

Highlights

An Efficient Adaptive Large Neighborhood Search Algorithm for the Flying Sidekick Traveling Salesman Problem

Duc D. Vu^{}, Le Thi Hong, Tran Nam Khanh^{}, Nguyen Dinh Cong, Vu Duc Minh^{}

- Combines two destroy–repair groups with six truck–drone local-search moves.
- A stage-based representation enables $O(1)$ move evaluation after pre-processing.
- SMD speeds up ALNS by about threefold at 250 customers.
- Matches all 72 MC-set optimal scenarios and has a 0.02% gap on 90 Agatz-S cases.
- Lowers mean objectives by 1.28–3.77% (150 Agatz-L) and 1.51–3.14% (60 Ha-set).

An Efficient Adaptive Large Neighborhood Search Algorithm for the Flying Sidekick Traveling Salesman Problem

Duc D. Vu^a, Le Thi Hong^b, Tran Nam Khanh^{c,e}, Nguyen Dinh Cong^b,
Vu Duc Minh^{d,e,*}

^a*School of Management, University of Michigan-Flint, 303 East Kearsley Street, Flint, MI, 48502-1950, USA*

^b*Hong Duc University, 565 Quang Trung, Hac Thanh, Thanh Hoa, Viet Nam*

^c*Hanoi University of Science, 334 Nguyen Trai, Thanh Xuan, Ha noi, Viet Nam*

^d*National Economics University, 207 Giai Phong, Hai Ba Trung, Ha noi, Viet Nam*

^e*Smart Logistics and Supply Chain Management Laboratory, National Economics University, Viet Nam*

Abstract

This paper proposes an efficient Adaptive Large Neighborhood Search (ALNS) algorithm for the Flying Sidekick Traveling Salesman Problem (FSTSP). The algorithm integrates a stage-based truck–drone solution representation, two complementary destroy–repair operator groups, and six local-search neighborhoods focusing on drone operations and truck–drone interactions. Computational efficiency is enhanced through constant-time move evaluation after preprocessing and Static Move Descriptors, while an adaptive penalty mechanism handles drone endurance constraints. Extensive experiments on 336 benchmark instances demonstrate strong performance across problem sizes. On small instances, ALNS recovers all 72 known MC-set optimal benchmark values across 720 independent runs, averaging only 0.76 seconds per run. On 90 Agatz-S instances, it matches or improves the dynamic-programming reference solutions in 83.9% of runs, with an average gap of just 0.02%. On 150 medium-to-large Agatz-L instances (50–250 customers), ALNS achieves average mean-objective improvements of 1.28%, 3.77%, and

*Corresponding author

Email address: minhvd@neu.edu.vn (Vu Duc Minh)

3.01% over MD-SPP-H, EP-All, and SPP-All, respectively, with statistically significant differences. On 60 Ha-set instances, it further improves mean solution quality by 1.51% over MD-SPP-H and 3.14% over HGA-TAC. For 100-customer Ha-set instances, ALNS also reduces runtime by 47.2% and 44.5% relative to these methods, respectively. Ablation and robustness analyses further confirm the effectiveness and reliability of the proposed framework.

Keywords: Flying Sidekick Traveling Salesman Problem, Adaptive Large Neighborhood Search, Static Move Descriptor, Truck-Drone Routing

1. Introduction

The last mile, i.e., the final stage of the supply chain in which goods are transported from a distribution facility to end customers, has become one of the most operationally challenging segments of modern logistics. This stage is marked by dispersed delivery destinations, fluctuating demand, urban congestion with associated concerns over air quality (Nguyen et al., 2022), and rising customer expectations. The rapid expansion of e-commerce has further intensified pressure on logistics providers to improve efficiency, responsiveness, and service quality in this critical segment.

To address these challenges, both researchers and practitioners have increasingly examined the role of emerging technologies in last-mile operations. Firms have begun integrating unmanned aerial vehicles, or drones, into their logistics networks, delivering even time-sensitive or perishable products such as food and medicines. For instance, Wendy noted that “Drone delivery is kind of a natural extension of the delivery platform”¹. This momentum is reflected in market forecasts, with the global drone delivery market expected to reach \$28 billion by 2028, signaling strong confidence in this technology’s future role in logistics.

Nevertheless, significant barriers remain. Drones continue to face tech-

¹<https://www.fiercesensors.com/electronics/wendys-beefs-drone-delivery-tech-cooks>

nical constraints, particularly in payload capacity and flight endurance. In addition, fully drone-based systems require considerable infrastructure investment and must comply with evolving regulatory requirements. Consequently, hybrid delivery systems in which drones complement conventional trucks have emerged as a practical alternative. By combining the flexibility of drones with established infrastructure and capacity of truck delivery, such systems can capitalize on the advantages of both delivery modes.

In this paper, we study the Flying Sidekick Traveling Salesman Problem (FSTSP) (Murray and Chu, 2015), in which a truck and a drone work in tandem to deliver packages to customers. We propose an Adaptive Large Neighborhood Search (ALNS) that directly explores coordinated truck–drone solutions while reducing the computational cost of repeated move evaluation. Our main contributions are summarized as follows.

- We combine complementary truck-only and mixed destroy–repair groups with six interaction-focused local-search neighborhoods. This design couples broad route exploration with targeted improvements to customer assignments and launch–recovery decisions.
- We develop a stage-based truck–drone representation that supports $O(1)$ candidate-move evaluation after linear-time preprocessing, accounting for synchronization and endurance effects without reevaluating the complete solution.
- Experiments on 336 benchmark instances assess solution quality, efficiency, and robustness. On 150 Agatz-L instances, mean objectives improve by 1.28%, 3.77%, and 3.01% over published MD–SPP–H, EP–All, and SPP–All results, respectively, with statistically significant differences.
- We adapt Static Move Descriptors (SMD) to retain unaffected drone–customer swap evaluations. Ablation results show roughly 6–8-fold acceleration of this neighborhood on larger instances and about three-fold speedup of the complete ALNS at 250 customers.

The remainder of the paper is organized as follows. Section 2 reviews related literature, while Section 3 formally defines our problem setting. Section 4 presents our ALNS framework, including the destroy–repair operator groups and local search with SMD. Section 5 reports computational results for the proposed solution against state-of-the-art algorithms. Section 6 concludes the paper.

2. Related work

The FSTSP, introduced by [Murray and Chu \(2015\)](#), couples truck routing, customer assignments, and drone launch–recovery decisions through synchronization and endurance constraints. Consequently, shortening the truck route alone need not reduce delivery completion time. Exact methods for the FSTSP and related Traveling Salesman Problem with Drone (TSP-D) variants include dynamic programming ([Bouman et al., 2018](#)), branch-and-bound ([Dell’Amico et al., 2022](#)), branch-and-price ([Roberti and Ruthmair, 2021](#)), and stage-based formulations ([Le et al., 2026](#)). Broader surveys cover other vehicle configurations and operational assumptions ([Macrina et al., 2020](#); [Moshref-Javadi and Winkenbach, 2021](#)).

A major heuristic strategy separates sequence search from the construction of synchronized truck–drone operations. [Agatz et al. \(2018\)](#); [Kundu et al. \(2022\)](#) develop partitioning and split procedures for a given customer order. [Schaumann et al. \(2025\)](#) extend this principle to multiple drops through MD–SPP–H, combining a polynomial-time split algorithm with sequence improvement and diversification; their method also addresses the single-drop FSTSP. [Mahmoudinazlou and Kwon \(2024\)](#) encode customer order and service assignments in type-aware chromosomes, then use dynamic programming to determine rendezvous points. These approaches use decoding to optimize coordinated operations for a given sequence or assignment, making decoder efficiency important during repeated search.

Another strategy modifies truck–drone solutions through local search or destruction and reconstruction. [De Freitas and Penna \(2020\)](#) develop a variable neighborhood search for the FSTSP. In multi-drop routing, [Gonzalez-](#)

R et al. (2020) propose an iterated-greedy heuristic, while Mara et al. (2022) develop an ALNS. The latter also motivates the LDR insertion rules used in our truck-route repair. Such methods provide a basis for exploring structural changes directly, but their effectiveness depends on how route modifications, service reassignment, and rendezvous changes are combined.

These studies motivate a methodological question for the classical FSTSP: how can broad structural exploration and focused synchronization improvements be combined while keeping repeated move evaluation affordable? A candidate move must account for its effects on parallel travel, waiting, and endurance. Yet local changes affect only part of the solution, and many candidate evaluations remain valid after an accepted move. This creates two complementary opportunities: reducing the cost of each evaluation and avoiding redundant evaluations. The established Static Move Descriptor (SMD) principle addresses the latter in routing local search (Zachariadis and Kiranoudis, 2010; Beek et al., 2018).

We address this question through a direct ALNS for the single-truck, single-drone FSTSP, with one customer per sortie and no revisits or drone loops. Complementary truck-only and mixed destroy–repair groups provide broad exploration, followed by six local-search neighborhoods targeting drone operations and truck–drone interactions. A stage-based representation supports $O(1)$ candidate-move evaluation after preprocessing, while SMD retains unaffected evaluations in the drone-customer swap neighborhood. The research focus is the integration of coordinated search with these complementary reductions in computational effort.

3. Problem statement

FSTSP can be modeled by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_0, v_1, \dots, v_{N-1}\}$ is a set of N nodes. Nodes v_0 and v_{N-1} represent the depot, while the remaining nodes are customers served by either the truck or the drone. $\mathcal{E}$ is the set of arcs connecting pairs of nodes. For each arc $(i, j) \in \mathcal{E}$, τ_{ij} and τ'_{ij} denote the truck and drone travel times between nodes i and j , respectively.

In FSTSP, one truck and one drone operate throughout the delivery

process. The truck route is the ordered sequence $\mathcal{C} = \{c_1, c_2, \dots, c_K\}$, with $c_1 = v_0$ and $c_K = v_{N-1}$, where K is the number of truck visits. A *stage* is the position k of the truck visit c_k in this sequence; the segment after stage k is the arc (c_k, c_{k+1}) . A drone may be launched at any non-final truck stage. Sortie d is the triple (c_{d1}, c_{d2}, c_{d3}) , consisting of its launch node, drone-served customer, and recovery node, respectively. The set of sorties is $\mathcal{D} = \{(c_{d1}, c_{d2}, c_{d3}) : d = 1, \dots, D\}$, where D is the number of sorties.

A solution to the FSTSP consists of the truck route $\mathcal{C}$ and the set of drone sorties $\mathcal{D}$. The objective is to minimize the *makespan*: the time by which all customers have been served and both vehicles have returned to the depot. Figure 1 illustrates a solution for an instance with five customers. The schedule starts and ends at the depot, represented by v_0 and v_6 , with customers 1–3 served by the truck and customers 4–5 served by the drone; that is, $\mathcal{C} = \{v_0, v_1, v_2, v_3, v_6\}$ and $\mathcal{D} = \{(v_0, v_4, v_1), (v_2, v_5, v_6)\}$.

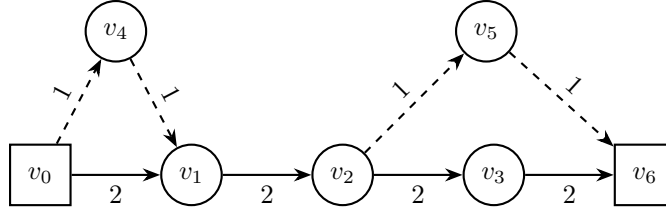


Figure 1: Example operation with five truck nodes $\{v_0(c_1), v_1(c_2), v_2(c_3), v_3(c_4), v_6(c_5)\}$ and two drone nodes $\{v_4, v_5\}$. The depot at $\{v_0, v_6\}$ is represented by a square. Solid arcs denote truck travel with distance 2; dashed arcs denote drone trips with distance 1. With zero launch and recovery service times, the objective value of this sample solution is 8.

We use the following assumptions:

1. Each customer is served exactly once, either by the truck or by the drone. Only customers in the drone-eligible subset may be assigned to a drone sortie.
2. The truck has no capacity limit. A single drone is carried by the truck when it is not deployed, and each sortie serves exactly one customer. While the drone is deployed, the truck may continue serving customers.
3. The launch and recovery nodes of a sortie are distinct nodes visited by the truck, with the launch preceding the recovery on the truck route.

Loop operations, in which the truck remains at the launch node until the drone returns, are not allowed.

4. The truck and drone are synchronized at recovery. The vehicle arriving first waits for the other; in particular, the drone remains airborne while waiting and therefore continues consuming endurance.

