Journal of *Marine Science and Engineering*

MDPI

Article

Multi-Sortie UAV Inspection Route Planning in Three-Dimensional Container Yards with Static Obstacles Using a Hybrid SA–ALNS–2OPT Algorithm

Haiyan Wang [1,2,*] and Xueni Tian [2]

[1] State Key Laboratory of Maritime Technology and Safety, Wuhan University of Technology, Wuhan 430063, China
[2] School of Transportation and Logistics Engineering, Wuhan University of Technology, Wuhan 430063, China; xntian@whut.edu.cn
* Correspondence: hywang777@whut.edu.cn

Abstract

To address the challenges posed by a large number of spatially distributed inspection points, dense obstacles, and limited unmanned aerial vehicle (UAV) endurance in container yard inspection, this study investigates a multi-sortie route planning problem for a single UAV operating from a fixed docking station. A mathematical model is formulated to minimize total flight time subject to obstacle avoidance and safety clearance requirements, per-sortie duration limits, and exactly-once visitation of each inspection point. A hybrid metaheuristic integrating simulated annealing (SA), adaptive large neighborhood search (ALNS), and 2-opt, hereafter referred to as SA–ALNS–2OPT, is developed based on a sequence-first, split-second strategy. ALNS optimizes the giant-tour visitation sequence, simulated annealing guides candidate solution acceptance, and 2-opt refines the local route structure. The optimized sequence is subsequently partitioned using a Split procedure to generate endurance-feasible multi-sortie solutions. Computational experiments in a simulated three-dimensional container-yard environment, including repeated trials, ablation studies, benchmark comparisons, and additional methodological validation, show that the proposed method produces high-quality inspection routes while satisfying safety and endurance constraints and maintaining stable solution quality. The proposed approach provides a methodological framework for large-scale UAV inspection planning in automated container yards.

Keywords: container yard; UAV inspection; three-dimensional path planning; multi-sortie route optimization; adaptive large neighborhood search

Check for updates

Academic Editor: Decheng Wan

Received: 25 August 2026
Revised: 20 September 2026
Accepted: 21 September 2026
Published: 24 September 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) license.

1. Introduction

Container yards are essential components of port and inland logistics systems [1], supporting key operations such as container transshipment, temporary storage, and operational scheduling. However, they are typically characterized by densely stacked containers, narrow traffic lanes, and various types of cargo handling equipment. Structural damage, container positioning errors, and other safety hazards may compromise operational efficiency and safety within the yard [2]. Regular inspection of container yards and associated facilities is therefore important for maintaining safe and efficient operations.

Conventional container yard inspection relies mainly on manual patrols, vehicle-mounted inspection systems, and elevated access equipment to examine yard lanes, site fa-

cilities, container stacking conditions, and container positions. However, these approaches are generally associated with low inspection efficiency, high labor intensity, considerable safety risks, and substantial operational costs [3]. Moreover, their inspection coverage is often limited to specific areas, making frequent and comprehensive inspection of large-scale container yards difficult. In contrast, unmanned aerial vehicles (UAVs) offer high maneuverability and flexible flight capabilities [4], enabling rapid aerial inspection over large yard areas and reducing inspection time. UAVs can also inspect in high-risk areas, such as zones with high container stacks and hazardous material storage, thereby reducing personnel exposure to hazardous environments and associated safety risks. Accordingly, UAV-based inspection has emerged as an important approach for intelligent container yard inspection.

Several studies have investigated UAV inspection path planning problems in port and container yard environments. For example, Tang et al. [4] proposed a UAV path planning framework integrating A, *an improved minimum snap method, and the intelligent bidirectional rapidly exploring random trees star (IB-RRT)* algorithm for the inspection of rail-mounted gantry crane tracks in container terminal yards. Tang et al. [5] further developed a UAV motion planning method based on an extended RRT* algorithm for port inspection scenarios to improve trajectory safety and planning efficiency in complex port environments.

UAV inspection in container yard environments is a multi-point routing problem rather than a simple point-to-point flight task. A UAV must visit numerous spatially distributed inspection points while satisfying obstacle avoidance, safety clearance, and per-sortie endurance constraints. Inappropriate route planning may result in excessively long flight paths, unnecessary detours, or violations of per sortie duration limits, thereby reducing inspection efficiency and increasing operational risks. Therefore, effective UAV route planning is important for automated container yard inspection and safe operations.

In recent years, UAV path planning has attracted considerable attention across a wide range of applications, including smart farming [6], urban mapping [7], sensing data acquisition [8], and marine operations [9]. Pourroostaei Ardakani and Cheshmehzangi [6] investigated reinforcement learning-based UAV itinerary planning for remote sensing applications in smart farming. Vo et al. [7] optimized urban LiDAR flight paths using a genetic algorithm and a dual parallel computing framework. Yang and Yoo [8] developed a multi-objective bio-inspired approach for UAV path planning in sensing data acquisition over IoT sensor networks. Yan et al. [9] investigated multi-UAV task allocation and route planning in marine environments using an improved particle swarm optimization algorithm. For infrastructure inspection, Rao et al. [10] investigated UAV inspection route optimization for photovoltaic stations using a genetic algorithm. Luo et al. [11] developed an improved particle swarm optimization method for UAV inspection route planning of ultra-high-voltage transmission lines. Zhu et al. [12] investigated UAV path planning for power line inspection based on ant colony optimization (ACO). Shen et al. [13] formulated a cooperative multi-UAV path planning model for ship emission monitoring in port environments. Pu et al. [14] proposed a dynamic coverage path planning method for UAV formations performing multi-region aerial tasks. Wang et al. [15] proposed a dual-layer path planning framework for plant protection UAVs by integrating deep deterministic policy gradient (DDPG) with an ILA optimization algorithm, enabling dynamic inter-field navigation and optimized coverage planning within irregular farmlands.

Overall, existing studies have established a broad methodological foundation for UAV path planning. However, many of these studies have focused on two-dimensional or relatively general environments, while static three-dimensional container yard environments with dense container stacks remain comparatively underexplored. UAV inspection research has also expanded beyond point-to-point path optimization to integrated routing and scheduling problems involving task assignment, visitation sequencing, and operational

constraints. For example, Zhen et al. [16] investigated UAV inspection routing and scheduling for engineering management, incorporating no-fly zones, monitoring interval time windows, multiple monitoring rounds, and charging decisions.

Compared with general inspection scenarios, container yard inspection involves more restrictive spatial and operational constraints. First, direct Euclidean flight between two inspection points may be infeasible because of obstacle distributions and prescribed safety clearances, requiring detours that increase the actual travel distance. Second, limited UAV endurance may prevent all inspection tasks from being completed within a single flight. The overall mission must therefore be divided into multiple sorties, each departing from and returning to a fixed docking station. For a similar endurance-constrained multi-trip setting, Wang and Wen [17] investigated UAV path planning with an autonomous docking station for geohazard monitoring. Owing to the maximum flight duration constraint, the UAV must return to the docking station for recharging after each trip, while each geohazard site is visited exactly once. Their model prioritizes more urgent sites by maximizing the total collected profit while minimizing the number of trips, and a tabu search-based metaheuristic was developed to solve the problem. In contrast, the present study focuses on a static three-dimensional container yard environment, with the objective of minimizing total flight time while accounting for obstacle-aware feasible flight costs and safety clearance requirements. Accordingly, the problem is formulated as a single-UAV multi-sortie combinatorial optimization problem subject to three-dimensional obstacle constraints, per sortie duration limits, exactly-once visitation of each inspection point, and fixed docking station departure and return requirements.

Recent studies have increasingly addressed UAV inspection path planning in complex three-dimensional environments, supported by advances in three-dimensional modeling and intelligent inspection technologies. Zhao et al. [18] proposed a two-stage optimization method for automated structural inspection in which a genetic algorithm determines inspection viewpoints and a greedy algorithm optimizes camera poses while ensuring inspection coverage and minimizing path length. An et al. [19] developed an automated UAV inspection mission planning method based on a geometric digital twin (GDT) model and voxelized obstacle information (VOI), combining waypoint adjustment with a hybrid GA–A* strategy for collision-avoiding flight path generation. Liu et al. [20] proposed an autonomous flight path planning method for fine-grained construction site inspection that integrates three-dimensional reconstruction with building information modeling (BIM). An improved artificial potential field method combined with a greedy algorithm is used for waypoint generation, while A* and simulated annealing are employed for path optimization. Li et al. [21] developed a coverage path planning method for UAV-based bridge inspection, using a Collaborative Greedy Neighborhood algorithm to reduce viewpoint redundancy and an improved particle swarm optimization algorithm to optimize flight paths while considering collision risk, path length, sharp turns, and rapid altitude changes. Collectively, these studies demonstrate that integrating three-dimensional environmental modeling, inspection viewpoint optimization, obstacle information, and intelligent path planning algorithms can improve the safety and efficiency of UAV inspection in complex three-dimensional environments.

In summary, substantial progress has been made in UAV path planning; however, several research gaps remain in the context of container yard inspection. Although single-UAV multi-trip routing with autonomous docking stations has been investigated in applications such as geohazard monitoring [17], comparatively few studies have jointly considered dense three-dimensional obstacles, safety clearance requirements, and obstacle-aware point-to-point flight costs in container yard environments. To address these gaps, this study investigates a single-UAV multi-sortie inspection route planning problem in a static

three-dimensional container yard environment. An optimization model is formulated to minimize total flight time, and a hybrid metaheuristic framework based on a sequence-first, split-second strategy is developed to balance solution quality, feasibility, and computational efficiency.

The main contributions of this study are summarized as follows:

(1) A single-UAV multi-sortie inspection route planning model with a fixed docking station is formulated for static three-dimensional container yard environments. The model incorporates obstacle avoidance and safety clearance requirements, per sortie duration limits, and exactly-once visitation of each inspection point.
(2) A hybrid metaheuristic integrating simulated annealing (SA), adaptive large neighborhood search (ALNS), and 2-opt, hereafter referred to as SA–ALNS–2OPT, is proposed based on a sequence-first, split-second strategy. Global search and local improvement are performed on a giant-tour visitation sequence, reducing the complexity of direct multi-sortie route optimization.
(3) A hierarchical solution strategy combining surrogate objective search with obstacle-aware feasibility evaluation is developed. To maintain search efficiency, the proposed method employs Split decoding and obstacle-aware route evaluation to generate feasible multi-sortie inspection solutions that satisfy practical flight constraints.
(4) The proposed method is validated in a simulated three-dimensional container yard environment through repeated trials, ablation studies, and benchmark comparisons, evaluating its stability, feasibility, and comparative performance.

The remainder of this paper is organized as follows: Section 2 introduces the UAV platform and its key technical parameters. Section 3 analyzes the inspection problem and formulates the mathematical model. Section 4 presents the proposed SA–ALNS–2OPT hybrid metaheuristic. Section 5 reports the simulation experiments and discusses the results. Finally, Section 6 summarizes the main conclusions and outlines directions for future research.

2. UAV Platform and Technical Specifications

2.1. UAV Platform Types and Applicability Analysis

UAV platforms differ considerably in endurance, flight speed, hovering capability, takeoff and landing requirements, and mission adaptability. Based on their flight configurations, UAVs can generally be categorized as multirotor, fixed-wing, or hybrid wing platforms. For container yard inspection in port environments, UAVs must operate safely in spatially constrained, obstacle-dense environments while maintaining low-speed maneuverability and stable hovering performance.

Multirotor UAVs offer advantages such as vertical takeoff and landing (VTOL), hovering capability, and flexible operation, making them well suited to inspection tasks that require close-range observation and precise data acquisition [22]. They are therefore suitable for applications such as container surface inspection, yard equipment condition monitoring, and port area safety patrols [4,22].

Fixed-wing UAVs offer advantages such as higher flight speeds, longer endurance, and greater operational range, making them suitable for large-area and long-distance monitoring missions [22]. However, they generally require relatively large takeoff and landing areas and lack hovering capability [22]. These characteristics make them less suitable for container yard inspection tasks that require operation in confined spaces and close-range observation of spatially distributed targets.

Hybrid wing UAVs combine the endurance advantages of fixed-wing UAVs with the vertical takeoff and landing capability of multirotor UAVs, providing a balance between efficient long-range cruising and flexible operation in complex environments. However,

their structural and control systems are relatively complex, which may result in higher equipment costs, greater maintenance demands, and increased deployment complexity. For container yard inspection, where the UAV operates within a relatively compact area, frequently hovers for image acquisition, and departs from and returns to a fixed docking station, multirotor UAVs provide greater operational flexibility.

In summary, a multirotor UAV is selected as the operational platform for the container yard inspection task considered in this study. This choice satisfies the requirements for flexible takeoff and landing, stable hovering, and reliable image acquisition in spatially constrained yard environments and provides the operational basis for the subsequent route planning formulation.

The main characteristics and applicability of different UAV types are summarized in Table 1.

Table 1. Characteristics and applicability of different UAV types for container yard inspection.

