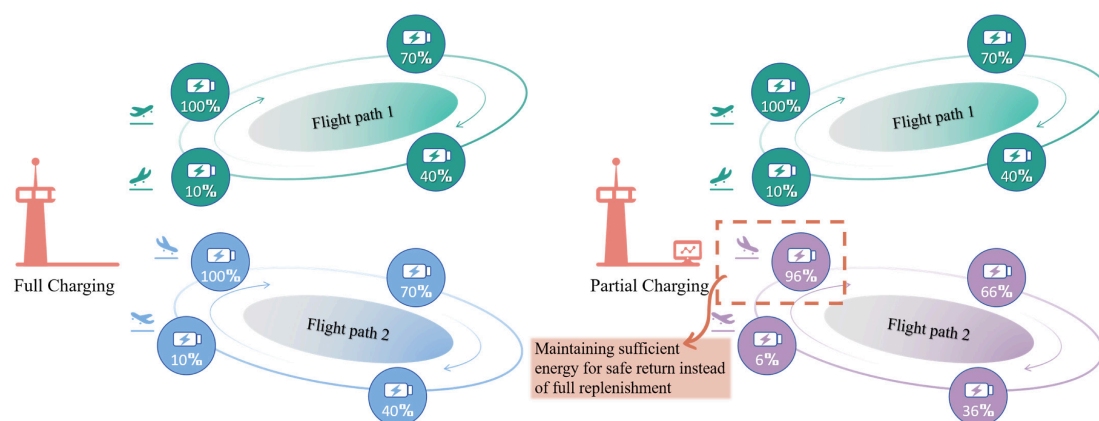


Figure 1. Full charging vs. partial charging.

Beyond mathematical programming and heuristics, recent studies have also explored learning-based approaches for routing with energy constraints. For instance, Q-learning and hierarchical reinforcement learning have been applied to energy-aware route planning for electric vehicles [26]. In the broader aviation domain, deep learning architectures such as multi-graph convolution networks have been developed for spatiotemporal prediction tasks like flight delay forecasting [27]. However, these learning-based methods have yet to be adapted to the joint charging–routing optimization under dual energy constraints. To further clarify the novelty, it is useful to position the BWI-D problem against several well-known routing problems from the literature. Classic Drone Routing Problems (DRPs) typically assume single-trip missions with full battery capacity or no recharging at all, and they rarely distinguish between propulsion and hovering energy. Electric Vehicle Routing Problems (EVRPs) with charging allow partial recharging in some variants, yet they deal with ground vehicles that do not consume energy while stationary, and none of them impose a hard deadline for returning collected samples to a depot. Drone Routing with Charging (DRC) usually models charging as a fixed-duration full charge or a post-routing

decision, and the energy model remains a single flight-based rate. These comparisons make it evident that existing models do not simultaneously combine partial charging strategy, a dual energy model, and a coupling between sampling time windows and a strict sample delivery deadline.

The BWI-D problem therefore integrates three critical advances: (i) charging duration is optimized concurrently with route planning as a core decision variable, rather than being treated as a fixed parameter or a post-routing adjustment; (ii) both propulsion energy consumption and hovering energy consumption are incorporated; and (iii) unique time constraints inherent to ballast water inspection are rigorously embedded. Consequently, deeply coupling these innovations represents an unresolved core challenge in the field.

3. Problem Statement and Formulation

This section defines the BWI-D problem and develops an arc-based integer programming model to address it.

3.1. Problem Statement

During port entry procedures, vessels engage in ballast water discharge or exchange operations. To enhance regulatory efficiency, maritime authorities require drones to replace traditional sampling vessels for collecting and transporting ballast water samples. This study addresses a drone scheduling problem where a base must deploy a fleet of $|D|$ drones over a planning horizon $|T|$ to perform comprehensive inspections for $|V|$ incoming vessels. Each vessel requires n independent sampling operations to ensure full coverage, resulting in $|P| = n \cdot |V|$ inspection points. Each point $i \in P$ has a predetermined sampling time window t_i^{ins} derived from vessel arrival schedules. These inspection points are fixed geographic coordinates on the projected vessel routes, and drones navigate to these stationary locations. Critically, collected samples must be delivered to the base laboratory within Δ^{ins} time units to maintain validity. Drones operate in flight paths p , defined as cycles departing from and returning to the base. Each path p commences with an initial energy level δ_p , which governs its endurance. During pre-flight waits of duration λ_p at the base, drones charge adaptively to gain $e^{cha} \cdot \lambda_p$ energy (e^c : charging rate), subject to the maximum battery capacity Δ^{max} ($\delta_p \leq \Delta^{max}$). Early arrivals at inspection points incur hovering energy consumption while awaiting t_i^{ins} . The core optimization problem is thus: Design scheduling strategies for $|D|$ drones to service all $|P|$ inspection points while respecting both sampling time windows and sample delivery deadlines, leveraging partial charging strategy and path planning to minimize total energy consumption.

Figure 2 illustrates a BWI-D solution through dual-layer graph representation. The outer graph abstracts drone scheduling using flight paths as nodes, while the inner graph details operational trajectories with inspection points as nodes. In this configuration, drone 1 executes two sequential flight paths: Path 1 originates from the base through inspection sequence $4 \rightarrow 3 \rightarrow 10$ before returning for intermediate charging, followed by path 3 conducting inspections $1 \rightarrow 8 \rightarrow 11 \rightarrow 12$. Concurrently, drone 2 completes a single flight path visiting inspections $7 \rightarrow 2 \rightarrow 9 \rightarrow 5 \rightarrow 6$.

3.2. Assumptions and Notations

To ensure model tractability for the ballast water inspection drone scheduling problem, the following operational assumptions are adopted:

1. Prevailing maritime conditions permit all drones to maintain a constant safe cruising speed between locations, with uniform velocity across the fleet and no airspace congestion. Consequently, travel times between any two points are deterministic.

2. Water sampling operations at inspection points and sample deposition at base stations incur negligible time.
3. The mass of the collected ballast water sample is small relative to the drone's takeoff weight, and its effect on energy consumption is accordingly neglected.

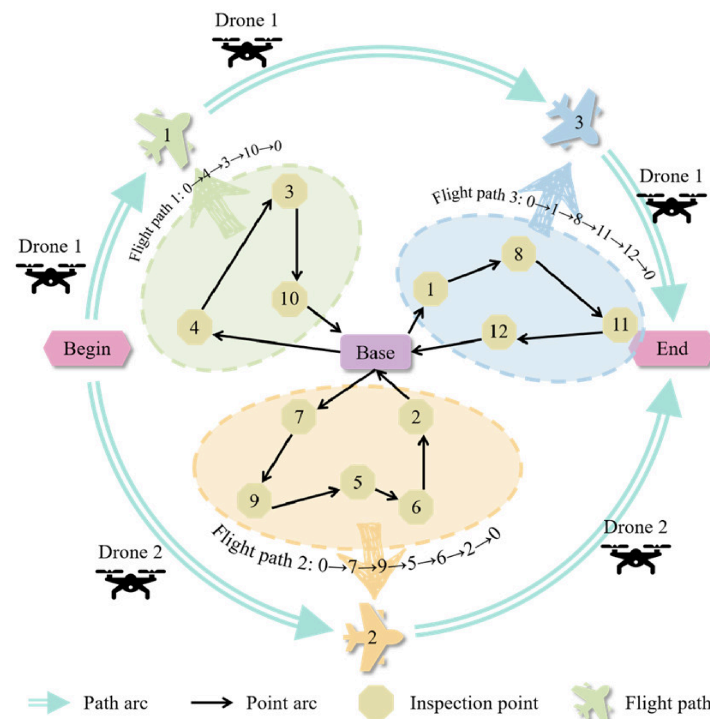


Figure 2. BWI-D solution representation.

All mathematical parameters and decision variables are formally defined in Table 1.

Table 1. Notations and descriptions for model formulation.

Notation	Description
Inputs	
D	The set of drones, $u \in 1, 2, \dots, D $.
F	The set of flight paths, $p, q, g \in 1, 2, \dots, F $.
P	The set of all inspection points, $i, j, h \in 1, 2, \dots, P $.
t_i^{ins}	The specified inspection time for inspection point i (min).
Δ^{ins}	The duration within which each sample must be returned to the base (min).
Δ^{max}	The max battery capacity of a drone ($\Delta^{max} = 100$ unit).
e^{cha}	The drone's charging rate at the base (unit/min).
e^{pro}	The drone's propulsion energy consumption rate (unit/min).
e^{hov}	The drone's hovering energy consumption rate (unit/min).
$TT(i, j)$	The travel time for a drone from inspection point i to point j (min).
M	A large positive number.
Decision variables	
$\alpha_{i,j,p}$	1 if flight path p includes point arc (i, j) ; 0 otherwise.
$\kappa_{p,q,u}$	1 if drone u includes path arc (p, q) ; 0 otherwise.
τ_i	The time when the drone arrives at inspection point i (min).
μ_p	The start time of flight path p (min).
η_p	The end time of flight path p (min).
λ_p	The waiting time at the base before the departure of flight path p (min).
δ_p	The initial energy level of flight path p (unit).

3.3. Arc-Based Model

Leveraging the notation defined in Table 1, the objective function of the BWI-D model is formulated as follows:

$$\min \sum_{p \in F} \left[(\eta_p - \mu_p) - \sum_{i \in P \cup \{0\}} \sum_{j \in P \cup \{0\}} \alpha_{i,j,p} TT(i,j) \right] \cdot e^{hov} + \sum_{p \in F} \sum_{i \in P \cup \{0\}} \sum_{j \in P \cup \{0\}} \alpha_{i,j,p} TT(i,j) \cdot e^{pro} \quad (1)$$

Objective function (1) minimizes the total energy consumption across all drone flight paths, where the first term represents hovering energy and the second term denotes propulsion energy. Here, $\eta_p - \mu_p$ represents the total flight time for path p , and $\sum_{i \in P \cup \{0\}} \sum_{j \in P \cup \{0\}} \alpha_{i,j,p} TT(i,j)$ quantifies its real propulsion duration. Their difference gives the total hovering duration in this flight path. Then, the model constraints are grouped into the following three categories.