Let SL , SR , and DTL denote the launch service time, recovery service time, and drone endurance limit, respectively. Following the Murray–Chu convention, launch preparation is performed while the drone is still carried by the truck. It delays both vehicles but does not consume drone endurance; no launch time is charged when the sortie starts at the initial depot. After the truck and drone meet, recovery service delays completion of the sortie and does consume endurance, including at the final depot. Therefore, the schedule adds the applicable launch time, the longer of the parallel truck travel and drone flight times, and SR . This description matches the original FSTSP convention and its later exact formulations (Murray and Chu, 2015; Dell’Amico et al., 2022).

4. Proposed ALNS Algorithm

Algorithm 1 adapts the ALNS framework of Ropke and Pisinger (2006) to the FSTSP. A stage-based truck–drone representation supports direct synchronization and endurance evaluation. Two destroy–repair groups diversify the search by modifying either the truck route alone or truck and drone operations jointly; six problem-specific local-search moves then intensify it. Constant-time candidate evaluation (after preprocessing) and Static Move Descriptors (SMD) reduce local-search effort, while an adaptive endurance penalty permits temporary infeasibility. Adaptive operator weights and simulated-annealing acceptance guide the search. Starting from a feasible truck-only tour, the algorithm maintains the current solution s and the best feasible solution s^* , which is returned.

At each iteration, the algorithm first selects uniformly between the enabled truck-only and mixed truck–drone operator groups. Within the selected group, the destroy operator is chosen by adaptive roulette selection.

Algorithm 1 Adaptive Large Neighborhood Search for Flying Sidekick Traveling Salesman Problem

Require: Instance I , ALNS parameters

Ensure: A best feasible solution s^*

#Block 1: Initialize the initial solution and parameters

- 1: $s^*, s \leftarrow$ Initial solution from I
 - 2: Initialize weights ω for operators and group operators
 - 3: **while** *stopping condition is not met* **do**

#Block 2: Apply destroy-repair operators and local search operators

 - 4: Select a group operator G from the set $\mathcal{G}$
 - 5: Select a destroy operator $q \in G$ and a repair operator $r \in G$
 - 6: $s' \leftarrow r(q(s))$
 - 7: Apply local search operators on s' to find improved solutions

#Block 3: Solution and parameter updates

 - 8: **if** $s'.\text{cost} \leq s.\text{cost}$ or $\mathcal{U}(0, 1) < \exp\left(\frac{-(s'.\text{cost} - s.\text{cost})}{BZ \cdot T_c}\right)$ **then** - 9: $s \leftarrow s'$
 - 10: **if** s' is feasible and $s'.\text{cost} < s^*.\text{cost}$ **then** - 11: $s^* \leftarrow s'$
 - 12: **end if**
 - 13: **end if**
 - 14: Update weights ω and acceptance temperature T_c
 - 15: **end while**
 - 16: **return** s^*
-

A truck-only repair operator is selected independently by another adaptive roulette, whereas each mixed destroy operator is tied to its corresponding repair operator. The resulting solution is then improved by local search. Temporary endurance infeasibility is allowed and evaluated through the penalized objective defined below.

The candidate s' is evaluated by a simulated-annealing acceptance rule. If its penalized cost is no greater than that of s , it is accepted directly; otherwise, it is accepted with probability $\exp\left(\frac{-[s'.\text{cost} - s.\text{cost}]}{BZ \cdot T_c}\right)$, where BZ is the Boltzmann scaling factor and T_c is the current temperature. If accepted, s' replaces s , and s^* is updated only when s' is feasible and improves the incumbent. Operator weights and the temperature are then updated. After each iteration, the temperature is cooled as $T_c \leftarrow \alpha T_c$. The search stops when

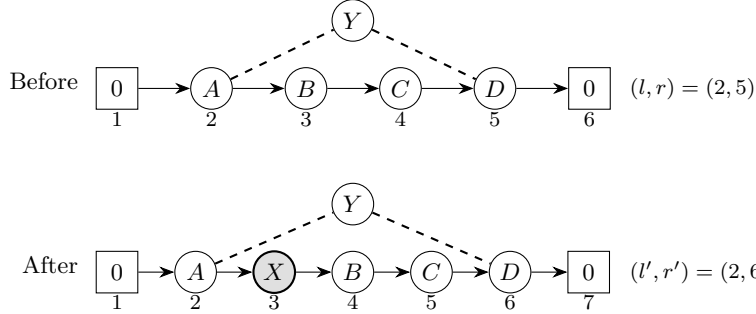


Figure 2: Stage-index updates after inserting truck customer X before B . The drone sortie $A \rightarrow Y \rightarrow D$ retains its endpoints, but its stage indices change from $(2, 5)$ to $(2, 6)$.

either the final temperature or the time limit is reached. Although penalized solutions guide the search, the algorithm returns s^* , the best feasible solution; an endurance-infeasible solution is never reported as the final result.

4.1. Objective evaluation

Auxiliary notation. Consider a solution $(\mathcal{C}, \mathcal{D})$, where $\mathcal{C} = (c_1, \dots, c_K)$ is the truck route and $\mathcal{D} = \{(c_{d1}, c_{d2}, c_{d3}) : d = 1, \dots, D\}$ is the set of drone sorties. Let l_d and r_d denote the launch and recovery stages of sortie d , respectively, with $c_{l_d} = c_{d1}$, $c_{r_d} = c_{d3}$, and $l_d < r_d$. The corresponding stage sets are $\mathcal{L} = \{l_d : d = 1, \dots, D\}$ and $\mathcal{R} = \{r_d : d = 1, \dots, D\}$.

Define the cumulative truck travel time $\theta_{1j} = \sum_{i=1}^{j-1} \tau_{c_i c_{i+1}}$ and the drone flight duration $\theta'_d = \tau'_{c_{d1} c_{d2}} + \tau'_{c_{d2} c_{d3}}$. The truck travel time between stages $i < j$ is $\theta_{ij} = \theta_{1j} - \theta_{1i}$, which can be obtained in $O(1)$ time after preprocessing.

In Figure 1, $\mathcal{L} = \{1, 3\}$, $\mathcal{R} = \{2, 5\}$, $\theta = \{0, 2, 4, 6, 8\}$, and $\theta' = \{2, 2\}$. Truck-route modifications require consistent updates to all affected stage indices and their associated references, as illustrated in Figure 2.

For simplicity of presentation, the objective expression below omits the launch and recovery service times. The implementation includes them according to the rules stated in Section 3. The search objective is:

$$\underbrace{\sum_{i=1}^{K-1} \tau_{c_i c_{i+1}}}_{\text{truck travel time}} + \underbrace{\sum_{d \in \mathcal{D}} [\theta'_d - \theta_{l_d r_d}]^+}_{\text{waiting time}} + \underbrace{\lambda \sum_{d \in \mathcal{D}} [\max\{\theta'_d, \theta_{l_d r_d}\} - DTL]^+}_{\text{drone endurance penalty}},$$

where $[a]^+ = \max\{a, 0\}$. The three terms represent truck travel time, truck waiting time, and an endurance-violation penalty, respectively (Gonzalez-R et al., 2020).

Following the adaptive penalty principle of Vidal et al. (2013), λ is adjusted according to the observed feasible-solution proportion ρ_t and its target ρ^* :

$$\lambda \leftarrow \begin{cases} \eta^+ \lambda, & \rho_t < \rho^*, \\ \eta^- \lambda, & \text{otherwise,} \end{cases}$$

where $\eta^+ > 1$ and $0 < \eta^- < 1$. This adjustment balances the exploration of feasible and endurance-infeasible solutions.

4.2. Initial solution generation

We construct a feasible initial solution by generating a random truck-only tour with the depot at both ends (a TSP tour). The solution is then improved using the local search described in Section 4.4.

4.3. Destroy–repair operator groups

To account for the interdependence between truck routes and drone sorties, we organize destroy–repair operators into two groups: (i) truck-only and (ii) mixed truck–drone modifications. Each group contains compatible operators selected through the adaptive mechanism.

Type 1: Truck-only group. This group diversifies the search by rearranging multiple truck customers while preserving the launch and recovery stage indices of existing sorties. It provides large-scale changes to the truck route, reducing the need for additional truck-only local-search operators, whose limited benefits relative to their computational cost are discussed in Section 4.4.

Three destroy operators are used: (i) *truck random removal*, removing randomly selected customers; (ii) *truck worst removal*, removing customers with the largest arc savings $\tau_{c_{k-1}c_k} + \tau_{c_k c_{k+1}} - \tau_{c_{k-1}c_{k+1}}$; and (iii) *truck block removal*, removing consecutive customers.

The repair operators are (i) *truck greedy insertion*, minimizing insertion cost; (ii) *LDR-1*, inserting customer x after the node minimizing $\tau_{c_k x}$ (Mara et al., 2022); and (iii) *LDR-2*, inserting x before the node minimizing τ_{xc_k} (Mara et al., 2022). Any destroy operator can be paired with any repair operator.

Type 2: Mixed truck-drone group. This group diversifies the search by jointly modifying the truck route and drone sorties, allowing changes in customer assignments and launch-recovery configurations. Unlike the truck-only group, it can restructure both components simultaneously, enabling coordinated changes that would be difficult to achieve through isolated moves.

Mixed block removal removes truck customers within an interval $[i, j]$ and sorties spanning its boundaries. Removed stages are retained as placeholders to preserve stage indices. *Mixed block repair* reconstructs the segment using nearest-neighbor truck insertion followed by greedy drone insertion.

Mixed outlier removal removes a randomized subset of distant customers, adjacent truck customers, and affected sorties. Remaining sortie indices are updated after removal. *Mixed outlier repair* reinserts customers in decreasing depot-distance order, choosing between truck and drone service.

Each mixed destroy operator is paired exclusively with its corresponding repair operator.

4.4. Local search

The local search focuses on drone operations and truck-drone interactions through six neighborhoods: AD, SD1-SD4, and RD. These moves modify drone sorties, customer assignments, or truck-drone synchronization, complementing the truck-only destroy-repair group that already explores large changes in truck-customer order.

We also tested conventional truck-only moves (2-opt, relocation, and swap) within truck segments between consecutive sorties on 150 instances ($n = 50-250$, ten seeds). These moves improved the mean objective by only 0.41% while increasing runtime by a factor of 3-4. As the truck-only destroy-repair group already diversifies truck routes (Section 4.3), we exclude these moves from local search.

Move definitions. Figure 3 illustrates the six moves:

1. *AD (Truck-to-Drone Relocation)*. Transfers a truck customer to a new, nonoverlapping drone sortie, retaining the endpoints of existing sorties.
2. *SD1 (Drone-Customer Swap)*. Exchanges the drone-served customers of two sorties without changing their launch and recovery stages.
3. *SD2 (Shared-Stage Relocation)*. Shifts the shared recovery/launch stage of two consecutive sorties to another feasible truck stage.
4. *SD3 (Launch-Recovery Customer Swap)*. Exchanges the truck customers at the launch and recovery stages of a sortie.
5. *SD4 (Truck-Drone Customer Swap)*. Exchanges a sortie’s drone-served customer with the truck customer at its launch or recovery stage.
6. *RD (Drone-to-Truck Relocation)*. Removes a sortie and reinserts its drone-served customer into the truck route, retaining the endpoints of other sorties.

The six moves are applied in the fixed cycle

$$AD \rightarrow SD1 \rightarrow SD2 \rightarrow SD3 \rightarrow SD4 \rightarrow RD.$$

Each move uses a deterministic improvement procedure. Local search performs at most two cycles, stopping earlier when a complete cycle yields no improvement. Candidate objective changes are evaluated in $O(1)$ time after preprocessing (Supplementary Material, Section S1).

4.5. Speeding up Local Search with Static Move Descriptors

To reduce repeated evaluations in local search, we incorporate a lightweight Static Move Descriptor (SMD) mechanism, originally introduced by Zachariadis and Kiranoudis (2010) and later adapted in routing algorithms such as FILO (Accorsi and Vigo, 2024). The key idea is that a local-search move usually modifies only a small part of the solution; hence, most candidate moves remain unchanged after an accepted move and do not need to be re-evaluated.

