IIMB-WP N0. 725/2025

WORKING PAPER NO: 725

Arc Routing Problems with Multiple Trucks and Drones: A Hybrid Genetic Algorithm

Abhay Sobhanan
Assistant Professor
Decision Sciences Area
Indian Institute of Management Bangalore
abhay.sobhanan@iimb.ac.in

Hadi Charkhgard
Associate Professor
Department of Industrial and Management Systems Engineering,
University of South Florida, Tampa, FL, USA
hcharkhgard@usf.edu

Changhyun Kwon
Associate Professor
Department of Industrial and Systems Engineering, KAIST,
Daejeon, Republic of Korea
chkwon@kaist.ac.kr

Year of Publication – September 2025

Arc Routing Problems with Multiple Trucks and Drones: A Hybrid Genetic Algorithm

Abhay Sobhanan[a], Hadi Charkhgard[b], and Changhyun Kwon[c]

[a]Indian Institute of Management Bangalore, Bannerghatta Road, Bengaluru, 560076, India
[b]Department of Industrial and Management Systems Engineering, University of South Florida, Tampa, FL, USA
[c]Department of Industrial and Systems Engineering, KAIST, Daejeon, 34141, Republic of Korea

Abstract

Arc-routing problems underpin numerous critical field operations, including power-line inspection, urban police patrolling, and traffic monitoring. In this domain, the Rural Postman Problem (RPP) is a fundamental variant in which a prescribed subset of edges or arcs in a network must be traversed. This paper investigates a generalized form of the RPP, called RPP-mTD, which involves a fleet of multiple trucks, each carrying multiple drones. The trucks act as mobile depots traversing a road network, from which drones are launched to execute simultaneous service, with the objective of minimizing the overall makespan. Given the combinatorial complexity of RPP-mTD, we propose a Hybrid Genetic Algorithm (HGA) that combines population-based exploration with targeted neighborhood searches. Solutions are encoded using a two-layer chromosome that represents: (i) an ordered, directed sequence of required edges, and (ii) their assignment to vehicles. A tailored segment-preserving crossover operator is introduced, along with multiple local search techniques to intensify the optimization. We benchmark the proposed HGA against established single truck-and-drone instances, demonstrating competitive performance. Additionally, we conduct extensive evaluations on new, larger-scale instances to demonstrate scalability. Our findings highlight the operational benefits of closely integrated truck-drone fleets, affirming the HGA's practical effectiveness as a decision-support tool in advanced mixed-fleet logistics.

Key words: Arc routing problem, hybrid genetic algorithm, mixed-fleet logistics, truck-drone routing

1 Introduction

Arc Routing Problems (ARPs) represent a fundamental class of optimization problems in logistics, where the objective is to determine optimal routes that traverse specific arcs or edges in a network to satisfy predefined service requirements. In contrast to node routing problems, which focus on visiting designated network nodes, ARPs require servicing along the network's edges or arcs. These problems arise in a wide range of real-world applications, including waste collection, postal delivery, road maintenance, snow plowing, and inspection of electric power transmission lines. Although node routing problems, such as the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP), are well-established as $\mathcal{NP}$-hard, ARPs introduce additional complexities, making them at least equally challenging and computationally demanding. As a result, they continue to be a topic of significant interest in combinatorial optimization and logistics research.

In this paper, we focus on a fundamental variant of the ARP known as the Rural Postman Problem (RPP). In the RPP, only a designated subset of edges in the network requires service,

while the remaining "unrequired" edges are traversed solely to enable more efficient routing. The RPP generalizes the classical Chinese Postman Problem (CPP), in which all edges must be serviced. This added flexibility makes the RPP particularly well-suited to applications such as urban road patrolling, traffic surveillance, and monitoring of energy transmission infrastructure, where selectively traversing unrequired arcs to reach required ones can significantly reduce overall operational costs.

Recent advancements in drone technology (Wired Magazine, 2017) have given rise to innovative hybrid logistics systems that integrate traditional ground vehicles, such as trucks, with Unmanned Aerial Vehicles (UAVs or drones). This mixed fleet approach significantly enhances operational efficiency (Bouman et al., 2018; Murray and Chu, 2015), particularly in environments where traditional ground-based logistics are impeded by obstacles or traffic congestion. While drones offer advantages in terms of speed and the ability to bypass ground obstacles, their limited battery capacity inherently restricts their operational range. Integrating drones with trucks addresses this limitation effectively, as trucks can simultaneously serve both as mobile depots, facilitating battery swaps or recharges, and as service vehicles themselves.

The integration of multiple trucks and drones, each autonomously deploying from and returning to its assigned truck, introduces additional complexities and optimization challenges. Although recent literature has extensively explored such hybrid vehicle-drone systems, existing studies predominantly address node routing problems. Research addressing truck-drone coordination from an arc-routing perspective remains sparse, primarily due to the added computational complexity. One notable exception is provided by Liu et al. (2025), which investigates a constrained variant of the problem featuring a single truck equipped with drones, serving as a foundation for our broader, multi-vehicle scenario.

Our study contributes to the literature by addressing a novel extension of the traditional RPP involving multiple trucks, each capable of carrying and deploying multiple drones. In this setting, each drone must launch from and return to the same truck, creating complex interdependency between truck routes and drone assignments. The primary objective is to minimize the operational makespan, i.e., the total time required to complete all designated required edge services and return all vehicles to the depot, through coordinated truck-drone routing. Furthermore, our approach incorporates realistic operational constraints, including drone flying time limitations imposed by the battery capacity.

Recognizing the inherent computational challenges posed by this problem, we propose a Hybrid Genetic Algorithm (HGA) specifically designed to optimize mixed-fleet coordinated routing for minimal makespan. Our HGA features a distinctive two-part chromosome representation that explicitly encodes the sequencing and directional traversal of required edges, along with their assignments to trucks and drones. Additionally, we enhance the algorithm with multiple effective local search heuristics to systematically guide the exploration of the solution space toward improved optimality. We validate the proposed method through computational experiments on benchmark instances and comparison with an existing algorithm.

The remainder of this paper is organized as follows. Section 2 reviews the related literature, beginning with arc routing problems and followed by drone-assisted routing in both node and arc routing contexts. Section 3 formally defines the problem, detailing the key assumptions and features. Section 4 describes the proposed hybrid genetic algorithm based solution approach. Computational experiments and performance evaluations are presented in Section 5. Finally, Section 6 summarizes the key contributions and outlines directions for future research.

2 Literature Review

In this section, we provide a comprehensive review of the existing literature relevant to our study. We begin by examining key research on arc routing problems, highlighting significant contributions in this area. Subsequently, we discuss the node routing literature that focuses on the integration of truck-drone mixed fleets. Finally, we explore the application of mixed truck-drone fleets within the context of arc routing problems, identifying the existing research gaps that our study aims to address.

2.1 Arc Routing Problems

The family of arc routing problems, arising in various logistics applications (Corberán and Laporte, 2015), is a crucial area within the routing literature. For an in-depth review of arc routing problems and their variants, readers may refer to Corberán et al. (2021). In this study, we focus on the RPP, a fundamental variant of arc routing problems that seeks the shortest tour for a postman to service a designated subset of edges in a network while returning to the starting point. Notably, when all edges must be traversed, this becomes the CPP, analogous to the TSP in node routing literature. Several exact methods have been developed for RPP, including Christofides et al. (1986) and Corberán and Sanchis (1994), as well as for its variants such as hierarchical (Colombi et al., 2017), profitable (Avila et al., 2016; Colombi and Mansini, 2014), min-max k-vehicle windy (Benavent et al., 2014), multi-depot (Fernández et al., 2018) and periodic (Benavent et al., 2019) RPPs.

