An Adaptive Large Neighborhood Search Method for the Drone-Truck Arc Routing Problem

Xufei Liu[1], Sung Hoon Chung[2], and Changhyun Kwon*[3]

[1]College of Management and E-Business, Zhejiang Gongshang University, Hangzhou, 310018, China, xufeiliu@zjgsu.edu.cn

[2]Department of System Science and Industrial Engineering, Binghamton University, Binghamton, NY 13902, U.S.A., schung@binghamton.edu

[3]Department of Industrial and Systems Engineering, KAIST, Daejeon, 34141, Republic of Korea, chkwon@kaist.ac.kr

December 18, 2024

Abstract

For applications such as traffic monitoring, infrastructure inspection, and security, ground vehicles (trucks) and unmanned aerial vehicles (drones) may collaborate to finish the task more efficiently. This paper considers an Arc Routing Problem (ARP) with a mixed fleet of a single truck and multiple homogeneous drones, called a Drone-Truck Arc Routing Problem (DT-ARP). While the truck must follow a road network, the drone can fly off of it. With a limited battery capacity, however, the drone has a length constraint, i.e., the maximum flight range. A truck driver can replace a battery for the drone after each flight trip. We first transform the DT-ARP into a node routing problem, for which we present a MIP formulation for the case with a truck and a drone. To solve large-size instances with multiple drones, a heuristic method based on Adaptive Large Neighborhood Search is proposed. The performance of ALNS is evaluated on small-size randomly generated instances and large-size undirected rural postman problem benchmark instances. In addition, an analysis is provided on the relationship between truck/drone speeds and the drone's flight range, which affects the difficulty level to solve. The robustness of ALNS is shown via numerical experiments.

Keywords: Drone-Truck; Arc Routing Problem; Adaptive Large Neighborhood Search

1 Introduction

The rapid development of unmanned aerial vehicles, also called drones, and the use of advanced technologies have provided new opportunities in many application areas such as aerial imaging (Rakha and Gorodetsky, 2018), traffic monitoring (Li et al., 2018), infrastructure inspection (Otto et al., 2018), policing and surveillance (Engberts and Gillissen, 2016; Zeng et al., 2022; Amorosi et al., 2023), rescue operations (Rabta et al., 2018), product deliveries (Boysen et al., 2018; Wang and Sheu, 2019), and agriculture (Mogili and Deepak, 2018; Ahirwar et al., 2019). Drones can improve service thanks to their high speed, low cost, and versatility,

*Corresponding author

and can also promote safety by replacing humans for dangerous operations such as wind turbine inspection.

A drone is not limited to the ground transportation infrastructure while servicing the edges. The cooperation of a truck and a drone can be particularly useful for large-scale operations where some edges require service but there are no access roads to them, such as inspection along the power lines/pipelines and wind farm monitoring/inspection (Yu et al., 2019). For example, some electric power lines in mountain areas may not be accessible by ground vehicles while other areas have access roads. In this case, some edges can be covered by drone only while some others can be serviced by drone, truck, or both of them. A truck can supply batteries for drones upon arrival from a flight trip, which makes it possible to overcome the drone's flight range limitation. In this paper, we study the drone-truck mixed fleet operations focused on arc routing applications, so-called the Drone-Truck Arc Routing Problem (DT-ARP). This extends the traditional arc routing problem where the service may not be limited within the road network and is done by the mixed fleet working in a cooperative fashion.

The DT-ARP optimizes the truck and drone routes to minimize the total time of completing all the tasks (all required edges are traversed at least once). Despite the benefits of DT-ARP, it is a complicated problem to make arc routing decisions. The cooperation between the truck and the drone poses multiple challenges. First, the decisions on the truck's route and the drone's route are hierarchical and interdependent. The decision on the drone's takeoff and landing nodes depends on the truck route. Meanwhile, the truck must move along the route that includes takeoff and landing nodes. Second, because the drone has limited battery capacity, each flight trip has a physical constraint, i.e., the maximum flight range. The drones must land on the truck frequently, and the driver replaces the battery for the drones. Third, it is allowed to fly over multiple arcs in one flight trip as long as the flight length is less than the maximum flight range. So, it is the uncertain number of arcs in each trip.