1. Constraints between flight paths and inspection points

$$\sum_{p \in F} \sum_{i \in P \cup \{0\}} \alpha_{i,j,p} = 1 \quad \forall j \in P \quad (2)$$

$$\sum_{j \in P} \alpha_{0,j,p} = 1 \quad \forall p \in F \quad (3)$$

$$\sum_{i \in P} \alpha_{i,0,p} = 1 \quad \forall p \in F \quad (4)$$

$$\sum_{i \in P \cup \{0\}} \alpha_{i,h,p} - \sum_{j \in P \cup \{0\}} \alpha_{h,j,p} = 0 \quad \forall h \in P, p \in F \quad (5)$$

Constraints (2) to (5) govern arcs between inspection points. Specifically, constraint (2) states that for every inspection point j , exactly one arc (i, j) from some flight path p enters it. Constraint (3) ensures each flight path p has exactly one departure arc $(0, j)$ from the base. Constraint (4) requires each flight path p to have exactly one return arc $(i, 0)$ to the base. Finally, constraint (5) enforces flow conservation so that every inspection point has exactly one incoming arc and one outgoing arc.

$$\tau_j \geq t_i^{ins} + TT(i, j) - M(1 - \sum_{p \in F} \alpha_{i,j,p}) \quad \forall i, j \in P \quad (6)$$

$$\tau_i \leq t_i^{ins} \quad \forall i \in P \quad (7)$$

$$\mu_p \leq t_j^{ins} - TT(0, j) + M(1 - \alpha_{0,j,p}) \quad \forall p \in F, j \in P \quad (8)$$

$$\eta_p \geq t_i^{ins} + TT(i, 0) - M(1 - \alpha_{i,0,p}) \quad \forall p \in F, i \in P \quad (9)$$

$$\eta_p \leq t_i^{ins} + \Delta^{ins} + M(1 - \sum_{j \in P \cup \{0\}} \alpha_{i,j,p}) \quad \forall i \in P, p \in F \quad (10)$$

Constraints (6) to (9) enforce temporal constraints related to the inspection points themselves. Constraint (6) requires that a drone must complete the inspection at point i before traversing arc (i, j) to the next point j . Constraint (7) ensures the drone arrives at point i before its scheduled inspection start time t_i^{ins} . Constraints (8) and (9) calculate the departure time μ_p from the base and return time η_p to the base for each flight path p . Finally, constraint (10) guarantees that ballast water samples from every point i are transported to the base within the specified time limit Δ^{ins} after sampling.

2. Constraints between drones with flight paths

$$\sum_{u \in D} \sum_{p \in F \cup 0} \kappa_{p,q,u} = 1 \quad \forall q \in F \quad (11)$$

$$\sum_{q \in F} \kappa_{0,q,u} \leq 1 \quad \forall u \in D \quad (12)$$

$$\sum_{p \in F} \kappa_{p,0,u} = 1 \quad \forall u \in D \quad (13)$$

$$\sum_{p \in F+0} \kappa_{g,h,u} - \sum_{q \in F+0} \kappa_{g,q,u} = 0 \quad \forall g \in F, u \in D \quad (14)$$

Constraints (11) to (14) define arc connectivity rules between flight paths. Constraint (11) enforces that every flight path q has exactly one incoming path arc (p, q) from a preceding path. Constraint (12) specifies that each drone u has at most one initial path arc $(0, q)$ leading into its first flight path. Constraint (13) requires that each drone u has exactly one terminal path arc $(p, 0)$ exiting its final flight path. Finally, constraint (14) ensures continuity by mandating that upon completing a flight path p , drone u must either immediately commence the next connected flight path or conclude its operations.

$$\mu_q - \eta_p - \lambda_q + M(1 - \sum_{u \in D} \kappa_{p,q,u}) \geq 0 \quad \forall p, q \in F \quad (15)$$

$$\delta_q \leq \delta_p - \left[(\eta_p - \mu_p) - \sum_{i \in P \cup \{0\}} \sum_{j \in P \cup \{0\}} \alpha_{i,j,p} TT(i, j) \right] \cdot e^{hov} - \sum_{i \in P \cup \{0\}} \sum_{j \in P \cup \{0\}} \alpha_{i,j,p} TT(i, j) \cdot e^{pro} + e^{cha} \cdot \lambda_q + M(1 - \sum_{u \in D} \kappa_{p,q,u}) \quad \forall p, q \in F \quad (16)$$

$$\delta_q \leq \Delta^{max} + M \sum_{u \in D} \kappa_{0,q,u} \quad \forall q \in F \quad (17)$$

$$\delta_q \geq \Delta^{max} \sum_{u \in D} \kappa_{0,q,u} \quad \forall q \in F \quad (18)$$

$$\delta_q \leq \Delta^{max} + \frac{1 - \sum_{u \in D} \kappa_{0,q,u}}{M} \quad \forall q \in F \quad (19)$$

$$\delta_p \geq \left[(\eta_p - \mu_p) - \sum_{i \in P \cup \{0\}} \sum_{j \in P \cup \{0\}} \alpha_{i,j,p} TT(i, j) \right] \cdot e^{hov} + \sum_{i \in P \cup \{0\}} \sum_{j \in P \cup \{0\}} \alpha_{i,j,p} TT(i, j) \cdot e^{pro} \quad \forall p \in F \quad (20)$$

Constraints (15) to (20) govern drone battery management protocols. Constraint (15) mandates pre-flight holding period λ_q at base prior to flight path q . Constraints (16) and (17) compute the pre-departure battery level for flight path q . Constraints (18) and (19) enforce full battery initialization Δ^{max} for all drones' first flights. Finally, constraint (20) guarantees that the combined propulsion and hovering energy consumption never exceeds instantaneous battery level.

3. Domains for the decision variables

$$\alpha_{i,j,p} = \{0, 1\} \quad \forall i, j \in P, p \in F \quad (21)$$

$$\kappa_{p,q,u} = \{0, 1\} \quad \forall p, q \in F, u \in D \quad (22)$$

$$\tau_i \geq 0 \quad \forall i \in P \quad (23)$$

$$\mu_p, \eta_p, \lambda_p, \delta_p \geq 0 \quad \forall p \in F \quad (24)$$

4. Solution Method

The drone routing problem is proven NP-Hard. Our BWI-D problem extends this complexity by incorporating three additional features: partial recharging, multi-trip operations, and complex spatiotemporal constraints. Consequently, BWI-D inherently remains NP-Hard. Given its stringent real-time requirements, we propose an AM algorithm to balance computational efficiency and solution quality. The framework operates in two phases. Firstly, Section 4.1 develops a bounded-width dynamic programming (BWDP) approach with backward recursion to rapidly generate high-quality initial solutions. Secondly, Section 4.2 integrates the mathematical programming model within an ALNS framework. Leveraging two decomposition operators, this framework efficiently generates tractable subproblems solvable to optimality at each iteration. This enables systematic refinement of the initial solution, yielding high-quality approximate optimal results within practical computation time.

4.1. Initial Solution: Bounded Width Dynamic Programming Approach

Firstly, to address the dense temporal constraints inherent in the BWI-D problem, we propose a BWDP algorithm based on backward recursion. BWDP treats each time step as a stage, with the minimum cumulative energy consumption from the schedule start to the current stage as a key state variable. The problem is decomposed via state transition equations, and subproblems are solved using the recursive relationship of the value function. Crucially, to curb state-space explosion, the algorithm strictly bounds the width of state expansion during subproblem solving: within each subproblem, any drone is permitted at most one takeoff operation and one landing operation. This approach efficiently generates high-quality initial solutions.

4.1.1. Stage Definition

Given a scheduling horizon $|T| = n$ for BWI-D, the algorithm treats each time step as a stage, resulting in n stages. The state variable $dp[t]$ denotes the minimum cumulative energy consumption from the scheduling start to time t . A composite state structure $arcdp[t]$ fully characterizes the system configuration at stage t . Taking t^{beg} as an example:

$$arcdp[t^{beg}] = \{D^{bas}, D^{fli}, P^{ins}, P_u^{com}, \Delta_u^{saf}, l_u\} \quad (25)$$

where D^{bas} and D^{fli} represent sets of drones waiting at base and in flight, respectively, while P^{ins} denotes the set of inspection points already completed by u in its current flight path. For each drone u : P_u^{com} is the set of inspection points already completed by u in its current flight path, l_u indicates the last visited location, and Δ_u^{bat} is the current remaining battery. Specifically, for drones waiting at base ($u \in D^{bas}$), Δ_u^{bat} is determined by waiting duration with $P_u^{com} = \emptyset$ and $l_u = 0$; for in-flight drones ($u \in D^{fli}$), Δ_u^{bat} depends on flight duration, with P_u^{com} and l_u dynamically updated.