UAV Type	Main Characteristics	Applicability to Container Yard Inspection
Multirotor UAV	Vertical takeoff and landing; stable hovering; high maneuverability, including at low speeds	Well suited to spatially constrained container yards requiring close-range, detailed inspection of distributed targets
Fixed-wing UAV	Long endurance; high cruising speed; wide area coverage	Suitable for peripheral port areas, large maritime areas, and long-distance inspection missions, but less suitable for close-range, detailed inspection within container yards
Hybrid wing/VTOL fixed-wing UAV	Combines vertical takeoff and landing capability with relatively high cruising efficiency	Suitable for medium- to long-range inspection missions, but with greater system complexity, higher costs, and greater maintenance demands

2.2. UAV Technical Parameters

Based on the operational requirements of container yard inspection, particularly cruising performance, wind resistance, operational safety, and mission feasibility [23], the DJI Mavic 3 Pro was selected as the reference UAV platform for the simulation experiments and inspection mission planning [24]. The main technical specifications of the UAV are summarized in Table 2.

Table 2. Main technical specifications of the UAV platform.

Parameter	Specification
UAV model	Mavic 3 Pro
Takeoff weight	958 g
Folded dimensions	231.1 × 98 × 95.4 mm (excluding propellers)
Unfolded dimensions	347.5 × 290.8 × 107.7 mm (excluding propellers)
Maximum flight time	43 min
Maximum horizontal flight speed	21 m/s (at sea level, no wind)
Maximum ascent speed	8 m/s
Maximum descent speed	6 m/s
Maximum wind resistance	Approximately 12 m/s
Maximum video transmission/remote control range	Approximately 15 km

Although the selected UAV has a maximum horizontal flight speed of 21 m/s, container yard inspection does not require operation at this limit. Instead, greater emphasis is

placed on image acquisition quality, flight stability, and route execution safety. Excessively high flight speeds may increase the likelihood of image blur, compromise trajectory tracking stability, and heighten operational risk in obstacle-dense environments. Accordingly, the UAV cruising speed is set to 12 m/s and used for subsequent flight time calculations.

In addition, the nominal maximum flight time of the Mavic 3 Pro is 43 min, which represents the upper limit under ideal test conditions. In practical inspection missions, however, an adequate safety reserve should be maintained for takeoff, ascent, landing, return-to-base operations, and emergency handling. The safety reserve should also account for obstacle avoidance detours, communication uncertainties, and external environmental disturbances. Accordingly, the maximum allowable flight time for mission planning is conservatively set to 1800 s (30 min). This setting corresponds approximately to a safety margin of $\beta = 0.30$ relative to the nominal maximum flight time. The UAV planning parameters used in the experiments are summarized in Table 3.

Table 3. UAV planning parameters used in the experiments.

Parameter	Value
v_{UAV}	12 m/s
$T_{\mathrm{fly}}^{\max}$	43 min
Safety margin, β	0.30
$T_{\max}$	1800 s (30 min)
s_0	0 s
s_i	2 s

3. Problem Analysis and Model Formulation

3.1. Problem Analysis

Container yard UAV inspection is conducted in a complex, spatially constrained operating environment and therefore differs substantially from path planning in open environments. Within a container yard, densely stacked containers, relatively regular yet spatially confined traffic lanes, and obstacles at essentially fixed locations form a static three-dimensional obstacle environment. In this setting, the UAV is required to sequentially visit multiple inspection points to perform image acquisition, visual observation, or condition monitoring. An inappropriate visitation sequence may cause redundant detours and local backtracking, increasing total flight time and reducing inspection efficiency.

In addition, owing to limited battery capacity and flight safety requirements, a single UAV cannot generally complete all inspection tasks within a single flight. Therefore, the overall mission must be divided into multiple sorties, each departing from and returning to a fixed docking station within the maximum allowable duration. Furthermore, dense container stacks may obstruct the direct connection between two inspection points, so a feasible flight path cannot always be represented by a straight Euclidean line segment. Consequently, obstacle avoidance and safety clearance requirements must be incorporated when determining the feasible travel distance between nodes.

In summary, the UAV route planning problem investigated in this study is a multi-sortie combinatorial optimization problem in a static three-dimensional obstacle environment, subject to per sortie duration limits and fixed docking station departure and return requirements, rather than a conventional shortest path problem. Accordingly, the autonomous docking station S_0 serves as the fixed origin and destination, and a single UAV performs multiple inspection sorties. The objective is to determine a feasible multi-sortie inspection plan that visits each inspection point exactly once; satisfies obstacle avoidance, safety clearance, and per sortie duration constraints; and minimizes total flight time.

3.2. Environment Modeling

A three-dimensional container yard environment was constructed in Unity 2022.3.62f1 (Unity Technologies, San Francisco, CA, USA) based on the typical spatial configuration of container terminals. The model primarily represents the arrangement and stacking patterns of containers. The top and side views of the constructed environment are shown in Figures 1 and 2, respectively.

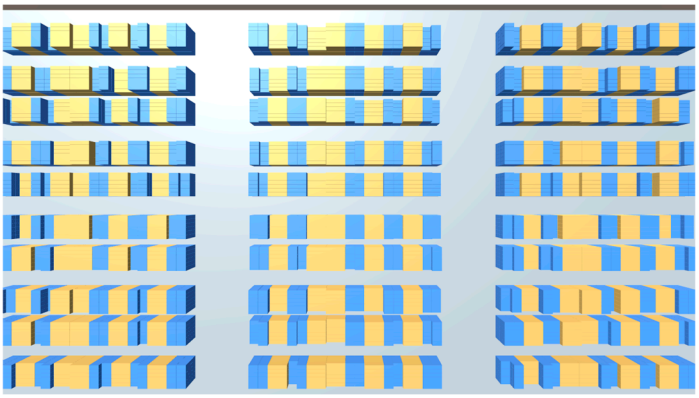

Figure 1. Top view of the three-dimensional container yard model. Blue and yellow blocks represent 20-foot and 40-foot containers, respectively.

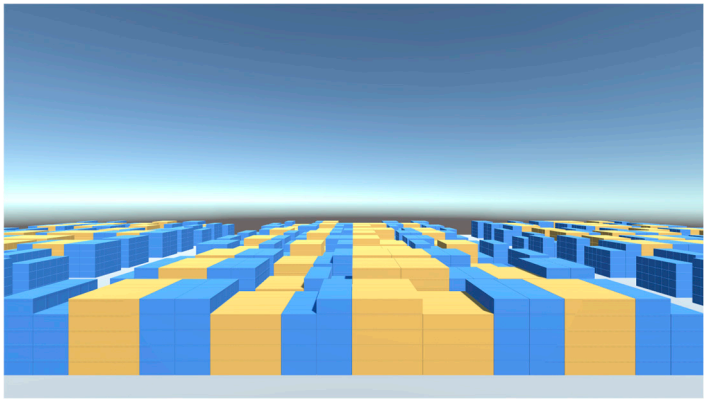

Figure 2. Side view of the three-dimensional container yard model. Blue and yellow blocks represent 20-foot and 40-foot containers, respectively.

Because this study focuses on multi-sortie inspection route optimization rather than high-fidelity physical simulation, nonessential physical details of the environment were simplified while preserving the spatial characteristics relevant to UAV route planning. In particular, the geometric layout of the container stacks and the obstacle configuration were retained to provide the spatial information required for UAV route planning. The resulting three-dimensional model is used for subsequent inspection point generation, feasible flight route estimation, and evaluation of the proposed route planning algorithm.

3.3. Inspection Node Generation

3.3.1. Camera Parameters

For image-based UAV inspection, effective coverage of target surfaces depends on appropriate UAV positions and camera orientations. Accordingly, inspection node generation should account for the camera field of view (FOV), imaging distance, and geometry of the target container surfaces.

For container yard inspection, a standard lens configuration is adopted to limit the image distortion associated with ultra-wide-angle lenses while avoiding the restricted field

of view of telephoto lenses. Let θ_H and θ_V denote the horizontal and vertical field-of-view angles, respectively, and let d denote the effective imaging distance between the UAV and the target surface. In this study, the effective imaging distance is uniformly set to 5 m. The 5 m imaging distance is selected to balance image clarity, coverage range, and flight safety and is applied consistently when placing inspection nodes on the top, side, and end surfaces of containers.

The main camera parameters used in this study are summarized in Table 4.

Table 4. Main camera parameters used for inspection node generation.

Lens Type	Horizontal Field of View (HFOV)	Vertical Field of View (VFOV)	Equivalent Focal Length
Standard lens	approximately 84° (±1°)	approximately 46° (±1°)	24 mm

When the UAV performs nadir imaging of the top surface of a container, the target plane can be regarded as horizontal. If the perpendicular distance between the UAV and the container top surface is d, the horizontal and vertical dimensions of the single-frame coverage footprint can be expressed as

$$L_H = 2dtan\left(\frac{\theta_H}{2}\right), L_V = 2dtan\left(\frac{\theta_V}{2}\right) \quad (1)$$

Accordingly, the single-frame coverage area for top surface imaging is

$$S_{\text{top}} = L_H L_V \quad (2)$$

When the UAV performs frontal imaging of a container side or end surface, the target plane can be regarded as vertical. If the horizontal distance between the UAV and the target surface is also set to d, the coverage width and height follow the same FOV-based geometric relationship. The single-frame coverage area is therefore given by

$$S_{\text{side}} = 2dtan\left(\frac{\theta_H}{2}\right) \times 2dtan\left(\frac{\theta_V}{2}\right) \quad (3)$$

Therefore, under a uniform effective imaging distance, the single-frame coverage areas for top surface and vertical surface imaging can be determined using the same trigonometric relationship. The resulting coverage dimensions are then compared with those of commonly used containers to determine inspection node placement. The relationships between the imaging modes, target surfaces, and coverage dimensions are summarized in Table 5.

Table 5. Relationship between imaging modes, target surfaces, and coverage dimensions.

Imaging Mode	Target Surface	Coverage-Dimension Relationship	Relationship Between FOV and Container Dimensions
Nadir imaging	Top surface; horizontal plane	Length × width in a horizontal plane	θ_H corresponds to the container length direction, and θ_V corresponds to the container width direction
Frontal imaging of side surface	Side surface; vertical plane	Length × height in a vertical plane	θ_H corresponds to the container length direction, and θ_V corresponds to the container height direction
Frontal imaging of end surface	End surface; vertical plane	Width × height in a vertical plane	θ_H corresponds to the container width direction, and θ_V corresponds to the container height direction

3.3.2. Coverage Calculation

For the adopted camera parameters:

$$\tan\left(\frac{84^\circ}{2}\right) = \tan(42^\circ) \approx 0.90040 \tag{4}$$

and

$$\tan\left(\frac{46^\circ}{2}\right) = \tan(23^\circ) \approx 0.42447 \tag{5}$$

For nadir imaging of the container top surface:

$$L_H = 2 \times 5\ \text{m} \times \tan(42^\circ) \approx 9.0040\ \text{m} \tag{6}$$

$$L_V = 2 \times 5\ \text{m} \times \tan(23^\circ) \approx 4.2447\ \text{m} \tag{7}$$

and

$$S_{top} = L_H \cdot L_V \approx 38.22\ \text{m}^2 \tag{8}$$

For frontal imaging of a vertical container surface:

$$L_H = 2 \times 5\ \text{m} \times \tan(42^\circ) \approx 9.0040\ \text{m} \tag{9}$$

$$L_V = 2 \times 5\ \text{m} \times \tan(23^\circ) \approx 4.2447\ \text{m} \tag{10}$$

and

$$S_{side} = L_H \cdot L_V \approx 38.22\ \text{m}^2 \tag{11}$$

Thus, with the adopted imaging distance and camera parameters, the theoretical single-frame coverage area is approximately 38.22 m^2 for both nadir top surface imaging and frontal side or end surface imaging.

3.3.3. Compatibility with Common Container Dimensions

To compare the camera coverage footprint with common container dimensions, 20-foot standard containers and 40-foot high cube containers are considered. Based on the theoretical single-frame coverage area of approximately 38.22 m^2, the area equivalent coverage ratio, defined as the single-frame coverage area divided by the target surface area, is calculated for each container surface and summarized in Table 6.

Table 6. Area-based compatibility between UAV single-frame coverage and common container surface dimensions.

Imaging Mode	Target Surface	Container Type	Surface Dimensions	Surface Area	Area Equivalent Coverage Ratio
Nadir imaging	Top surface	20-foot standard container	6.06 m × 2.44 m	≈14.8 m^2	≈2.6
		40-foot high cube container	12.19 m × 2.44 m	≈29.7 m^2	≈1.3
Frontal imaging	Side surface	20-foot standard container	6.06 m × 2.59 m	≈15.70 m^2	≈2.4
		40-foot high cube container	12.19 m × 2.90 m	≈35.35 m^2	≈1.1
	End surface	20-foot standard container	2.44 m × 2.59 m	≈6.32 m^2	≈6.0
		40-foot high cube container	2.44 m × 2.90 m	≈7.08 m^2	≈5.4

The area equivalent coverage ratio provides only a theoretical surface area comparison and does not imply complete single-frame coverage of the target surface. Complete single-frame coverage also depends on the geometric dimensions and orientation of the target surface, as well as its relative position within the camera coverage footprint.