Our DT-ARP is NP-hard because it is a generalization of another well-known arc routing problem, the Rural Postman Problem (RPP), proven to be NP-hard (Lenstra and Kan, 1976). As a starting point, we first consider the simple case of a single drone and a single truck, for which we develop mathematical formulations and a computational method. Then, we extend our attention to the more complicated cases of multiple drones and a single truck.

Developing a mathematical formulation for the arc routing problems of interest is not straightforward. The objective function in this paper is to minimize the total time of completing all tasks, not the total traveling time; therefore, one vital decision variable is the arrival times at each node. While most current research assumes that the required edges are required to be traversed once, this paper assumes that the required edge is allowed to be traversed at least once. The multiple revisit on one node happens when the edge is traversed several times. Because we do not know how many times the same node will be visited, it is also difficult to determine the variable index that represents the n-th arrival time at the same node. If we just create a large enough number of dummy nodes for each node in the original graph, the graph size becomes very large, and in the meantime, quite a lot of dummy nodes will not be visited. This leads to a situation where the computation time increases a lot, but the efficiency goes down. It is also impossible to associate a unique starting and completion time for an edge; hence, the arc routing problem problems with the time are very hard to model directly without extensive graph modification (Monroy-Licht et al., 2014).

Instead of developing an *arc* routing formulation, we develop *node* routing formulations using two kinds of transformations. The idea is to create *side nodes* for each node corresponding to every required edge and only keep these dummy nodes to compose a complete graph. The traveling salesman problem on this graph is equivalent to the arc routing problem with visiting all required edges at least once. Pearn et al. (1987) added two side nodes and one middle node over each required edge and ensured that each edge is traversed when all nodes are visited once. Longo et al. (2006) added two side nodes over each required edge and ensured that each edge is traversed when the side nodes are visited in sequence. Then, our single-drone-single-truck DT-ARP can be transformed into a node routing problem, which becomes a traveling salesman problem with a drone (TSP-D), wherein the drone can visit multiple nodes in a fly-out; the resulting problem is called the multi-visit TSP-D (Poikonen and Golden, 2020; Luo et al., 2021). The formulations are mixed integer program problems that may be solved by off-the-shelf optimization solvers such as CPLEX or Gurobi.

For large-scale problems, it is essential to develop an efficient heuristic method to solve DT-ARP. In this paper, we develop an adaptive large neighborhood search (ALNS) algorithm to solve the problem. ALNS, first proposed by Ropke and Pisinger (2006), is a well-known iterative metaheuristic framework that has been popularly applied to solving various vehicle routing problems. ALNS was first applied to the arc routing problem by Laporte et al. (2010), who solved the capacitated arc routing problem with stochastic demands and multiple vehicles to minimize the total cost. The key characteristic of ALNS is to *destroy* an incumbent solution and *repair* it to construct a new solution in each iteration (Pisinger and Ropke, 2019). However, applying ALNS on DT-ARP is not straightforward because two decisions for the truck routes and drone routes are tangled. We propose efficient destroy and repair methods to handle such complexity within the ALNS framework in this paper. We also extend our ALNS algorithm to the more complex case of multiple drones and a truck.

We demonstrate the effectiveness and efficiency of the ALNS algorithm via extensive numerical experiments. We use the multi-start tabu search (MSTS) algorithm (Luo et al., 2021) devised for the multi-visit TSP-D as a benchmark, as well as the MIP formulations we develop. Numerical experiments are conducted for randomly generated instances of various sizes and a set of large-size benchmark Undirected Rural Postman Problem instances (Corberán et al., 2021b).