However, RPP is $\mathcal{NP}$-hard (Lenstra and Kan, 1976), necessitating development of near-optimal computational approaches for various practical applications. Numerous heuristics have been proposed for arc routing problems, such as tabu search (Hertz et al., 2000), adaptive large neighborhood search (Monroy-Licht et al., 2017), ant colony optimization (Santos et al., 2010), and genetic algorithms (Arakaki and Usberti, 2018; Lacomme et al., 2001). Genetic algorithms, in particular, have gained popularity as one of the most efficient metaheuristics for routing problems. Vidal (2017) explore extended neighborhoods and solve the CARP using two approaches: iterated local search and hybrid genetic search. A recent study, Mahmoudinazlou et al. (2024) propose a genetic algorithm for solving arc routing problems, leveraging dynamic programming decisions to assign arcs to vehicles.

2.2 Truck-Drone Routing Problems

We classify the collaborative truck-and-drone routing problems found in the literature into two main categories: node routing and arc routing.

2.2.1 Node Routing

The increasing prevalence of drone-assisted deliveries and surveillance has fueled research interest in vehicle routing with drones. Murray and Chu (2015) introduced truck routing with drone assistance as the Flying Sidekick Traveling Salesman Problem (FSTSP), presenting two mixed-integer linear programming (MILP) formulations and heuristics. A related variant, the Traveling Salesman Problem with Drone (TSP-D), emerged subsequently in Agatz et al. (2018). Multiple solution methods, including both exact (Roberti and Ruthmair, 2021) and heuristic (Bogyrbayeva et al., 2023; El-Adle et al., 2023; Lee et al., 2025; Mahmoudinazlou and Kwon, 2024) approaches, have been proposed for the TSP-D. More generalized extensions of the TSP-D, which incorporate multiple vehicles, have gained attention due to their potential to significantly reduce the makespan. Tamke and Buscher (2021) solve the vehicle routing problem with

drones, consisting of multiple truck-drone pairs, using a branch-and-cut method. They solve instances with up to 30 nodes to optimality. In a recent study, Sobhanan et al. (2024b) address a humanitarian routing problem involving a truck and multiple drones, introducing an exact branch-and-price algorithm tailored to maximize the coverage in emergency logistics. Their results demonstrate the effectiveness of dynamic programming with dominance rules in solving small-sized instances, while also highlighting the scalability limitations of exact optimization approaches for such complex truck-drone coordination problems. To address larger and more practical instances, heuristic and metaheuristic approaches play a crucial role. Sacramento et al. (2019) propose an adaptive large neighborhood search method for the vehicle routing problem with multiple trucks, each equipped with one drone. Similarly, Euchi and Sadok (2021) present a hybrid genetic algorithm for this problem. In this work, we focus on an arc routing equivalent of this vehicle routing problem with drones, involving a fleet of multiple trucks and drones. For comprehensive reviews of collaborative truck and drone routing, one can refer to Chung et al. (2020) and Macrina et al. (2020).

2.2.2 Arc Routing

Campbell et al. (2018) addresses the drone arc routing problem, where a drone, unlike traditional vehicles, can travel directly between any two points in the network. This inherent flexibility renders arc routing with drones a continuous optimization problem. To tackle this, Campbell et al. (2018) discretizes the problem by representing each edge as a polygonal chain. A similar approach is extended for the length-constrained k-drones RPP in Campbell et al. (2021). We incorporate this flexibility of drone traversal into our study by utilizing a distinct graph where edges are constructed based on Euclidean distance. While the drone arc routing problem has garnered attention, studies on the collaborative truck-drone arc routing problem remain limited. Liu et al. (2025) employs a joint system of one truck and one drone to solve the rural postman problem. The author proposes a formulation for the node routing equivalent of the problem and provide tabu search (TS) and adaptive large neighborhood search (ALNS) algorithms to solve large-scale instances. Additional heuristic methods have been explored in applications like powerline inspection (Liu et al., 2019) and traffic patrolling (Luo et al., 2019).

Table 1: Comparison of related truck-drone routing literature

Reference	**Node/Arc Routing**	**Trucks**	**Drones**	**Drone Capacity**	**Method**	**Problem Size**
Liu et al. (2019)	Arc	1	1	D	Heuristic	100 nodes
Luo et al. (2019)	Arc	1	1	D	Heuristic	24 nodes
Schermer et al. (2019)	Node	K	M	1	Matheuristic	100 nodes
Sacramento et al. (2019)	Node	K	M	1	ALNS	250 nodes
Tamke and Buscher (2021)	Node	K	M	D	Exact	30 nodes
Euchi and Sadok (2021)	Node	K	M	1	GA	200 nodes
Liu et al. (2025)	Arc	1	2	D	TS, ALNS	542 nodes
Mahmoudinazlou and Kwon (2024)	Node	1	1	1	GA	250 nodes
This work	Arc	K	M	D	GA	500 nodes

Table 1 presents a comparison of existing studies on truck-drone routing problems. Drone capacity D represents the number of nodes or arcs a drone can visit during an independent flight from the truck. To the best of our knowledge, no previous work has proposed a solution approach or solved an arc routing problem involving both multiple trucks and multiple drones. Our study fills this gap by developing a tailored genetic algorithm to solve this problem. Additionally, we

extend a traditional assumption of truck-drone routing by allowing drones to traverse multiple arcs, subject to a predefined flying time limit.

3 Problem Description

Consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consisting of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$ that connect the vertices in $\mathcal{V}$. Each edge $(i, j) \in \mathcal{E}$ is associated with a positive distance or cost, denoted by $\rho(i, j)$. Furthermore, a subset of edges $\mathcal{R} \subseteq \mathcal{E}$ is designated as "required", meaning these edges must be included in any feasible route, while the remaining edges are optional. RPP seeks to find a closed tour in $\mathcal{G}$ that traverses every required edge in $\mathcal{R}$ at least once, while minimizing the total cost of the route. The total time is the sum of the travel time of all traversed edges, including any edges that are traversed multiple times. Since the RPP is known to be a $\mathcal{NP}$-hard problem, finding an optimal solution efficiently is computationally challenging for large instances.

Numerous algorithms (Corberán et al., 2000; Monroy-Licht et al., 2014; Pearn and Wu, 1995) have been devised to address the RPP, motivated by its practical significance in various routing contexts. This study extends the classical RPP to include multiple drone assistance for ground vehicles, a variant we refer to as the RPP with multiple Trucks and multiple Drones (RPP-mTD). In this model, the truck acts as a mobile station for each drone designated to it when the drones are not in flight. This variant of RPP enhances operational efficiency through coordinated truck-drone operations. We assume a depot serves as both the starting and ending point for the truck-drone fleet. While incorporating a depot node technically transforms the arc routing problem into a general routing problem, we treat it as a special case given the sole involvement of the depot.

The objective function of RPP-mTD is to minimize the makespan, which represents the total time required to complete all services and return the fleet to the depot, while determining the optimal cooperative routes for the truck and drones. Every required edge in the network must be serviced at least once by either a truck or a drone. Here, multiple traversals of edges are permitted and is accounted for in the makespan.

3.1 Key Assumptions

We explore the RPP-mTD under the following additional assumptions:

1. *Fleet Composition:* The fleet consists of one or more trucks (K), where each truck carries a fixed number (M) of identical drones. The drones are homogeneous in terms of battery capacity and speed, which together determine the maximum allowable flying time for each drone.

2. *Truck–Drone Operational Rules:* A drone may take off from or land on its associated truck only at vertices where the truck is stationed at that moment. Take-off and landing times are assumed to be negligible. In this setup, trucks function as mobile launch and recovery platforms. While drones operate independently during flight, trucks provide essential logistical support, such as instantaneous battery replacement or recharging upon drone landing. Furthermore, when trucks traverse a required edge, they directly service it.

3. *Drone Routing Flexibility:* Unlike the trucks, drones are capable of flying directly between vertices without following the predefined road network. A drone can thus bypass ground obstacles by directly flying between any two vertices. To accommodate drone movement, an auxiliary drone-specific graph is implicitly considered, constructed by establishing Euclidean direct edges between all vertex pairs, regardless of road connectivity.