4.1.2. State Transition Equation

Directly solving for $dp[t]$ using the model defined in Section 3 becomes computationally intractable as t increases. To address this, we decompose the problem into two temporally adjacent subproblems: $dp[t^{beg}]$ capturing the minimum cumulative energy con-

sumption from initialization to t^{beg} , and $f(t^{beg}, t)$ representing the optimal consumption from t^{beg} to current time t . The state transition is then established by enumerating all viable segmentation points:

$$dp[t] = \min_{t^{beg} \in (0, t)} dp[t^{beg}] + f(t^{beg}, t) \quad (26)$$

4.1.3. Value Function

To narrow the feasible region of t^{beg} , we impose that each drone may launch or land at the base at most once during the interval (t^{beg}, t) . Any configuration violating this is infeasible. Because of this, the value function differs from the arc-based model in Section 3. The model in Section 3 uses a two-layer arc structure: the outer layer connects flight paths p, q ; and the inner layer sequences inspection points i, j . The decision variable $\kappa_{p,q,u}$ assigns drone u to path (p, q) and selects a point arc at the same time. The value function here simplifies this to a single layer. The decision variable is $\chi_{i,j,u} = \alpha_{i,j,p} \sum_{q \in F} \kappa_{p,q,u}$. Since each drone is allowed at most one launching and one landing, we no longer need to track flight paths and can reduce the variable to choices over inspection point arcs only. Building upon the notations defined in Tables 1 and 2, the value function is:

$$f(t^{beg}, t) = \min_{u \in D^{bas}} \left(\varepsilon_u - \beta_u - \sum_{i \in P^{ins} \cup \{0\}} \sum_{j \in P^{ins} \cup \{0\}} TT(i, j) \chi_{i,j,u} \right) \cdot \omega_u e^{pro} + \sum_{u \in D^{fli}} \sum_{i \in P^{ins} \cup \{l_u\}} \sum_{j \in P^{ins} \cup \{0\}} TT(i, j) \chi_{i,j,u} \cdot e^{pro} + \sum_{u \in D^{fli}} \left(\varepsilon_u - t_{l_u}^{ins} - \sum_{i \in P^{ins} \cup \{l_u\}} \sum_{j \in P^{ins} \cup \{0\}} TT(i, j) \chi_{i,j,u} \right) \cdot e^{hov} \quad (27)$$

Table 2. Notations and descriptions for value function.

Notation	Description
Inputs	
D^{bas}, D^{fli}	The set of drones at the base and in flight, respectively, $D^{bas} \cup D^{fli} = D$.
P^{ins}	The set of inspection points that have not been inspected, $P^{ins} \subset P$.
P_u^{com}	The set of inspection points that have been inspected by drone u , $P_u^{com} \subset P$.
Δ_u^{bat}	The current remaining battery of drone u .
l_u	The last visited inspection point of drone $u \in D^{fli}$.
t^{beg}	The beginning time of the scheduling period.
Decision variables	
$\chi_{i,j,u}$	1 if drone u travels arc (i, j) from node i to node j ; 0 otherwise.
ω_u	1 if drone u has an inspection task during the scheduling period; 0 otherwise.
β_u	The time at which drone $u \in D^{bas}$ takes off from the base.
ε_u	The time at which drone u returns to the base.

The objective function (27) minimizes the total energy consumption for all drones to complete scheduled missions within the time horizon (t^{beg}, t) . The first two terms represent the energy consumption of drones on standby at the base, while the latter two terms account for drones in flight. Then, the constraints are grouped into the following three categories.

1. Constraints about arcs

$$\sum_{u \in D} \sum_{i \in P \cup 0} \chi_{i,j,u} = 1 \quad \forall j \in P^{ins} \quad (28)$$

$$\sum_{j \in P^{ins}} \chi_{0,j,u} = \omega_u \quad \forall u \in D^{bas} \quad (29)$$

$$\sum_{j \in P^{ins} \cup 0} \chi_{l_u,j,u} = 1 \quad \forall u \in D^{fli} \quad (30)$$

$$\sum_{i \in P^{ins} \cup \{l_u\}} \chi_{i,0,u} = \omega_u \quad \forall u \in D \quad (31)$$

$$\chi_{l_u,0,u} = 1 - \omega_u \quad \forall u \in D^{fli} \quad (32)$$

$$\sum_{i \in P \cup 0} \chi_{i,h,u} - \sum_{j \in P \cup 0} \chi_{h,j,u} = 0 \quad \forall h \in P^{ins}, u \in D \quad (33)$$

Constraints (28) to (33) govern drone routing through arc-based constraints. Constraint (28) mandates exactly one incoming arc to every inspection point j . For origin constraints, constraint (29) specifies that drones at base depart via exactly one arc $(0, j)$ if assigned a mission ($\omega_u = 1$), otherwise no departure occurs, while constraint (30) ensures all in-flight drones (regardless of ω_u) originate exactly one arc (l_u, j) from their current location l_u . For destination constraints, constraint (31) requires drones with missions ($\omega_u = 1$) to have exactly one return arc $(i, 0)$, whereas constraint (32) directs mission-free in-flight drones ($\omega_u = 0$) to return directly via arc $(l_u, 0)$. Finally, constraint (33) enforces flow continuity by necessitating departure from every visited inspection point.

2. Constraints about time and battery

$$\tau_j - t_i^{ins} + M(1 - \sum_{u \in D} \chi_{i,j,u}) \geq TT(i, j) \quad \forall i, j \in P \quad (34)$$

$$\tau_i \leq t_i^{ins} \quad \forall i \in P^{ins} \quad (35)$$

$$\varepsilon_u - t_i^{ins} \leq \Delta^{ins} \quad \forall u \in D^{fli}, i \in P_u^{com} \quad (36)$$

$$\varepsilon_u - t_i^{ins} - M(1 - \sum_{j \in P^{ins} \cup 0} \chi_{i,j,u}) \leq \Delta^{ins} \quad \forall u \in D, i \in P^{ins} \quad (37)$$

$$\beta_u \leq t_j^{ins} - TT(0, j) + M(1 - \chi_{0,j,u}) \quad \forall j \in P^{ins}, u \in D^{bas} \quad (38)$$

$$\varepsilon_u \geq t_i^{ins} + TT(i, 0) + M(\chi_{i,0,u} - 1) \quad \forall i \in P, u \in D \quad (39)$$

$$\sum_{i \in P^{ins} \cup \{0\}} \sum_{j \in P^{ins} \cup \{0\}} TT(i, j) \chi_{i,j,u} \cdot e^{pro} \leq \Delta_u^{bat} + M(1 - w_u) \quad \forall u \in D^{bas} \quad (40)$$

$$\left(\varepsilon_u - \beta_u - \sum_{i \in P^{ins} \cup \{0\}} \sum_{j \in P^{ins} \cup \{0\}} TT(i, j) \chi_{i,j,u} \right) \cdot e^{hov} +$$

$$\left(\varepsilon_u - t_{l_u}^{ins} - \sum_{i \in P^{ins} \cup \{l_u\}} \sum_{j \in P^{ins} \cup \{0\}} TT(i, j) \chi_{i,j,u} \right) \cdot e^{hov} + \sum_{i \in P^{ins} \cup \{l_u\}} \sum_{j \in P^{ins} \cup \{0\}} TT(i, j) \chi_{i,j,u} \cdot e^{pro} \leq \Delta_u^{bat} \quad \forall u \in D^{fli} \quad (41)$$

Constraints (34) to (41) formulate temporal and energy constraints. Constraint (34) enforces that the flight duration between any two inspection points must be less than the inspection interval. Constraint (35) mandates that drones arrive at sampling points before their scheduled sampling time. Constraints (36) and (37) jointly ensure all collected ballast water samples are returned to base within the time limit. Constraint (38) explicitly computes the departure time for drones launching from base, while constraint (39) calculates the mission completion time for each drone in the current cycle. Finally, constraints (40) to (41) guarantee the total energy consumption throughout the route remains within battery capacity limits.

3. Domains for the decision variables:

$$\chi_{i,j,u}, \omega_u = \{0, 1\} \quad \forall i, j \in P, u \in D \quad (42)$$

$$\tau_i, \beta_u, \varepsilon_u \geq 0 \quad \forall i \in P, u \in D \quad (43)$$

4.1.4. Approximate Dynamic Programming

The described exact algorithm suffers from the curse of dimensionality as its computational complexity scales exponentially with state quantities, particularly under large temporal scales. This may hinder rapid decision-making in BWI-D scenarios. To accelerate solution generation, we impose a state expansion width limit BW that constrains computational iterations in state transitions. Specifically, instead of enumerating all t^{beg} values in $[0, t]$, we only consider the last BW values: $t^{beg} \in (t - BW, t)$. Here, BW must satisfy $BW < \Delta^{max} / e^{pro}$ (maximum safe flight duration). This restriction reduces the solution space, yielding an approximate optimal solution. We feed this approximation as the initial solution into the ALNS algorithm in Section 4.2 for further refinement.

The state transition equation reveals that the original DP algorithm exhibits $T(n) \sim O(n!)$. When solved via backward recursion, the per-stage computational costs progress as $1, 2, \dots, n$ operations, yielding $T(n) \sim O\left(\frac{n(n+1)}{2}\right)$ complexity (equivalent to $O(n^2)$). For our BWDP, the first BW stages require 1 to BW operations respectively during backward recursion, while all subsequent stages demand exactly BW operations per recursion step. By conservatively bounding the initial stages' cost to BW , the BWDP achieves $T(n) \sim O(BW \cdot n)$ complexity.

4.2. Improvement of Initial Solution

Although the BWDP algorithm yields a high-quality initial solution, a non-trivial optimality gap persists. To bridge this gap, we integrate the proposed mathematical model within an ALNS framework. To accelerate convergence, we implement two specialized decomposition operators whose selection is dynamically adapted during the search process. The algorithm continuously evaluates each operator combination's performance based on achieved solution quality improvements and computational time expenditure, then adjusts their selection weights accordingly. At each iteration, operators are probabilistically selected via roulette wheel selection proportional to these dynamically updated weights, enabling autonomous adaptation to solution landscape characteristics.

4.2.1. Decomposition Operators

To ensure effective exploration of novel solution regions beyond local optima, we implement two complementary decomposition operators:

1. Random inspection-point decomposition: This operator randomly selects m inspection points (each associated with specific time windows) and resets all decision variables $\alpha_{i,j,p}$ spanning the corresponding time intervals to unassigned states. An illustration of an inspection-point decomposition is depicted in Figure 3.
2. Random flight-path decomposition: This operator randomly selects m flight paths and resets all decision variables $\alpha_{i,j,p}$ intersecting the time windows of inspection points along these paths. An illustration of a flight-path decomposition is depicted in Figure 4.