In our ALNS, SMD is applied to the SD1 neighborhood, which is the most time-consuming local-search move. Each descriptor stores the two sorties

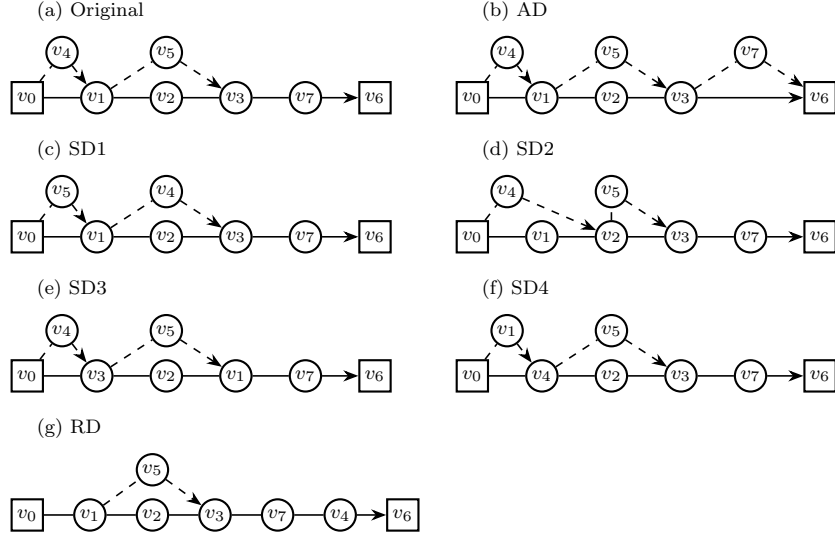


Figure 3: Illustration of the original solution and six local-search moves. Solid arcs denote truck movements and dashed arcs denote drone sorties: (a) Current solution, (b) AD creates a sortie for v_7 ; (c) SD1 swaps drone customers v_4 and v_5 ; (d) SD2 shifts the shared stage from v_1 to v_2 ; (e) SD3 swaps launch/recovery nodes v_1 and v_3 ; (f) SD4 swaps truck customer v_1 with drone customer v_4 ; and (g) RD reinserts drone customer v_4 into the truck route.

involved in the swap and the corresponding precomputed objective variation Δ_{SD1} . At the beginning of the SD1 search, all candidate moves are evaluated once, and only improving moves are kept in an active list. When the best descriptor is applied, descriptors that conflict with it, i.e., those involving affected sorties, are removed and re-evaluated. Unaffected descriptors are retained without recomputation.

This implementation uses a simple linear container rather than specialized data structures. Although this does not change the worst-case complexity, it substantially reduces unnecessary move evaluations in practice while preserving the simplicity of the proposed local-search framework.

5. Computational Experiments and Analysis

To guide the computational study, we formulate four research questions covering the competitive performance, algorithmic components, drone

characteristics, and robustness of the proposed ALNS.

- *RQ1. How competitive is the proposed ALNS?* We assess solution quality and runtime against exact and heuristic benchmarks.
- *RQ2. What is the contribution of each algorithmic component?* We examine the destroy–repair operator groups, local search, SMD, the number of local-search cycles, and the dynamic penalty.
- *RQ3. How do drone speed and endurance affect the solutions?* We evaluate the changes in objective value and the number of customers served by the drone under different speed and endurance settings.
- *RQ4. How robust is ALNS across repeated runs and instance classes?* We assess repeated-run stability across the comparative Agatz-L instances.

5.1. Benchmark instances and experimental setup

We conduct computational experiments on three benchmark datasets: the Murray–Chu set (*MC-set*) (Murray and Chu, 2015)², the Agatz set (*Agatz-set*) (Agatz et al., 2018; Bouman et al., 2018)³, and the Ha set (*Ha-set*) (Ha et al., 2018)⁴. The Agatz-set is divided into two subsets: *Agatz-S* (small instances) and *Agatz-L* (medium and large instances). Together, these datasets comprise 336 distinct benchmark instances and 372 instance–configuration scenarios: 72 from the MC-set, 90 from Agatz-S, 150 from Agatz-L, and 60 from the Ha-set. Table 1 summarizes the main settings and the research questions (RQs) addressed by each dataset. Further details are provided in Supplementary Material, Section S3.

The experiments were conducted on Ubuntu 24.10 using one core of a 13th Gen Intel(R) Core(TM) i5-13500 CPU with 62 GB RAM. The algorithm

²<https://www.chasemurray.com/research/180-2/>

³<https://github.com/pcbouman-eur/TSP-D-Instances>

⁴<https://orlab.com.vn/download/instances-for-the-traveling-salesman-problem-with-drone-tsp-d/>

Table 1: Benchmark datasets, main experimental settings, and associated research questions.

Dataset	Inst.	Scen.	n	Travel (truck/drone)	Drone speed	DTL	(SL, SR)	RQs
MC-set	36	72	10	Original	Original	20, 40	(1, 1)	1
Agatz-S	90	90	11–19	Euc./Euc.	2× truck	∞	(0, 0)	1
Agatz-L	150	150	50–250	Euc./Euc.	2× truck 40 km/h	∞	(0, 0)	1–4
Ha-set	60	60	50, 100	Man./Euc.	(both)	20 min	(0, 0)	1
Total	336	372						

Note: Inst. and Scen. denote instances and scenarios; Euc. and Man. denote Euclidean and Manhattan distances. Each scenario represents an instance evaluated under a main experimental configuration. MC-set instances are evaluated at two endurance limits. Additional Agatz-L configurations used in RQ2 and RQ3 are excluded from these totals. SL , SR , and DTL denote launch service time, recovery service time, and drone endurance limit, respectively.

was implemented in C++ and built using CMake 3.30.3. Parameter settings are reported in Supplementary Material, Section S2.

Each ALNS configuration was run independently ten times. The initial temperature is set to $T_1 = 200$ and decreases geometrically by a cooling factor of $\alpha = 0.99999$ after each iteration. The search terminates when the temperature reaches the final threshold $T_0 = 10$. For each instance, we record the best and mean objective values over the ten runs, denoted by z^{best} and $\bar{z}$, respectively. For the deterministic EP–All and SPP–All algorithms, their single published objective value is used in both comparisons. For each benchmark method Z , the best and the average objective gaps of ALNS are defined as

$$\Delta_Z^{\text{best}} = 100 \times \frac{z_{\text{ALNS}}^{\text{best}} - z_Z^{\text{best}}}{z_Z^{\text{best}}}(\%), \quad \Delta_Z^{\text{avg}} = 100 \times \frac{\bar{z}_{\text{ALNS}} - \bar{z}_Z}{\bar{z}_Z}(\%).$$

Negative comparative gaps favor ALNS. Group summaries are arithmetic means of the instance-level percentage gaps, denoted by $\bar{\Delta}_Z^{\text{best}}$ and $\bar{\Delta}_Z^{\text{avg}}$. The comparator subscript is omitted when it is clear from the table headings.

5.2. Comparison with benchmark methods - RQ1

5.2.1. Comparison with exact solutions on small instances

We first assess ALNS on the MC-set and Agatz-S. The former provides 72 scenarios with proven PS2 optimal values, while the latter contains 90 small instances with published DP reference solutions from [Bouman et al. \(2018\)](#). Each scenario is solved independently ten times. Tables 2 and 3 present the results.

For the MC-set, ALNS recovers all 72 PS2 optima reported by [Dell’Amico et al. \(2021\)](#) in every run, yielding 720/720 optimum matches and zero best and mean gaps. The mean ALNS runtime is 0.69 s at $DTL = 20$ and 0.83 s at $DTL = 40$, or 0.76 s overall. For comparison, the 2IF formulation of [Dell’Amico et al. \(2022\)](#) requires mean times of 57.0 s and 233.4 s, respectively, to certify optimality.

Table 2: ALNS recovery of the [Dell’Amico et al. \(2021\)](#) PS2 optima on the MC-set.

DTL	Scenarios	Optimum matches	Mean time (s)
20	36	360/360	0.69
40	36	360/360	0.83
Overall	72	720/720	0.76

On Agatz-S, ALNS finds optimal or near-optimal solutions for instances with $n \leq 17$. For $n \geq 18$, ALNS even finds better solutions for some instances where the DP method could not prove optimality. Overall, ALNS matches or improves the DP reference solutions in 755 out of 900 runs (83.9%), with an average gap of only 0.02% and a mean runtime of 1.44 s. For reference, Table 3 of [Bouman et al. \(2018\)](#) reports that the unrestricted exact DP method requires an average of 4 seconds for $n = 11$, approximately 5 minutes for $n = 14$, 26 minutes for $n = 15$, and over 2.5 hours for $n = 16$. For $n \geq 17$, the method does not complete within the 12-hour time limit.

5.2.2. Medium and large instances: heuristic benchmarks

Agatz-L instances. Figure 4 compares ALNS with the published results of MD-SPP-H ([Schaumann et al., 2025](#)), EP-All, and SPP-All ([Kundu et al., 2020](#)) on 150 Agatz-L instances under $DTL = \infty$. Each size group contains

Table 3: ALNS comparison with the Bouman et al. (2018) DP reference objectives on Agatz-S, by instance size (n). Reference values for $n = 18, 19$ are not proven optimal.

Size (n)	Instances	At/below reference	Avg gap (%)	Mean time (s)
$n = 11$	10	80/100	+0.12	1.09
$n = 12$	10	70/100	+0.21	1.15
$n = 13$	10	69/100	+0.30	1.22
$n = 14$	10	100/100	-0.00	1.38
$n = 15$	10	86/100	+0.03	1.45
$n = 16$	10	70/100	+0.08	1.52
$n = 17$	10	85/100	+0.03	1.53
$n = 18$	10	100/100	-0.48	1.73
$n = 19$	10	95/100	-0.10	1.86
Overall	90	755/900	+0.02	1.44

30 instances, with detailed results provided in Supplementary Material, Section S4.

ALNS achieves lower overall best and mean objective values than all three benchmarks. Against MD-SPP-H, the closest competitor, ALNS performs better for $n \geq 100$, while MD-SPP-H yields better average best objectives at $n = 50$ and 75. Table 4 summarizes the paired comparisons, with all six one-sided Wilcoxon signed-rank tests favoring ALNS after Holm correction. Differences within 0.05 are treated as ties to account for the rounding of published results.

Table 4: Paired objective comparisons over 150 Agatz-L instances. W/T/L counts are from the ALNS perspective; p values are Holm-adjusted across six tests.

Comparator	ALNS value	Mean gap (%)	W/T/L	Adjusted p
MD-SPP-H	Best	-0.63	87/3/60	1.24×10^{-4}
	Mean	-1.28	117/1/32	2.87×10^{-18}
EP-All	Best	-4.75	146/0/4	9.32×10^{-26}
	Mean	-3.77	140/0/10	6.34×10^{-25}
SPP-All	Best	-4.01	144/0/6	2.78×10^{-25}
	Mean	-3.01	133/0/17	1.81×10^{-22}

For runtime comparison, the reported MD-SPP-H times are adjusted using PassMark single-thread ratings (PassMark Software, 2020), with a factor of $2334.5/3853 \approx 0.606$. The former is a proxy for its Xeon E3-1284L

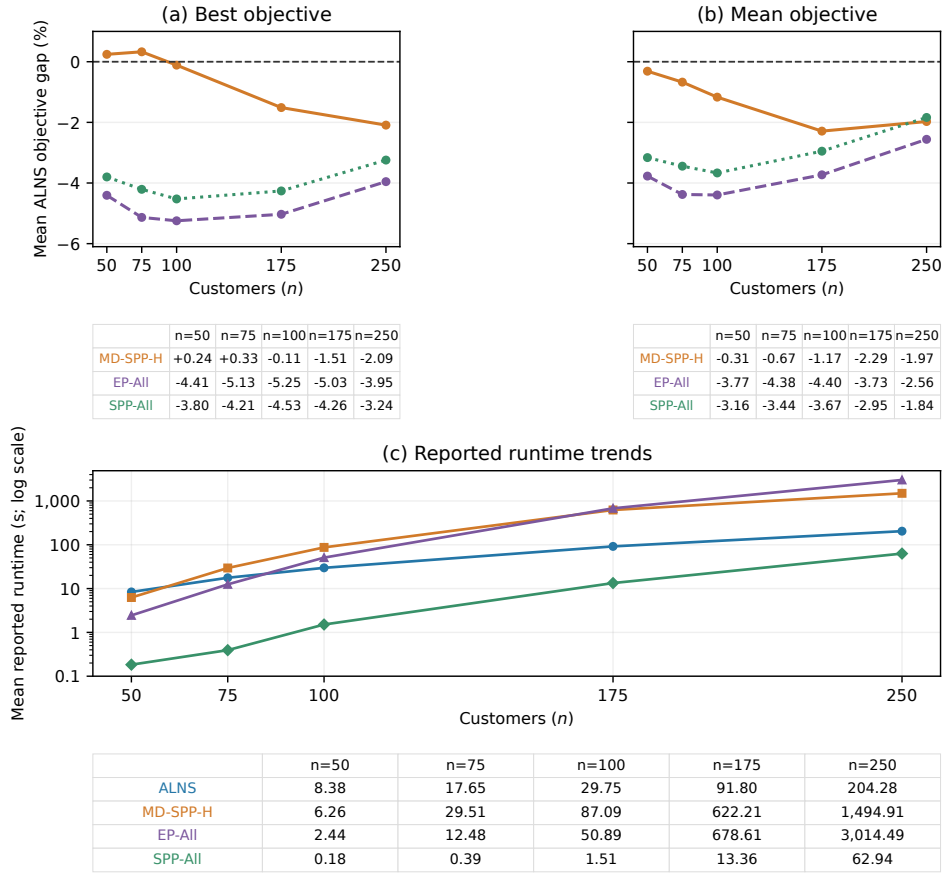


Figure 4: Comparison on Agatz-L: (a) best-objective gaps, (b) mean-objective gaps, and (c) reported mean runtimes (MD-SPP-H adjusted; logarithmic scale).

v4 processor, obtained by averaging ratings of the E3-1285L v4 and E3-1285 v4 models, while the latter corresponds to our i5-13500. EP-All and SPP-All times remain unadjusted due to unavailable processor information.