When traversing non-required edges, the drone is assumed to follow the corresponding arc in this auxiliary graph. However, when servicing a required arc, the drone instead follows the *shadow* of that arc in the original graph to ensure proper coverage.

4. *Flight Range Constraint:* Drone flights are restricted by their maximum flight time, limited by battery capacity. The flight duration of any drone sortie cannot exceed this predefined limit. Upon returning to a truck, a drone's battery is instantly replaced or recharged, resetting its available flight time for subsequent operations.

5. *Drone Multi-Arc Visit Flexibility:* Unlike in some known routing problems involving drone assistance, such as the TSP-D, each drone in the RPP-mTD may service multiple required edges consecutively during a single flight, subject to its flight range constraint.

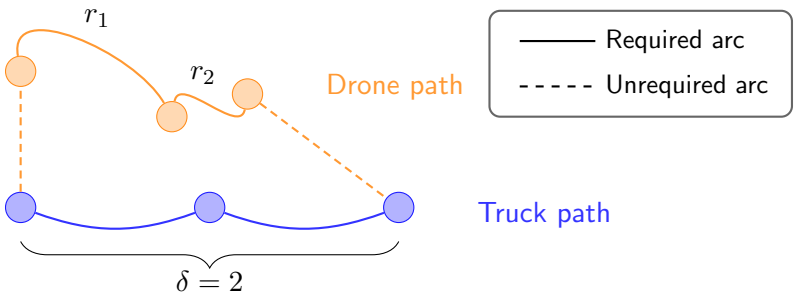


Figure 1: An example of the δ-hop rendezvous mechanism with $\delta = 2$. Here, the drone lands on the last permissible node within the allowed limit.

6. *Drone–Truck Coordination Flexibility:* We introduce a hyperparameter, δ, to control the extent of independence between a drone's movement and its originating truck. When $\delta = 1$, the model adheres to the restricted assumption: the drone is launched and retrieved at consecutive truck nodes, with the truck traversing only one arc during the drone's flight. In contrast, when $\delta = L$, the truck is permitted to travel up to L arcs after launching the drone before rendezvousing for retrieval. Figure 1 illustrates a scenario where $\delta = 2$ involving one truck and one drone. In this example, the drone returns to the truck at the farthest node permitted within the δ-hop limit. Increasing δ significantly enlarges the solution space, as the drone can potentially land at any node visited by the truck within the allowed δ-hop window. This added flexibility can lead to more efficient routing, leading to a reduction in overall makespan. However, practitioners should note that very large values of δ may introduce practical challenges, such as increased difficulty in real-time communication and a higher risk of coordination failures.

3.2 An illustrative Example

An illustrative example of a feasible RPP-mTD solution with seven required arcs is shown in Figure 2. In this scenario, two trucks, k_1 and k_2, are each equipped with two drones. Specifically, k_1 carries drones d_1 and d_2, while k_2 carries drones d_3 and d_4. Notably, drone d_4 remains idle throughout the operation. Among the seven required arcs, two are serviced by drone d_1, one by drone d_2, one by drone d_3, and one by each truck. In this example, the coordination parameter is set to $\delta = 2$, allowing a maximum separation of two hops between drone launch and retrieval. Drones d_1 and d_2 fully utilize this δ-hop flexibility, while drone d_3 does not. The figure illustrates the coordinated interaction between trucks and their assigned drones, highlighting how the RPP-mTD framework leverages ground–aerial collaboration to improve efficiency and reduce makespan.

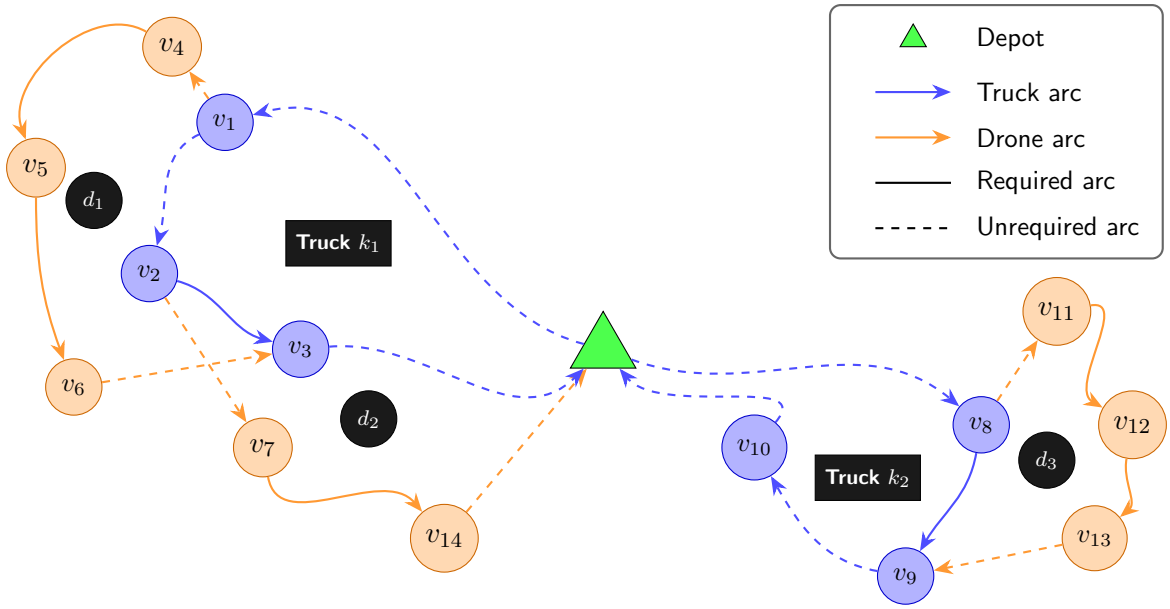


Figure 2: An illustrative RPP-mTD solution: two trucks k_1 and k_2, each carrying two drones—k_1 carries d_1 and d_2; k_2 carries d_3 and d_4. Notably, d_4 remains idle in this routing scenario.

3.3 Problem Complexity and Formulation Challenges

The assumptions made in the previous subsection significantly generalize the problem under study compared to the existing body of literature (see arc routing papers in Table 1). This is due to the introduction of several additional flexibility dimensions, including but not limited to the involvement of multiple trucks, multiple drones, drone multi-arc visit flexibility, and drone–truck coordination flexibility. Unlike classical arc-routing problems or single-truck drone-assisted routing models, RPP-mTD encompasses multiple intertwined decision layers, ranging from vehicle routing and positioning to service assignment and drone endurance constraints. This intricate interplay greatly expands the solution space and makes the problem considerably more challenging.

We note that developing an integer programming model that fully captures these complexities is highly complicated and provides limited practical value, as such models quickly become intractable for state-of-the-art solvers. Therefore, we do not present an exact optimization model in this paper. Instead, we note that Liu et al. (2025) demonstrate the inherent modeling complexity even for a very special case of our problem (i.e., one truck and one drone), where the computational burden already renders exact methods impractical. In their study, Liu et al. (2025) transform arc-routing problems into equivalent node-routing formulations (Longo et al., 2006) to address truck–drone coordination, and subsequently design heuristic solution methods due to its computational complexity. While the formulation is effective for their restricted setting, this approach introduces substantial computational overhead by enlarging the underlying graph, which further limits scalability.

Given that our problem setting is far broader, exact optimization techniques are unlikely to be meaningfully applied in real-world scenarios. Additionally, unlike Liu et al. (2025), we do not rely on node-transformation techniques, which allows us to avoid the additional complexity and overhead associated with graph expansion and an increased fleet size. This observation underscores the need for efficient heuristic approaches capable of producing high-quality solutions within reasonable computational times, which is the main goal of our study.

4 Methodology

In this section, we present the Hybrid Genetic Algorithm (HGA) developed to solve the RPP-mTD. Our approach builds on established evolutionary computation frameworks while incorporating several novel components that set it apart from standard implementations. These innovations, essential to the algorithm's effectiveness, are specifically designed to address the unique challenges of coordinated makespan optimization in multi-fleet truck-and-drone routing. The algorithm iteratively applies a combination of traditional and custom genetic operators, complemented by powerful local search procedures, to refine solutions and guide the search toward a global optimum.