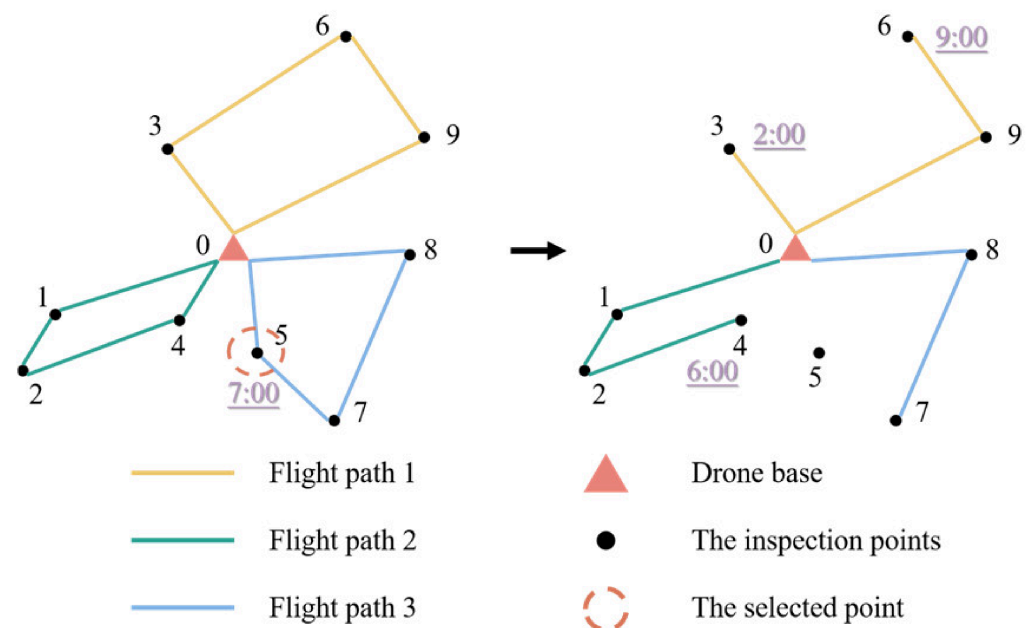


Figure 3. Schematic diagram of random inspection-point decomposition operator.

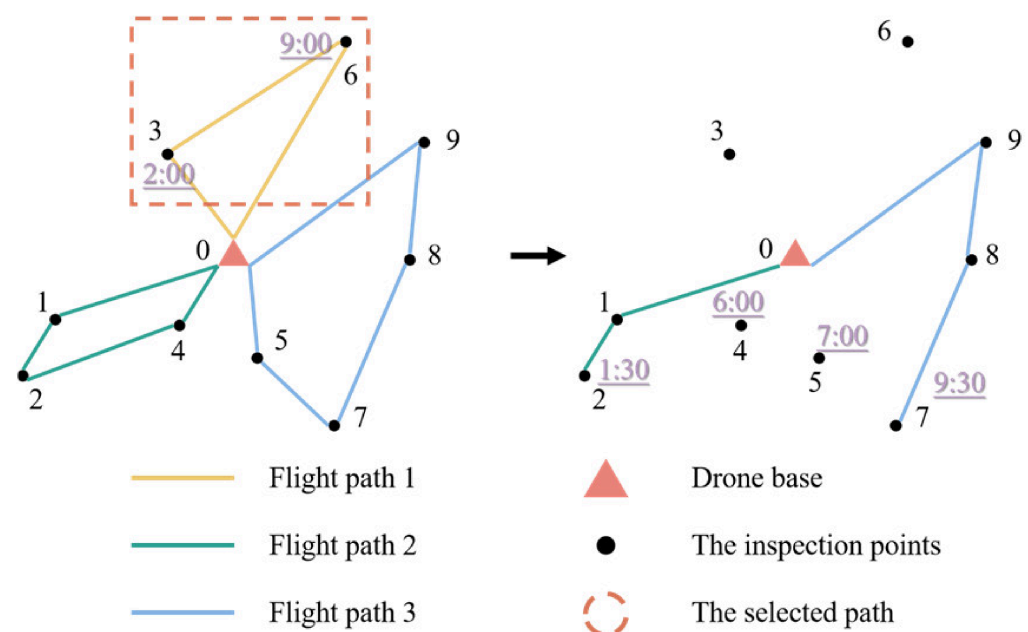


Figure 4. Schematic diagram of random flight-path decomposition operator.

4.2.2. Solution Repair

Following solution decomposition, a subset of decision variables $\alpha_{i,j,p}$ and $\kappa_{p,q,u}$ are reset to unassigned states. To reconstruct a complete solution, we formulate a reduced-scale optimization submodel. This submodel exclusively focuses on reassigning values to these reset variables while preserving the fixed values of unaffected variables. The significant reduction in free variables transforms the original NP-hard problem into computationally tractable subproblems. Given the limited scale of decomposition at each iteration, these submodels can be optimally solved within practical timeframes using exact mathematical programming solvers. In our implementation, GUROBI is employed to solve the integer programming submodel. Upon solver termination, all previously unassigned variables obtain optimal values, yielding a complete renewed solution for subsequent ALNS iterations.

4.2.3. Adaptive Operator Selection Mechanism

During the initialization phase, all decomposition operators ω are assigned equal selection weights $\Omega[\omega] = 1$. The adaptive mechanism operates in cycles of L iterations (denoted as the learning interval). Within each cycle, the algorithm records for each operator ω the cumulative improvement in objective value $I[\omega]$ and the total computational time $\Gamma[\omega]$. Based on these metrics, a performance score $P[\omega] = \alpha \cdot \tilde{I}[\omega] + (1 - \alpha) \cdot \tilde{r}[\omega]$ is computed, where $\tilde{I}[\omega] = I[\omega] / \sum_{\omega'} I[\omega']$ normalizes the improvement, $\tilde{r}[\omega] = (I[\omega] / \Gamma[\omega]) / \sum_{\omega'} (I[\omega'] / \Gamma[\omega'])$ normalizes the computational efficiency, and $\alpha \in [0, 1]$ trades off solution quality against speed. After each learning interval, operator weights are updated via an exponential smoothing rule:

$$\Omega[\omega]_{new} = (1 - \beta)\Omega[\omega]_{old} + \beta P[\omega] \quad (44)$$

where β is the smoothing factor, and $\tilde{P}[\omega] = P[\omega] / \sum_{\omega'} P[\omega']$. At each iteration, an operator is selected using roulette wheel selection with probabilities proportional to the updated weights.

4.2.4. Algorithmic Procedure of the AM Framework

The complete procedure of the AM algorithm is formalized in Algorithm 1. The algorithm comprises two phases: (i) construction of an initial solution S_0 via the BWDP procedure (Section 4.1), and (ii) iterative improvement using the ALNS framework with adaptive operator selection. The main loop executes for I_{max} iterations. At each iteration, a decomposition operator ω is selected by roulette wheel, applied to destroy part of the current solution S , and the resulting partial solution is repaired by solving the reduced submodel with GUROBI to obtain a candidate solution S' . The acceptance criterion follows a simulated annealing scheme: an improving S' is always accepted, while a non-improving S' is accepted with probability $\exp(-(f(S') - f(S))/T)$, where T denotes the current temperature, initialized to T_0 and decreased by a cooling rate c every iteration. Operator weights are updated every L iterations according to the mechanism described in Section 4.2.3.

Algorithm 1: Adaptive Metaheuristics**In:** *Instance*, I_{max} , L , β , α , T_0 , c , BW **Out:** S_{best} —the best solution

```

1:  /* Phase 1: Initial solution via BWDP */
2:   $S_0 \leftarrow BWDP(instance, BW)$        $\triangleleft$  Section 4.1
3:   $S \leftarrow S_0$ ,  $S_{best} \leftarrow S_0$ ,  $T \leftarrow T_0$        $\triangleleft$  Initialize temperature
4:   $\Omega[\omega_1] \leftarrow 1$ ,  $\Omega[\omega_2] \leftarrow 1$ ,  $I[\omega_1] \leftarrow 0$ ,  $I[\omega_2] \leftarrow 0$ ,  $\Gamma[\omega_1] \leftarrow 0$ ,  $\Gamma[\omega_2] \leftarrow 0$ ,  $k \leftarrow 1$ 
5:  /* Phase 2: ALNS with adaptive operator selection */
6:  while  $k \leq I_{max}$  do
7:       $\omega \leftarrow RouletteWheelSelect(\Omega[\omega])$        $\triangleleft$  Select a decomposition operator by
          roulette wheel
8:       $S_{destroyed} \leftarrow Decompose(S, \omega)$        $\triangleleft$  Section 4.2.1: decompose current
          solution using operator  $\omega$ 
9:       $S' \leftarrow SolveSubmodel(S_{destroyed})$        $\triangleleft$  Section 4.2.2: solve the reduced
          submodel to optimality
10:     if  $f(S') < f(S)$  then       $\triangleleft$  Acceptance criterion (simulated annealing)
11:          $S \leftarrow S'$ 
12:     else
13:          $r \leftarrow Random(0, 1)$ 
14:         if  $r < \exp(-(f(S') - f(S))/T)$  then
15:              $S \leftarrow S'$ 
16:         end if
17:     end if
18:      $S_{best} \leftarrow \max(S, S_{best})$        $\triangleleft$  Update best solution
19:     if  $f(S') < f(S)$  then       $\triangleleft$  Record operator performance
20:          $I[\omega] \leftarrow I[\omega] + (f(S) - f(S'))$ 
21:     end if
22:      $\Gamma[\omega] \leftarrow \Gamma[\omega] + TimeSpent(\omega)$ 
23:      $T \leftarrow c \cdot T$        $\triangleleft$  Cooling
24:     if  $k \bmod L = 0$  then       $\triangleleft$  Update operator weights every  $L$  iterations
25:          $Update(\Omega[\omega])$        $\triangleleft$  Section 4.2.3
26:          $I[\omega] \leftarrow 0$ ,  $\Gamma[\omega] \leftarrow 0$ 
27:     end if
28:      $k \leftarrow k + 1$ 
29: end while
30: return  $S_{best}$ 

```

5. Computational Experiments

All heuristic algorithms were implemented in Python 3.7 and executed on a Windows 11 system equipped with an Intel® Core™ i7-1065G7 CPU (1.50 GHz max frequency) and 16 GB RAM. The integrated mathematical model was coded in Python 3.7 and solved using the GUROBI 9.1.2 optimization solver.