For $n \geq 100$, ALNS achieves better solution quality while requiring 43.6–77.4% less mean runtime than adjusted MD-SPP-H and 41.5–93.2% less than reported EP-All. SPP-All remains the fastest method, but ALNS improves its best and mean objectives by 4.01% and 3.01%, respectively, on average.

Ha-set instances. We compare ALNS with the official implementations of MD-SPP-H (Schaumann et al., 2025)⁵ and HGA-TAC (Mahmoudinazlou and Kwon, 2024)⁶. All methods are run ten times per instance on the same machine, using their respective default stopping conditions. Figure 5 and Table 5 summarize the results over 60 instances.

ALNS achieves lower overall best and mean objectives than both benchmarks, with average gaps of -0.62% and -1.51% against MD-SPP-H, and -2.44% and -3.14% against HGA-TAC, respectively. Its advantage increases at $n = 100$. All four overall one-sided Wilcoxon signed-rank tests favor ALNS after Holm correction.

Table 5: Paired objective comparisons over 60 Ha-set instances. W/T/L counts are from the ALNS perspective; p values are Holm-adjusted across 12 tests.

Comparator	ALNS value	Mean gap (%)	W/T/L	Adjusted p
MD-SPP-H	Best	-0.62	44/13/3	2.93×10^{-8}
	Mean	-1.51	59/0/1	1.10×10^{-10}
HGA-TAC	Best	-2.44	57/3/0	2.57×10^{-10}
	Mean	-3.14	60/0/0	9.78×10^{-11}

Regarding runtime, ALNS requires 13.76 s at $n = 50$, compared with 7.39 s for MD-SPP-H and 37.45 s for HGA-TAC. At $n = 100$, ALNS is faster than both competitors, requiring 43.39 s versus 82.15 s and 78.21 s, respectively, corresponding to reductions of 47.2% and 44.5%. Thus, ALNS combines better solution quality with shorter runtimes on larger instances.

⁵<https://github.com/schasarah/FSTSP-MD-Heuristic>

⁶<https://github.com/Sasanm88/TSPDroneHGATAC.jl>

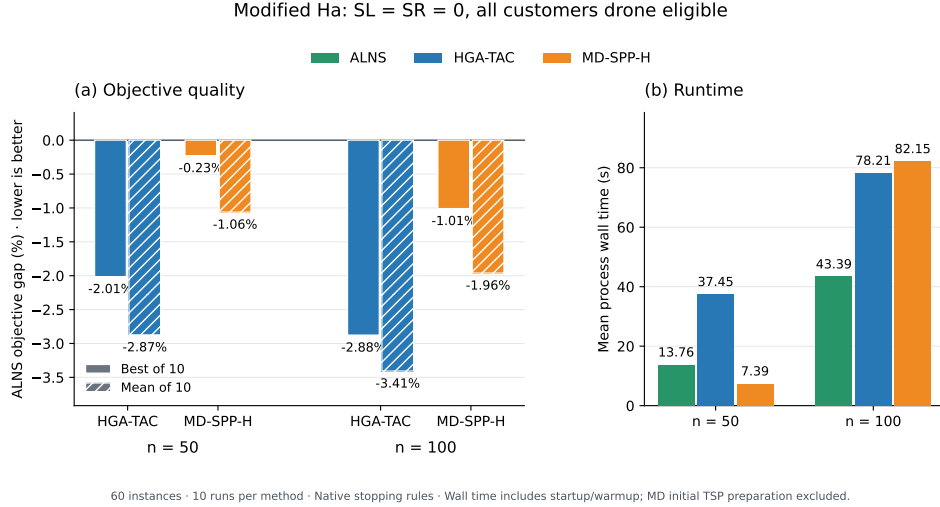


Figure 5: Comparison on Ha-set: (a) best and mean objective gaps and (b) mean process wall times.

Detailed per-instance results are provided in Supplementary Material, Section S5.

5.3. Ablation study - RQ2

We assess the contributions of the main ALNS components on Agatz-L. Table 6 reports mean-objective gaps relative to the reference configuration, the proposed ALNS, computed as $100(\bar{z}_{\text{variant}} - \bar{z}_{\text{reference}})/\bar{z}_{\text{reference}}$ and averaged across instances. Positive gaps favor the reference ALNS; W/T/L counts are also reported from its perspective.

5.3.1. Contributions of local search and destroy-repair groups

Local search contributes most strongly to solution quality: disabling it increases the mean objective by 25.12% across all 150 instances. Removing the truck-only and mixed truck-drone destroy-repair groups increases the mean objective by 4.66% and 1.62%, respectively. These results confirm the contribution of both operator groups, with a larger observed impact from the truck-only group.

Table 6: Combined ablation results relative to the reference ALNS configuration.

Variant	DTL	Inst.	Gap (%)	W/T/L
No local search	∞	150	25.12	150/0/0
No truck-only group	∞	150	4.66	150/0/0
No mixed group	∞	150	1.62	147/0/3
SMD disabled	∞	50	-0.01	23/0/27
One local-search cycle	∞	150	0.63	145/0/5
Three local-search cycles	∞	150	-0.08	57/0/93
Fixed big- M	20	150	71.12	149/0/1
Fixed big- M	40	150	50.84	147/0/3

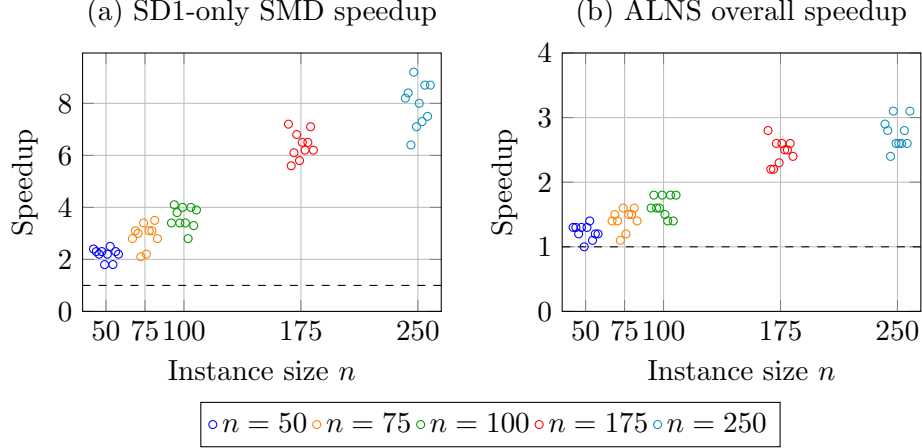


Figure 6: Impact of SMD on speedup for SD1 alone and for the complete ALNS, by instance size n .

5.3.2. Impact of Static Move Descriptors on runtime

Disabling SMD has essentially no systematic effect on objective value (Table 6 - row 'SMD disabled'), as expected for an acceleration mechanism. Figure 6 shows its runtime contribution to SD1 and the complete ALNS. Speedup is the runtime without SMD divided by the runtime with SMD; values above one indicate a reduction in time.

For SD1 alone, SMD yields a speedup of about 2 on 50-customer instances and roughly 6–8 on larger instances (Figure 6(a)). Its benefit increases with instance size as repeated move evaluations become more expensive. The overall ALNS speedup is smaller but remains above one at every tested size

(Figure 6(b)), showing that the SD1 acceleration reduces total runtime.

5.3.3. Impact of the dynamic penalty mechanism

Replacing the dynamic penalty with a fixed big- M penalty increases the mean objective by 71.12% at $DTL = 20$ and 50.84% at $DTL = 40$ (Table 6). Table 7 further breaks down these gaps by spatial distribution.

Table 7: Mean-objective gaps of the fixed big- M variant relative to dynamic penalty.

Instance class	DTL	#Inst.	Mean gap (%)
Single-center	20	50	11.98
Single-center	40	50	20.67
Double-center	20	50	73.74
Double-center	40	50	82.94
Uniform	20	50	127.65
Uniform	40	50	48.92

Dynamic penalty improves the mean objective in every spatial distribution and endurance setting. The gains are largest for uniform instances at $DTL = 20$ and double-center instances at $DTL = 40$, confirming its importance under finite endurance.

5.3.4. Choice of two local-search cycles

Compared with the default two-cycle configuration, one cycle increases the mean objective by 0.63% on average, while three cycles improve it by only 0.08% (Table 6). Table 8 shows that two cycles consistently outperform one cycle across all distribution-size groups. A third cycle yields only small, mixed improvements at a substantial runtime cost. We therefore retain two cycles as the default quality-time compromise.

5.4. Effects of Drone Speed and Endurance - RQ3

Figure 7 shows the mean makespan reductions on Agatz-L when the drone travels two or three times faster than the truck, relative to equal speeds.

For $DTL = 20$, speed ratios of 2 and 3 reduce makespan by 23.3–28.5% and 33.6–37.8%, respectively. For $DTL = 40$, the corresponding reductions

Table 8: Comparison of one-cycle, two-cycle, and three-cycle local-search configurations.

Distribution	n	1-cycle vs. 2-cycle		3-cycle vs. 2-cycle		Average runtime (s)		
		$\bar{\Delta}_{\text{best}}$	$\bar{\Delta}_{\text{avg}}$	$\bar{\Delta}_{\text{best}}$	$\bar{\Delta}_{\text{avg}}$	1-cycle	2-cycle	3-cycle
Uniform	50	0.33	0.27	0.02	-0.08	5.11	7.63	9.40
Uniform	75	0.28	0.47	0.04	-0.04	11.26	16.59	21.78
Uniform	100	0.18	0.45	0.02	-0.16	18.12	27.01	35.99
Uniform	175	0.48	0.42	-0.14	-0.05	56.27	82.37	108.44
Uniform	250	0.64	0.57	0.04	-0.04	128.12	186.11	237.44
1-center	50	0.14	0.26	0.01	-0.13	6.36	9.21	11.72
1-center	75	0.55	0.63	-0.15	-0.11	12.82	19.18	23.98
1-center	100	0.29	0.69	0.16	0.04	21.59	32.06	42.13
1-center	175	0.57	0.56	-0.00	-0.11	66.82	98.18	125.61
1-center	250	0.56	0.86	-0.34	-0.10	147.16	211.58	256.61
2-center	50	0.30	0.44	-0.07	0.04	5.70	8.30	11.00
2-center	75	0.37	0.40	-0.13	-0.16	11.62	17.18	22.29
2-center	100	0.33	0.51	-0.06	-0.11	20.35	30.16	39.02
2-center	175	0.71	1.26	0.08	0.09	64.64	94.84	117.56
2-center	250	0.33	1.67	-0.07	-0.27	149.25	215.15	261.66

Note: Gap values are percentages for the listed one- or three-cycle variant relative to the two-cycle reference. A positive value favors the two-cycle reference.