4.1 Graph Abstraction

We adopt a graph abstraction to enable straightforward solution representation and to appropriately capture truck and drone movements during evaluation. The underlying network consists of both required and unrequired edges. Trucks are constrained to travel along this network topology. Drones, however, operate in a hybrid mode: they fly unconstrained, point-to-point (Euclidean distance) between launch and recovery nodes, but must explicitly traverse the path of any required edge they are assigned to service.

To support the efficient evaluation of several candidate solutions, we preprocess the graph to compute the shortest paths between the endpoints of all required edges. This abstraction reduces the complex pathfinding challenge to a high-level sequencing problem. As a result, the traversal time between any two consecutive service tasks can be retrieved from a precomputed lookup table rather than being repeatedly recomputed, which is critical for the algorithm's performance.

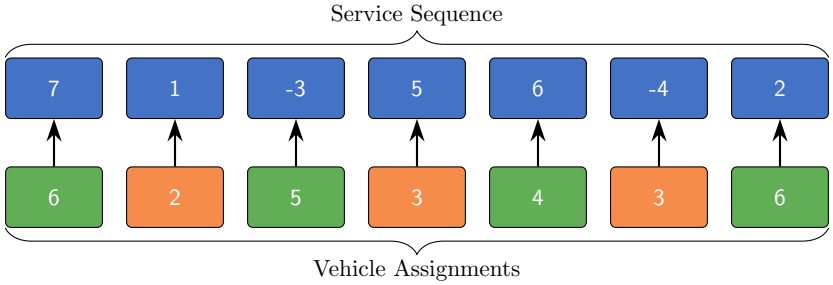
4.2 Solution Representation

For an effective optimization process, it is crucial to establish a valid representation of the solution. To capture the distinctive characteristics of the solution space, we adopt a two-part chromosome encoding approach:

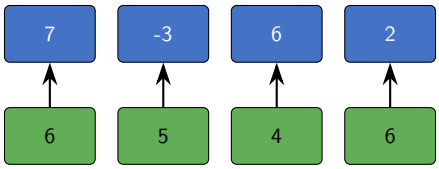
1. *Service Sequence:* This component is an ordered list representing all required edges that must be traversed, each uniquely indexed. With R required edges, this chromosome segment has length R. Each required edge is accompanied by a direction marker to indicate traversal direction: a positive index denotes forward traversal, while a negative sign indicates traversal in the opposite direction. Collectively, this chromosome part defines the sequence of service and traversal orientations.

2. *Vehicle Assignment:* This part complements the service-order sequence by specifying the vehicle responsible for each service step. Each vehicle is assigned a unique identifier to facilitate differentiation. For example, in a configuration with $K = 2$ trucks, each equipped with $M = 2$ drones, the numbering proceeds as follows: Truck 1 is labeled as 1, its drones as 2 and 3, Truck 2 as 4, followed by its drones as 5 and 6, respectively. Figure 3 shows an example of the chromosome encoding for a problem with $R = 7$, illustrating how the representation can be decomposed for a truck system. Tasks assigned to a truck and its associated drones collectively constitute the tour of that truck-drone system, and those allocated specifically to individual drones form their respective sorties.

Since both parts of the chromosome are of equal length and jointly capture all necessary information, a complete routing plan for the vehicles can be derived directly by interpreting the



(a) Two-part chromosome encoding: the service sequence of required edges (top) and vehicle assignments (bottom).



(b) Decomposition for Truck-Drone System 2

Figure 3: An example solution representation of an RPP-mTD with $R = 7$, $K = 2$ and $M = 2$

chromosome. This representation clearly separates *what* must be done (the exact sequence and orientation of required arcs) from *which* vehicle performs it, while maintaining full synchronization between the two components.

4.3 Route Construction for Evaluation

To translate a chromosome into a complete routing plan and evaluate its quality, we follow a clear decoding process. First, the full route for each vehicle is constructed by connecting its assigned service tasks. When connecting two consecutive tasks, if multiple precomputed shortest paths of equal length (co-optimal paths) exist, one is randomly sampled. This random choice occurs during the creation and modification of solutions, introducing valuable structural diversity into the population by exploring different intermediate nodes for drone operations.

Second, the process handles structural inefficiencies implicitly. In cases where the shortest path between required edges r_i and r_j happens to pass through another required edge r_k, the resulting route is valid but contains a redundancy. Such solutions are not explicitly forbidden or repaired. Instead, their naturally longer makespan results in a poorer fitness score, and they are less likely to be selected for propagation, allowing the evolutionary search to organically favor more efficient orderings (e.g., $...r_i, r_k, r_j...$). The quality of a decoded plan is ultimately measured by its makespan, which serves as the primary input to the fitness function described in the following section.

4.4 Fitness Function

The fitness of each candidate solution is measured by its makespan, i.e., the total time required to service all designated edges and return to the depot. To compute this makespan, we first determine each truck's route duration by summing the travel times along the shortest paths that connect the pairs of required arcs in its service sequence (including any necessary repeats of edges). Next, we calculate each drone's flight time using the straight-line distance between non-required edges, based on the potential launch and recovery vertices. For required edges, the drone's travel time is computed along the given edges. We also verify that every drone route

respects the maximum allowable flight time, τ, and does not exceed the permitted number of intervening truck-traversal arcs, δ, between take-off and landing points. Finally, the drone paths are selected based on their effect on the objective value.

After the makespan $T(I)$ of an individual I has been calculated, we translate it into a fitness score $F(I)$. Note that any solution violating the drone-range constraint incurs a range-violation penalty

$$w_{\text{inf}} \sum_{d=1}^{D} \big(\tau_{\max}(d) - \tau\big)^{+}$$

added directly to its makespan. Here, $\tau_{\max}(d)$ denotes the highest independent flight time of drone d, and $(x)^{+} = \max\{0, x\}$. w_{inf} is the penalty that we dynamically adjust throughout the genetic algorithm. To encourage population diversity, we also compute each individual's diversity score $\delta(I)$ as the normalized Hamming distance to its two nearest neighbors (Sobhanan et al., 2024a). If n_E denotes the number of elite individuals in the population and n_P is the total population size, then the fitness function for a chromosome I is defined as

$$F(I) \;=\; T(I) \times \left(\frac{n_E}{n_P}\right)^{\delta(I)}.$$

Therefore, solutions with lower fitness values, reflecting both shorter effective makespans and greater niche differentiation, are preferred in the selection process.

4.5 Initial Population

The initial population of solutions is generated using a hybrid strategy that combines purely random initialization with targeted initialization. Specifically, a fraction p_t of the population is seeded by a problem-specific heuristic that identifies promising drone–truck routing plans. By incorporating these makespan-aware individuals alongside randomly generated solutions, the genetic algorithm benefits from both high-quality starting points and broad population diversity.

Each targeted individual is constructed in two steps. First, we build a simple RPP service sequence: starting at the depot, we iteratively append the as-yet-unserved required edge whose nearest endpoint to the current truck location that minimizes the incremental truck travel time, while respecting edge directionality by choosing the traversal order (u, v) or (v, u) that is most efficient. Second, we apply a greedy makespan-minimization assignment. We traverse the resulting service sequence and, for each arc (u, v), compute the projected completion time if served by each of the K trucks or any of their $K \times M$ drones, tracking each truck's current node and accumulated drive time, along with each drone's cumulative flight time. We then assign the arc to the vehicle that yields the smallest overall makespan.

4.6 Evolutionary Process

The population evolves through an iterative evolutionary process involving selection, crossover, and mutation, executed for a predefined maximum number of generations G. At each generation, offspring solutions are generated by combining existing solutions through genetic operations, ensuring exploration of diverse and potentially superior solutions.