5.1. Instance Experiment

Field investigations at a major northern Chinese port established our experimental scenario. Vessel navigation routes were obtained from publicly available nautical charts of the port area, with distances measured directly from the charts; the in-port sailing speed was set to 7 knots based on operational experience shared by port professionals. Travel times were therefore computed from chart-measured distances and this speed. The arrival times

of the 10 vessels, listed in Table 3, represent a typical 48 h snapshot provided by the port authority. To account for operational uncertainty, initial vessel positions were randomized within a 0.5-nautical-mile radius of the chart-derived boarding points. Shore-based drone bases dispatch drones to conduct randomized ballast water sampling along these navigation paths as depicted in Figure 5. Each vessel route was divided into equidistant sampling locations as illustrated in Figure 6. Collected samples require delivery to the shore base within a 1 h time window ($\Delta^{ins} = 60$) from collection time.

Table 3. The data of vessels.

Vessel ID	Position	Arrival Time
Vessel_001	38°52.7' N, 121°47.7' E	5:25
Vessel_002	38°53.1' N, 121°47.2' E	8:59
Vessel_003	38°53.3' N, 121°47.8' E	10:06
Vessel_004	38°53.0' N, 121°47.4' E	15:25
Vessel_005	38°53.4' N, 121°47.4' E	17:13
Vessel_006	38°53.0' N, 121°48.1' E	20:46
Vessel_007	38°52.8' N, 121°47.2' E	23:37
Vessel_008	38°53.3' N, 121°47.0' E	28:29
Vessel_009	38°52.7' N, 121°47.4' E	31:18
Vessel_010	38°52.7' N, 121°48.0' E	34:44

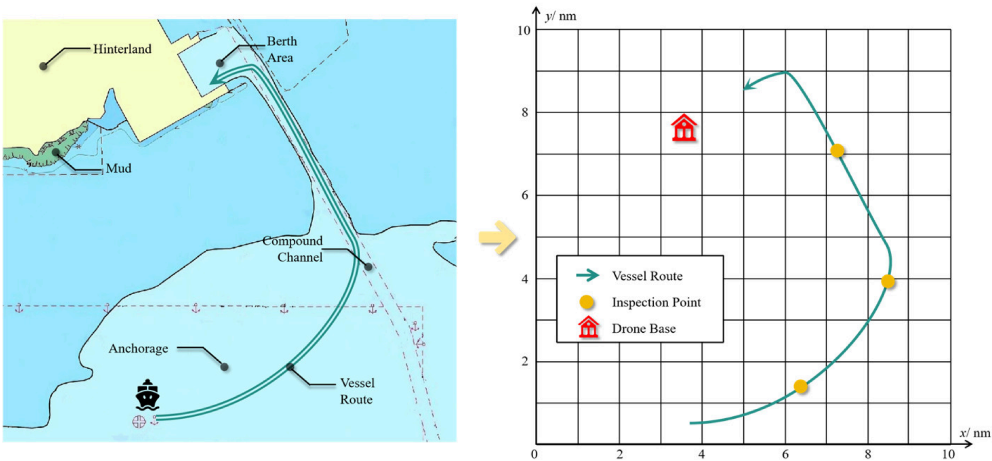


Figure 5. Layout of the port area.

Drones are deployed from the shore-based bases to collect ballast water samples at predetermined vessel locations. The drone flight speed was set to a constant safe cruising speed of $10\text{ m}\cdot\text{s}^{-1}$, which is typical for multi-rotor inspection drones and was consistent with the flight behavior observed on site. The drone energy parameters were calibrated through controlled flight tests conducted at the port using the actual water-sampling drone. With the port’s permission, we measured the full-battery endurance of the drone under two conditions: constant-speed cruising at $10\text{ m}\cdot\text{s}^{-1}$ and steady hovering. Normalizing the full battery capacity to 100 units ($\Delta^{max} = 100$), the propulsion consumption was determined to be $1\text{ unit}\cdot\text{min}^{-1}$ ($e^{pro} = 1$) and the hovering consumption to be $1.5\text{ units}\cdot\text{min}^{-1}$ ($e^{hov} = 1.5$). The charging rate of $0.4\text{ units}\cdot\text{min}^{-1}$ ($e^{cha} = 0.4$) was derived from the observed charging profile during these tests. Although the specific drone model cannot be disclosed because of port confidentiality requirements, these normalized values faithfully represent the real energy behavior of the tested aircraft. The resulting inspection plan for the base instance is presented in Table 4. Analysis reveals frequent drone returns to shore bases, attributable to three operational constraints. Firstly, drones must return before energy depletion to ensure operational safety. Secondly, when sampling intervals exceed threshold durations,

returning for concurrent charging proves more efficient than hovering on-site. Finally, ballast water samples require delivery within the 1 h time window from collection to maintain biological integrity.

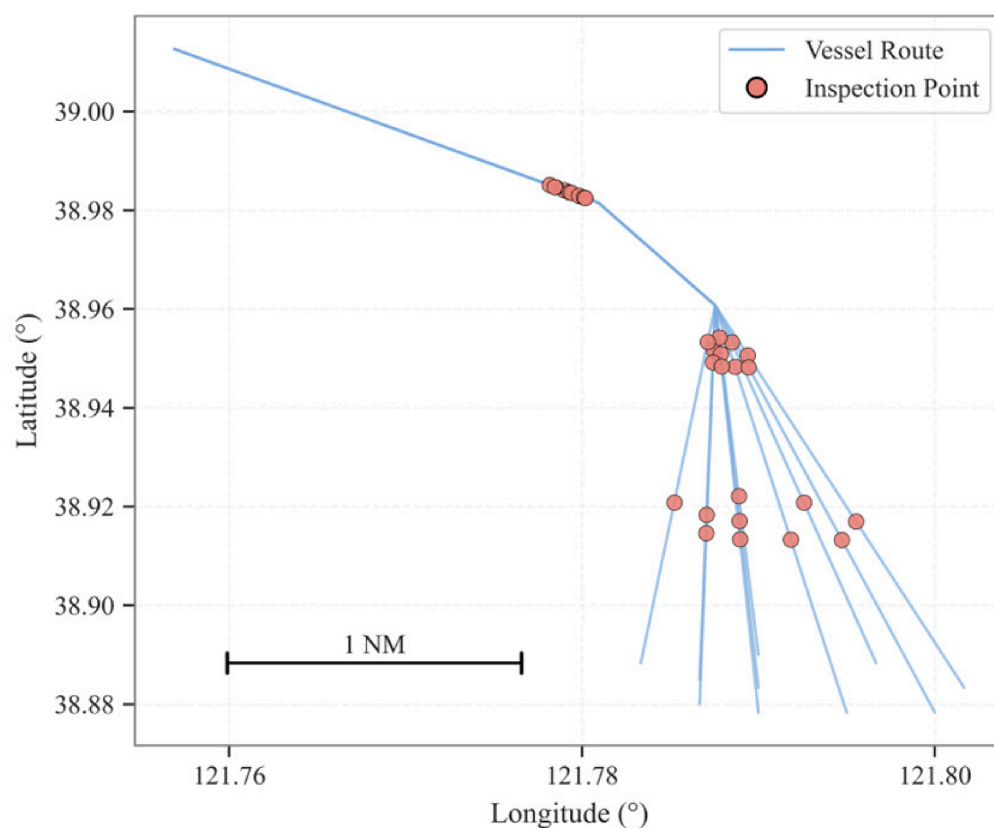


Figure 6. Location of inspection points.

Table 4. Detailed scheduling scheme for the ballast water inspection planning of 10 vessels.

Drone	Path No.	Sequence	Departure	Return	Energy (Unit)
Drone 1	1	0→1→2→0	05:33	06:13	44.5
	2	0→8→9→0	10:29	11:08	44.6
	3	0→10→0	15:30	15:54	24.0
	4	0→15→0	17:50	18:14	23.5
	5	0→21→0	24:19	24:41	22.8
	6	0→22→23→0	28:35	29:14	43.7
Drone 2	1	0→3→0	06:07	06:31	24.5
	2	0→6→7→0	09:39	10:33	66.9
	3	0→13→14→0	17:19	17:58	44.2
	4	0→16→17→0	20:52	21:32	44.8
	5	0→27→28→0	32:02	32:32	30.0
Drone 3	1	0→4→5→0	09:05	09:45	27.0
	2	0→11→12→0	15:48	16:28	27.0
	3	0→18→0	21:27	21:49	22.0
	4	0→19→20→0	23:42	24:22	28.0
	5	0→24→0	28:58	29:19	22.0
	6	0→25→26→0	31:25	32:05	27.0
	7	0→29→30→0	35:09	35:49	27.3

5.2. Comparison with the Exciting Charge Strategy

This section evaluates our partial charge strategy (PCS) against two charge strategies.