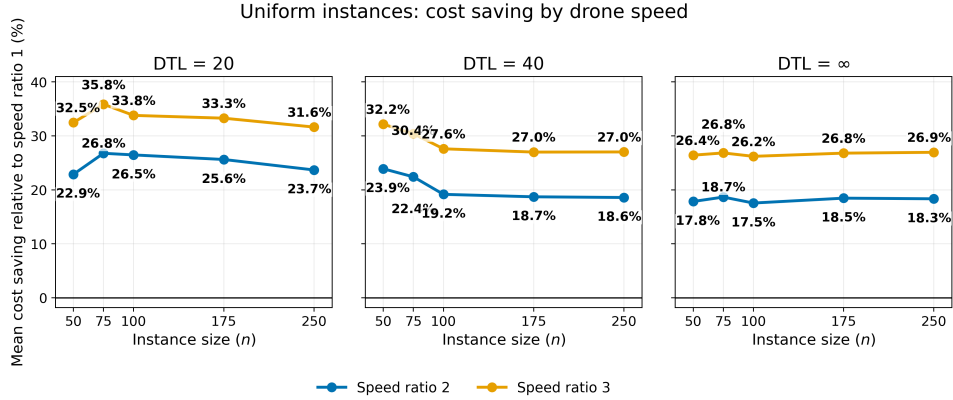


Figure 7: Makespan reductions under different drone speeds and endurance limits.

are 18.8–25.2% and 27.4–33.7%, generally decreasing with instance size. With unlimited endurance, savings remain stable across sizes at 18.5–19.2% and 27.3–28.1%, respectively.

Faster drones also serve more customers: the average drone-served fraction increases from 18–21% at speed ratio 1 to 36–38% at ratio 2 and approximately 42% at ratio 3. However, the additional benefit from increasing the ratio from 2 to 3 is smaller, suggesting diminishing returns as truck–drone synchronization becomes a limiting factor.

5.5. Robustness across Independent Runs - RQ4

To assess robustness, ALNS is run independently ten times on each of the 150 Agatz-L instances under $DTL = \infty$. For run r on instance i , the deviation from the best objective obtained across the ten runs is defined as

$$\Delta_{ir} = 100 \frac{z_{ir} - z_i^{\text{best}}}{z_i^{\text{best}}}, \quad z_i^{\text{best}} = \min_r z_{ir}.$$

Let p_ϵ denote the fraction of runs with $\Delta_{ir} \leq \epsilon$, where ϵ is a percentage tolerance. These deviations measure run-to-run stability, not optimality gaps.

Table 9 shows that 88.27% of the 1500 runs are within 2% of their instance’s best solution, and 99.40% are within 5%. All runs on uniform and single-center instances remain within 5%, while double-center instances exhibit a heavier tail, with a maximum deviation of 12.51%. Overall, these results indicate consistent solution quality across independent runs.

Table 9: Run-to-best dispersion over ten independent ALNS runs per Agatz-L instance ($DTL = \infty$).

Distribution	Runs	$100p_{1\%}$	$100p_{2\%}$	$100p_{5\%}$	$\Delta^{\max}$ (%)
Uniform	500	55.40	88.80	100.00	4.58
Single-center	500	56.80	87.60	100.00	4.38
Double-center	500	58.00	88.40	98.20	12.51
Overall	1500	56.73	88.27	99.40	12.51

Note: $100p_\epsilon$ is the percentage of runs within $\epsilon\%$ of the per-instance best ALNS objective; $\Delta^{\max}$ is the maximum observed deviation.

6. Conclusion

This paper develops an efficient Adaptive Large Neighborhood Search (ALNS) algorithm for the Flying Sidekick Traveling Salesman Problem (FSTSP). The proposed framework integrates complementary truck-only and mixed truck-drone destroy-repair operator groups with an intensive local-search procedure. To improve computational efficiency, Static Move Descriptors (SMD) reduce redundant move evaluations, while a dynamic penalty mechanism facilitates the exploration of solutions under drone endurance constraints. A two-cycle local-search strategy further balances solution quality and computational effort.

Extensive experiments demonstrate the effectiveness of these components. On the MC-set, ALNS recovers all 72 known optimal solutions in every one of 720 runs, averaging only 0.76 s per run. On Agatz-S, it matches or improves the DP reference objectives in 83.9% of runs, including improvements on some larger instances for which optimality had not been established. On Agatz-L, ALNS achieves statistically significant improvements over MD-SPP-H, EP-All, and SPP-All. Comparisons on the Ha-set, using the same computing environment, further confirm its competitive solution quality and runtime performance. Ablation experiments validate the contributions of the proposed components, while additional analyses demonstrate the algorithm’s robustness and reveal the effects of drone speed and endurance on operational performance.

Future research may extend the proposed framework to more complex truck-drone routing settings, including multiple drones, multiple trucks, multiple deliveries per flight, and customer time windows. Further improvements in neighborhood evaluation and parallel search could also enhance scalability to larger instances.

References

Accorsi, L., Vigo, D., 2024. Routing one million customers in a handful of minutes. *Computers & Operations Research* 164, 106562.

- Agatz, N., Bouman, P., Schmidt, M., 2018. Optimization approaches for the traveling salesman problem with drone. *Transportation Science* 52, 965–981.
- Beek, O., Raa, B., Dullaert, W., Vigo, D., 2018. An efficient implementation of a static move descriptor-based local search heuristic. *Computers & Operations Research* 94, 1–10.
- Bouman, P., Agatz, N., Schmidt, M., 2018. Dynamic programming approaches for the traveling salesman problem with drone. *Networks* 72, 528–542.
- De Freitas, J.C., Penna, P.H.V., 2020. A variable neighborhood search for flying sidekick traveling salesman problem. *International Transactions in Operational Research* 27, 267–290.
- Dell’Amico, M., Montemanni, R., Novellani, S., 2021. Benchmark instances and optimal solutions for the traveling salesman problem with drone. URL: <https://arxiv.org/abs/2107.13275>, [arXiv:2107.13275](https://arxiv.org/abs/2107.13275).
- Dell’Amico, M., Montemanni, R., Novellani, S., 2022. Exact models for the flying sidekick traveling salesman problem. *International Transactions in Operational Research* 29, 1360–1393.
- Gonzalez-R, P.L., Canca, D., Andrade-Pineda, J.L., Calle, M., Leon-Blanco, J.M., 2020. Truck-drone team logistics: A heuristic approach to multi-drop route planning. *Transportation Research Part C: Emerging Technologies* 114, 657–680.
- Ha, Q.M., Deville, Y., Pham, Q.D., Hà, M.H., 2018. On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies* 86, 597–621.
- Kundu, A., Gatica, R., Matis, T.I., 2020. Results for the implementation of EP-All and SPP-All heuristic.

- Kundu, A., Gatica Escobar, R., Matis, T.I., 2022. An efficient routing heuristic for a drone-assisted delivery problem. *IMA Journal of Management Mathematics* 33, 583–601.
- Le, T.H., Vu, D.M., Tran, N.K., Nguyen, D.C., Le, H.T., 2026. A 2-index stage-based formulation and a construct-merge-solve and adapt algorithm for the flying sidekick traveling salesman problem, in: *Lecture Notes in Networks and Systems*, pp. 73–85.
- Macrina, G., Di Puglia Pugliese, L., Guerriero, F., Laporte, G., 2020. Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies* 120, 102762.
- Mahmoudinazlou, S., Kwon, C., 2024. A hybrid genetic algorithm with type-aware chromosomes for traveling salesman problems with drone. *European Journal of Operational Research* 318, 719–739.
- Mara, S.T.W., Rifai, A.P., Sopha, B.M., 2022. An adaptive large neighborhood search heuristic for the flying sidekick traveling salesman problem with multiple drops. *Expert Systems with Applications* 205, 117647.
- Moshref-Javadi, M., Winkenbach, M., 2021. Applications and research avenues for drone-based models in logistics: A classification and review. *Expert Systems with Applications* 177, 114854.
- Murray, C.C., Chu, A.G., 2015. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies* 54, 86–109.
- Nguyen, T.C., Nguyen, H.D., Le, H.T., Kaneko, S., 2022. Residents’ preferred measures and willingness-to-pay for improving urban air quality: A case study of Hanoi city, Vietnam. *Journal of Economics and Development* 24, 262–275.
- PassMark Software, 2020. Professional CPU benchmarks. URL: <https://www.passmark.com/index.html>. accessed September 26, 2026; single-thread ratings from the processor-specific pages at cpubenchmark.net.

- Roberti, R., Ruthmair, M., 2021. Exact methods for the traveling salesman problem with drone. *Transportation Science* 55, 315–335.
- Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science* 40, 455–472.
- Schaumann, S.K., Kundu, A., Pina-Pardo, J.C., Winkenbach, M., Gatica, R.A., Wagner, S.M., Matis, T.I., 2025. The flying sidekick traveling salesman problem with multiple drops: An effective heuristic approach. *Computers & Industrial Engineering* 210, 111523.
- Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research* 40, 475–489.
- Zachariadis, E.E., Kiranoudis, C.T., 2010. A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. *Computers & Operations Research* 37, 2089–2105.

Supplementary Material for *An Efficient Adaptive Large Neighborhood Search Algorithm for the Flying Sidekick Traveling Salesman Problem*

Duc D. Vu, Le Thi Hong, Tran Nam Khanh, Nguyen Dinh Cong, and Vu Duc Minh

S1. Move Evaluation

Each neighborhood operator modifies either the truck route, the set of drone sorties, or both. To support efficient neighborhood exploration, we evaluate the objective variation of every candidate move in $O(1)$ time after an $O(n)$ preprocessing step on the current solution.

More precisely, once a solution is fixed, we precompute: (i) the truck travel time between any two truck stages, so that θ_{ij} can be queried in constant time for any pair of stages $i < j$; (ii) the drone travel time θ'_s of each sortie s ; and (iii) the constant-time references needed to identify the few sorties that may be affected by a local modification, such as the sortie covering a given truck stage or the sorties immediately adjacent to a given sortie. Therefore, each move only changes a constant number of truck arcs and a constant number of sortie-delay terms.

For convenience, we use the following local truck-route primitives throughout this section. Removing the customer at truck stage k induces the change

$$\delta_k^{\text{rm}} = \tau_{c_{k-1}c_{k+1}} - \tau_{c_{k-1}c_k} - \tau_{c_kc_{k+1}}.$$

Inserting a customer x between truck stages $i - 1$ and i induces the change

$$\delta_i^{\text{ins}}(x) = \tau_{c_{i-1}x} + \tau_{xc_i} - \tau_{c_{i-1}c_i}.$$

When needed, a direct truck-route change can be written as a combination of one removal term and one insertion term, in that order.

Definition 1 (delay function). For readability, the move-evaluation expressions below are written for the Agatz configuration $SL = SR = 0$. For the Murray MC-set configuration, the implementation evaluates the same expressions with the applicable SL and SR terms described in Section 3 of the main article; these are constant-time additions and do not change the

complexity of a move evaluation. Let $\theta_{l_s r_s}$ and θ'_s be the truck and drone travel durations, respectively, between the launch stage l_s and the recovery stage r_s of a sortie s . The delay contribution of sortie s is

$$\mathcal{D}(\theta_{l_s r_s}, \theta'_s) = [\theta'_s - \theta_{l_s r_s}]^+ + \lambda[\max\{\theta_{l_s r_s}, \theta'_s\} - DTL]^+. \quad (\text{S1})$$

Proposition 1 (change in delay under perturbations). Let $\Delta t, \Delta t' \in \mathbb{R}$ be the changes in truck and drone travel durations of a sortie s . The corresponding change in its delay contribution is

$$\begin{aligned} \omega(s, \Delta t, \Delta t') &= \mathcal{D}(\theta_{l_s r_s} + \Delta t, \theta'_s + \Delta t') - \mathcal{D}(\theta_{l_s r_s}, \theta'_s) \\ &= [\theta'_s + \Delta t' - (\theta_{l_s r_s} + \Delta t)]^+ - [\theta'_s - \theta_{l_s r_s}]^+ \\ &\quad + \lambda \left([\max\{\theta'_s + \Delta t', \theta_{l_s r_s} + \Delta t\} - DTL]^+ - [\max\{\theta'_s, \theta_{l_s r_s}\} - DTL]^+ \right). \end{aligned} \quad (\text{S2})$$

For convenience, let $\text{cover}(k)$ denote the unique sortie whose launch–recovery interval contains truck stage k , when such a sortie exists. Similarly, let $\text{prev}(s)$ and $\text{next}(s)$ denote the sorties immediately before and after s that share its launch or recovery stage, respectively, when such sorties exist.