3.4. Model Formulation

3.4.1. Basic Assumptions and Route Representation

Consider a container yard with n inspection points and one autonomous UAV docking station, denoted by S_0. The following assumptions are adopted for the multi-sortie route planning problem:

(1) The obstacles in the container yard remain stationary throughout the inspection mission; therefore, the operating environment is modeled as a static three-dimensional obstacle environment.
(2) A single UAV performs all inspection tasks, and each sortie departs from and returns to the docking station S_0.
(3) Each inspection point is visited exactly once and is assigned a constant service time of $s_i = 2$ s, $i \in N$.
(4) Charging, maintenance, and data transmission times at the docking station are excluded from the optimization model; accordingly, $s_0 = 0$.
(5) The UAV flies at a prescribed cruising speed v_{UAV}, and the travel time between any pair of nodes is computed from the corresponding feasible flight route length.

Let

$$R = (R_1, \ldots, R_h)$$

denote a feasible solution to the multi-sortie UAV inspection route planning problem, where $h \leq H$ is the actual number of sorties used. The route of the r-th sortie is represented as

$$R_r = \left(0, i_1, i_2, \ldots, i_{\mu_r}, 0\right),$$

where node 0 denotes the autonomous docking station S_0, and μ_r denotes the number of inspection points visited during sortie r.

Let

$$W_r = \left\{i_1, i_2, \ldots, i_{\mu_r}\right\}$$

denote the set of inspection points visited during sortie r. The complete visitation requirement is expressed as

$$N = \bigcup_{r=1}^{h} W_r,$$

where the sets W_r assigned to different sorties are pairwise disjoint.

3.4.2. Sets, Parameters, and Decision Variables

The sets and basic notation used in the mathematical model are defined as follows:

- $N = \{1, 2, \ldots, n\}$: set of inspection points;
- S_0: autonomous docking station used for takeoff, landing, and charging;
- $V = \{0\} \cup N$: set of all nodes, where node 0 represents S_0;
- $K = \{1, 2, \ldots, H\}$: set of potential sorties, where H denotes the predefined maximum number of sorties;
- $O = \{1, 2, \ldots, \mid O \mid\}$: set of obstacles.

The model parameters are defined as follows:

- $p_i = (x_i, y_i, z_i)$, $i \in V$: three-dimensional coordinates of node i;

- $v_{\mathrm{UAV}} = 12$ m/s: prescribed UAV cruising speed;
- $s_i = 2$ s, $i \in N$: service time at inspection point i;
- $s_0 = 0$: service time at the docking station;
- $T_{\max} = 1800$ s: maximum allowable duration of each sortie;
- $d_{\mathrm{safe}} = 0.30$ m: prescribed minimum safety clearance distance used to construct the safety-expanded obstacle regions;
- $d_{\mathrm{lift}} = 0.30$ m: additional vertical clearance margin applied above the top of a safety-expanded obstacle during overflight;
- $\widetilde{O}_k$, $k \in O$: safety-expanded three-dimensional region of obstacle k;
- $\mathrm{Line}(i, j)$: straight line segment connecting nodes i and j;
- $\widetilde{z}_k^{\max}$: maximum z-coordinate of the safety-expanded obstacle region $\widetilde{O}_k$;
- z_{ij}^{fly}: overflight altitude used when the direct path between nodes i and j is obstructed;
- $w_{ij}^{(1)}, w_{ij}^{(2)}$: intermediate detour waypoints generated for the obstacle-avoiding route from node i to node j;
- L_{ij}: feasible flight route length from node i to node j;
- $t_{ij} = L_{ij}/v_{\mathrm{UAV}}$: travel time from node i to node j.

The binary decision variables are defined as follows:

$$x_{ij}^r = \begin{cases} 1, & \text{if the UAV travels from node } i \text{ to node } j \text{ during sortie } r, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$y_i^r = \begin{cases} 1, & \text{if inspection point } i \text{ is visited during sortie } r, \\ 0, & \text{otherwise.} \end{cases}$$

The auxiliary variable u_i^r, $i \in N$, $r \in K$, is introduced for Miller–Tucker–Zemlin (MTZ) subtour elimination. Self-loops are prohibited by imposing

$$x_{ii}^r = 0, \forall i \in V, \ \forall r \in K.$$

3.4.3. Feasible Flight Route and Travel Time Calculation

Because densely stacked containers form three-dimensional obstacle regions within the yard, the straight line connection between two nodes is not always feasible. The UAV is assumed to fly at a constant cruising speed v_{UAV}, while obstacle avoidance and safety clearance requirements are accounted for when calculating the pairwise feasible flight-route length L_{ij}.

The minimum safety clearance d_{safe} is applied by expanding each physical obstacle in three-dimensional space to form a safety-expanded obstacle region. For obstructed direct connections, an additional vertical clearance margin d_{lift} is imposed when constructing the overflight segment. A feasible flight route between nodes i and j must neither intersect any safety-expanded obstacle region nor leave the predefined flyable space.

Let $\mathcal{P}_{ij}$ denote the set of candidate feasible routes generated between nodes i and j. The pairwise feasible flight route length is defined as

$$L_{ij} = \min_{P \in \mathcal{P}_{ij}} \mathrm{len}(P), \forall i, j \in V, i \neq j \tag{12}$$

where $\mathrm{len}(P)$ denotes the geometric length of route P.

If $\mathcal{P}_{ij} = \varnothing$, no feasible connection exists between nodes i and j, and the corresponding arc is regarded as unreachable and excluded from the feasible arc set used in the subsequent route optimization.

The corresponding point-to-point travel time is calculated as

$$t_{ij} = \frac{L_{ij}}{v_{\mathrm{UAV}}}, \forall i, j \in V, i \neq j \tag{13}$$

If the straight line segment connecting nodes i and j lies entirely within the flyable space and does not intersect any safety-expanded obstacle region, the feasible route reduces to a direct flight path, and its length is given by the Euclidean distance

$$L_{ij} = \| \boldsymbol{p}_i - \boldsymbol{p}_j \|_2 = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{14}$$

Here, p_i and p_j denote the three-dimensional coordinates of nodes i and j, respectively.

For the line-of-sight test, the straight line segment between nodes i and j is represented as

$$\mathrm{Line}(i, j) = \{ p_i + \lambda (p_j - p_i) \mid 0 \leq \lambda \leq 1 \}.$$

For obstacle $k \in O$, let $\widetilde{O}_k$ denote its safety-expanded region defined using the prescribed clearance d_{safe}. The direct line of sight between nodes i and j is regarded as collision-free with respect to the obstacles if

$$\mathrm{Line}(i, j) \cap \widetilde{O}_k = \emptyset, \forall k \in O.$$

Otherwise, the direct connection is regarded as obstructed.

In the implementation, if the straight line segment $\mathrm{Line}(i, j)$ is feasible, L_{ij} is given by the Euclidean distance. If the direct segment is obstructed, a piecewise obstacle-avoiding route consisting of vertical ascent, horizontal traversal above the obstacle region, and vertical descent is constructed, and its total geometric length defines L_{ij}.

When the direct segment is obstructed, let

$$I_{ij} = \left\{ k \in O \mid \mathrm{Line}(i|j) \cap \widetilde{O}_k \neq \emptyset \right\}$$

denote the set of obstacles intersecting the direct path. Let $\widetilde{z}_k^{max}$ denote the maximum z-coordinate of the corresponding safety-expanded obstacle region. The overflight altitude is determined as

$$z_{ij}^{\mathrm{fly}} = max \left\{ z_i, z_j, \underset{k \in I_{ij}}{max} \widetilde{z}_k^{\max} + d_{\mathrm{lift}} \right\},$$

where d_{lift} denotes the additional vertical clearance margin above the top of the safety-expanded obstacle region.

Two intermediate detour waypoints are then generated as

$$w_{ij}^{(1)} = \left(x_i, y_i, z_{ij}^{\mathrm{fly}} \right), w_{ij}^{(2)} = \left(x_j, y_j, z_{ij}^{\mathrm{fly}} \right).$$

Accordingly, the obstacle-avoiding route is constructed as

$$p_i \rightarrow w_{ij}^{(1)} \rightarrow w_{ij}^{(2)} \rightarrow p_j,$$

corresponding to vertical ascent, horizontal traversal above the intersecting obstacle regions, and vertical descent, respectively.

The subsequent combinatorial model and solution algorithm therefore use the precomputed L_{ij} and corresponding t_{ij} values, with obstacle avoidance and safety clearance requirements embedded in the pairwise feasible route evaluation rather than represented by explicit geometric constraints in the upper-level routing model.

3.4.4. Objective Function

The objective of the proposed model is to minimize the total UAV travel time over all sorties, thereby completing the entire inspection mission as efficiently as possible. The objective function is formulated as

$$minZ = \sum_{r=1}^{H} \sum_{i \in V} \sum_{j \in V} t_{ij} x_{ij}^{r} \tag{15}$$

Here, t_{ij} denotes the travel time associated with the feasible flight route from node i to node j, and $x_{ij}^{r} = 1$ indicates that the corresponding arc is traversed during sortie r.

The objective function considers only UAV travel time, while the service time s_i is included in the sortie feasibility constraint. Because each inspection point is visited exactly once and has the same service time, $s_i = 2$ s, the total service time is identical for all feasible solutions. Accordingly, the fixed inspection service time is intentionally omitted from the objective function in Equation (15), since its inclusion would not alter the optimal routing decisions.

3.4.5. Constraints

The mathematical model is subject to the following constraints:

$$s.t. \sum_{j=1}^{n} x_{0j}^{r} = \sum_{i=1}^{n} x_{i0}^{r} \leq 1, \forall r \in K \tag{16}$$

$$\sum_{r=1}^{H} y_{i}^{r} = 1, \forall i \in N \tag{17}$$

$$\sum_{j=0, j \neq i}^{n} x_{ji}^{r} = \sum_{j=0, j \neq i}^{n} x_{ij}^{r} = y_{i}^{r}, \forall i \in N, \forall r \in K \tag{18}$$

$$u_{i}^{r} - u_{j}^{r} + (n) x_{ij}^{r} \leq n - 1, \forall i \in N, \forall j \in N, i \neq j, \forall r \in K \tag{19}$$

$$1 \leq u_{i}^{r} \leq n, \forall i \in N, \forall r \in K \tag{20}$$

$$\sum_{i \in V} \sum_{\substack{j \in V \\ j \neq i}} t_{ij} x_{ij}^{r} + \sum_{i \in N} s_i y_{i}^{r} \leq T_{\max}, \forall r \in K \tag{21}$$

$$\begin{cases} y_{i}^{r} \in \{0,1\}, & \forall i \in N, \forall r \in K, \\ x_{ij}^{r} \in \{0,1\}, & \forall i, j \in V, i \neq j, \forall r \in K. \end{cases} \tag{22}$$

Constraint (16) enforces equal departure and return flows at the docking station, with at most one departure and one return for each sortie. The inequality allows for unused sortie indices to remain empty.

Constraint (17) guarantees that each inspection point is assigned to exactly one sortie and is therefore visited exactly once over the entire inspection mission, preventing both repeated visits and task omissions.

Constraint (18) imposes flow conservation and links the routing variables x_{ij}^{r} to the visit variables y_{i}^{r}. If inspection point i is assigned to sortie r, exactly one incoming arc and one outgoing arc must be associated with that inspection point in the corresponding sortie.

Constraints (19) and (20) are the Miller–Tucker–Zemlin (MTZ) subtour elimination constraints. Constraint (19) prevents the formation of disconnected cycles that do not include the docking station, while Constraint (20) specifies the allowable range of the auxiliary variables u_{i}^{r}. Together with the preceding route connectivity constraints, these MTZ constraints ensure that the inspection points visited during each sortie form a single connected route associated with the docking station rather than isolated subtours.

Constraint (21) imposes the maximum sortie duration limit. For each sortie, the sum of the travel time and inspection service time must not exceed the maximum allowable duration $T_{\max}$.

Finally, Constraint (22) specifies the binary domains of the decision variables x_{ij}^r and y_i^r. Here, x_{ij}^r indicates whether the arc from node i to node j is selected during sortie r, whereas y_i^r indicates whether inspection point i is visited during sortie r. Together, these binary variables characterize the multi-sortie inspection problem as a 0–1 combinatorial optimization model.

3.4.6. Relationship Between the Mathematical Model and the Solution Framework

The mathematical formulation above defines the optimization objective and constraint structure of the multi-sortie UAV inspection problem investigated in this study. In the computational implementation, however, the resulting multi-sortie 0–1 combinatorial optimization model is not solved directly using an exact optimization method. Instead, a two-stage approximate solution framework is developed in which a giant tour sequence is optimized first and subsequently split into multiple sorties.