1. **Full Charge Strategy (FCS):** This strategy employs a time-prioritized task allocation mechanism operating under rigid protocols. All pending inspection points are chronologically sequenced based on task initiation times to form a strict execution order. For each target point, the strategy filters available drones possessing sufficient charge for round-trip transit and sampling operations. The drone requiring minimal flight energy consumption is selected, with its latest safe return timestamp recorded. During continuous assignments, any drone exceeding its designated return timestamp must immediately return for charging upon task completion. Above all, all drones must mandate full replenishment to 100% battery capacity before a new flight path.
2. **70%-Threshold Strategy (70TS):** Building upon the FCS's task allocation framework, this variant introduces fixed charging thresholds. While maintaining identical sequencing and drone selection procedures, it modifies the charging protocol: drones returning for charging need only reach 70% battery capacity before being deployed on subsequent flight paths.

To comprehensively compare the proposed PCS with the aforementioned benchmark strategies, we establish two experimental scales: small-scale with 10 to 40 inspection points, and large-scale with 90 to 200 inspection points. The number of vessels is configured as $|V|^{sm} = 5, 6, 7, 8, 9, 10$ and $|V|^{la} = 30, 32, 34, 36, 38, 40$, respectively. The inspection frequency per vessel is defined as $n^{sm} = 2, 3, 4$ and $n^{la} = 3, 4, 5$. This configuration yields 36 unique experimental instances. For simplified referencing, we adopt the naming convention '[instance ID]_V[vessel count]n[frequency]'. For example, the 17th experimental instance configured with 10 vessels and an inspection frequency of 3 is denoted as 17_V10n3. Comparative results between the PCS and exciting charge strategies are systematically presented in Tables 5 and 6, which respectively showcase the performance comparisons for small-scale and large-scale experimental scenarios.

Table 5. Comparison with existing charge strategies for the small-scale instances (PCS, 70TS, and FCS all solved using the proposed AM algorithm).

Instance ID	Drone Count $ D $	Obj. (Unit)		
		FCS	70TS	PCS
1_V5n2	3	237.9	217.9	182.6
2_V5n3	3	359.7	326.8	274.8
3_V5n4	3	405.8	372.6	314.6
4_V6n2	3	318.0	287.7	241.3
5_V6n3	3	448.3	412.0	345.6
6_V6n4	3	509.9	477.9	394.6
7_V7n2	3	333.5	302.2	254
8_V7n3	3	461.0	430.0	356.2
9_V7n4	3	529.1	489.3	407.8
10_V8n2	3	437.2	409.3	339.3
11_V8n3	3	612.5	567.1	465.9
12_V8n4	3	697.1	641.1	528.7
13_V9n2	3	451.8	420.8	347.9
14_V9n3	3	676.4	624.0	516
15_V9n4	3	760.8	712.6	584.1
16_V10n2	3	563.8	519.7	434
17_V10n3	3	775.1	732.5	593.8
18_V10n4	3	877.2	832.6	676.4

Table 6. Comparison with existing charge strategies for the large-scale instances (same setting as Table 5).

Instance ID	Drone Count $ D $	Obj. (Unit)		
		FCS	70TS	PCS
19_V30n3	15	1946.5	1757.7	1427.2
20_V30n4	15	2239.2	2067.6	1666.7
21_V30n5	15	-	-	1987
22_V32n3	15	1863.1	1722.2	1398.5
23_V32n4	15	2354.7	2157.3	1751.3
24_V32n5	15	-	-	1953.4
25_V34n3	15	1877.2	1761.1	1411
26_V34n4	15	-	2090.3	1696.1
27_V34n5	15	-	-	2057.7
28_V36n3	15	2156.4	2004.1	1613.2
29_V36n4	15	2773.3	2476.2	2041.4
30_V36n5	15	-	-	2337.5
31_V38n3	15	2109.5	1930.0	1566.7
32_V38n4	15	-	2421.0	1936.5
33_V38n5	15	-	-	2152.6
34_V40n3	15	-	2111.2	1703.2
35_V40n4	15	-	-	2075.9
36_V40n5	15	-	-	2427.7

The experimental results clearly demonstrate that the 70TS consistently outperforms the FCS in both small-scale and large-scale instances. As shown in Table 5, the PCS shows minimal differences compared to the two benchmark strategies when processing limited inspection points. This is because drone path scheduling dominates efficiency under such conditions. However, as inspection points increase (Table 6), the FCS becomes the first strategy unable to complete inspection tasks under identical drone counts, followed by the 70TS. This failure stems from mandatory charging durations, which raise drone idle rates and prevent timely sampling. Critically, in large-scale instances, PCS achieves a 25% reduction in energy consumption versus 70TS while successfully completing tasks with fewer drones. These efficiencies translate to significant operational cost savings for ballast water inspection.

To put this in concrete operational terms, consider instance 34_V40n3: with 15 drones, FCS fails to complete all inspections within the 48 h planning horizon, whereas PCS successfully finishes the same tasks with 15 drones within 48 h. More broadly, across large-scale instances where FCS and 70TS fail, PCS completes all inspections with the same fleet size and within the same time horizon, demonstrating the practical advantage of adaptive partial charging in resource-constrained port operations.

5.3. Model Sensitivity Analysis

To further examine the model's behavior under different operational conditions, this section conducts a sensitivity analysis on two key decision-related parameters: the drone fleet size and flight speed. They significantly influence the energy consumption and planning outcome. Using the 17_V10n3 instance as a benchmark case, we evaluated the total energy consumption of the ballast water inspection planning solutions for fleet sizes ranging from three to nine drones. Depicted in Figure 7, the results reveal a distinct pattern: total energy consumption decreases rapidly initially as the number of drones increases, but this decline gradually plateaus at larger fleet sizes. Concurrently, the marginal energy consumption exhibits a diminishing trend. This pattern clearly demonstrates the marginal utility associated with increasing the drone fleet size. Notably, the marginal

energy consumption converges to near zero when the fleet size reaches seven drones. This observation suggests that, for this specific vessel scenario under consideration, a fleet of seven drones operates within the optimal range for effectively meeting the ballast water inspection demand.

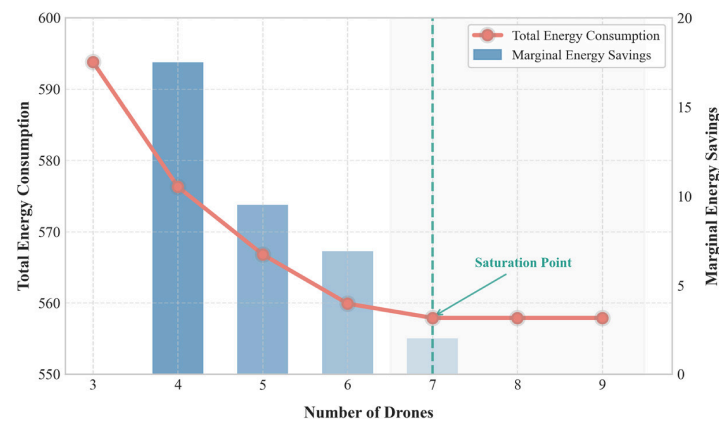


Figure 7. Objective results under different numbers of drones.

Contrary to the intuition that higher flight speed invariably leads to greater energy consumption, the relationship between drone speed and energy usage is non-monotonic, exhibiting an initial increase followed by a decrease as speed rises [28]. To investigate the impact of varying flight speeds on scheduling outcomes, we evaluated speeds of 8, 9, 10, 11, and 12 (m/s), corresponding to flight power consumption rates of 1.1, 1.0, 1.0, 1.2, and 1.5 (units/min), respectively. This analysis was conducted using both the small-scale benchmark instance 17_V10n3 and the large-scale instance 21_V30n5. Detailed numerical results are presented in Table 7. A critical factor influencing total energy consumption is the dual energy expenditure during drone operations: energy is consumed not only during flight but also significantly during hovering. Consequently, total energy consumption is not solely proportional to total flight time. The results in Table 7 demonstrate that for the specific sensor deployment scenarios represented by these two instances, once sufficient drones are deployed to complete the ballast water inspection plan, indiscriminately increasing flight speed does not yield a reduction in total energy consumption. This finding highlights that the model and algorithm developed in this study can serve as a decision support tool for maritime authorities to determine the energy-optimal flight speed for drones under varying operational conditions.

Table 7. Objective values under different drone flight speeds.

Drone Flight Speed	Propulsion Energy Consumption e^{pro}	21_V30n5		17_V10n3	
		Obj.	Total Flight Time	Obj.	Total Flight Time
8	1.1	2398.4	1557.5	733.5	494.2
9	1	2061.3	1405.4	627.1	415.1
10	1	1987	1412.8	593.8	469.1
11	1.2	2054.7	1125.6	643.3	426.9
12	1.5	2206.5	995.9	721.8	377

5.4. Solution Quality and Ablation Study

In this subsection, we conduct a detailed comparative analysis of our proposed AM algorithm against several established benchmark methods: the GUROBI solver, the variable neighborhood search (VNS) algorithm, and the ALNS algorithm. To ensure a fair comparison, the heuristic algorithms utilize the same set of neighborhood operators implemented

in our AM algorithm and share identical termination criteria. The specific configurations for the heuristic benchmarks are detailed below:

1. VNS begins by generating a high-quality initial solution using the 70TS described in Section 5.2. Subsequently, in each iteration, the algorithm randomly selects a single neighborhood operator from the shared operator pool. This selected operator is applied to perturb the current solution, and 20 candidate solutions are generated and evaluated within the resulting neighborhood. The best solution found among these 20 candidates is accepted as the new current solution for the next iteration. The algorithm terminates after reaching a maximum of 200 iterations and returns the global best solution discovered throughout the entire search process.
2. ALNS also starts by generating its initial solution using the 70TS. In each iteration, ALNS adaptively selects a destroy operator based on the adaptive weight selection mechanism defined within the ALNS component of our AM algorithm. This operator partially dismantles the current solution. The partially destroyed solution is then repaired using a corresponding repair operator. During this repair process, 20 candidate solutions are generated and evaluated. The best solution found among these 20 candidates is accepted as the new current solution. The algorithm runs for a maximum of 200 iterations and ultimately returns the global best solution found during the search.