The remainder of the paper is written as follows. In Section 2, the related literature is reviewed. The problem statement and mathematical model are presented in Section 3. Adaptive Large Neighborhood Search is described in detail in Section 4. In Section 5, the experiment results validate the performance of ALNS to solve the Drone-Truck Arc Routing Problem. Conclusions are summarized in Section 6.

2 Related Works

There have been numerous studies to investigate optimization for the arc routing problem. Although there is rapidly growing literature on the arc routing problems with trucks or drones, the research on the cooperation between the truck and the drone has been previously assessed only to a limited extent.

Some papers discussed the arc routing problem only with a single vehicle or a fleet of homogeneous vehicles. Classical arc routing problems include the Chinese Postman Problem, the Rural Postman Problem, and the Capacitated Arc Routing Problem (Golden and Wong, 1981); see surveys (Eiselt et al., 1995a,b; Corberán and Laporte, 2015; Corberán et al., 2021a) for rich developments for arc routing problems. Most relevant to our work, Monroy-Licht et al. (2017) proposed an ALNS algorithm to solve the rural postman problem with time windows of serving some required edges with one vehicle and solved a set of large instances with up to 104 required edges. Since our problem considers both drones and a truck, our ALNS algorithm should handle additional complexities.

Some papers consider arc routing problems with drones in various contexts, which include patrolling (Oh et al., 2011, 2014; Momeni et al., 2022), sensor covering (Sipahioglu et al., 2010; Dille and Singh, 2013), and traffic monitoring (Chow, 2016; Li et al., 2018; De Maio et al., 2021). Drone arc routing problems are unique since drones can fly not only over the existing road networks but also between any two points freely unless prohibited by law or obstructed by buildings. In this sense, Campbell et al. (2018) studied drone arc routing problems to minimize the total cost, where drones can travel directly between *any* two points and approximate each curve in the plane by a polygonal chain. The drones leave and enter at the points on the polygonal chain. An iterative algorithm is proposed to solve RPP instances with an increasing number of points of the polygonal chain. Campbell et al. (2021) also digitized the Length Constrained K-Drones Rural Postman Problem by a polygonal chain with a finite number of points. Since we can use similar ideas to introduce polygonal chains in our DT-ARP, we will restrict drone movements to a predefined graph only in this paper. Altin and Sipahioglu (2024) considered an extension with multiple service drones and proposed a simulated annealing algorithm.

Since Murray and Chu (2015) introduced the Traveling Salesman Problem with Flying Sidekicks, the interaction between trucks and drones—to reduce the total travel cost or the total completion time of services while considering the drone's limited service capacity and flying range—has been popularly considered in the *node* routing literature (Macrina et al., 2020; Chung et al., 2020; Moshref-Javadi et al., 2020). For a simpler problem specification, Agatz et al. (2018) used the term, the Traveling Salesman Problem with Drones (TSP-D). Some assumed the drone can visit only one node at a time (Agatz et al., 2018; Poikonen et al., 2019; Wang et al., 2019; Tamke and Buscher, 2021; Bogyrbayeva et al., 2023; Ghiasvand et al., 2024), after which the drone should come back to the truck, while some assumed drones can visit multiple nodes in a single fly-out (Poikonen and Golden, 2020; Luo et al., 2021; Jiang et al., 2024).