During the main search stage, a giant tour sequence containing all inspection points is used as the solution representation. The length of the closed Euclidean tour is used as a surrogate objective to improve search efficiency. During the decoding stage, the resulting sequence is reevaluated using the obstacle-aware feasible flight route lengths and corresponding travel times. A sequential Split procedure is then applied to partition the giant tour into multiple feasible sorties, each satisfying the maximum sortie duration constraint.

Consequently, the SA–ALNS–2OPT algorithm presented in Section 4 serves as an efficient approximate solution method for the mathematical model formulated above. A computationally inexpensive surrogate objective is used during the main search stage, whereas obstacle-aware route evaluation and sortie feasibility verification are performed during the decoding stage to ensure the feasibility of the resulting multi-sortie solution.

4. Algorithm Design

The single-UAV multi-sortie inspection route planning problem considered in this study involves inspection point sequencing, obstacle-aware travel costs, per sortie duration limits, and fixed docking station departure and return requirements. The problem is NP-hard because it contains the traveling salesman problem as a special case when only one sortie is allowed and the sortie duration constraint is nonbinding. As the number of inspection points increases, obtaining provably optimal solutions with exact methods can become computationally prohibitive within practical computation times. To balance solution quality and computational efficiency, this study develops a hybrid metaheuristic that integrates adaptive large neighborhood search (ALNS), simulated annealing (SA), and 2-opt local improvement, hereafter referred to as SA–ALNS–2OPT.

4.1. Overall Algorithmic Framework

Unlike methods that directly search over multiple sortie routes, the proposed algorithm adopts a giant tour visitation sequence as the core solution representation during the main search stage. Specifically, a global visitation sequence π, containing all inspection points is first constructed, with the docking station S_0 treated as the fixed origin and destination. A subsequent Split procedure is then applied to partition the sequence into a feasible multi-sortie solution

$$R = (R_1, \ldots, R_h)$$

such that each resulting sortie satisfies the maximum sortie duration constraint. This sequence-first, split-second strategy [25] reduces search complexity by separating visitation sequence optimization from sortie partitioning.

Because evaluating obstacle-aware point-to-point routes in a three-dimensional environment is computationally expensive, the algorithm combines surrogate objective search

with obstacle-aware feasible route evaluation. During the main search stage, the closed Euclidean tour length is used as the surrogate objective for candidate evaluation. Candidate sequences generated during the ALNS search are evaluated and accepted on the basis of this surrogate objective. After the main search terminates, the best giant tour visitation sequence is reevaluated using the obstacle-aware feasible flight route lengths L_{ij} and the corresponding travel times t_{ij}. The Split procedure is then applied to decode the sequence into a feasible multi-sortie inspection solution in which each sortie satisfies the maximum sortie duration constraint.

Overall, the basic procedure of the proposed SA–ALNS–2OPT algorithm is summarized in Figure 3 and described as follows:

(1) Construct an initial giant tour visitation sequence $\pi^{(0)}$.
(2) At each iteration, select a destroy operator and a repair operator according to their adaptive weights, and apply large-neighborhood reconstruction to the current sequence to generate a candidate sequence π'.
(3) Periodically apply 2-opt local improvement to π' to reduce route crossings and local redundancies.
(4) Evaluate the candidate sequence using the surrogate objective and determine whether it is accepted using the simulated annealing acceptance criterion.
(5) Update the weights of the destroy and repair operators according to their search performance.
(6) After the iterative search terminates, the best giant tour sequence is reevaluated using the obstacle-aware feasible flight route lengths L_{ij} and the corresponding travel times t_{ij}. The Split procedure is then applied to obtain a feasible multi-sortie inspection solution satisfying the maximum sortie duration constraint T_{max}, which is subsequently used for route statistics and visualization.

4.2. Initial Solution Generation

The initial solution determines the starting point of the metaheuristic search and can influence its early search behavior. To enable rapid initialization while maintaining good scalability, this study adopts a random initialization strategy based on a giant tour visitation sequence to generate the initial solution for the main search stage. Multi-sortie feasibility is subsequently ensured by the Split decoding procedure after the main search terminates.

4.2.1. Construction of the Giant Tour Visitation Sequence

A giant tour visitation sequence π, containing all inspection points and starting from the docking station S_0, is constructed. For implementation simplicity and computational efficiency, a random permutation strategy is adopted to generate the initial giant tour sequence.

Specifically, node 0, corresponding to the docking station S_0, is fixed as the starting node, while all inspection points in the set N are randomly permuted. The initial sequence is therefore represented as

$$\pi^{(0)} = [0, \pi_1, \pi_2, \ldots, \pi_n],$$

where $(\pi_1, \pi_2, \ldots, \pi_n)$ is a random permutation of the inspection point set N, and $\pi_\ell \in N$ denotes the inspection point at position ℓ in the giant tour sequence.

The initial sequence is required to satisfy only two basic conditions: each inspection point appears exactly once, and the starting node is fixed at the docking station. The maximum sortie duration constraint is not explicitly enforced during initialization; instead, the corresponding feasibility is guaranteed by the subsequent Split decoding procedure.

The random initialization strategy is adopted for two main reasons. First, it is simple to implement and computationally inexpensive, making it suitable for the rapid initialization

of large-scale problem instances. Second, random permutation provides a diverse initial state for the subsequent ALNS process and avoids excessive dependence on a fixed initial structure, thereby supporting broader exploration of the solution space.

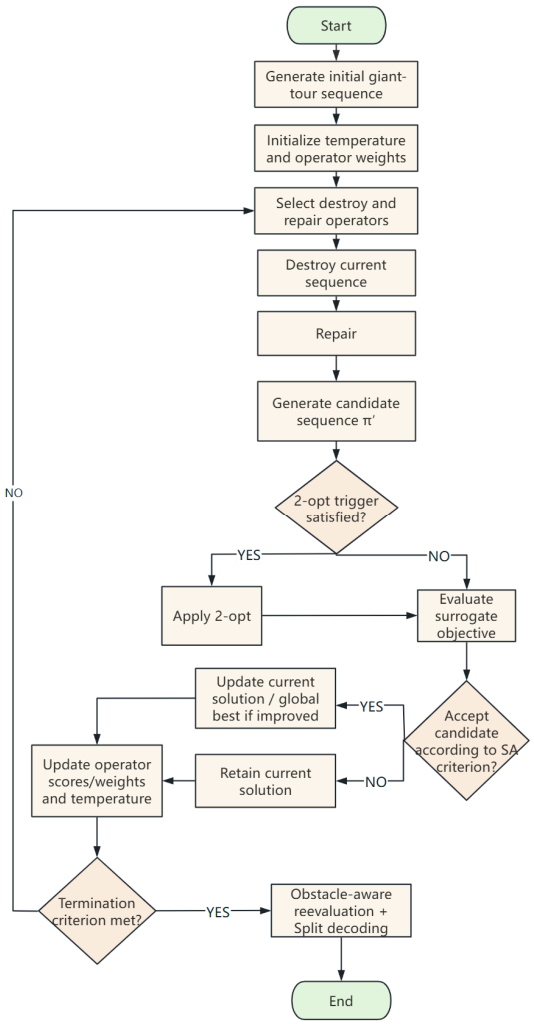

Figure 3. Flowchart of the proposed SA–ALNS–2OPT algorithm.

4.2.2. Hierarchical Objective Evaluation Mechanism

In a three-dimensional obstacle environment, point-to-point feasible route evaluation requires obstacle intersection checks against the safety-expanded obstacle regions and, when necessary, construction of a piecewise obstacle-avoiding route. If obstacle-aware feasible route evaluation were performed for every candidate solution at each ALNS iteration, the computational cost would increase substantially, thereby reducing the search efficiency of the algorithm.

To balance the computational efficiency of the main search with the practical feasibility of the final solution, this study adopts a hierarchical objective evaluation mechanism that combines surrogate objective search with obstacle-aware feasibility evaluation.

During the main search stage, the closed Euclidean length of the giant tour visitation sequence is used as a computationally efficient surrogate objective. Let the current sequence be

$$\pi = [0, \pi_1, \pi_2, \ldots, \pi_n].$$

By defining $\pi_0 = 0$ and $\pi_{n+1} = 0$, with $\pi_\ell \in N$ for $\ell = 1, \ldots, n$, the surrogate objective function is expressed as

$$F_p(\pi) = \sum_{\ell=0}^{n} d^E_{\pi_\ell, \pi_{\ell+1}},$$

where $d^E_{\pi_\ell, \pi_{\ell+1}}$ denotes the Euclidean distance between two consecutive nodes in the sequence. The definition $\pi_{n+1} = 0$ represents the return from the final inspection point to the docking station. This surrogate objective is primarily used for candidate sequence comparison, simulated annealing acceptance decisions, and local search evaluation within the main ALNS loop.

During the final evaluation stage, the best giant tour visitation sequence is reevaluated using the obstacle-aware feasible flight route lengths L_{ij} and the corresponding travel times t_{ij} defined in Section 3.4.3. The Split decoding procedure is then applied to partition the sequence into multiple feasible sorties, each satisfying the maximum allowable sortie duration constraint.

Accordingly, the proposed method improves search efficiency by employing a computationally inexpensive surrogate objective during the main search stage, while obstacle-aware flight cost evaluation and Split decoding are subsequently used to ensure the feasibility of the final multi-sortie solution. This hierarchical evaluation mechanism allows for the final inspection solution to account for obstacle avoidance, safety clearance, and sortie duration requirements without requiring obstacle-aware route evaluation for every candidate sequence during the main search. To assess whether the surrogate preserves the relative quality ordering of candidate solutions, the consistency between the Euclidean and obstacle-aware objectives is evaluated empirically in Section 5.4.1.

4.3. Optimization Algorithm Design

After constructing the initial giant tour visitation sequence and establishing the hierarchical objective evaluation mechanism, an ALNS-based optimization algorithm is developed to optimize the visitation sequence. The algorithm iteratively applies destroy and repair operators to reconstruct the current sequence within a large neighborhood and incorporates a simulated annealing acceptance criterion, adaptive operator weight updating, and 2-opt local improvement to efficiently explore the solution space. Overall, the optimization process aims to progressively improve both the global visitation order and the local route structure of the giant tour sequence while maintaining computational efficiency, thereby providing a high-quality sequence for subsequent obstacle-aware evaluation and multi-sortie decoding.

4.3.1. ALNS Framework

Large neighborhood search (LNS) was originally proposed by Shaw [26]. Its core idea is to iteratively apply destroy and repair operations to the current solution. Specifically, a subset of nodes is first removed from the current solution to partially disrupt the existing route structure, after which the removed nodes are reinserted to construct a new solution. This mechanism enables the search to explore large neighborhoods through substantial modifications to the solution structure.

Compared with conventional local search methods based on small-scale exchanges or local adjustments, this broader search scope helps LNS escape from local optima. Ropke and Pisinger [27] subsequently introduced an adaptive operator selection mechanism into the LNS framework, leading to adaptive large neighborhood search (ALNS). In ALNS, the selection probabilities of different destroy and repair operators are dynamically adjusted according to their historical performance, enabling the search to balance exploration and exploitation. Pisinger and Ropke [28] further demonstrated that adaptive operator selection allows for better-performing operators to be selected more frequently while preserving sufficient diversification during the search process. Owing to its flexibility and robustness,

ALNS has been widely applied to routing and scheduling problems in transportation and logistics and has also been successfully extended to UAV-related routing problems, including monitoring, delivery, pickup-and-delivery, and recharging operations [29–32].

Considering the characteristics of the single-UAV multi-sortie inspection route planning problem investigated in this study, including a large number of inspection points, spatially distributed task locations, and the strong influence of the global visitation order on route quality, ALNS is adopted as the main search framework for optimizing the giant tour visitation sequence [33]. Within the proposed SA–ALNS–2OPT framework, ALNS performs large-scale reconstruction of the visitation sequence, simulated annealing enhances search diversification by allowing for the probabilistic acceptance of non-improving solutions, and 2-opt local improvement refines candidate sequences through cost-reducing edge exchanges.

Based on this framework, three destroy operators and two repair operators are designed. These operators are combined with the simulated annealing acceptance criterion, adaptive operator weight updating mechanism, and 2-opt local improvement to form the complete optimization procedure.

4.3.2. Destroy Operators

Let the current visitation sequence be

$$\pi = [0, \pi_1, \pi_2, \ldots, \pi_n].$$

Given a removal ratio ρ, the number of inspection points to be removed is defined as

$$q = \max\{1, \mathrm{round}(n\rho)\},$$

where the docking station S_0 is excluded from the removal operation. Three complementary destroy operators are adopted in this study.

(1) Random Removal.
The random removal operator randomly selects q inspection points from all non-S_0 nodes and removes them from the current sequence. Because this operator is independent of the existing route structure, it introduces relatively strong random perturbations. It therefore promotes broader exploration of the solution space and helps reduce the risk of premature convergence.