We now apply Proposition 1 to the six neighborhood moves.

AD move. This move removes the truck customer at stage k and creates a new drone sortie with launch stage k_1 and recovery stage k_2 .

Objective change. $\Delta_{\text{AD}} = \delta_k^{\text{rm}} + A_k^{\text{AD}} + \mathcal{D}(\theta_{k_1 k_2} + B_{k_1, k, k_2}^{\text{AD}}, \tau'_{c_{k_1} c_k} + \tau'_{c_k c_{k_2}})$, where

$$\begin{aligned} A_k^{\text{AD}} &= \begin{cases} \omega(\text{cover}(k), \delta_k^{\text{rm}}, 0), & \text{if } \text{cover}(k) \text{ exists,} \\ 0, & \text{otherwise,} \end{cases} \\ B_{k_1, k, k_2}^{\text{AD}} &= \begin{cases} \delta_k^{\text{rm}}, & \text{if } k_1 < k < k_2, \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

Interpretation. The term δ_k^{rm} is the truck shortcut after removing c_k . If stage k lies inside another sortie interval, its truck duration is updated by A_k^{AD} . The last term is the delay of the new sortie, with truck duration adjusted by $B_{k_1, k, k_2}^{\text{AD}}$ when $k_1 < k < k_2$. Hence Δ_{AD} is evaluated in $O(1)$.

SD1 move. The SD1 operator exchanges the drone-served customers of two distinct sorties u and v while keeping their launch and recovery stages unchanged. Let $u = (c_{u1}, c_{u2}, c_{u3})$, $v = (c_{v1}, c_{v2}, c_{v3})$.

Objective change. Define

$$\begin{aligned}\Delta'_u &= \tau'_{c_{u1}c_{v2}} + \tau'_{c_{v2}c_{u3}} - \tau'_{c_{u1}c_{u2}} - \tau'_{c_{u2}c_{u3}}, \\ \Delta'_v &= \tau'_{c_{v1}c_{u2}} + \tau'_{c_{u2}c_{v3}} - \tau'_{c_{v1}c_{v2}} - \tau'_{c_{v2}c_{v3}}.\end{aligned}$$

Then $\Delta_{\text{SD1}} = \omega(u, 0, \Delta'_u) + \omega(v, 0, \Delta'_v)$.

Interpretation. SD1 leaves the truck route unchanged and only updates the drone durations of sorties u and v . Hence Δ_{SD1} is the sum of two delay updates and is computed in $O(1)$ time.

SD2 move. The SD2 move applies to two consecutive sorties u and v with $v = u + 1$ and $r_u = l_v$. The move shifts their shared truck stage to a new feasible stage k , with $l_u < k < r_v$. Before the move, $u = (c_{u1}, c_{u2}, c_{u3})$, $v = (c_{v1}, c_{v2}, c_{v3})$, where $c_{u3} = c_{v1}$. After the move, the updated sorties are $u' = (c_{u1}, c_{u2}, c_k)$, $v' = (c_k, c_{v2}, c_{v3})$. The truck- and drone-duration changes are $\Delta t_u = \theta_{l_u k} - \theta_{l_u r_u}$; $\Delta'_u = \tau'_{c_{u2}c_k} - \tau'_{c_{u2}c_{u3}}$, $\Delta t_v = \theta_{k r_v} - \theta_{l_v r_v}$, $\Delta'_v = \tau'_{c_k c_{v2}} - \tau'_{c_{v1} c_{v2}}$.

Objective change. $\Delta_{\text{SD2}} = \omega(u, \Delta t_u, \Delta'_u) + \omega(v, \Delta t_v, \Delta'_v)$.

Interpretation. SD2 changes the shared truck stage between two consecutive sorties. Each sortie may therefore change its truck duration and one drone arc. No other part of the solution is affected, so the move is evaluated in $O(1)$ time.

SD3 move. The SD3 move swaps the launch-stage customer and the recovery-stage customer of the same sortie s . Let $l = l_s$ and $r = r_s$, and let the original sortie be $s = (c_l, c_{s2}, c_r)$. After the move, the sortie becomes $s' = (c_r, c_{s2}, c_l)$.

Objective change. The objective variation is decomposed into four terms:

$$\Delta_{\text{SD3}} = \Delta_{\text{trk}}^{\text{SD3}} + \omega(s, \Delta t_s^{\text{SD3}}, \Delta'_s{}^{\text{SD3}}) + A_{\text{prev}}^{\text{SD3}} + A_{\text{next}}^{\text{SD3}}.$$

The direct truck-route contribution is

$$\Delta_{\text{trk}}^{\text{SD3}} = \begin{cases} \delta_r^{\text{rm}} + \delta_l^{\text{ins}}(c_r), & \text{if } r - l = 1, \\ \delta_r^{\text{rm}} + \delta_l^{\text{ins}}(c_r) + \delta_r^{\text{ins}}(c_l) - \tau_{c_{r-1}c_{r+1}}, & \text{if } r - l > 1. \end{cases}$$

The truck- and drone-duration changes of sortie s are

$$\begin{aligned}\Delta t_s^{\text{SD3}} &= \begin{cases} \tau_{c_r c_l} - \tau_{c_l c_r}, & \text{if } r - l = 1, \\ \tau_{c_r c_{l+1}} + \tau_{c_{r-1} c_l} - \tau_{c_l c_{l+1}} - \tau_{c_{r-1} c_r}, & \text{if } r - l > 1, \end{cases} \\ \Delta'_s{}^{\text{SD3}} &= \tau'_{c_r c_{s2}} + \tau'_{c_{s2} c_l} - \tau'_{c_l c_{s2}} - \tau'_{c_{s2} c_r}.\end{aligned}$$

The additional adjustment for a previous neighboring sortie is

$$A_{\text{prev}}^{\text{SD3}} = \begin{cases} \omega\left(\text{prev}(s), \tau_{c_{l-1}c_r} - \tau_{c_{l-1}c_l}, \tau'_{c_{p2}c_r} - \tau'_{c_{p2}c_l}\right), & \text{if prev}(s) \text{ exists,} \\ 0, & \text{otherwise,} \end{cases}$$

where c_{p2} denotes the drone-served customer of $\text{prev}(s)$. The additional adjustment for a next neighboring sortie is

$$A_{\text{next}}^{\text{SD3}} = \begin{cases} \omega\left(\text{next}(s), \tau_{c_l c_{r+1}} - \tau_{c_r c_{r+1}}, \tau'_{c_l c_{n2}} - \tau'_{c_r c_{n2}}\right), & \text{if next}(s) \text{ exists,} \\ 0, & \text{otherwise,} \end{cases}$$

where c_{n2} denotes the drone-served customer of $\text{next}(s)$.

Interpretation. SD3 removes c_r from stage r and inserts it at stage l ; when $r - l > 1$, customer c_l is then inserted at stage r . The same swap also changes the truck and drone durations of sortie s . If the launch or recovery stage is shared with a neighboring sortie, that sortie is updated as well. Hence the move is evaluated in $O(1)$ time.

SD4 move. The SD4 move swaps the drone-served customer of a sortie with either its launch-stage customer or its recovery-stage customer. We present the launch-stage case; the recovery-stage case follows symmetrically. Let s be the considered sortie, with launch stage $l = l_s$ and recovery stage $r = r_s$. Before the move, $s = (c_l, c_{s2}, c_r)$, and after swapping c_l with c_{s2} , the updated sortie becomes $s' = (c_{s2}, c_l, c_r)$.

Objective change. The objective variation of the launch-stage case is

$$\Delta_{\text{SD4,L}} = \Delta_{\text{trk}}^{\text{SD4,L}} + \omega(s, \Delta t_s^{\text{SD4,L}}, \Delta'_s{}^{\text{SD4,L}}) + A_{\text{prev}}^{\text{SD4,L}}.$$

The direct truck-route contribution is $\Delta_{\text{trk}}^{\text{SD4,L}} = \delta_l^{\text{rm}} + \delta_l^{\text{ins}}(c_{s2})$. The truck- and drone-duration changes of sortie s are

$$\begin{aligned} \Delta t_s^{\text{SD4,L}} &= \tau_{c_{s2}c_{l+1}} - \tau_{c_l c_{l+1}}, \\ \Delta'_s{}^{\text{SD4,L}} &= \tau'_{c_{s2}c_l} + \tau'_{c_l c_r} - \tau'_{c_l c_{s2}} - \tau'_{c_{s2}c_r}. \end{aligned}$$

The additional correction for a previous neighboring sortie is

$$A_{\text{prev}}^{\text{SD4,L}} = \begin{cases} \omega\left(\text{prev}(s), \tau_{c_{l-1}c_{s2}} - \tau_{c_{l-1}c_l}, \tau'_{c_{p2}c_{s2}} - \tau'_{c_{p2}c_l}\right), & \text{if prev}(s) \text{ exists,} \\ 0, & \text{otherwise,} \end{cases}$$

where c_{p2} denotes the drone-served customer of $\text{prev}(s)$.

Interpretation. SD4 removes c_l from stage l and inserts c_{s2} at the same stage. It also changes the truck and drone durations of sortie s , and it may affect one previous neighboring sortie if that sortie recovers at stage l . Hence the move is evaluated in $O(1)$ time.

RD move. The RD move removes an existing drone sortie s and reinserts its drone-served customer into the truck route. Let $s = (c_{s1}, c_{s2}, c_{s3})$, where c_{s2} is the customer currently served by the drone. Suppose that c_{s2} is inserted between truck stages $k - 1$ and k .

Objective change. The objective variation is $\Delta_{\text{RD}} = -\mathcal{D}(\theta_{l_s r_s}, \theta'_s) + \delta_k^{\text{ins}}(c_{s2}) + A_k^{\text{RD}}$, where

$$A_k^{\text{RD}} = \begin{cases} \omega(\text{cover}(k), \delta_k^{\text{ins}}(c_{s2}), 0), & \text{if } \text{cover}(k) \text{ exists and } \text{cover}(k) \neq s, \\ 0, & \text{otherwise.} \end{cases}$$

Interpretation. The first term removes the delay contribution of sortie s ; the second inserts c_{s2} back into the truck route. If the insertion position lies inside another sortie interval, its truck duration is updated by A_k^{RD} . Hence Δ_{RD} is evaluated in $O(1)$ time.

S2. Algorithm and Penalty Parameter Settings

The ALNS parameters are listed in Tables S1 and S2. The settings in Table S1 largely follow Mara et al. (2022), with minor adjustments to the final temperature and destroy ratio.

Table S1: ALNS parameter configuration

Parameter	Description	Value
T_1	Initial temperature	200
T_0	Final temperature	10
α	Cooling rate	0.99999
BZ	Boltzmann scaling factor	0.1
$\Omega_1, \Omega_2, \Omega_3, \Omega_4$	Weight rewards	0.30, 0.15, 0, 0
λ	Initial penalty coefficient	1
λ'	Decay parameter	0.9
f	Destroy ratio	Random in [0.25, 0.75]

Table S2 reports the initial ranges and calibrated values, tuned by `irace`, used in the final experiments for the dynamic penalty mechanism: update interval, target proportion of feasible candidate solutions, penalty increase

factor, and penalty decrease factor. The tuning budget was 10,000 algorithm runs on a single machine. The training set was derived from the benchmark instances used in the main experiments. For each instance, four additional perturbed instances were generated. In each perturbed instance, every customer coordinate (x, y) was independently multiplied by a random factor sampled uniformly from $[0.9, 1.1]$. Only the perturbed instances were shuffled before being passed to `irace`. During tuning, the algorithm seed was fixed to isolate the effect of the penalty parameters from stochastic variation in the search.

Table S2: Dynamic penalty parameters calibrated by `irace`

Parameter	Description	Initial range	Selected value
I_λ	Penalty update interval	(10, 500)	300
$\rho^\star$	Target feasible proportion	(0.05, 1.00)	0.1
η^+	Penalty increase factor	(1.05, 2.00)	1.2
η^-	Penalty decrease factor	(0.50, 0.95)	0.8

S3. Benchmark Dataset Details

This supplementary material describes the benchmark instances and the settings used in our computational study. The dataset sources are provided in Section 5.1 of the main article.