Multi-visit problems are relevant to our DT-ARP since we can transform DT-ARP into a multi-visit TSP-D. Although we will provide MIP formulations for such transformed multi-visit TSP-D, we will develop an algorithm directly for DT-ARP. For multi-visit problems, we introduce the existing literature. Ha et al. (2020) developed a hybrid genetic search to solve the TSP with a truck and a drone for delivering parcels to customers with minimizing the completion time. Di Puglia Pugliese et al. (2021) built an MILP and developed a heuristic method for a last-mile

Table 1: A Summary of the Drone-Truck Arc Routing Problem Literature (MILP = Mixed Integer Linear Programming; MINLP = Mixed Integer Non-Linear Programming)

Reference	Truck	Drone	Objective	Visit Req-Edge	Req-Edge Visited by	Model	Method
Amorosi et al. (2021)	1	1	Total distance	Required Coverage	Drone	MINLP	Heuristic
Petitprez et al. (2021)	M	M	Cost; Inspection performance	At least once	Drone	MINLP	Hybrid GA
Wu et al. (2022)	1	M	Completion time	At least once	Drone	MINLP	ALNS
Xu et al. (2023)	1	M	Completion time	At least once	Drone	MILP	RVND-SA
Amorosi et al. (2023)	1	M	Completion time	Required Coverage	Drone	MINLP	Heuristic
Xue et al. (2023)	M	M	Travel cost	Exactly once	Drone	MILP	Saving heuristic -TS
Sun et al. (2024)	M	M	Travel cost	At least once	Drone	MILP	VNS
Zandieh et al. (2024)	M	M	Travel cost	At most once (zone)	Drone	MINLP	ILS-VND
This study	1	M	Completion time	At least once	Drone or Truck	MILP	ALNS

delivery routing problem under time window, capacity, and flying endurance constraints to minimize the cost. Luo et al. (2022) proposed an iterated local search algorithm for the pickup and delivery problem. Kuo et al. (2022) developed a mixed-integer programming model for the cooperation of trucks and drones in the delivery task and considered the customer time windows. A variable neighborhood search is proposed to solve it to minimize the total traveling costs. Li et al. (2022) studies how to find suitable drone launch and retrieval locations with moving trucks and developed ALNS to solve it. Zhou et al. (2023) studied a two-echelon vehicle-drone routing problem where multiple vehicles and drones work collaboratively to serve customers. They provided an MILP model and a set-partitioning model and developed a branch-and-price algorithm. Morandi et al. (2023) solved the traveling salesman problem with a truck and multiple drones to serve customers in the minimum time and discussed the consequence of allowing the revisit on each node. Rave et al. (2023) studied the decisions on the vehicle fleet mix and drone station locations and developed specialized ALNS for the practice-sized instances. Xia et al. (2023) also developed a branch-and-price-and-cut algorithm based on the Danzig-Wolfe decomposition framework to solve the vehicle routing problem with load-dependent drones. Jiang et al. (2024) formulated a MILP for a multi-visit flexible-docking vehciel pickup and delivery routing problem with a truck and a drone in rural areas and proposed a tailored adaptive large neighborhood search metaheuristic.