(2) Worst Removal.
For any inspection point $i \in N$ in the visitation sequence, let $\mathrm{pred}(i)$ and $\mathrm{succ}(i)$ denote its predecessor and successor, respectively. The reduction in edge cost resulting from the removal of node i is defined as

$$\Delta_i = c_{\mathrm{pred}(i),i} + c_{i,\mathrm{succ}(i)} - c_{\mathrm{pred}(i),\mathrm{succ}(i)}.$$

where c_{ij} denotes the edge cost from node i to node j. Under surrogate objective evaluation, c_{ij} is represented by the Euclidean distance, whereas under obstacle-aware evaluation, c_{ij} is represented by the corresponding feasible flight route length L_{ij}.

The operator removes the q nodes with the highest Δ_i values. It therefore targets nodes with large marginal contributions to the current route cost, thereby disrupting locally expensive route structures.

(3) Segment Removal.
The segment removal operator randomly selects a contiguous subsequence from the non-S_0 portion of the visitation sequence and removes the entire segment. Compared with random removal, this operator introduces a stronger perturbation to the local

route structure and facilitates the reorganization of subpaths. It is particularly useful for disrupting relatively stable but suboptimal local structures.

In summary, the three destroy operators play complementary roles in the optimization process. Random removal promotes global exploration, worst removal targets high-cost local structures, and segment removal facilitates the structural reorganization of local subpaths. Their combined use improves the flexibility of the search process throughout the optimization procedure.

4.3.3. Repair Operators

After a subset of inspection points has been removed during the destroy stage, a repair operator is applied to reinsert the removed nodes into the partial sequence, thereby restoring a complete giant tour visitation sequence. The repair process reconstructs the visitation sequence based on the remaining route structure by selecting insertion positions according to incremental edge costs. Two repair operators are adopted in this study.

(1) Greedy Insertion.
For each removed node u, all candidate insertion positions (a, b) in the current partial sequence are examined. The incremental cost of inserting node u between nodes a and b is calculated as

$$\Delta(a, u, b) = c_{a,u} + c_{u,b} - c_{a,b} ,$$

where c_{ij} denotes the edge cost from node i to node j under the current evaluation mode. Node u is then inserted at the position that minimizes $\Delta(a, u, b)$. This strategy is straightforward to implement and incurs relatively low computational cost, making it suitable for rapid iterative search in large-scale problem instances.

(2) Best Insertion.
For all combinations of currently uninserted nodes and candidate insertion positions, the node–position pair with the minimum incremental cost is selected at each insertion step. The selected node is inserted at the corresponding position, after which the partial sequence and the set of remaining uninserted nodes are updated. This procedure is repeated until all removed nodes have been reinserted.

Compared with greedy insertion, the best insertion operator performs a more comprehensive evaluation of the available node–position combinations at each insertion step and can therefore provide more refined local reconstruction, although at the expense of greater computational effort.

To prevent excessive computational overhead when a large number of nodes are removed, a threshold mechanism is introduced. When the number of removed nodes exceeds the predefined threshold q_{th}, greedy insertion is used instead of best insertion to control the computational cost of the repair process. In the implementation, $q_{th} = 40$.

4.4. Simulated Annealing Acceptance Criterion and Adaptive Operator Weight Updating

4.4.1. Simulated Annealing Acceptance Criterion

Although the destroy–repair operations iteratively generate new candidate sequences, accepting only improving candidates may cause the search to become trapped in local optima. To enhance search diversification and improve the ability to escape from local optima, a simulated annealing (SA) acceptance criterion is incorporated into the ALNS framework.

Let the current sequence and candidate sequence be denoted by π and π', respectively, with corresponding surrogate objective values $F_p(\pi)$ and $F_p(\pi')$. If

$$F_p(\pi') \leq F_p(\pi),$$

the candidate sequence is accepted unconditionally. Otherwise, the non-improving candidate is accepted with probability

$$P(\text{accept}) = \exp\left(-\frac{F_p(\pi') - F_p(\pi)}{T}\right),$$

where T denotes the current temperature. After each iteration, the temperature is updated according to the cooling rate α:

$$T \leftarrow \alpha T.$$

This mechanism allows for non-improving candidate sequences to be accepted with a relatively high probability at higher temperatures, thereby promoting exploration and reducing the likelihood of premature convergence. As the temperature gradually decreases, the probability of accepting non-improving candidates declines accordingly, causing the search to progressively shift from diversification toward intensification. In this manner, the simulated annealing acceptance mechanism complements the adaptive large neighborhood search by providing a controlled means of escaping local optima while gradually focusing the search on promising regions of the solution space.

4.4.2. Adaptive Operator Weight Updating

To enable the algorithm to dynamically adjust the selection probabilities of different operators according to their performance during the search, separate performance-driven adaptive weight-updating mechanisms are employed for the destroy and repair operators.

Following the adaptive operator scoring and weight-updating strategy of Pisinger and Ropke [28], the selected operators are assigned different scores according to the quality of the candidate sequence. Specifically, the score depends on whether the candidate sequence improves the global best sequence, improves the current sequence, or is accepted by the simulated annealing criterion despite being non-improving.

Suppose that one destroy operator and one repair operator are selected in a given iteration. Their scores are assigned according to the following rules:

(1) If the candidate sequence improves the global best sequence, the selected operators receive a score of

$$\sigma_1 = 3.$$

(2) If the candidate sequence improves the current sequence but does not improve the global best sequence, the selected operators receive a score of

$$\sigma_2 = 2.$$

(3) If the candidate sequence does not improve the current sequence but is accepted according to the SA acceptance criterion, the selected operators receive a score of

$$\sigma_3 = 1.$$

(4) If the candidate sequence is rejected, the selected operators receive a score of 0.

Within each weight update period of length L, the scores obtained by each destroy and repair operator are accumulated separately. Let S_g denote the cumulative score of operator g during the current update period. Its normalized performance score is defined as

$$p_g = \frac{S_g}{\sum_{g'} S_{g'}},$$

where the summation is taken over operators of the same type, i.e., destroy operators and repair operators are normalized separately. If the cumulative score of all operators

of a given type is zero during an update period, the corresponding operator weights remain unchanged.

The operator weight is then updated using exponential smoothing:

$$w_g \leftarrow max\{w_{min}, (1-\beta)w_g + \beta p_g\},$$

where $\beta \in (0,1)$ is the smoothing coefficient and w_{min} denotes the minimum allowable operator weight, preventing an operator from being completely excluded because of poor short-term performance.

This adaptive mechanism serves two main purposes. First, operators that perform well during the search are assigned greater influence in subsequent operator selection decisions, allowing for the algorithm to exploit more effective search strategies. Second, the weight update process retains historical performance information through exponential smoothing while maintaining a minimum weight for each operator. This mechanism preserves search diversity and reduces the likelihood that the search becomes dominated by a single operator or restricted to a narrow local search pattern.

4.5. Local Improvement and Multi-Sortie Splitting

4.5.1. 2-Opt Local Improvement

Although ALNS can effectively modify the overall route structure through large-neighborhood reconstruction, candidate sequences generated after the repair operation may still contain edge crossings, local backtracking, or other short-range redundancies. To further improve route quality, a 2-opt local search procedure is periodically applied to candidate sequences for local refinement.

Suppose that two edges in the current sequence are (a,b) and (c,d). If replacing them with (a,c) and (b,d) reduces the corresponding edge cost, i.e.,:

$$c_{a,c} + c_{b,d} < c_{a,b} + c_{c,d},$$

the subsequence between the selected edges is reversed to perform the 2-opt exchange, thereby improving the local route structure.

A first improvement strategy is adopted in this study. Once an improving exchange is identified, it is immediately applied, thereby reducing the computational effort required for local search. During the main ALNS loop, a candidate set-based 2-opt procedure is periodically applied to the candidate sequence. After the main search terminates, a stronger post-optimization stage applies both random 2-opt and candidate set-based 2-opt to the best sequence. Random 2-opt samples edge pairs randomly to broaden the range of local exchanges examined, whereas candidate set-based 2-opt selects candidate edges using a nearest neighbor list to focus the search on potentially promising exchanges.

As a classical local search operator for combinatorial routing problems, 2-opt is computationally simple and effective for improving local route structures. It is therefore well suited to the local intensification of the giant tour visitation sequence considered in this study.

4.5.2. Multi-Sortie Splitting and Feasible Solution Construction

The best giant tour visitation sequence π^* is decoded into a multi-sortie inspection solution subject to the maximum sortie duration constraint:

$$R = (R_1, R_2, \ldots, R_h).$$

The splitting process starts from the docking station S_0 and sequentially scans the inspection points according to their order in the best giant tour sequence π^*. Let i denote

the last node currently included in the sortie, j denote the next inspection point to be considered, and curTime denote the cumulative duration of the current sortie up to node i.

When node j is considered for inclusion in the current sortie, both the travel time t_{ij} from node i to node j and the service time s_j at node j must be taken into account. In addition, sufficient time must remain for the UAV to return to the docking station after visiting node j. Accordingly, the feasibility condition is defined as

$$\text{curTime} + t_{ij} + s_j + t_{j0} \leq T_{\max},$$

where t_{j0} denotes the travel time from node j back to the docking station S_0, and $T_{\max}$ represents the maximum allowable duration of a single sortie.

If the above condition is satisfied, node j is included in the current sortie, and the cumulative sortie duration is updated as

$$\text{curTime} \leftarrow \text{curTime} + t_{ij} + s_j.$$

Otherwise, the current sortie is terminated by returning to the docking station S_0. A new sortie is then initiated from S_0, with node j considered as the first inspection point of the new sortie, after which the sequential scanning process continues.

The travel time t_{ij} used during the splitting stage is calculated from the obstacle-aware feasible flight route length L_{ij} defined in Section 3.4.3. At this stage, the best giant tour visitation sequence is evaluated using the corresponding obstacle-aware travel costs, and the maximum sortie duration constraint is incorporated into the node insertion test during the splitting process. After splitting, the resulting sorties are validated to confirm that each sortie starts from and returns to S_0 and that its total duration does not exceed T_{max}. In this way, the resulting multi-sortie solution accounts for both obstacle-aware flight costs and endurance limitations in the three-dimensional container yard environment.

Through the sequential splitting and validation procedure described above, the optimized global visitation sequence is transformed into a feasible multi-sortie inspection plan. The resulting plan specifies the visitation sequence of each sortie, the feasible flight-route length of each segment, the corresponding travel and service times, and the total duration of each sortie, thereby providing the required input for subsequent simulation analysis and three-dimensional visualization.

The sequential Split adopted in the proposed framework is a heuristic decoding procedure. Although it ensures feasibility with respect to the maximum sortie duration constraint, it does not necessarily minimize the total obstacle-aware flight distance for a fixed giant tour visitation sequence. To quantify the approximation introduced by this decoding step, the sequential Greedy Split is compared with an exact dynamic programming (DP) Split in Section 5.4.2.

4.6. Termination Criteria

To balance search effectiveness and computational efficiency, a combined termination mechanism incorporating a maximum iteration limit and adaptive stagnation detection is adopted in this study. Specifically, a maximum number of iterations, $K_{\max}$, is specified to prevent the search from continuing indefinitely. In addition, after a predefined minimum number of iterations has been completed, the relative improvement in the global best surrogate objective value is periodically evaluated over fixed iteration intervals. If the improvement rate remains below a specified threshold for several consecutive checks, the search is considered to have entered a stable stage and is terminated early.

Let $F_{\text{best}}^{(m)}$ denote the global best surrogate objective value recorded at the m-th convergence check. The relative improvement rate between two consecutive checks is defined as

$$\delta^{(m)} = \frac{F_{\text{best}}^{(m-1)} - F_{\text{best}}^{(m)}}{\max\left(\left|F_{best}^{(m-1)}\right|, \varepsilon\right)},$$

where ε is a sufficiently small positive constant introduced to avoid division by zero. If

$$\delta^{(m)} < \tau$$

is satisfied for several consecutive checks, the search is regarded as having reached a stable stage, where τ denotes the predefined improvement threshold.

The proposed termination mechanism offers two main advantages. First, the predefined minimum iteration requirement ensures a baseline level of search effort, while the maximum iteration limit provides a deterministic upper bound on the number of iterations. Second, the adaptive stagnation criterion avoids unnecessary computation when only marginal improvements are obtained during the later stages of the search. In the implementation used in this study, adaptive stopping is activated after 1500 iterations, the relative improvement rate is evaluated every 50 iterations, and the search terminates when the improvement rate remains below 5×10^{-4} for five consecutive checks, subject to a maximum of 3000 iterations.

5. Simulation Experiments and Results

The performance of the proposed SA–ALNS–2OPT algorithm was evaluated in a simulated three-dimensional container yard environment, where a single UAV was required to perform multiple inspection sorties. The experiments were designed to assess route quality, solution stability, and computational efficiency under the specified operational constraints. The SA–ALNS–2OPT algorithm was implemented in MATLAB R2025b (MathWorks, Natick, MA, USA) and executed on a computer equipped with an Intel® Core™ Ultra 5 225H processor (Intel Corporation, Santa Clara, CA, USA) and 32 GB of RAM.