MC-set. The MC-set comprises 36 ten-customer FSTSP instances introduced by [Murray and Chu \(2015\)](#). The original test problems are available from the author upon request. Each instance is evaluated at $DTL \in \{20, 40\}$, giving 72 scenarios. Launch and recovery service times are one time unit, except that launch service at the initial depot is zero. Launch preparation does not consume drone endurance, whereas recovery service and drone waiting at the recovery location do. We compare ALNS with the PS2 optimal values reported by [Dell’Amico et al. \(2021\)](#). This set is used for the exact-solution validation in RQ1.

Agatz-S. The Agatz-S subset contains 90 small, uniformly distributed instances, with 10 instances for each size $n = 11, \dots, 19$. The coordinates and DP reference results are taken from the benchmark instances studied by [Bouman et al. \(2018\)](#). The main setting uses Euclidean travel times, a drone speed twice that of the truck, no endurance limit, and zero launch and recovery service times. We use the published DP solutions as reference

objectives in RQ1. The DP study did not establish optimal solutions for some larger instances ($n \geq 18$), so not all reported reference values can be treated as proven optima.

Agatz-L. The Agatz-L subset comprises 150 instances (Agatz et al., 2018). The five tested sizes are $n = 50, 75, 100, 175$, and 250. Each size includes 10 uniform, 10 single-center, and 10 double-center instances. The main comparison uses Euclidean truck and drone travel times, a drone twice as fast as the truck, unlimited endurance ($DTL = \infty$), and $SL = SR = 0$. Published FSTSP-configured MD-SPP-H (Schaumann et al., 2025), EP-All, and SPP-All (Kundu et al., 2020) values are used for the RQ1 comparison. This subset is also used for the ablation studies (RQ2), the drone speed and endurance analysis (RQ3), and the repeated-run robustness analysis (RQ4). The finite-endurance and alternative-speed settings in RQ2 and RQ3 are evaluated separately from the main comparison scenarios.

Ha-set. The Ha-set contains 60 instances (Ha et al., 2018): B1–D10 ($n = 50$) and E1–G10 ($n = 100$). We retain the original coordinates, Manhattan truck travel, Euclidean drone travel, a speed of 40 km/h for both vehicles, and 20-minute drone endurance. We set $SL = SR = 0$, make all customers drone-eligible, forbid loops and truck-node revisits, and count drone waiting toward endurance. We also use this set to compare with ALNS, MD-SPP-H, and HGA-TAC for RQ1 comparisons of solution quality and runtime.

S4. Per-instance Agatz-L results

Table S3 reports the 150 Agatz-L instances. “ALNS best” is the best solution found over the ten ALNS runs. The MD-SPP-H, EP-All, and SPP-All columns reproduce their published values; they were not rerun locally. Boldface separately identifies the lowest displayed value in the best-objective and mean-objective comparison groups for each instance. It does not assert best-known status beyond this comparison set.

Table S3: Per-instance Agatz-L comparison. ALNS results are measured locally; MD-SPP-H, EP-All, and SPP-All values are published results. Bold indicates the lowest displayed value within each objective-summary group.

Distribution	n	ID	Best objective				Mean objective	
			ALNS	MD-SPP-H	EP-All	SPP-All	ALNS	MD-SPP-H
Uniform	50	71	385.53	387.70	393.70	389.90	389.01	390.80
Uniform	50	72	423.57	420.60	454.70	454.70	426.23	425.80

Table S3 – continued

Distribution	n	ID	Best objective				Mean objective	
			ALNS	MD-SPP-H	EP-All	SPP-All	ALNS	MD-SPP-H
Uniform	50	73	393.76	393.80	410.60	409.80	398.35	396.50
Uniform	50	74	412.12	409.00	422.60	420.50	413.79	412.00
Uniform	50	75	414.73	413.00	441.00	441.00	416.85	416.30
Uniform	50	76	372.54	370.70	376.10	376.10	373.78	374.80
Uniform	50	77	415.24	415.80	439.60	434.20	420.90	419.90
Uniform	50	78	413.61	410.50	438.20	438.20	415.79	422.00
Uniform	50	79	377.84	374.40	397.30	397.30	381.10	380.00
Uniform	50	80	364.57	361.80	376.00	376.00	368.10	364.20
1-center	50	71	427.90	428.20	446.60	446.60	429.66	432.70
1-center	50	72	484.52	483.00	484.30	484.30	488.49	484.20
1-center	50	73	360.89	361.50	376.80	376.80	362.14	365.90
1-center	50	74	530.51	528.20	582.50	550.70	534.17	532.10
1-center	50	75	552.13	552.10	589.10	565.40	556.33	561.10
1-center	50	76	540.27	539.00	573.70	563.50	544.17	550.00
1-center	50	77	426.77	425.30	442.10	442.10	428.72	430.60
1-center	50	78	541.38	541.40	558.10	558.10	543.24	557.00
1-center	50	79	433.24	432.70	465.40	451.60	436.82	441.50
1-center	50	80	551.41	550.80	573.80	571.10	556.55	558.50
2-center	50	71	803.60	802.00	846.30	846.30	807.35	805.40
2-center	50	72	789.92	788.50	832.70	832.70	792.80	797.50
2-center	50	73	717.02	717.30	762.20	762.20	722.07	730.00
2-center	50	74	702.15	702.00	707.50	707.50	706.93	705.50
2-center	50	75	824.28	825.30	887.40	883.90	830.09	833.50
2-center	50	76	798.01	795.50	847.20	847.20	801.30	803.60
2-center	50	77	730.27	728.00	742.90	736.90	734.36	734.80
2-center	50	78	738.47	734.90	756.90	756.90	743.15	748.30
2-center	50	79	760.49	757.50	823.90	823.90	768.65	772.80
2-center	50	80	862.73	861.70	883.80	883.80	865.73	865.70
Uniform	75	81	464.38	459.10	476.60	470.40	468.32	465.90
Uniform	75	82	436.86	432.90	454.40	454.40	439.64	441.80
Uniform	75	83	439.34	435.20	456.20	456.10	442.82	442.70
Uniform	75	84	469.44	463.10	467.80	463.80	471.90	469.20
Uniform	75	85	481.06	478.40	506.90	499.20	484.15	486.80
Uniform	75	86	456.50	451.50	493.70	493.00	461.29	460.40
Uniform	75	87	464.51	460.30	475.10	470.80	469.54	464.60
Uniform	75	88	457.81	453.90	520.20	517.90	462.71	460.40
Uniform	75	89	424.96	421.80	467.00	467.00	427.09	426.50
Uniform	75	90	453.73	452.50	470.00	468.90	456.65	457.20
1-center	75	81	687.05	695.60	767.70	701.30	690.25	715.30
1-center	75	82	564.32	560.50	624.40	624.40	569.71	580.10
1-center	75	83	618.03	616.60	631.90	631.90	620.60	624.80
1-center	75	84	679.05	684.20	715.50	710.80	693.48	706.50
1-center	75	85	585.03	580.30	588.10	585.60	587.06	589.80
1-center	75	86	702.45	720.90	769.30	762.50	709.96	737.40
1-center	75	87	686.28	687.00	739.20	738.30	689.86	702.00
1-center	75	88	689.58	689.90	695.30	692.50	695.68	699.90
1-center	75	89	631.90	639.10	697.40	665.40	636.92	650.00
1-center	75	90	700.43	693.10	776.80	733.60	707.23	711.90
2-center	75	81	863.43	858.30	871.40	864.40	872.88	867.60
2-center	75	82	886.95	886.10	936.40	936.40	893.33	902.00
2-center	75	83	872.17	862.60	881.00	878.60	876.61	872.40
2-center	75	84	1119.99	1115.10	1220.50	1220.50	1126.35	1138.50

Table S3 – continued

Distribution	n	ID	Best objective				Mean objective	
			ALNS	MD-SPP-H	EP-All	SPP-All	ALNS	MD-SPP-H
2-center	75	85	1055.39	1054.60	1088.80	1088.50	1064.91	1064.20
2-center	75	86	924.58	913.20	943.80	937.90	929.18	933.50
2-center	75	87	942.32	941.70	963.00	963.00	947.47	947.20
2-center	75	88	1062.25	1069.00	1122.50	1119.30	1074.75	1101.50
2-center	75	89	911.32	908.50	1017.30	1017.30	914.98	931.50
2-center	75	90	842.38	836.50	865.20	865.20	852.45	848.50
Uniform	100	91	551.05	547.40	575.70	569.70	554.97	558.90
Uniform	100	92	486.18	487.40	529.30	523.60	492.55	498.50
Uniform	100	93	497.58	491.20	547.00	535.60	502.36	503.50
Uniform	100	94	518.97	512.20	543.80	543.80	522.73	526.60
Uniform	100	95	537.62	528.30	591.70	591.10	545.39	541.00
Uniform	100	96	540.79	546.50	558.00	556.10	545.58	553.30
Uniform	100	97	552.35	554.40	592.80	590.50	556.16	565.20
Uniform	100	98	514.20	514.70	517.00	516.80	519.41	520.40
Uniform	100	99	544.54	536.90	574.50	558.70	546.75	546.20
Uniform	100	100	522.22	516.50	578.70	577.30	524.50	530.80
1-center	100	91	777.87	778.50	827.40	826.50	781.63	793.90
1-center	100	92	851.15	854.70	890.70	887.20	856.94	877.00
1-center	100	93	843.35	844.90	934.40	934.30	849.66	861.50
1-center	100	94	929.11	947.80	1015.30	997.80	939.47	971.00
1-center	100	95	729.46	738.90	752.70	748.90	734.00	746.00
1-center	100	96	806.94	809.50	866.10	861.80	813.62	829.70
1-center	100	97	824.06	831.60	851.40	848.00	835.18	845.50
1-center	100	98	758.90	785.70	840.90	833.10	768.69	803.50
1-center	100	99	661.56	662.10	679.70	679.10	671.16	678.20
1-center	100	100	752.14	758.30	797.20	793.90	760.17	768.10
2-center	100	91	1033.40	1042.20	1080.70	1073.60	1044.95	1055.00
2-center	100	92	977.08	988.90	1028.40	1018.70	987.34	1009.70
2-center	100	93	1000.75	990.90	1052.50	1049.90	1011.14	1005.90
2-center	100	94	1054.54	1056.30	1076.60	1074.00	1063.78	1077.00
2-center	100	95	1102.34	1091.80	1132.50	1130.80	1104.49	1115.70
2-center	100	96	1153.24	1149.40	1237.70	1182.80	1165.45	1174.80
2-center	100	97	1209.06	1214.60	1259.90	1259.90	1219.30	1237.00
2-center	100	98	1131.39	1118.80	1164.90	1160.80	1138.22	1149.00
2-center	100	99	1001.96	1003.00	1028.50	1025.40	1014.62	1015.10
2-center	100	100	1118.59	1122.90	1137.00	1113.60	1126.55	1138.00
Uniform	175	101	675.44	684.00	734.90	730.40	683.81	699.20
Uniform	175	102	671.62	683.10	728.30	727.80	676.50	704.70
Uniform	175	103	667.23	674.30	727.00	725.80	677.42	697.30
Uniform	175	104	654.57	657.30	703.60	685.60	663.99	670.20
Uniform	175	105	682.35	695.80	726.90	725.60	690.29	708.30
Uniform	175	106	673.71	678.10	691.70	688.50	683.12	689.40
Uniform	175	107	653.62	654.10	661.40	653.70	660.24	659.00
Uniform	175	108	684.13	677.40	728.40	718.70	689.29	689.60
Uniform	175	109	684.35	686.20	716.90	715.60	690.43	705.60
Uniform	175	110	641.96	637.00	654.10	650.00	651.14	644.20
1-center	175	101	1007.73	1025.90	1051.60	1021.80	1024.80	1049.30
1-center	175	102	1039.49	1052.70	1071.60	1058.50	1057.69	1065.80
1-center	175	103	1073.60	1117.80	1071.60	1058.50	1088.50	1152.00
1-center	175	104	1045.13	1073.40	1132.80	1119.40	1053.86	1088.80
1-center	175	105	1064.19	1074.10	1093.70	1060.60	1074.61	1115.20
1-center	175	106	1092.52	1107.60	1143.00	1138.10	1103.71	1132.50