So far, we have discussed research works for drone-truck combined *node* routing problems. We introduce the literature of drone-truck *arc* routing problem now, as summarized in Table 1. Amorosi et al. (2021) formulated an MINLP for the arc routing problem with a mothership truck and a drone to minimize the total distance. In contrast to other studies, they did not require every task edge must be visited and just wanted to guarantee the required coverage of the target graphs. They further extended the problem by considering multiple drones and a truck to minimize the makespan, and a heuristic was proposed for a large-size real case of inspecting the people flow during the Cordoba Courtyards Festival (Amorosi et al., 2023). Petitprez et al. (2021) solved a Two-Echelon vehicle routing problem in which drones fly over linear infrastructures by Hybrid Genetic Algorithm (HGA). They only allowed the revisitation over an edge by the drone and used k-th drone index to distinguish the arrival times, while our study allows it by either the truck or the drone. They ignored the truck's or the drone's waiting time at each rendezvous node. Wu et al. (2022) studied a truck-drone arc routing problem with ALNS where the predefined number of the drone sortie and k-th drone index distinguish the arrival times at the same node. Xu et al. (2023) studied vehicle-drone arc routing problems, which coordinated vehicles and drones to monitor road networks, while the drones are used to inspect some specific road segments. A hybrid algorithm was proposed by combining the randomized variable neighborhood descent search and simulated annealing algorithm (RVND-SA). They allowed every target arc to be visited only by drones at least once and prevented one drone from visiting a node multiple times in a flight. All possible combinations of rendezvous nodes and the drone index were used to distinguish the multiple arrival times at the same node. Xue et al. (2023) investigated a two-stage vessel-UAV arc routing problem for offshore oil and gas pipeline inspection. A two-stage formulation was established to minimize the total travel cost where. The first stage was a UAV routing problem with fuzzy service time, and the second stage solved the vessel routing problem. A saving heuristic and a genetic-tabu search heuristic were proposed to solve the first- and second-stage models, respectively. Sun et al. (2024) studied an integrated ground vehicles routing problem and an arc routing problem with arc window time where the drones conducted a pre-reconnaissance of the traveling routes of the ground vehicles. They built an MILP model and developed a Variable Neighborhood Search algorithm integrated with a greedy search and simulated annealing. Zandieh et al. (2024) studied a similar problem where the drones must surveil the ground vehicle through the link ahead before the ground vehicle leaves the customers' points. An improved iterated local search (ILS) algorithm was designed to optimize the proposed problem. The ground vehicles are responsible for delivering goods to customer points, and the drones surveil the road arcs (Sun et al., 2024) and zones (Zandieh et al., 2024). Corberán et al. (2025) considered a combined truck-drone arc routing problem considering the makespan minimization objective and simultaneous location decisions.

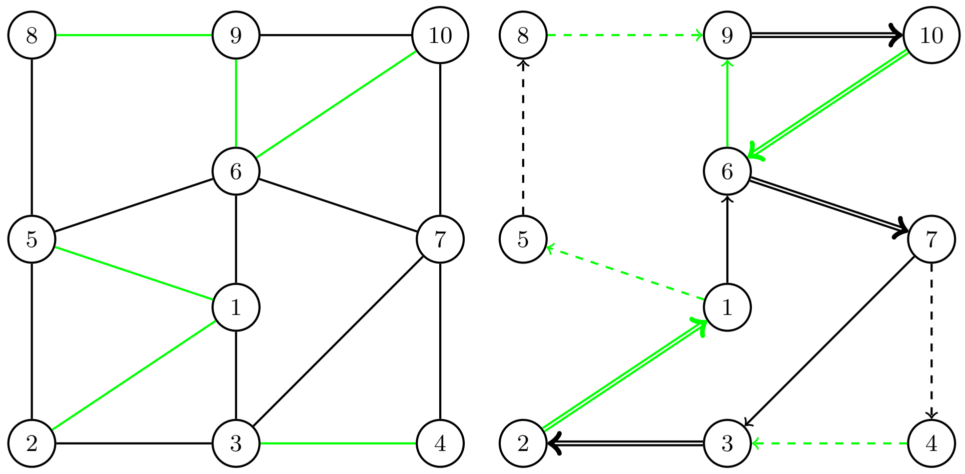
Our problem is similar to the paper by Wu et al. (2022) that studied a coordinated vehicle-drone arc routing problem with an adaptive large neighborhood search. The first difference is that they restricted some edges required to be serviced only by the drone, while we do not have this assumption in our paper. The second difference is that they only established a nonlinear mathematical model and did not try to solve the model to validate it. We will build a mixed-integer linear program, and the optimal solution can be obtained by solving MIP with Gurobi solver. The solutions obtained from MIP can be considered as Upper-Bound to validate the effectiveness of our proposed approach. The third difference is the encoding rule of the ALNS method. They encoded a route as a matrix. In our ALNS, a route is encoded by the string of arcs that represent the sequence of traversing the arcs. The encoding form of the string enables simplifying the computations.