These three configurations also constitute an ablation hierarchy, where each successive variant adds a single algorithmic component. Specifically, VNS provides a baseline with random operator selection and 70TS initialization; ALNS adds the adaptive operator selection mechanism; and AM further introduces the BWDP initial solution. Comparing them quantifies the individual contribution of each component.

Given that the proposed AM algorithm belongs to the category of stochastic search algorithms, we performed 10 independent replications for each small-scale problem instance to obtain reliable average values for both the objective function and computational time. To first establish the absolute solution quality of the AM algorithm, we compare it with the exact optimal solutions obtained by GUROBI on small-scale instances, where GUROBI remains tractable. The results are reported in Table 8. Analysis of Table 8 reveals that the GUROBI solver can find optimal solutions within very short computation times for instances with a small number of inspection points. However, its performance degrades significantly as the problem size increases, becoming unable to find optimal solutions within reasonable time limits for larger instances. This observation strongly indicates that the mathematical model for the BWI-D problem possesses high computational complexity, necessitating the development of effective heuristic approaches for tackling large-scale instances. The experimental results demonstrate the effectiveness of the AM algorithm. Although AM did not always attain the global optimum identified by GUROBI on small instances where the optimum is known, the maximum optimality gap observed across all instances did not exceed 1.06%.

We then conduct an ablation study on large-scale instances, comparing the three algorithmic configurations VNS, ALNS, and AM to isolate the contribution of each component. These results are presented in Table 9. To ensure statistical reliability, each algorithm was executed for 10 independent replications per instance, enabling the calculation of robust average values for both the objective function and computational time. As detailed in Table 9, the results reveal several significant insights. From an ablation perspective, these three algorithms allow us to isolate the individual contributions of the two key components in AM: the adaptive operator selection mechanism and the BWDP initialization. Firstly, the ALNS algorithm consistently outperformed the baseline VNS in solution quality, achieving an average improvement of 2% in the objective function value. Since the only difference between VNS and ALNS is that

VNS selects operators randomly while ALNS uses the adaptive weight update, this improvement quantifies the contribution of the adaptive operator selection mechanism. Secondly, the AM algorithm adds the BWDP initialization on top of ALNS, producing solutions that are on average 2.08% better than ALNS. This gain directly measures the contribution of the high-quality initial solution generated by BWDP. Third, AM also outperforms VNS by a larger margin, confirming that the two components are complementary. In summary, this ablation analysis demonstrates that both the adaptive operator selection and the BWDP initialization contribute meaningfully to the superior performance of AM, with their effects being additive and consistent across all large-scale instances.

Table 8. Comparison of objective values and computation times between GUROBI and AM on small-scale instances.

Instance ID	Drone Count $ D $	GUROBI		AM	
		Obj. (Unit)	Time (s)	Obj. (Unit)	Time (s)
1_V5n2	3	182.6	0.47	182.6	0.54
2_V5n3	3	274.5	0.80	274.8	3.15
3_V5n4	3	314.3	9.52	314.6	2.93
4_V6n2	3	241.3	0.77	241.3	0.55
5_V6n3	3	344.1	54.27	345.6	2.46
6_V6n4	3	391.2	3524.83	394.6	6.80
7_V7n2	3	253.7	0.63	254	3.23
8_V7n3	3	353.6	28.46	356.2	2.05
9_V7n4	3	404.2	35.92	407.8	7.53
10_V8n2	3	339.0	1.36	339.3	2.23
11_V8n3	3	457.7	16.93	465.9	5.70
12_V8n4	3	525.6	6660.93	528.7	6.65
13_V9n2	3	347.6	19.40	347.9	3.13
14_V9n3	3	509.3	538.04	516	6.06
15_V9n4	3	579.7	7035.89	584.1	8.69
16_V10n2	3	429.4	5.48	434	6.78
17_V10n3	3	590.3	1283.00	593.8	7.39
18_V10n4	3	690.8	7200.00	676.4	7.82

Table 9. Comparison of objective values and computation times among VNS, ALNS, and AM on large-scale instances.

Instance ID	Drone Count $ D $	VNS		ALNS		AM	
		Obj. (Unit)	Time (s)	Obj. (Unit)	Time (s)	Obj. (Unit)	Time (s)
19_V30n3	15	1482.8	97.80	1478.9	101.68	1427.2	158.45
20_V30n4	15	1740.1	178.65	1692.8	171.48	1666.7	293.95
21_V30n5	15	2069.4	157.15	2043.0	149.45	1987	296.50
22_V32n3	15	1451.4	123.39	1446.9	128.96	1398.5	183.36
23_V32n4	15	1828.6	153.02	1788.2	150.30	1751.3	237.83
24_V32n5	15	2026.9	111.15	1977.5	102.02	1953.4	210.48
25_V34n3	15	1464.9	119.60	1411.8	121.34	1411	175.74
26_V34n4	15	1768.8	114.48	1709.8	112.45	1696.1	208.60
27_V34n5	15	2139.2	142.26	2063.3	129.96	2057.7	226.17
28_V36n3	15	1676.1	104.98	1659.1	99.93	1613.2	203.92
29_V36n4	15	2122.1	154.58	2084.0	144.55	2041.4	273.47
30_V36n5	15	2442.5	177.03	2417.7	168.02	2337.5	338.14
31_V38n3	15	1630.6	98.37	1624.9	95.06	1566.7	192.22
32_V38n4	15	2023.9	139.23	1949.4	127.56	1936.5	220.94
33_V38n5	15	2239.1	232.87	2229.0	220.04	2152.6	399.73
34_V40n3	15	1769.1	150.88	1708.1	147.09	1703.2	216.73
35_V40n4	15	2167.6	255.89	2132.1	233.56	2075.9	399.93
36_V40n5	15	2528.8	257.13	2479.6	241.36	2427.7	370.43

A closer examination of the computational times reported in Tables 8 and 9 reveals the scalability characteristics of the algorithms. The AM algorithm maintains its computational time within approximately 400 s even for the largest instances. In contrast, GUROBI exhibits

an exponential growth in solution time as instance size increases, exceeding 7000 s for instances with 40 inspection points. This divergence confirms that while exact methods are suitable for small-scale planning, the AM algorithm provides a scalable and practical alternative for large-scale port scenarios.

5.5. Convergence Behavior Analysis

To evaluate the convergence behavior of the AM algorithm, we further quantified the convergence behavior by computing the relative improvement after the first 50 iterations and after the first 100 iterations across all 36 experimental instances. The results are summarized in Table 10.

Table 10. Average relative improvement over initial solution (%).

Algorithm	After 50 Iterations	After 100 Iterations	After 150 Iterations
AM (small-scale)	3.2	4.1	4.1
AM (large-scale)	4.8	5.9	6.2
ALNS (small-scale)	8.7	10.2	10.5
ALNS (large-scale)	11.3	13.1	13.5

As shown in Table 10, ALNS achieves larger relative improvements over its initial solution, which is expected because its 70TS initialization is of lower quality than the BWDP initialization used by AM. However, AM consistently attains a better absolute final objective value, confirming that the BWDP initialization places the search in a more promising region of the solution space.

5.6. Sensitivity to Algorithm Parameters

The bounded width parameter in the BWDP initialization phase influences the performance of AM algorithm. This parameter controls the number of time-step values considered during state expansions in the BWDP phase. We tested values of 5, 10, 15, and 20 on three large-scale instances. The results presented in Table 11 show the objective value and computational time for each configuration.

Table 11. Impact of bounded width parameter.

Instance ID	BW = 5		BW = 10		BW = 15		BW = 20	
	Obj. (Unit)	Time (s)	Obj. (Unit)	Time (s)	Obj. (Unit)	Time (s)	Obj. (Unit)	Time (s)
21_V30n5	1981	202.4	1987	223.1	1987	296.5	1987	398.7
27_V34n5	2051	185.3	2058	198.7	2057	226.2	2057	312.4
30_V36n5	2337	262.1	2337	283.5	2337	338.1	2337	467.2

The results indicate that the solution quality is relatively insensitive once it reaches 10, while the computational time increases approximately linearly. Based on this analysis, we selected as a balanced choice for all experiments reported in Section 5.

6. Conclusions

This study presents ballast water inspection planning with drones considering a partial charging problem. To overcome drone energy limitation, we develop a novel partial charging strategy (PCS) enabling adaptive energy replenishment. This strategy is integrated in a MIP model, which simultaneously optimizes drone routing and adaptive charging durations. Crucially, the model incorporates dual energy consumption (propulsion and hovering) and enforces complex spatiotemporal constraints for sampling operations.

To efficiently solve this NP-Hard problem, an AM algorithm combining BWDP and ALNS is proposed. Experimental validation using real-world port data demonstrates significant advancements: The PCS achieves 25% energy reduction compared to conventional 70%-Threshold Strategies in the large-scale instance, while successfully completing inspections where benchmark methods failed. Algorithmically, AM yields near-optimal solutions within 1.06% of GUROBI's objective values for the small-scale instance and outperforms ALNS by an average of 2.08% for the large-scale instance. Sensitivity analysis further reveals that the ballast water inspection planning is significantly influenced by the fleet size and flight speed of drones, providing maritime authorities with a quantifiable decision-making tool to optimize drone configurations based on port-specific constraints.