Crossover operators play a crucial role by combining chromosome sequences from pairs of parent solutions to produce offspring inheriting traits from both parents. Parent solutions are selected using binary tournament selection based on their fitness scores. We randomly apply one of three crossover methods. The *Order Crossover (OX)* method involves selecting two

cut points in the chromosome sequence, copying the segment between these points from one parent into the offspring, and filling the remaining positions sequentially with genes from the other parent, skipping genes already copied. The *Partially Mapped Crossover (PMX)* exchanges segments between two cut points while resolving duplicate genes through a mapping process between parents, ensuring valid offspring permutations.

Additionally, we propose a novel, problem-specific crossover operator termed *Segment-Preserving Crossover*, tailored explicitly for the truck-drone system context. This operator selects a complete segment corresponding to all visits handled by a single truck system and its associated drones from one parent, and extracts the corresponding segment from the other parent. Subsequently, either OX or PMX is applied exclusively to these segments. The generated segment is then reintegrated into copies of the original parents, followed by a repair procedure. The repair step corrects any duplicate or missing visits by incorporating appropriate arcs from the opposite parent. Consequently, this method yields two feasible offspring preserving intact truck system-level segments.

Mutation operations occur with an initial probability p_m, introducing minor random variations within chromosome structures to explore new solution spaces and maintain genetic diversity. To mitigate stagnation, if no improvement in solution quality is observed for G_m consecutive generations, we dynamically increase the mutation probability to p_m^+. Mutation is executed by randomly selecting one of three operators applied to both the service sequence and vehicle assignment components of the chromosome: (1) *Swap Mutation* selects two positions at random and exchanges their genes, (2) *Inversion Mutation* selects a random subsequence between indices $i < j$ and reverses it in place, and (3) *Reassignment Mutation*, designed specifically for vehicle assignments, randomly selects a single gene and assigns it to a different vehicle.

Throughout the evolutionary process, the population size is maintained within the interval $[P_L, P_H]$. After generating offspring solutions at each generation, the population is sorted based on the fitness function, and only the best-performing P_L individuals are retained for subsequent generations. This strategy effectively balances diversity with exploration of high-quality solutions. Additionally, a maximum of P_H candidate solutions is generated at each iteration through combined evolutionary operations and local search refinements. To further promote convergence, the top 1% of the solutions are explicitly preserved across generations. This elitist strategy ensures that superior solutions are not lost due to stochastic variation.

4.7 Local Search and Refinement

To further enhance exploration and mitigate the risk of becoming trapped in local optima, each newly generated population undergoes a dedicated local improvement phase. Following the evaluation of offspring fitness, individuals are sorted in ascending order of their fitness scores, and the top 20% are selected for further refinement. Each selected individual is subjected to a bounded iterative local search process, limited to a number of steps (`ls_steps`). This process involves randomized exploration of the solution's neighborhood, and unlike the mutation procedure, only strictly improving moves are accepted. This ensures a monotonically non-increasing makespan and steady progression toward local optimality.

At each iteration of the local search, a candidate solution is modified using one randomly chosen neighborhood search operator from the following five strategies:

1. *Subsequence Reversal*: Selects and reverses a subsequence of the overall service sequence and the corresponding vehicle assignments, to potentially reveal more efficient orderings.

2. *Or-opt*: Relocates a contiguous block of up to b required arcs to a different position in the sequence, preserving their internal order, to exploit local structural improvements.

3. *Drone Sortie Optimization*: For each drone, attempts to improve the internal ordering of assigned arcs when the sortie length exceeds a threshold (e.g., ≥ 3), effectively solving small intra-sortie Traveling Salesman subproblems.

4. *Greedy Vehicle Reassignment*: Evaluates alternative vehicle assignments for an arc, identifies the reassignment yielding the greatest reduction in makespan, and applies the most beneficial changes.

5. *Ruin-and-Construct*: Temporarily removes a proportion p_{ruin} of the required arcs and reinserts them into positions and assigns vehicles that minimize the resulting makespan, leading to additional exploration of the solution space.

After applying a neighborhood move, the solution's fitness is re-evaluated. If the resulting makespan is strictly better than that of the incumbent solution, the candidate replaces it. This process is repeated for up to `ls_steps` iterations, after which the locally refined individual is returned. Once all selected candidates in the current generation have undergone local refinement, the resulting pool is trimmed based on fitness to restore the original population size.

By integrating broad global search via genetic evolution with focused local refinement, our hybrid genetic algorithm (HGA) strikes a robust balance between diversification and exploitation. This balance leads to high-quality solutions with minimized makespan. Upon convergence, the best solution found across all generations is selected as the final output, representing a practically deployable multi-fleet truck-drone routing plan.

5 Computational Study

In this section, we present a series of experiments to evaluate the performance of the proposed heuristic for the RPP-mTD under various problem settings. All algorithms and experimental procedures are implemented in Julia 1.11.5 and executed on a MacBook Pro equipped with an Apple M1 chip, 16 GB of RAM, and running macOS Sequoia 15.5. The parameters used in our experiments are as follows: $n_E = 0.8n_P$, $G = 100$, $G_m = 10$, $P_L = 100$, $P_H = 200$, $w_{\text{inf}} \in [0.01, 100.0]$, $p_t = 0.1$, $p_m = 0.1$, $p_m^+ = 0.3$, $p_{\text{ruin}} = 0.2$, and `ls_steps` $= 30$.

We conduct two types of experiments. First, we benchmark our algorithm using the instances provided in Liu et al. (2025), focusing on a restricted setting with $K = 1$ truck and $M = 1$ drone. This allows for a controlled comparison of a special case with the existing method, using one of their parameter configurations with the drone speed ($s_{\text{drone}} = 2$) and the truck speed ($s_{\text{truck}} = 1$), the most common situation in practical applications (Harrison, 2025). Second, we generate a set of large-scale RPP-mTD instances to analyze the algorithm's scalability and responsiveness to different problem characteristics. Specifically, we examine the effects of varying vehicle compositions, drone ranges, and the δ-hop window. In these experiments, truck travel time on an edge is calculated by dividing its arc length by a constant cruise speed of 40 $\text{km}\,\text{h}^{-1}$. Since drones are generally faster than trucks (Harrison, 2025), especially in urban environments, their flight time is computed using a cruise speed of 80 $\text{km}\,\text{h}^{-1}$ (Aurambout et al., 2022). Each sortie is further limited by a maximum airborne time of 60 minutes, reflecting the battery capacity constraints.

For the second set of experiments, we generate RPP test instances based on a 10 km $\times$ 10 km square grid that approximates an urban district. First, the number of nodes N is selected such that $N \in [50, 500]$, and node coordinates are generated randomly on the square grid. A graph is then constructed by connecting the nodes using random edges. To ensure that routes exist, we retain the largest connected component and discard any isolated nodes or subgraphs. A subset of these edges is then designated as required, with its cardinality selected from the range

[15, 100]. All edge lengths in the ground network are measured using the Manhattan metric to realistically capture rectilinear street layouts. For each instance size considered, we generate five test instances per instance class.