5.1. Repeated Trials and Stability Analysis of the Main Algorithm

To evaluate the stability of the proposed SA–ALNS–2OPT algorithm under stochastic search conditions, ten independent runs with different random seeds were conducted on the same test instance containing 5164 inspection points and one fixed docking station. Except for the random seed, all algorithmic and experimental parameters were kept identical across the ten runs.

These repeated trials primarily assess stochastic stability on the same large-scale instance rather than generalizability across different container yard layouts. To provide additional evidence at a different problem scale and under contrasting local obstacle complexity conditions, two fixed 500-task validation instances are further examined in Section 5.4.1.

The main results of the repeated experiments are summarized in Table 7, including the stopping iteration, surrogate closed tour length, obstacle-aware feasible closed tour length, pure flight time, and total computational runtime. Figures 4 and 5 illustrate the obstacle-aware giant tour route and the resulting three-dimensional multi-sortie inspection routes, respectively.

Table 7. Results of ten repeated SA–ALNS–2OPT experiments with different random seeds.

Random Seed	Stopping Iteration	Surrogate Closed Tour Length (m)	Obstacle-Aware Closed Tour Length (m)	Pure Flight Time (s)	Pure Flight Time (min)	Total Runtime (s)
1	1900	21,243.35	25,321.44	2110.12	35.17	74,506.69
2	1750	21,389.04	25,492.11	2124.34	35.41	25,428.67
3	1850	21,222.39	25,314.84	2109.57	35.16	66,449.80
4	1750	21,171.52	25,300.72	2108.39	35.14	27,839.85
5	1750	21,228.06	25,333.05	2111.09	35.18	65,321.14
6	1800	21,513.52	25,612.83	2134.4	35.57	63,298.06
7	1800	21,325.97	25,414.32	2117.86	35.30	63,639.73
8	1750	21,270.24	25,320.65	2110.05	35.17	66,639.78
9	1750	21,517.08	25,628.06	2135.67	35.59	56,821.83
10	1750	21,324.72	25,366.90	2113.91	35.23	59,211.89

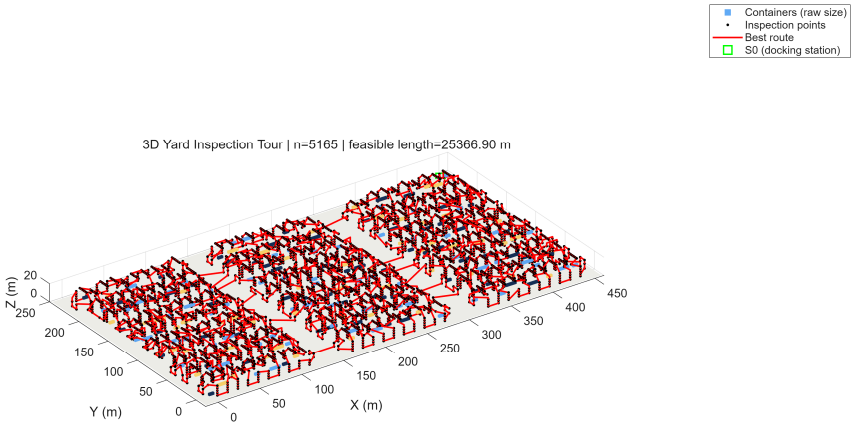

Figure 4. Feasible giant tour route generated by SA–ALNS–2OPT.

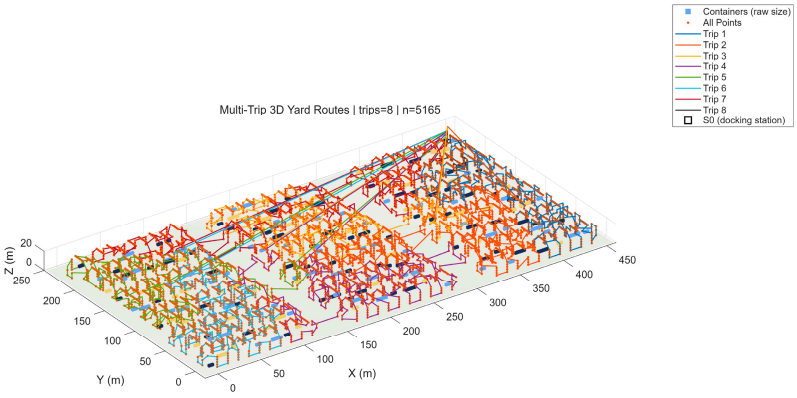

Figure 5. Three-dimensional multi-sortie inspection routes.

In terms of solution quality, the mean surrogate closed tour length obtained over the ten independent runs was 21,320.59 m, with a standard deviation of 119.95 m and a coefficient of variation (CV) of 0.56%. The mean obstacle-aware closed tour length was 25,410.49 m, with a standard deviation of 124.74 m and a CV of 0.49%. Correspondingly, the mean pure flight time was 2117.54 s (35.29 min), with a standard deviation of 10.39 s and a CV of 0.49%. The low CV values indicate that the route quality metrics varied only slightly across different random seeds, demonstrating good stability and repeatability of the proposed algorithm.

Among the ten runs, the shortest obstacle-aware closed tour length was 25,300.72 m, whereas the longest was 25,628.06 m, resulting in a difference of 327.34 m. This relatively narrow range, together with the low CV, further indicates that the proposed algorithm exhibits limited sensitivity to stochastic variations and consistently produces solutions of similar quality.

In contrast, computational runtime varied considerably across the ten runs, with a mean of 56,915.74 s, a standard deviation of 16,645.38 s, and a CV of 29.25%. Compared with the low CVs of the route quality metrics, this result indicates substantially greater variability in computational effort than in solution quality. Because the mean runtime remains substantial despite the stable route quality, Section 5.4.3 further analyzes the computational cost associated with obstacle-aware evaluation and the subsequent decoding procedure.

Overall, the SA–ALNS–2OPT algorithm produced consistent route quality results across the ten independent runs, demonstrating good stability and repeatability for the large-scale three-dimensional UAV inspection instance with 5164 inspection points and one fixed docking station considered in this study.

5.2. Ablation Study

To evaluate the contribution of the key algorithmic components to the overall performance of the proposed method, four algorithm configurations, denoted as Groups A, B, C, and D, were tested on the same problem instance. For each configuration, ten independent runs with different random seeds were conducted. Except for the algorithmic component removed in each ablation configuration, all other algorithmic and experimental settings were kept identical to ensure a fair comparison. Table 8 summarizes the statistical results in terms of stopping iteration, surrogate closed tour length, obstacle-aware closed tour length, pure flight time, and total optimization runtime.

Table 8. Comparison of repeated experimental results under different algorithm configurations.

Group	Algorithm Configuration	Mean Stopping Iteration	Stopping Iteration CV (%)	Mean Surrogate Closed-Tour Length (m)	Mean Obstacle-Aware Closed-Tour Length (m)	Obstacle-Aware Length CV (%)	Mean Pure Flight Time (s)	Mean Total Runtime (s)
A	SA–ALNS–2OPT	1785	2.97	21,320.59	25,410.49	0.49	2117.54	56,915.74
B	Without SA	1515	12.26	21,477.44	25,577.53	0.61	2131.46	56,028.58
C	Without 2-opt	1565	18.63	22,112.77	26,295.50	0.60	2191.29	67,463.65
D	Without adaptive operator-weight updating	1595	11.20	21,271.45	25,370.08	0.49	2114.17	72,815.76

Group A yielded a mean obstacle-aware closed tour length of 25,410.49 m, a mean pure flight time of 2117.54 s, and a mean stopping iteration of 1785. Its stopping iteration CV was 2.97%, the lowest among the four configurations, while the obstacle-aware closed tour length CV was 0.49%, indicating stable stopping behavior and route quality across independent runs.

When the simulated annealing acceptance mechanism was removed, the mean obstacle-aware closed tour length of Group B increased to 25,577.53 m, which was 167.04 m longer than that of Group A, while the mean pure flight time increased by 13.92 s. The mean stopping iteration decreased to 1515, whereas the stopping iteration CV increased from 2.97% to 12.26%. These results indicate that SA contributes to route quality and more consistent convergence behavior, supporting its role in reducing premature convergence.

The removal of 2-opt produced the most pronounced deterioration in solution quality. Group C yielded a mean obstacle-aware closed tour length of 26,295.50 m, representing an increase of 885.01 m relative to Group A, while the mean pure flight time increased by

73.75 s. Its mean surrogate closed tour length also increased substantially to 22,112.77 m. Although the obstacle-aware length CV remained relatively low at 0.60%, the mean route quality was clearly inferior to that of the other configurations. These results indicate that 2-opt plays an important role in the local refinement of the giant tour visitation sequence.

After adaptive operator weight updating was disabled, Group D achieved a mean obstacle-aware closed tour length of 25,370.08 m, which was 40.41 m shorter than that of Group A, while the mean pure flight time decreased by 3.37 s. However, its mean total runtime increased to 72,815.76 s, compared with 56,915.74 s for Group A, and its stopping iteration CV increased from 2.97% to 11.20%. Meanwhile, the obstacle-aware length CV remained identical to that of Group A at 0.49%. These results indicate that adaptive operator weight updating did not improve the mean route length in the present experiments; however, the complete configuration achieved comparable route quality with a lower mean time and more consistent stopping behavior.

Further comparison shows that removing either SA or 2-opt deteriorated the mean route quality, with the removal of 2-opt producing the largest degradation. Among the tested components, 2-opt had the largest effect on mean route quality. The more moderate effect of removing SA is consistent with its role in maintaining search diversification and reducing premature convergence. By contrast, disabling adaptive operator weight updating did not deteriorate mean route quality but increased computational runtime and stopping iteration variability, indicating that its main benefit in the present experiments was improved search efficiency and consistency rather than shorter mean routes.

Overall, considering solution quality, search stability, and computational cost jointly, the complete SA–ALNS–2OPT configuration in Group A provided the most balanced performance for the large-scale three-dimensional inspection instance considered in this study.

5.3. Comparison with Benchmark Algorithms

The whale optimization algorithm (WOA) and its variants have been applied to three-dimensional UAV path planning problems in complex environments. For example, Chen et al. [34] developed an adaptive multi-population cooperative WOA and validated its performance in several three-dimensional UAV path planning scenarios. To further evaluate the performance of the proposed method for the large-scale three-dimensional inspection routing problem considered in this study, four benchmark algorithms—standard ALNS, WOA [35], DBO [36], and PSO [37]—were implemented and evaluated on the same problem instance using consistent route cost definitions and the same final obstacle-aware evaluation procedure.

Because WOA, DBO, and PSO were originally designed for continuous search spaces, a common priority-based random key representation was adopted to adapt them to the large-scale permutation problem. Let D denote the number of inspection nodes to be sequenced, excluding the fixed docking station S_0. Each candidate solution was represented by a D-dimensional continuous vector $y = (y_1, y_2, \ldots, y_D)$, where $y_i \in [0, 1]$, and each component was permanently associated with one inspection node and interpreted as its visitation priority. WOA, DBO, and PSO applied their respective continuous space update rules to these priority values, after which the updated variables were bounded within $[0, 1]$. The discretization step was performed by rank-based decoding: the priority values were sorted in ascending order, and the corresponding node order defined the inspection-node permutation. Because each inspection node was represented exactly once, the decoding procedure produced a complete permutation without duplicated or omitted nodes and required neither direct integer rounding nor an additional permutation repair procedure. The same encoding and decoding procedure was used for all three continuous baselines to maintain a consistent mapping scheme.

The comparative results are summarized in Table 9. Among the evaluated algorithms, SA–ALNS–2OPT achieved the shortest mean obstacle-aware closed tour length of 25,410.49 m, with a corresponding mean pure flight time of 2117.54 s (35.29 min).

Table 9. Comparison of repeated experimental results for different algorithms.

Algorithm	Mean Euclidean Closed Tour Length (m)	Mean Obstacle-Aware Closed Tour Length (m)	Difference from SA–ALNS–2OPT (m)	Mean Pure Flight Time (s)	Difference from SA–ALNS–2OPT (s)
SA–ALNS–2OPT	21,320.59	25,410.49	—	2117.54	—
ALNS	22,211.31	26,415.88	1005.39	2201.32	83.78
WOA	45,163.99	55,418.62	30,008.13	4618.22	2500.68
DBO	45,161.32	55,411.84	30,001.35	4617.65	2500.11
PSO	63,790.33	77,444.68	52,034.19	6453.72	4336.18

To improve the transparency and reproducibility of the benchmark comparison, the principal parameter settings and search budgets adopted for the five algorithms are summarized in Table 10.

Table 10. Main parameter settings and search budgets of the compared algorithms.