Several promising directions emerge from this study. First, real-time environmental disturbances such as wind gusts and sea state variability can substantially affect drone energy consumption and flight trajectories. Incorporating these stochastic factors into the model would improve robustness for operational deployment. Second, the current work assumes that drones operate independently of surface vessels; future research could investigate hybrid drone–vessel coordination, where patrol boats serve as mobile charging platforms or sample relay points, potentially reducing the number of required drone trips. Third, extending the single-base scenario to multi-port collaborative inspection would align the framework with the practical reality that maritime regulatory authorities often oversee inspection operations across geographically distributed facilities, creating opportunities for inter-port resource sharing. Fourth, the linear energy consumption model adopted here neglects payload effects. For sampling missions requiring larger ballast water volumes, payload-dependent energy models should be integrated, which would make the partial charging decisions even more critical since the energy budget becomes time-varying and load-sensitive. Fifth, the current model treats vessels as stationary inspection points; a moving-target formulation where drones rendezvous with vessels underway would more accurately reflect on-the-fly sampling operations and introduce a new layer of spatiotemporal coordination complexity.

Author Contributions: Conceptualization, X.F. and H.Z.; methodology, X.F.; software, X.F. and Z.W.; validation, X.F. and Z.W.; formal analysis, X.F.; investigation, X.F.; resources, H.Z.; data curation, H.Z.; writing—original draft preparation, X.F.; writing—review and editing, X.F., Z.W. and H.Z.; visualization, X.F.; supervision, H.Z.; project administration, X.F.; funding acquisition, H.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, under Grant no. 71872025.

Data Availability Statement: Data are available upon request.

Acknowledgments: The authors would like to thank the developers of the GUROBI solver used in this study.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang, Z.; Saebi, M.; Grey, E.K.; Corbett, J.J.; Chen, D.; Yang, D.; Wan, Z. Ballast Water-Mediated Species Spread Risk Dynamics and Policy Implications to Reduce the Invasion Risk to the Mediterranean Sea. *Mar. Pollut. Bull.* **2022**, *174*, 113285. [[CrossRef](#)] [[PubMed](#)]
2. Duan, C.; Yang, M.; Wang, Q.; Xue, J.; Yuan, L.; Wu, H. Impacts of Salinity Stress Caused by Ballast Water Discharge on Freshwater Ecosystems. *Reg. Stud. Mar. Sci.* **2023**, *65*, 103079. [[CrossRef](#)]
3. International Maritime Organization. *International Convention for the Control and Management of Ships' Ballast Water and Sediments (BWM)*; International Maritime Organization: London, UK, 2004.
4. Yang, Z.; Yu, X.; Dedman, S.; Rosso, M.; Zhu, J.; Yang, J.; Xia, Y.; Tian, Y.; Zhang, G.; Wang, J. UAV Remote Sensing Applications in Marine Monitoring: Knowledge Visualization and Review. *Sci. Total Environ.* **2022**, *838*, 155939. [[CrossRef](#)] [[PubMed](#)]

5. Yuan, S.; Li, Y.; Bao, F.; Xu, H.; Yang, Y.; Yan, Q.; Zhong, S.; Yin, H.; Xu, J.; Huang, Z.; et al. Marine Environmental Monitoring with Unmanned Vehicle Platforms: Present Applications and Future Prospects. *Sci. Total Environ.* **2023**, *858*, 159741. [[CrossRef](#)] [[PubMed](#)]
6. Guban, G.; Haque, A. Path Planning for Autonomous Drones: Challenges and Future Directions. *Drones* **2023**, *7*, 169. [[CrossRef](#)]
7. Wang, Q.; Lin, L.; Chen, X.; Wu, W.; Wu, H. Transportation of Bloom Forming Species in Ballast Water by Commercial Vessels at Yangshan Deep Water Port. *Ocean Coast. Manag.* **2022**, *219*, 106045. [[CrossRef](#)]
8. Nkouefuth Nfongmo, Y.; Onana, F.M.; Masseret, E.; Nana, P.A.; Ewoukem, T.E.; Kacimi, A. Estimation of the Introduction Risk of Non-Indigenous Species through Ship Ballast Water in the Port of Douala (Cameroon). *Mar. Pollut. Bull.* **2024**, *198*, 115794. [[CrossRef](#)] [[PubMed](#)]
9. Nosrati-Ghods, N.; Ghadiri, M.; Früh, W.-G. Management and Environmental Risk Study of the Physicochemical Parameters of Ballast Water. *Mar. Pollut. Bull.* **2017**, *114*, 428–438. [[CrossRef](#)] [[PubMed](#)]
10. Shomar, B.; Solano, J.R. Probabilistic Human Health Risk Assessment of Trace Elements in Ballast Water Treated by Reverse Osmosis Desalination Plants. *Mar. Pollut. Bull.* **2023**, *188*, 114667. [[CrossRef](#)] [[PubMed](#)]
11. Saglam, H.; Duzgunes, E. Effect of Ballast Water on Marine Ecosystem. In *Exergy for A Better Environment and Improved Sustainability 2*; Aloui, F., Dincer, I., Eds.; Green Energy and Technology; Springer International Publishing: Cham, Switzerland, 2018; pp. 373–382.
12. Nwigwe, T.I.; Kiyokazu, M. Investigation of Ballast Water Quality in Onne Harbor-Physiochemical Assessment. *Int. J. Environ. Sci. Technol.* **2023**, *20*, 13799–13808. [[CrossRef](#)]
13. Yaakob, O.; Mohamed, Z.; Hanafiah, M.S.; Suprayogi, D.T.; Abdul Ghani, M.P.; Adnan, F.A.; Mukti, M.A.A.; Din, J. Development of Unmanned Surface Vehicle (USV) for Sea Patrol and Environmental Monitoring. In Proceedings of the International Conference on Marine Technology, Kuala Terengganu, Malaysia, 20 October 2012.
14. Sendner, F.-M. An Energy-Autonomous UAV Swarm Concept to Support Sea-Rescue and Maritime Patrol Missions in the Mediterranean Sea. *Aircr. Eng. Aerosp. Technol.* **2022**, *94*, 112–123.
15. Villa, T.; Gonzalez, F.; Miljevic, B.; Ristovski, Z.; Morawska, L. An Overview of Small Unmanned Aerial Vehicles for Air Quality Measurements: Present Applications and Future Prospectives. *Sensors* **2016**, *16*, 1072. [[CrossRef](#)] [[PubMed](#)]
16. Padró, J.-C.; Carabassa, V.; Balagué, J.; Brotons, L.; Alcañiz, J.M.; Pons, X. Monitoring Opencast Mine Restorations Using Unmanned Aerial System (UAS) Imagery. *Sci. Total Environ.* **2019**, *657*, 1602–1614. [[CrossRef](#)] [[PubMed](#)]
17. Velusamy, P.; Rajendran, S.; Mahendran, R.K.; Naseer, S.; Shafiq, M.; Choi, J.-G. Unmanned Aerial Vehicles (UAV) in Precision Agriculture: Applications and Challenges. *Energies* **2021**, *15*, 217. [[CrossRef](#)]
18. Corbau, C.; Buoninsegni, J.; Olivo, E.; Vaccaro, C.; Nardin, W.; Simeoni, U. Understanding through Drone Image Analysis the Interactions between Geomorphology, Vegetation and Marine Debris along a Sandy Spit. *Mar. Pollut. Bull.* **2023**, *187*, 114515. [[CrossRef](#)] [[PubMed](#)]
19. Sun, L.; Zhang, Y.; Ouyang, C.; Yin, S.; Ren, X.; Fu, S. A Portable UAV-Based Laser-Induced Fluorescence Lidar System for Oil Pollution and Aquatic Environment Monitoring. *Opt. Commun.* **2023**, *527*, 128914. [[CrossRef](#)]
20. Horricks, R.A.; Bannister, C.; Lewis-McCrea, L.M.; Hicks, J.; Watson, K.; Reid, G.K. Comparison of Drone and Vessel-Based Collection of Microbiological Water Samples in Marine Environments. *Environ. Monit. Assess.* **2022**, *194*, 439. [[CrossRef](#)] [[PubMed](#)]
21. Zhang, J.; Campbell, J.F.; Sweeney, D.C.; Hupman, A.C. Energy Consumption Models for Delivery Drones: A Comparison and Assessment. *Transp. Res. Part D Transp. Environ.* **2021**, *90*, 102668. [[CrossRef](#)]
22. Alyassi, R.; Khonji, M.; Karapetyan, A.; Chau, S.C.-K.; Elbassioni, K.; Tseng, C.-M. Autonomous Recharging and Flight Mission Planning for Battery-Operated Autonomous Drones. *IEEE Trans. Autom. Sci. Eng.* **2023**, *20*, 1034–1046. [[CrossRef](#)]
23. Wang, K.; Zhang, X.; Duan, L.; Tie, J. Multi-UAV Cooperative Trajectory for Servicing Dynamic Demands and Charging Battery. *IEEE Trans. Mob. Comput.* **2023**, *22*, 1599–1614.
24. Arafat, M.Y.; Moh, S. JRCs: Joint Routing and Charging Strategy for Logistics Drones. *IEEE Internet Things J.* **2022**, *9*, 21751–21764. [[CrossRef](#)]
25. Cheng, C.; Adulyasak, Y.; Rousseau, L.-M. Drone Routing with Energy Function: Formulation and Exact Algorithm. *Transp. Res. Part B Methodol.* **2020**, *139*, 364–387. [[CrossRef](#)]
26. Metin, A.; Dikici, S. Q-Learning-Based Energy-Aware Route Planning for Electric Vehicles on Real Road Networks with Charging Constraints. In Proceedings of the 2025 Innovations in Intelligent Systems and Applications Conference (ASYU), Bursa, Turkey, 10 September 2025; pp. 1–6.
27. Sun, B.; Xu, Z.; Ranjekar, P.; Wei, M. DS-MGCSTNet: Multi-Graph Convolution Spatial-Temporal Network with Dynamic and Static Graph Fusion for Flight Delay Prediction. *J. Air Transp. Manag.* **2026**, *136*, 103047. [[CrossRef](#)]
28. Bauersfeld, L.; Scaramuzza, D. Range, Endurance, and Optimal Speed Estimates for Multicopters. *IEEE Robot. Autom. Lett.* **2022**, *7*, 2953–2960. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.