5.1 Benchmark Evaluation

In Table 2, we present a benchmark evaluation of our proposed approach applied to the special case of the RPP-mTD problem with a single truck carrying one drone. We utilize benchmark instances originally introduced in Liu et al. (2025). In this experiment, we do not impose a limit on the δ-hop window. Additionally, we analyze the impact of varying values of β, a hyperparameter from their study, to determine the drone flying time limit, defined as follows:

$$\tau = \frac{\beta}{s_{\text{drone}}} \times \text{Average edge distance in the drone graph} \tag{1}$$

Table 2: Benchmark evaluation results for RPP-mTD with 1 truck and 1 drone

Instance Class	β	Best Known	ALNS			HGA		
			Obj	Time* (sec)	Gap (%)	Obj	Time (sec)	Gap (%)
N10E20R5	1	358.22	365.32	11.23	1.98	368.82	2.82	2.96
	2	256.43	263.23	7.32	2.65	260.80	3.15	1.70
	3	196.29	200.42	5.43	2.10	196.76	3.15	0.24
N10E20R7	1	461.53	470.34	16.23	1.91	475.62	6.92	3.05
	2	319.89	326.43	34.23	2.04	321.96	7.97	0.65
	3	237.08	242.53	43.42	2.30	242.82	8.48	2.42
N10E20R10	1	591.06	599.42	284.55	1.41	629.20	16.39	6.45
	2	394.15	400.23	204.32	1.54	399.86	22.12	1.45
	3	284.21	289.53	221.23	1.87	289.26	24.99	1.78
N15E30R5	1	446.42	452.32	4.63	1.32	456.50	2.77	2.26
	2	340.98	344.52	7.83	1.04	349.48	2.08	2.49
	3	253.60	259.32	6.58	2.26	263.64	1.94	3.96
N15E30R7	1	533.58	547.63	19.43	2.63	552.46	6.44	3.54
	2	399.35	410.23	35.64	2.72	404.54	5.21	1.30
	3	282.83	289.53	55.47	2.37	287.80	4.78	1.76
N15E30R10	1	660.18	675.73	210.24	2.36	690.34	16.45	4.57
	2	481.08	510.23	199.53	6.06	481.08	20.73	0.00
	3	337.74	359.34	178.46	6.40	337.56	22.19	-0.05
Average		379.70	389.24	85.88	2.50	389.36	9.92	2.25

The results indicate that our method achieves competitive performance compared to the ALNS. Specifically, our method has an average optimality gap of 2.25%, closely matching ALNS, which achieves a gap of 2.5%. It is worth emphasizing that while ALNS is specifically tailored for single-truck routing scenarios, our solution method is more general, capable of addressing broader configurations of the problem. Moreover, our approach demonstrates notable advantages over existing benchmarks. For instance, in one specific scenario (instance class N15E30R10 with $\beta = 3$), our method outperforms the best-known solution obtained using a MILP solver that was executed with a one-hour time limit. Additionally, our algorithm significantly reduces computational time compared to ALNS, delivering solutions significantly faster. Note, however, that the ALNS results were reported in the benchmark study using a workstation equipped with a 2.2 GHz Intel Xeon E5-2630 processor and 32 GB of RAM. Using data from `https://www.cpubenchmark.net/`, we compared the single-thread performance of the Xeon processor to that of our system. Based on this comparison, we estimate that if ALNS were executed on our hardware, its average runtime would be approximately 40.3 seconds, as opposed to the value reported in the benchmark study. Nevertheless, even under this adjusted

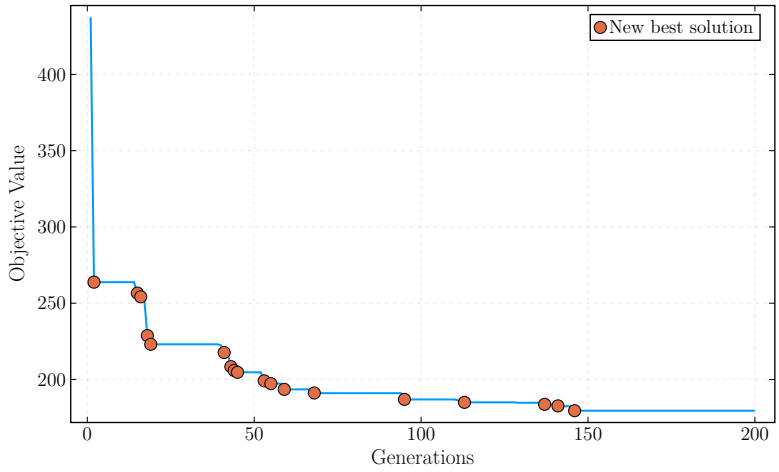


Figure 4: Convergence of the proposed HGA across generations for an instance with $R = 100$

estimate, our HGA algorithm maintains a clear advantage in computational efficiency, delivering solutions in significantly less time.

5.2 Scalability and Responsiveness Evaluation

We now conduct experiments on larger instances to better understand and empirically evaluate the factors that influence both the RPP-mTD and our proposed HGA. Before presenting the various experiments and results in this subsection, we first highlight the performance of the proposed HGA on a representative instance. This provides evidence that the algorithm does not waste computational resources; rather, it continuously searches for improved solutions until convergence. Figure 4 illustrates the convergence behavior of the HGA over generations for an instance with $R = 100$, using a configuration of $K = 2$ trucks, $M = 2$ drones per truck, and a drone flying time limit of $\tau = 1.0$ hour. As observed, the proposed method quickly identifies a feasible solution early in the search with an objective value of approximately 265. At approximately 150 generations, it finds the best solution with an objective value near 180, corresponding to an improvement of roughly 32%. After 200 iterations, the algorithm terminates and reports the best solution found.

5.2.1 Analysis of the Impact of Vehicle Composition

In this experiment, we analyze how the makespan of the system changes with respect to different combinations of trucks and drones, in addition to the computational complexity it possesses as we increase the fleet size. Tables 3 and 4 present the results of our method applied to varying vehicle compositions, specifically with $K \in \{1, 2, 3\}$ trucks and $M \in \{1, 2, 3\}$ drones. All the experiments were conducted using fixed parameters $\delta = 5$ and $\tau = 1.0$ hour. Using our proposed HGA, we examine how changes in the number of trucks and drones influence solution quality and computational runtime.

Across all instance classes, increasing the number of trucks consistently led to substantial reductions in the makespan. For example, in instance N200E400R50, transitioning from a single truck with one drone to a three-truck setup, each still equipped with one drone, reduced the makespan by approximately 57.6%, from 139.2 to 59.0. Similar patterns were observed across all problem sizes, underscoring the advantage of using multiple trucks for parallel coverage, particularly in larger networks. Adding drones per truck generally improved solution quality

Table 3: Impact of vehicle composition on solution quality and runtime - Part 1

Instance	1 Truck						2 Trucks					
	$M = 1$		$M = 2$		$M = 3$		$M = 1$		$M = 2$		$M = 3$	
	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)
N50E100R15	33.8	0.2	32.2	0.3	32.2	0.3	20.8	0.3	20.5	0.4	20.4	0.5
N50E100R20	45.4	0.4	44.3	0.5	42.2	0.6	27.0	0.6	26.8	0.8	25.2	1.0
N100E200R25	70.1	0.6	68.4	0.9	65.7	1.0	42.2	1.1	38.9	1.2	38.9	1.6
N100E200R30	80.2	0.9	77.1	1.4	76.9	1.7	48.4	1.8	45.1	2.3	44.4	3.0
N200E400R40	122.2	2.1	112.1	3.3	109.9	4.0	67.9	4.1	63.3	5.0	61.9	6.5
N200E400R50	139.2	4.2	136.1	6.3	130.5	7.6	81.6	7.6	78.0	9.4	74.7	11.9
N300E600R50	148.6	4.4	142.2	6.7	139.6	8.1	85.8	8.0	83.0	10.3	81.3	12.6
N300E700R70	203.7	11.8	189.4	17.6	188.2	20.8	112.8	20.7	108.0	25.7	104.3	32.0
N400E800R80	267.7	18.4	251.8	27.5	242.5	32.0	149.8	32.8	143.5	40.1	135.8	49.8
N500E1000R100	321.0	38.5	312.8	57.2	302.6	65.8	186.0	66.7	174.5	83.2	168.3	98.6

Table 4: Impact of vehicle composition on solution quality and runtime - Part 2

Instance	3 Trucks					
	$M = 1$		$M = 2$		$M = 3$	
	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)
N50E100R15	16.4	0.5	16.3	0.6	16.1	0.6
N50E100R20	20.7	0.9	20.2	1.0	19.7	1.3
N100E200R25	31.7	1.7	29.8	1.9	29.4	2.3
N100E200R30	35.4	2.8	34.4	3.2	34.1	3.8
N200E400R40	48.5	6.3	46.2	7.1	46.0	8.8
N200E400R50	59.0	11.9	56.0	13.4	55.5	16.3
N300E600R50	64.6	12.3	58.9	14.1	58.9	17.2
N300E700R70	82.4	32.8	76.1	36.1	76.1	43.8
N400E800R80	109.5	49.1	102.2	55.2	100.5	65.3
N500E1000R100	131.2	97.8	125.2	112.0	121.2	131.3