Algorithm	Representation	Population/Search Structure	Search Budget	Main Parameters	Stopping Rule
SA–ALNS–2OPT	Direct permutation	Single incumbent solution	Up to 3000 outer iterations with embedded 2-opt	removal ratio = 0.06; $T_0 = 80$; cooling rate = 0.995; $K = 15$; candidate 2-opt every 5 iterations; 1500 trials; max. 80 improvements	Adaptive stopping after 1500 iterations; checked every 50 iterations; stop after 5 consecutive checks below 5×10^{-4}; max. 3000
ALNS	Direct permutation	Single incumbent solution	Up to 3000 outer iterations	removal ratio = 0.06; adaptive destroy/repair weights; no SA; no 2-opt	Adaptive stopping after 1000 iterations; checked every 50 iterations; stop after 5 consecutive checks below 5×10^{-4}; max. 3000
WOA	Random key + rank decoding	Population = 30	500 iterations; 15,030 full tour fitness evaluations	$a : 2 \rightarrow 0$; spiral parameter $b = 1$; no NN-seeded individual	Fixed 500 iterations
DBO	Random key + rank decoding	Population = 30	500 iterations; 15,030 full tour fitness evaluations	$k = 0.1$; $b = 0.3$; $S = 0.5$; subgroup sizes = 6/6/7/11	Fixed 500 iterations
PSO	Random key + rank decoding	Population = 30	500 iterations; 15,030 full tour fitness evaluations	$w = 0.9$; $c_1 = c_2 = 2.0$; velocity bound $[-, 11]$	Fixed 500 iterations

Note: Each algorithm was independently executed 10 times. All algorithms were evaluated on the same inspection instance using the same route cost definitions and final obstacle-aware evaluation procedure. Because the algorithms have different search structures, the adopted search budgets are algorithm-specific and are not strictly identical in terms of elementary objective function evaluations.

Compared with standard ALNS, the proposed algorithm reduced the mean obstacle-aware closed tour length by 1005.39 m and the mean pure flight time by 83.78 s. The mean optimization runtime was also lower for SA–ALNS–2OPT (56,915.74 s) than for standard ALNS (71,020.35 s). Thus, under the current experimental settings, the proposed method achieved better route quality with a lower mean runtime than standard ALNS.

The route quality advantage of the proposed algorithm was more pronounced when compared with WOA and DBO. The mean obstacle-aware closed tour lengths obtained by WOA and DBO were 55,418.62 m and 55,411.84 m, respectively, with corresponding mean pure flight times of 4618.22 s and 4617.65 s. These values were substantially higher than those obtained by SA–ALNS–2OPT, indicating considerably longer inspection routes and greater flight time requirements. Although WOA and DBO required substantially less optimization time than SA–ALNS–2OPT, they produced considerably longer routes under the adopted random key encoding and experimental settings, indicating a clear trade-off between runtime and route quality in the present comparison.

The PSO results showed an even larger route quality gap. PSO produced a mean obstacle-aware closed tour length of 77,444.68 m and a mean pure flight time of 6453.72 s (107.56 min), representing increases of 52,034.19 m and 4336.18 s, respectively, relative to SA–ALNS–2OPT. Among the compared algorithms, PSO produced the longest mean route and the highest mean pure flight time under the adopted random key encoding and experimental settings.

Overall, under the experimental settings considered in this study, none of the four benchmark algorithms achieved mean route quality comparable to that of SA–ALNS–2OPT. Among the benchmark algorithms, standard ALNS exhibited the smallest route quality gap, suggesting that the ALNS framework is well suited to the routing structure of the problem considered in this study. Although WOA, DBO, and PSO required considerably less computational time, their resulting routes were substantially longer. Thus, SA–ALNS–2OPT achieved the best mean route quality in the present comparison and a lower mean runtime than standard ALNS, whereas WOA, DBO, and PSO achieved shorter runtimes at the expense of substantially longer routes.

5.4. Additional Methodological Validation and Computational Analysis

To provide additional evidence on the methodological validity and computational characteristics of the proposed framework, this section examines performance at a smaller problem scale, the consistency between the Euclidean surrogate and obstacle-aware evaluation, the approximation introduced by the sequential Greedy Split, and the computational cost of the evaluation and decoding procedures.

For the smaller-scale assessment, two fixed 500-inspection-point validation instances were considered to examine the performance of SA–ALNS–2OPT under contrasting local obstacle complexity conditions. Sparse-500 and Dense-500 were extracted from relatively sparse and relatively dense regions, respectively, within the same underlying simulated yard. Three independent runs were conducted for each instance. The resulting performance statistics, together with the full-scale results for reference, are summarized in Table 11.

The results in Table 11 show that relatively stable final route quality values were obtained for both 500-inspection-point validation instances across the three independent runs, whereas computational runtime exhibited substantially greater variation. This pattern is qualitatively consistent with the full-scale repeated experiments, in which route quality was also considerably more stable than runtime.

Table 11. Performance of SA–ALNS–2OPT across the full-scale and 500-inspection-point validation instances (mean ± standard deviation).

Instance	Independent Runs	Euclidean Closed-Tour Length (m)	Obstacle-Aware Closed Tour Length (m)	Stopping Iteration	Runtime (s)
Full-scale	10	21,320.59 ± 119.95	25,410.49 ± 124.74	1785.00 ± 52.97	56,915.74 ± 16,645.38
Sparse-500	3	4873.90 ± 134.27	5483.47 ± 138.84	2233.33 ± 596.52	4650.67 ± 5133.40
Dense-500	3	4832.36 ± 120.52	5345.67 ± 131.35	2283.33 ± 472.58	5829.13 ± 2577.07

Note: Sparse-500 and Dense-500 were extracted from regions with contrasting local obstacle complexity within the same underlying simulated yard. They therefore provide supplementary evidence across problem scales and local obstacle conditions, but do not constitute independent yard geometries. Absolute route lengths should not be directly compared across instances with different numbers and spatial distributions of inspection points.

5.4.1. Validation of the Euclidean Surrogate Objective

Because the main SA–ALNS–2OPT search employs Euclidean closed tour length as a computationally efficient surrogate, whereas the final solution is evaluated using obstacle-aware flight distances, an additional validation was conducted to quantify the consistency between the two objectives.

First, the ten independent full-scale runs on the 5164-inspection-point instance were analyzed. The Euclidean and obstacle-aware closed tour lengths showed a Pearson correlation coefficient of $r = 0.9768$ and a Spearman rank correlation coefficient of $\rho = 0.9515$, indicating a strong positive association at the final solution level.

To further examine candidate-level consistency, three independent SA–ALNS–2OPT runs were conducted on each of the Sparse-500 and Dense-500 instances introduced above. During these experiments, the original Euclidean-guided search procedure was retained without modification. Every 10 iterations, the generated candidate tour was additionally evaluated using a precomputed obstacle-aware distance matrix. These additional evaluations were used only for diagnostic recording and did not affect candidate acceptance, operator selection, adaptive weights, or best-solution updates.

An obstacle-aware re-ranking analysis was also conducted within the same sampled candidate sets. For each run, the sampled candidates were ranked separately according to the Euclidean surrogate and the obstacle-aware closed tour length. The obstacle-aware re-ranking gain was defined as the relative reduction in obstacle-aware closed tour length achieved by selecting the obstacle-aware best candidate rather than the Euclidean surrogate best candidate from the same sampled set. The correlation and re-ranking results are summarized in Table 12.

Table 12. Summary of the Euclidean surrogate validation results.

Validation Set	Pearson Correlation, r	Spearman Rank Correlation, ρ	Mean OA Re-Ranking Gain (%)	Maximum Gain (%)
Full-scale final solutions	0.9768	0.9515	—	—
Sparse-500 candidate samples	0.999971 ± 0.000005	0.9467 ± 0.0455	0.0692 ± 0.0626	0.1326
Dense-500 candidate samples	0.999948 ± 0.000028	0.9784 ± 0.0291	0.0440 ± 0.0763	0.1321

Note: The full-scale correlation was calculated from the final solutions of 10 independent runs. For Sparse-500 and Dense-500, the reported candidate-level correlation coefficients and mean re-ranking gains are presented as mean ± standard deviation over three independent runs. The maximum OA re-ranking gain denotes the largest value observed across the three runs for each instance. Obstacle-aware candidate evaluations were used only for diagnostic analysis and did not affect the search process.

As shown in Table 12, the Euclidean and obstacle-aware objectives exhibited strong positive association at both the full-scale final solution level and the candidate level. In particular, the candidate-level Pearson coefficients were close to unity for both Sparse-500 and Dense-500, while the Spearman coefficients also indicated strong rank consistency. The obstacle-aware re-ranking gains remained small across the sampled candidate pools. These results indicate that the Euclidean surrogate does not reproduce the obstacle-aware objective exactly but preserves the relative ordering of candidate solutions to a high degree under the tested static yard conditions.

5.4.2. Greedy Split Versus Exact Dynamic Programming Split

To evaluate the approximation introduced by the sequential Split procedure, the ten independently obtained full-scale giant tour sequences were repartitioned using both the original Greedy Split and an exact dynamic programming (DP) Split. The DP procedure minimizes total obstacle-aware flight distance for a fixed visitation order subject to the same 1800 s sortie duration constraint; therefore, its exactness applies only to the partitioning of a given giant tour and not to the overall routing problem. The comparison results for the ten full-scale giant tours are summarized in Table 13.

Table 13. Comparison of the sequential Greedy Split and exact dynamic programming Split.

Split Method	Mean Total Flight Distance (km)	Standard Deviation (km)	Feasible Partitions
Sequential Greedy Split	29.271	0.292	10/10
Exact DP Split	28.698	0.195	10/10

Note: Across the ten full-scale giant tours, the mean Greedy-to-DP relative gap was $1.997 \pm 0.928\%$, with a range of 0.645–3.624%. The reported distances represent the obstacle-aware total flight distance of the decoded multi-sortie solutions, including repeated departures from and returns to S_0, and therefore differ from the pre-Split obstacle-aware closed tour lengths reported in Table 7.

As shown in Table 13, the exact DP Split consistently reduced the total obstacle-aware flight distance relative to the sequential Greedy Split while maintaining feasibility under the same 1800 s sortie duration constraint. The exact DP Split reduced the total obstacle-aware flight distance by 572.65 ± 264.79 m on average, corresponding to a mean Greedy-to-DP relative gap of 1.997%. This modest but nonzero gap confirms that the sequential Greedy Split is an approximate decoding strategy rather than an optimal partitioning method. The Greedy Split is retained in the present framework for consistency with the original algorithmic design, while exact or improved partitioning represents a possible enhancement of the decoding stage.

5.4.3. Computational Cost Analysis

The full-scale SA–ALNS–2OPT runs required a mean optimization time of 56,915.74 s (approximately 15.81 h), indicating that computational cost remains a major limitation of the proposed framework. To determine whether the final obstacle-aware evaluation and decoding stages account for a substantial share of this cost, an additional postprocessing timing experiment was conducted using the ten previously obtained full-scale giant tours without rerunning the optimization procedure. The component-wise timing results are summarized in Table 14.

Table 14 shows that the final obstacle-aware evaluation and decoding stages account for only a very small fraction of the mean full-scale optimization time. The combined obstacle-aware evaluation, depot leg preprocessing, and Greedy Split required 14.158 s on average, corresponding to approximately 0.025% of the mean full-scale runtime. This indicates that the computational burden is concentrated primarily in the iterative search stage

rather than in final obstacle-aware evaluation or Split decoding. Because candidate-level 2-opt is embedded within the main ALNS loop, the historical timing records do not permit a finer retrospective separation of neighborhood reconstruction and local search costs.

Table 14. Computational cost breakdown of the postprocessing components.

Component	Mean Runtime (s)	Remark
Obstacle model construction	0.316	One-time preprocessing
Euclidean closed tour evaluation	0.00092	Negligible relative to full optimization
Obstacle-aware closed tour evaluation	0.22	Standalone final route evaluation
Obstacle-aware depot leg preprocessing	13.938	Largest measured postprocessing component
Sequential Greedy Split	0.00056	Negligible
OA evaluation + depot preprocessing + Greedy Split	14.158	$\approx$ 0.025% of mean full-scale optimization runtime

Note: The timing analysis was conducted using the ten previously obtained full-scale giant tours without rerunning the optimization process. Candidate-level 2-opt is embedded within the main ALNS loop and therefore cannot be separated retrospectively from the historical full-scale timing records.

The adaptive stopping criterion partially mitigates this computational burden by terminating the search when further improvement becomes marginal. Nevertheless, the present method should be regarded primarily as an offline planning framework for static or slowly varying container yard environments rather than as a real-time replanning method. Further acceleration through parallel neighborhood evaluation, incremental cost updates, more efficient local search, and implementation-level optimization remains an important direction for future work.