Table S3 – continued

Distribution	n	ID	Best objective				Mean objective	
			ALNS	MD-SPP-H	EP-All	SPP-All	ALNS	MD-SPP-H
1-center	175	107	1115.75	1171.30	1217.90	1217.90	1145.29	1200.90
1-center	175	108	1075.43	1115.80	1142.60	1126.90	1091.08	1132.40
1-center	175	109	1063.43	1095.30	1129.90	1121.50	1086.52	1112.30
1-center	175	110	1040.79	1077.00	1103.90	1103.20	1048.63	1093.20
2-center	175	101	1254.09	1269.10	1344.90	1344.30	1273.78	1307.30
2-center	175	102	1456.36	1491.40	1520.60	1502.20	1464.53	1526.00
2-center	175	103	1510.41	1524.30	1596.20	1592.90	1522.64	1544.40
2-center	175	104	1366.65	1397.00	1455.00	1445.40	1388.44	1443.70
2-center	175	105	1496.72	1543.50	1604.70	1602.60	1540.98	1572.00
2-center	175	106	1534.35	1549.10	1597.60	1597.10	1558.74	1573.30
2-center	175	107	1388.39	1389.70	1430.20	1426.30	1399.41	1405.20
2-center	175	108	1488.39	1511.50	1555.50	1548.10	1515.38	1545.60
2-center	175	109	1425.30	1447.50	1518.10	1516.20	1450.13	1522.30
2-center	175	110	1548.42	1541.10	1602.00	1583.30	1571.19	1567.80
Uniform	250	111	797.96	813.20	822.00	820.20	811.81	822.70
Uniform	250	112	797.41	804.90	806.30	802.00	808.68	812.40
Uniform	250	113	787.71	795.30	820.90	818.90	796.88	814.20
Uniform	250	114	784.89	805.80	830.90	830.60	795.58	817.90
Uniform	250	115	798.01	818.00	845.30	844.60	814.75	830.30
Uniform	250	116	807.60	807.70	831.10	826.40	817.02	819.40
Uniform	250	117	798.83	823.00	840.90	840.90	818.62	828.40
Uniform	250	118	764.88	761.30	790.10	780.20	770.10	774.20
Uniform	250	119	813.66	833.90	871.40	868.80	825.27	851.80
Uniform	250	120	793.67	789.00	811.00	805.20	803.50	801.90
1-center	250	111	1255.99	1279.50	1279.90	1270.80	1273.08	1284.10
1-center	250	112	1233.62	1278.60	1300.10	1264.30	1245.35	1300.10
1-center	250	113	1341.07	1379.70	1405.40	1389.20	1357.80	1396.10
1-center	250	114	1290.73	1358.50	1369.00	1364.00	1298.78	1363.90
1-center	250	115	1344.32	1342.80	1335.80	1328.90	1354.27	1353.30
1-center	250	116	1299.77	1303.70	1358.20	1335.40	1309.76	1351.40
1-center	250	117	1236.67	1277.10	1295.40	1291.20	1246.83	1287.50
1-center	250	118	1244.69	1263.30	1277.50	1271.30	1265.20	1272.60
1-center	250	119	1188.66	1189.00	1194.70	1190.60	1202.14	1199.60
1-center	250	120	1211.87	1272.90	1313.40	1312.30	1228.48	1303.50
2-center	250	111	1716.11	1762.60	1774.70	1764.50	1730.39	1779.90
2-center	250	112	1712.75	1807.60	1819.30	1802.90	1791.79	1829.30
2-center	250	113	1661.20	1706.30	1747.20	1728.60	1686.99	1730.30
2-center	250	114	1699.41	1737.60	1772.80	1758.50	1726.74	1752.10
2-center	250	115	1828.04	1869.50	1973.50	1949.40	1862.28	1897.60
2-center	250	116	1802.50	1807.10	1808.40	1808.00	1822.78	1832.60
2-center	250	117	1762.10	1817.40	1831.70	1821.20	1799.97	1836.30
2-center	250	118	1775.68	1813.80	1841.10	1814.20	1807.34	1823.20
2-center	250	119	1747.74	1820.40	1823.40	1805.10	1762.47	1851.30
2-center	250	120	1718.07	1761.90	1827.00	1790.20	1751.11	1787.40

S5. Per-instance Ha-set results

Table S4 reports best and mean objectives and mean process wall times for all 60 Ha-set instances. Each method has ten runs per instance under

$SL = SR = 0$, all-customer drone eligibility, and $DTL = 20$ minutes. Objectives are computed from independently checked schedules. Boldface identifies the lowest displayed objective within each best or mean comparison group; it does not assert optimality or best-known status outside this cohort. Process times include startup and warmup under the methods’ native stopping rules. MD–SPP–H’s initial-TSP preparation is cached once per instance and excluded from its process time.

Table S4: Per-instance Ha-set comparison. All methods are rerun on the same worker. Bold indicates the lowest displayed value within each objective-summary group. Process times include startup and warmup; MD TSP preparation is cached once per instance and excluded from its process time.

n	ID	Best objective			Mean objective			Mean process time (s)		
		ALNS	MD–SPP–H	HGA–TAC	ALNS	MD–SPP–H	HGA–TAC	ALNS	MD–SPP–H	HGA–TAC
50	B1	94.26	95.33	97.69	94.46	96.50	98.53	12.88	10.36	39.25
50	B2	92.52	92.72	93.17	92.63	93.19	94.11	12.95	6.14	35.76
50	B3	93.36	93.37	93.41	93.39	93.96	93.51	12.92	8.28	41.36
50	B4	93.25	93.33	97.74	93.29	95.02	98.17	12.77	10.42	41.22
50	B5	90.47	91.73	91.98	90.68	92.61	92.34	12.71	8.59	39.59
50	B6	93.22	93.50	96.24	93.35	95.19	96.51	13.00	8.85	33.74
50	B7	93.25	93.25	95.36	93.25	94.46	97.84	12.65	9.20	38.01
50	B8	93.28	93.30	94.84	93.29	93.98	96.05	12.86	8.39	37.54
50	B9	87.83	87.83	90.72	87.83	89.40	94.47	12.97	7.88	37.63
50	B10	91.81	91.65	94.27	91.97	92.91	94.97	12.88	6.97	34.99
50	C1	187.52	187.52	187.52	187.52	187.66	187.98	12.78	6.43	36.10
50	C2	175.26	175.26	178.90	175.32	176.62	179.05	12.84	6.98	33.53
50	C3	185.18	186.39	186.39	185.44	186.52	187.28	14.13	5.88	34.44
50	C4	185.69	186.54	190.59	185.69	189.00	192.09	13.12	9.16	42.76
50	C5	191.35	191.35	193.20	191.96	193.88	196.70	14.00	7.47	48.74
50	C6	204.26	204.26	212.12	204.80	206.86	216.59	15.10	9.17	37.00
50	C7	196.24	197.73	197.81	196.57	197.92	198.17	14.00	5.35	36.31
50	C8	198.97	198.97	210.28	200.11	203.74	211.08	14.08	9.46	34.77
50	C9	196.33	197.26	200.38	197.00	199.42	200.71	14.13	7.57	35.51
50	C10	190.78	191.17	193.92	190.88	192.59	195.31	13.98	7.68	40.28
50	D1	271.75	272.46	292.33	271.75	276.44	294.93	14.29	7.76	34.71
50	D2	276.07	278.60	278.60	276.08	281.31	279.52	14.81	4.37	33.91
50	D3	263.83	263.83	263.83	263.83	265.59	269.99	14.34	6.97	35.30
50	D4	291.53	291.53	291.53	291.53	292.02	292.20	15.13	6.70	39.97
50	D5	286.73	286.73	287.50	286.73	289.14	289.61	14.48	4.94	36.76
50	D6	279.26	279.26	279.52	279.37	279.77	280.04	14.48	4.84	34.43
50	D7	288.39	288.39	290.41	289.56	288.60	290.90	14.84	4.63	39.42
50	D8	253.98	254.20	265.29	254.05	258.78	277.13	14.65	7.01	38.00
50	D9	279.27	279.48	290.52	279.29	281.67	294.34	14.50	8.65	38.86
50	D10	268.95	268.95	269.55	268.95	270.27	272.14	14.51	5.59	33.55
100	E1	135.79	136.12	139.15	137.25	138.40	140.27	43.04	84.31	73.69
100	E2	136.84	137.14	143.85	138.13	141.29	146.15	43.32	125.52	76.50
100	E3	135.92	138.88	143.23	136.48	140.82	143.90	43.22	104.57	65.33
100	E4	137.58	138.51	142.24	138.05	140.35	144.20	43.72	89.33	86.93
100	E5	136.63	138.06	140.25	137.81	139.78	142.20	43.51	71.86	78.50
100	E6	135.89	137.21	140.59	137.01	138.66	142.13	43.64	142.12	89.63

Table S4 – continued

n	ID	Best objective			Mean objective			Mean process time (s)		
		ALNS	MD– SPP–H	HGA– TAC	ALNS	MD– SPP–H	HGA– TAC	ALNS	MD– SPP–H	HGA– TAC
100	E7	135.70	136.70	141.20	137.04	139.62	143.67	42.89	100.10	88.11
100	E8	136.39	138.60	142.27	137.29	140.70	144.12	43.20	80.09	71.17
100	E9	138.55	138.79	141.74	139.27	140.37	144.15	43.51	108.27	79.24
100	E10	137.50	137.09	139.65	138.46	139.07	140.74	43.65	88.94	73.31
100	F1	265.06	265.91	274.79	266.74	271.31	276.84	42.92	90.15	96.36
100	F2	256.84	257.71	259.83	258.84	262.27	266.00	43.28	85.94	94.20
100	F3	257.86	260.40	265.42	258.80	265.71	268.55	42.53	56.68	66.28
100	F4	261.03	261.95	262.33	262.04	265.43	266.12	42.75	77.96	99.27
100	F5	263.56	267.44	281.90	265.41	275.96	282.29	42.83	77.83	69.89
100	F6	248.79	248.72	251.94	249.93	251.24	253.29	43.06	81.17	63.63
100	F7	262.67	265.06	262.92	264.33	265.93	264.54	42.77	71.28	82.28
100	F8	267.73	268.38	270.18	268.41	269.39	271.27	41.91	55.11	75.26
100	F9	261.57	263.63	268.19	262.39	269.08	273.77	42.87	82.45	82.25
100	F10	257.84	262.98	266.04	259.21	265.36	270.95	42.43	93.75	70.39
100	G1	367.44	371.33	367.85	368.78	373.23	371.83	43.80	79.87	79.89
100	G2	337.01	338.93	341.99	337.74	343.31	345.80	42.58	61.07	93.94
100	G3	364.31	376.03	378.77	364.86	380.16	380.49	44.04	62.75	77.45
100	G4	365.40	368.45	381.30	366.46	377.10	386.09	44.79	77.58	60.98
100	G5	357.83	363.58	370.44	360.49	368.86	371.96	42.65	63.96	59.48
100	G6	364.56	373.24	374.57	365.71	375.43	377.51	44.51	63.32	70.17
100	G7	352.12	353.20	355.17	352.25	355.66	357.42	43.69	52.52	60.41
100	G8	354.54	357.49	373.17	355.87	368.21	375.70	45.42	96.20	93.53
100	G9	374.61	388.44	392.64	375.35	390.36	397.86	44.86	66.90	81.03
100	G10	360.45	365.37	377.22	361.86	374.07	379.80	44.22	72.91	87.16

References

- Agatz, N., Bouman, P., Schmidt, M., 2018. Optimization approaches for the traveling salesman problem with drone. *Transportation Science* 52, 965–981.
- Bouman, P., Agatz, N., Schmidt, M., 2018. Dynamic programming approaches for the traveling salesman problem with drone. *Networks* 72, 528–542.
- Dell’Amico, M., Montemanni, R., Novellani, S., 2021. Benchmark instances and optimal solutions for the traveling salesman problem with drone. URL: <https://arxiv.org/abs/2107.13275>, [arXiv:2107.13275](https://arxiv.org/abs/2107.13275).
- Ha, Q.M., Deville, Y., Pham, Q.D., Hà, M.H., 2018. On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies* 86, 597–621.
- Kundu, A., Gatica, R., Matis, T.I., 2020. Results for the implementation of EP-All and SPP-All heuristic.

- Mara, S.T.W., Rifai, A.P., Sopha, B.M., 2022. An adaptive large neighborhood search heuristic for the flying sidekick traveling salesman problem with multiple drops. *Expert Systems with Applications* 205, 117647.
- Murray, C.C., Chu, A.G., 2015. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies* 54, 86–109.
- Schaumann, S.K., Kundu, A., Pina-Pardo, J.C., Winkenbach, M., Gatica, R.A., Wagner, S.M., Matis, T.I., 2025. The flying sidekick traveling salesman problem with multiple drops: An effective heuristic approach. *Computers & Industrial Engineering* 210, 111523.