Table 5: Impact of drone flying time on the solution and runtime

Instance	$\tau = 0.5$		$\tau = 1.0$		$\tau = 1.5$		$\tau = 2.0$	
	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)
N50E100R15	23.7	0.3	20.5	0.4	13.8	0.5	9.1	0.5
N50E100R20	29.8	0.6	26.8	0.8	18.7	0.9	12.1	1.0
N100E200R25	48.1	1.0	38.9	1.2	27.9	1.7	19.7	1.7
N100E200R30	52.8	1.7	45.1	2.3	32.5	2.6	21.6	2.8
N200E400R40	79.2	3.7	63.3	5.0	42.0	5.9	29.2	6.3
N200E400R50	91.5	6.7	78.0	9.4	52.8	10.9	36.3	11.5
N300E600R50	101.3	7.1	83.0	10.3	56.0	11.5	40.6	12.0
N300E700R70	129.8	17.6	108.0	25.7	74.7	28.1	54.5	30.3
N400E800R80	170.9	26.4	143.5	40.1	101.8	43.1	71.2	46.1
N500E1000R100	206.3	53.4	174.5	83.2	122.1	85.9	86.9	91.5

as well, although the marginal gains diminished with each additional drone due to drone range limitations. Introducing a second drone per truck consistently yielded noticeable improvements in the objective value. However, the benefit of adding a third drone was often limited, especially in larger, multi-truck configurations. For instance, in the N400E800R80 case with three trucks, the improvement in the objective was only 1.7%, when increasing from two to three drones per truck, despite a significant increase in computational runtime from 55.2 to 65.3 minutes.

From a computational perspective, the runtime of our HGA algorithm increases significantly with the number of trucks and drones. This overhead becomes especially noticeable as more drones or trucks are added, reflecting the increased complexity of coordinating multiple drone routes with corresponding truck schedules. This effect is especially evident in larger instances. In the N500E1000R100 scenario, for example, the runtime nearly doubled, from 57.2 to 112.0 minutes, when moving from a single-truck, two-drone setup to a three-truck configuration with two drones each. These empirical results clearly highlight the trade-offs between solution quality and computational complexity. While deploying multiple trucks consistently enhances performance by enabling parallel operations, drone allocation requires greater caution due to their limited flight range. Given the diminishing marginal gains and the sharp increase in computational burden, a balanced strategy is essential when configuring larger fleets.

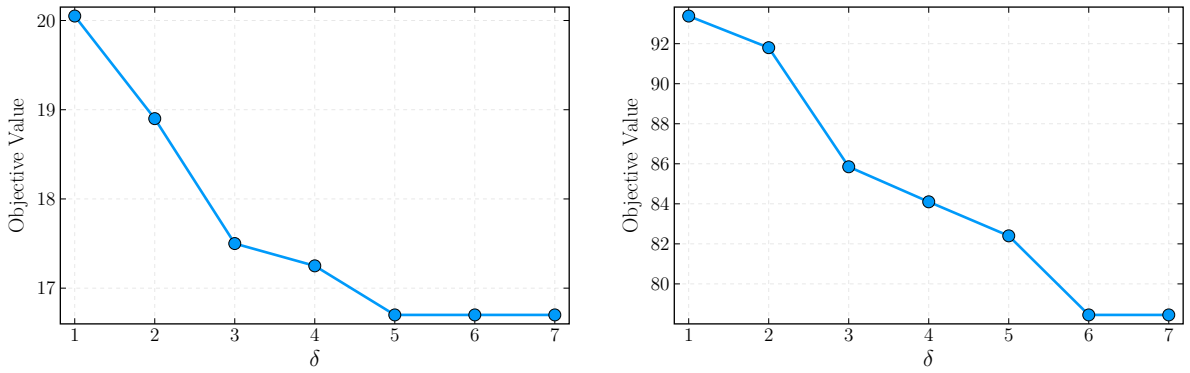
5.2.2 Analysis of Drone Flying Time Limit

Table 5 reports the impact of increasing the drone flying time limit τ on both makespan and computational runtime, with $\delta = 5$ and a fixed fleet of two trucks each carrying two drones. As τ increases from 0.5 to 2.0 hours, drones can service more distant arcs and the makespan falls steadily in every instance. In the smallest network class (N50E100R15), for example, the objective decreases from 23.7 at $\tau = 0.5$ to 9.1 at $\tau = 2.0$, with a 61.6 % reduction. However, the runtime increases from 0.3 minutes to 0.5 minutes. In the largest instance (N500E1000R100), extending τ from 0.5 to 2.0 hours cuts the makespan by 57.9 % but raises runtime by 71.3 %.

On average across all ten instance classes, increasing τ from 0.5 to 1.0 hour yields a 15.8 % reduction in makespan at the cost of a 39.6 % longer runtime. Similarly, increasing τ further from 1.0 to 1.5 delivers a 30.8 % drop in makespan with a 15.8 % runtime penalty, and raising τ from 1.5 to 2.0 brings another 30.7 % improvement in routing efficiency for a 5.7 % increase in runtime. Although today's commercial delivery drones typically operate flights of around one hour, ongoing advances in drone battery capacity are rapidly extending their operational range, making higher τ values increasingly realistic in the near future.

5.2.3 Analysis of δ-hop Window

Using our proposed HGA, we analyze the effect of varying the parameter δ. Figures 5a and 5b illustrate how adjusting the δ-hop window influences the makespan for two instances belonging to classes N50E100R15 and N200E400N50, respectively. Recall that $\delta = 1$ implies each drone sortie departs from a truck stop and returns at the immediately subsequent truck stop, whereas higher δ values allow the truck to traverse up to δ arcs before rendezvousing with the drone. The experiments are conducted with a fixed fleet configuration $K = 2$ and $M = 2$, and drone flying time limit $\tau = 1.0$.



(a) Instance with $N = 50$, $E = 100$, and $R = 15$ (b) Instance with $N = 200$, $E = 400$, and $N = 50$

Figure 5: Effect of varying δ on the makespan of two instances

Generally, increasing δ reduces the makespan since drones can rendezvous further from their launch points. However, the incremental benefit diminishes as δ increases. Thus, practitioners should select an appropriate δ value by carefully balancing routing flexibility and makespan savings against operational complexity and synchronization constraints, as excessively large δ values may be practically infeasible.

6 Conclusions

This paper introduced the Rural Postman Problem with multiple Trucks and Drones (RPP-mTD), a new arc-routing variant that captures multi-vehicle coordination in emerging truck–drone logistics. We developed a Hybrid Genetic Algorithm (HGA) with multiple neighborhood search methods to effectively solve this problem. Benchmark experiments on small-scale instances with one truck and one drone show that HGA achieves an average optimality gap of 2.25%, outperforming the state-of-the-art ALNS (2.5%) while delivering solutions significantly faster. On larger networks, up to 500 nodes and 100 required arcs, the algorithm scales well and produces good quality solutions within minutes.

Our experiments yield three managerial insights. (1) Adding trucks consistently reduces makespan, whereas addition of drones per truck may not be useful for larger networks due to limited drone flying time. (2) Extending drone flying time (τ) sharply lowers makespan but can inflate runtime on very large instances. (3) Relaxing the drone launch node constraint through a δ-hop window further decreases the makespan; however, beyond moderate δ values, the incremental benefit diminishes. Future research could explore extensions involving stochastic travel times and non-linear battery consumption. Another promising direction is to study scenarios where drones are not specifically associated with individual trucks, allowing them to fly in and out of different trucks throughout the operation.