6. Conclusions

This study investigated a single-UAV multi-sortie inspection route planning problem with a fixed docking station in a static three-dimensional container yard environment. The problem was motivated by the need to efficiently organize inspection routes under conditions involving a large number of spatially distributed inspection points, dense obstacles, and limited UAV endurance. To address these challenges, obstacle avoidance, the maximum allowable duration of each sortie, and the requirement that each inspection point be visited exactly once were incorporated into a unified optimization framework. A hybrid SA–ALNS–2OPT metaheuristic based on a sequence-first, split-second strategy was then developed to generate feasible and high-quality multi-sortie inspection solutions.

The simulation results support the effectiveness of the proposed framework for the tested static container yard scenarios. The proposed SA–ALNS–2OPT framework generated high-quality inspection routes while satisfying obstacle avoidance, safety clearance, and endurance constraints. Repeated experiments showed stable route quality performance across different random seeds, while the ablation study demonstrated the complementary roles of simulated annealing, 2-opt local improvement, and adaptive operator-weight updating within the hybrid framework. Benchmark comparisons further showed that the proposed method achieved substantially better mean route quality than standard ALNS, WOA, DBO, and PSO under the experimental settings considered in this study. Overall, the results indicate that the sequence-first, split-second strategy, combined with adaptive large neighborhood search, simulated annealing, and 2-opt local improvement, provides an effective solution framework for large-scale single-UAV multi-sortie inspection route planning in complex three-dimensional container yard environments.

The contributions of this study can be summarized from three perspectives. From a modeling perspective, the container yard UAV inspection problem was formulated as a multi-sortie combinatorial optimization problem incorporating three-dimensional obsta-

cle constraints, fixed docking station departure and return requirements, and per sortie duration limits, thereby providing a unified formulation for UAV inspection route planning under these combined operational constraints. From a methodological perspective, a hierarchical solution mechanism combining surrogate objective search with obstacle-aware feasibility evaluation was developed. This mechanism reduces the need for repeated obstacle-aware evaluations during the main search, while obstacle-aware evaluation, Split decoding, and final validation are used to obtain a feasible multi-sortie solution. The resulting hierarchical structure may also provide a useful reference for route optimization problems involving similarly complex constraints. From an application perspective, the proposed approach provides methodological support for UAV route generation and mission scheduling in automated container yard inspection and may support future efforts to improve inspection efficiency, reduce personnel exposure to hazardous operating environments, and advance intelligent port operation and maintenance.

Several limitations of the present study should be acknowledged. First, the investigated scenario is restricted to a static three-dimensional container yard environment, and the effects of dynamic objects, such as vehicles, personnel, and cargo handling equipment, on route execution are not considered. Second, only the single-UAV multi-sortie inspection problem is investigated, while cooperative task allocation, route coordination, and resource scheduling for multiple UAVs remain beyond the scope of this study. Third, the UAV flight process is represented using several idealized assumptions, and practical operational factors, including wind disturbances, communication delays, perception errors, and flight control deviations, have not yet been systematically incorporated. In addition, the proposed method has been validated through simulation; field experiments in real container yard environments are therefore needed to assess its practical performance.

Future research can proceed in several directions. First, real-time environmental perception and online replanning mechanisms can be incorporated to accommodate dynamic operating environments and improve the adaptability and robustness of the proposed approach. At the mission management level, runtime prediction models may also be explored to estimate mission completion time and support adaptive resource allocation, following recent work on predictive formal modelling in human–swarm operations [38]. Second, the problem can be extended to cooperative multi-UAV inspection, with particular attention to task allocation, inter-UAV collision avoidance, and charging scheduling. Third, route planning can be further integrated with inspection viewpoint placement, image acquisition quality constraints, and target recognition requirements to establish an integrated route–perception optimization framework. For example, Liu et al. [39] proposed the PD-YOLOv11 model to address multiscale detection, small-component localization, and overlapping target detection in UAV-based power transmission tower inspection imagery. By improving feature extraction, feature fusion, and the detection head architecture, their method enhanced object detection performance in complex inspection environments. Building on this direction, recognition-related imaging requirements, such as viewing distance, viewing angle, and target recognizability, could be incorporated into the joint optimization of inspection viewpoints and UAV routes. Finally, future studies should incorporate real port operational data and conduct field experiments to systematically evaluate the real-time responsiveness, operational stability, and engineering deployability of the extended framework.

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

Funding: This research received no external funding.

Data Availability Statement: The data and code that support the findings of this study are available on request from the second author.

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

References

1. Sun, H. Digital architecture of port container yards. *Port Sci. Technol.* **2022**, *10*, 1–5, 24. (In Chinese)
2. Li, J.; Li, L.; Song, H.; Wu, C.; Lin, Y. Research on detection algorithm of operators and vehicles in container yard for port area security. *J. Saf. Sci. Technol.* **2025**, *21*, 158–164. (In Chinese)
3. De Arriba López, V.; Maboudi, M.; Achanccaray, P.; Gerke, M. Automatic non-destructive UAV-based structural health monitoring of steel container cranes. *Appl. Geomat.* **2024**, *16*, 125–145. [CrossRef]
4. Tang, G.; Wang, C.; Zhang, Z.; Men, S. UAV path planning for container terminal yard inspection in a port environment. *J. Mar. Sci. Eng.* **2024**, *12*, 128. [CrossRef]
5. Tang, G.; Liu, P.; Hou, Z.; Claramunt, C.; Zhou, P. Motion planning of UAV for port inspection based on extended RRT* algorithm. *J. Mar. Sci. Eng.* **2023**, *11*, 702. [CrossRef]
6. Pourroostaei Ardakani, S.; Cheshmehzangi, A. Reinforcement learning-enabled UAV itinerary planning for remote sensing applications in smart farming. *Telecom* **2021**, *2*, 255–270. [CrossRef]
7. Vo, A.V.; Laefer, D.F.; Byrne, J. Optimizing urban LiDAR flight path planning using a genetic algorithm and a dual parallel computing framework. *Remote Sens.* **2021**, *13*, 4437. [CrossRef]
8. Yang, Q.; Yoo, S.-J. Optimal UAV path planning: Sensing data acquisition over IoT sensor networks using multi-objective bio-inspired algorithms. *IEEE Access* **2018**, *6*, 13671–13684. [CrossRef]
9. Yan, M.; Yuan, H.; Xu, J.; Yu, Y.; Jin, L. Task allocation and route planning of multiple UAVs in a marine environment based on an improved particle swarm optimization algorithm. *EURASIP J. Adv. Signal Process.* **2021**, *2021*, 94. [CrossRef]
10. Rao, L.; Zhou, C.; Xu, X.; Zhang, Y.; Zhu, Y. UAV inspection path optimization for photovoltaic stations based on a genetic algorithm. *China New Technol. Prod.* **2025**, *11*, 41–43. (In Chinese)
11. Luo, L.; Zheng, X.; Pan, C.; Su, W.; Zhang, X. UAV inspection path optimization for ultra-high-voltage transmission lines based on improved particle swarm optimization. *Electr. Technol. Econ.* **2025**, *8*, 289–292. (In Chinese)
12. Zhu, C.; Huang, X.; Gong, Q.; Wang, B.; Ma, H. Ant colony optimization algorithm based path planning research for UAV power line inspection. *Power Syst. Clean Energy* **2021**, *37*, 71–77. (In Chinese)
13. Shen, L.; Wang, Y.; Liu, K.; Yang, Z.; Shi, X.; Yang, X.; Jing, K. Synergistic path planning of multi-UAVs for air pollution detection of ships in ports. *Transp. Res. Part E Logist. Transp. Rev.* **2020**, *144*, 102128. [CrossRef]
14. Pu, Q.; Yang, L.; Li, T. Dynamic coverage path planning method for UAV formations in multi-region aerial tasks. *Aerosp. Sci. Technol.* **2025**, *167*, 110683. [CrossRef]
15. Wang, P.; He, P.; Ma, C.; Niu, C.; Gao, H.; Wang, H.; Muyeen, S.M.; Zhou, D. A novel path planning approach for plant protection UAV based on DDPG and ILA optimization algorithm. *Comput. Electron. Agric.* **2025**, *239*, 111006. [CrossRef]
16. Zhen, L.; Yang, Z.; Laporte, G.; Yi, W.; Fan, T. Unmanned aerial vehicle inspection routing and scheduling for engineering management. *Engineering* **2024**, *36*, 223–239. [CrossRef]
17. Wang, W.; Wen, C. Multi-trip path planning for UAV with autonomous docking station in geohazards monitoring. *Comput. Ind. Eng.* **2026**, *211*, 111578. [CrossRef]
18. Zhao, Y.; Lu, B.; Alipour, M. Optimized structural inspection path planning for automated unmanned aerial systems. *Autom. Constr.* **2024**, *168*, 105764. [CrossRef]
19. An, Y.; Ning, J.; Hou, C.; Ou, J. Efficient low-collision UAV-based automated structural surface inspection using geometric digital twin and voxelized obstacle information. *Autom. Constr.* **2025**, *171*, 105972. [CrossRef]
20. Liu, X.; Yi, W.; Chen, P.; Tan, Y. Flight path planning of UAV-driven refinement inspection for construction sites based on 3D reconstruction. *Autom. Constr.* **2025**, *177*, 106360. [CrossRef]
21. Li, X.; Chen, Y.; Chen, Z.; Huang, Z. Coverage path planning of bridge inspection with unmanned aerial vehicle. *Eng. Appl. Artif. Intell.* **2025**, *156*, 111253. [CrossRef]
22. Wang, J.; Zhou, K.; Xing, W.; Li, H.; Yang, Z. Applications, evolutions, and challenges of drones in maritime transport. *J. Mar. Sci. Eng.* **2023**, *11*, 2056. [CrossRef]
23. Ma, T.; Tian, M.; Wang, Z.; Dong, J.; Zhu, J. Research on multi-object detection algorithm for routine highway inspection based on UAV. *J. China Foreign Highw.* **2026**, *46*, 229–239. (In Chinese)
24. Hong, G.; Tang, F.; Zhong, S.; Qin, H. Path planning method for active inspection of distribution networks using rotary-wing UAVs. *Electron. Des. Eng.* **2025**, *33*, 160–165. (In Chinese)
25. Beasley, J.E. Route first—Cluster second methods for vehicle routing. *Omega* **1983**, *11*, 403–408. [CrossRef]

26. Shaw, P. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming—CP98*; Maher, M., Puget, J.-F., Eds.; Springer: Berlin/Heidelberg, Germany, 1998; Volume 1520, pp. 417–431.
27. Ropke, S.; Pisinger, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **2006**, *40*, 455–472. [CrossRef]
28. Pisinger, D.; Ropke, S. A general heuristic for vehicle routing problems. *Comput. Oper. Res.* **2007**, *34*, 2403–2435. [CrossRef]
29. Zhou, B.; Liu, W.; Yang, H. Unmanned aerial vehicle service network design for urban monitoring. *Transp. Res. Part C Emerg. Technol.* **2023**, *157*, 104406. [CrossRef]
30. Liu, Z.; Yu, B.; Chen, T.; Zhang, L. Integrating battery-related decisions into truck-drone tandem delivery problem with limited battery resources. *Transp. Res. Part C Emerg. Technol.* **2025**, *174*, 105082. [CrossRef]
31. Mulumba, T.; Najy, W.; Diabat, A. The drone-assisted pickup and delivery problem: An adaptive large neighborhood search metaheuristic. *Comput. Oper. Res.* **2024**, *161*, 106435. [CrossRef]
32. Meng, S.; Chen, Y.; Li, D. The multi-visit drone-assisted pickup and delivery problem with time windows. *Eur. J. Oper. Res.* **2024**, *314*, 685–702. [CrossRef]
33. Shi, J.; Mao, H.; Zhou, Z.; Zheng, L. Adaptive large neighborhood search algorithm for the unmanned aerial vehicle routing problem with recharging. *Appl. Soft Comput.* **2023**, *147*, 110831. [CrossRef]
34. Chen, Y.; Zhang, D.; Jing, H.; Huang, Y.; Li, X. An adaptive multi-population cooperative whale optimization algorithm for global optimization and 3D UAV path planning. *Adv. Eng. Inform.* **2026**, *69*, 104003. [CrossRef]
35. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
36. Xue, J.; Shen, B. Dung beetle optimizer: A new meta-heuristic algorithm for global optimization. *J. Supercomput.* **2023**, *79*, 7305–7336. [CrossRef]
37. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. [CrossRef]
38. Abioye, A.O.; Hunt, W.; Gu, Y.; Schneiders, E.; Naiseh, M.; Archibald, B.; Sevegnani, M.; Ramchurn, S.D.; Fischer, J.E.; Soorati, M.D. A user study evaluation of predictive formal modelling at runtime in human-swarm interaction. *ACM Trans. Hum.-Robot Interact.* **2025**, *14*, 58. [CrossRef]
39. Liu, L.; Meng, L.; Li, A.; Lv, Y.; Zhao, B. PD-YOLOv11: A power distribution enabled YOLOv11 algorithm for power transmission tower component detection in UAV inspection. *Alex. Eng. J.* **2025**, *131*, 312–324. [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.