Acknowledgments

This work was supported by the National Science Foundation under award number 2032458 and the National Research Foundation of Korea grant (RS-2023-00259550) funded by the Ministry of Science and ICT. Part of this research was conducted during the first author's doctoral program at the University of South Florida.

References

Agatz, N., P. Bouman, M. Schmidt. 2018. Optimization approaches for the traveling salesman problem with drone. *Transportation Science* **52**(4) 965–981.

Arakaki, R. K., F. L. Usberti. 2018. Hybrid genetic algorithm for the open capacitated arc routing problem. *Computers & Operations Research* **90** 221–231.

Aurambout, J.-P., K. Gkoumas, B. Ciuffo. 2022. A drone hop from the local shop? where could drone delivery as a service happen in europe and the usa, and how many people could benefit from it? *Transportation Research Interdisciplinary Perspectives* **16** 100708.

Avila, T., A. Corberan, I. Plana, J. M. Sanchis. 2016. A branch-and-cut algorithm for the profitable windy rural postman problem. *European Journal of Operational Research* **249**(3) 1092–1101.

Benavent, E., Á. Corberán, G. Desaulniers, F. Lessard, I. Plana, J. M. Sanchis. 2014. A branch-price-and-cut algorithm for the min-max k-vehicle windy rural postman problem. *Networks* **63**(1) 34–45.

Benavent, E., Á. Corberán, D. Laganà, F. Vocaturo. 2019. The periodic rural postman problem with irregular services on mixed graphs. *European Journal of Operational Research* **276**(3) 826–839.

Bogyrbayeva, A., T. Yoon, H. Ko, S. Lim, H. Yun, C. Kwon. 2023. A deep reinforcement learning approach for solving the traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies* **148** 103981.

Bouman, P., N. Agatz, M. Schmidt. 2018. Dynamic programming approaches for the traveling salesman problem with drone. *Networks* **72**(4) 528–542.

Campbell, J. F., Á. Corberán, I. Plana, J. M. Sanchis. 2018. Drone arc routing problems. *Networks* **72**(4) 543–559.

Campbell, J. F., Á. Corberán, I. Plana, J. M. Sanchis, P. Segura. 2021. Solving the length constrained K-drones rural postman problem. *European Journal of Operational Research* **292**(1) 60–72.

Christofides, N., V. Campos, A. Corberán, E. Mota. 1986. An algorithm for the rural postman problem on a directed graph. *Netflow at pisa* 155–166.

Chung, S. H., B. Sah, J. Lee. 2020. Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers & Operations Research* **123** 105004.

Colombi, M., Á. Corberán, R. Mansini, I. Plana, J. M. Sanchis. 2017. The hierarchical mixed rural postman problem. *Transportation Science* **51**(2) 755–770.

Colombi, M., R. Mansini. 2014. New results for the directed profitable rural postman problem. *European Journal of Operational Research* **238**(3) 760–773.

Corberán, A., R. Eglese, G. Hasle, I. Plana, J. M. Sanchis. 2021. Arc routing problems: A review of the past, present, and future. *Networks* **77**(1) 88–115.

Corberán, Á., G. Laporte. 2015. *Arc routing: problems, methods, and applications*. SIAM.

Corberán, A., R. Martí, A. Romero. 2000. Heuristics for the mixed rural postman problem. *Computers & Operations Research* **27**(2) 183–203.

Corberán, A., J. Sanchis. 1994. A polyhedral approach to the rural postman problem. *European Journal of Operational Research* **79**(1) 95–114.

El-Adle, A. M., A. Ghoniem, M. Haouari. 2023. A variable neighborhood search for parcel delivery by vehicle with drone cycles. *Computers & Operations Research* **159** 106319.

Euchi, J., A. Sadok. 2021. Hybrid genetic-sweep algorithm to solve the vehicle routing problem with drones. *Physical Communication* **44** 101236.

Fernández, E., G. Laporte, J. Rodríguez-Pereira. 2018. A branch-and-cut algorithm for the multidepot rural postman problem. *Transportation Science* **52**(2) 353–369.

Harrison, J. 2025. Amazon mk30 vs ehang eh216-s: Delivery drone comparison. `https://dronewaz.com/amazon-mk30-vs-ehang-eh216-s-delivery-drone-comparison/`. Accessed: 07 Jun 2025.

Hertz, A., G. Laporte, M. Mittaz. 2000. A tabu search heuristic for the capacitated arc routing problem. *Operations Research* **48**(1) 129–135.

Lacomme, P., C. Prins, W. Ramdane-Chérif. 2001. A genetic algorithm for the capacitated arc routing problem and its extensions. *Workshops on Applications of Evolutionary Computation*. Springer, 473–483.

Lee, J. H., M. Kim, J. Park, C. Kwon. 2025. The iterative chainlet partitioning algorithm for the traveling salesman problem with drone and neural acceleration. *arXiv preprint arXiv:2504.15147* .

Lenstra, J. K., A. R. Kan. 1976. On general routing problems. *Networks* **6**(3) 273–280.

Liu, X., S. H. Chung, C. Kwon. 2025. An adaptive large neighborhood search method for the drone–truck arc routing problem. *Computers & Operations Research* **176** 106959.

Liu, Y., J. Shi, Z. Liu, J. Huang, T. Zhou. 2019. Two-layer routing for high-voltage powerline inspection by cooperated ground vehicle and drone. *Energies* **12**(7) 1385.

Longo, H., M. P. De Aragao, E. Uchoa. 2006. Solving capacitated arc routing problems using a transformation to the CVRP. *Computers & Operations Research* **33**(6) 1823–1837.

Luo, H., P. Zhang, J. Wang, G. Wang, F. Meng. 2019. Traffic patrolling routing problem with drones in an urban road system. *Sensors* **19**(23) 5164.

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

Mahmoudinazlou, S., C. Kwon. 2024. A hybrid genetic algorithm with type-aware chromosomes for traveling salesman problems with drone. *European Journal of Operational Research* **318**(3) 719–739.

Mahmoudinazlou, S., C. Kwon, H. Charkhgard. 2024. Solving various classes of arc routing problems with a memetic algorithm-based framework .

Monroy-Licht, M., C. A. Amaya, A. Langevin. 2014. The rural postman problem with time windows. *Networks* **64**(3) 169–180.

Monroy-Licht, M., C. A. Amaya, A. Langevin. 2017. Adaptive large neighborhood search algorithm for the rural postman problem with time windows. *Networks* **70**(1) 44–59.

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

Pearn, W. L., T. Wu. 1995. Algorithms for the rural postman problem. *Computers & Operations Research* **22**(8) 819–828.

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

Sacramento, D., D. Pisinger, S. Ropke. 2019. An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies* **102** 289–315.

Santos, L., J. Coutinho-Rodrigues, J. R. Current. 2010. An improved ant colony optimization based algorithm for the capacitated arc routing problem. *Transportation Research Part B: Methodological* **44**(2) 246–266.

Schermer, D., M. Moeini, O. Wendt. 2019. A matheuristic for the vehicle routing problem with drones and its variants. *Transportation Research Part C: Emerging Technologies* **106** 166–204.

Sobhanan, A., H. Charkhgard, I. Dayarian. 2024a. Equity-driven workload allocation for crowd-sourced last-mile delivery .

Sobhanan, A., S. Mahmoudinazlou, H. Charkhgard, C. Kwon. 2024b. A branch-and-price algorithm for emergency humanitarian logistics with a mixed truck-drone fleet. *Proceedings of the IISE Annual Conference & Expo*.

Tamke, F., U. Buscher. 2021. A branch-and-cut algorithm for the vehicle routing problem with drones. *Transportation Research Part B: Methodological* **144** 174–203.

Vidal, T. 2017. Node, edge, arc routing and turn penalties: Multiple problems—one neighborhood extension. *Operations Research* **65**(4) 992–1010.

Wired Magazine. 2017. A drone-slinging UPS van delivers the future. URL `https://mendoza.nd.edu/news/a-drone-slinging-ups-van-delivers-the-future/`. Accessed: 2025-06-06.