3 Problem Formulation

For a single drone and a single truck case, the DT-ARP can be described as follows using the notation given in Table 2. Let $G = (\mathcal{N}, \mathcal{E})$ be an undirected connected graph. The depot is labeled as node 1. As stated in Section 1, for some applications such as traffic monitoring and security inspection, specific road segments and power lines/pipelines are required to be serviced. Due to the high speed, safety, low cost, and versatility, the drone working in tandem with the truck can improve service. The truck and the drone cooperatively complete the task in which

Table 2: Mathematical Notation

Graphs and Sets	
G	Original undirected graph, $G = (\mathcal{N}, \mathcal{E})$
$\mathcal{N}$	Set of original N nodes, $\mathcal{N} = \{1, 2, ..., N\}$; node 1 is the depot.
$\mathcal{E}$	Set of original undirected edges, $\mathcal{E} \in \mathcal{E}^{DR}$
$\mathcal{E}^{\text{DR}}$	Set of original undirected edges, including extended edges available for the drone
$\mathcal{R}$	Set of original undirected required edges $\mathcal{R} \subset \mathcal{E}$
$\mathcal{V}$	Set of nodes after a transformation with $V = \lvert\mathcal{V}\rvert$
$\mathcal{V}'$	Set of nodes with a dummy node, $\mathcal{V}' = \mathcal{V} \cup \{V + 1\}$
$\mathcal{V}_1$	Set of nodes excluding the depot, $\mathcal{V}_1 = \mathcal{V} \setminus \{1\}$
$\mathcal{Q}$	Set of undirected required edges after a transformation
$\mathcal{A}$	Set of directed arcs after a transformation, induced by $\mathcal{V}'$
Constants	
c_{ij}	Distance of edge $(i, j) \in \mathcal{E}$
$d^{\text{TR}}(i, j)$	Distance of arc $(i, j) \in \mathcal{A}$ for the truck after a transformation
$d^{\text{DR}}(i, j)$	Distance of arc $(i, j) \in \mathcal{A}$ for the drone after a transformation
v^{TR}	Truck speed
v^{DR}	Drone speed
t^{TR}_{ij}	Time of traversing arc $(i, j) \in \mathcal{A}$ for the truck
t^{DR}_{ij}	Time of traversing arc $(i, j) \in \mathcal{A}$ for the drone
e	Maximum flight time, $e = \frac{\text{Maximum Drone flight distance}}{v^{\text{DR}}}$
Variables	
x^{TR}_{ij}	1, if the truck traverses through arc $(i, j) \in \mathcal{A}$; Otherwise, 0.
x^{DR}_{ij}	1, if the drone traverses through arc $(i, j) \in \mathcal{A}$; Otherwise, 0.
y^{TR}_{i}	1, if node $i \in \mathcal{V}_1$ is visited only by the truck; Otherwise, 0.
y^{DR}_{i}	1, if node $i \in \mathcal{V}_1$ is visited only by the drone; Otherwise, 0.
y^{CB}_{i}	1, if node $i \in \mathcal{V}_1$ is combined node where a drone launches or lands; Otherwise, 0.
n^{TR}_{i}	The ordered visit sequence of nodes for the truck
n^{DR}_{i}	The ordered visit sequence of nodes for the drone
f_i	The flight duration when the drone arrives at node $i \in \mathcal{V}$.
a_i	The arrival time of the truck or the drone at node $i \in \mathcal{V}$.



(a) An Example of Original ARP Graph G (b) An Arc-Solution of Truck-Drone Routes

Figure 1: An Example of An Arc-Solution (green lines are the required edges; In Figure b: one truck and one drone depart from and return to the depot Node 1; dashed lines are drone flight route; solid lines are truck route; double solid lines represent the routes of truck carrying the drone)

the required edges must be traversed at least once. Define $\mathcal{R}$ as the set of required edges that must be served either by the truck or the drone. The aim is to find the truck and drone routes to minimize the makespan, i.e., the time between departure from and return to the depot.

We introduce some simplifying assumptions about the drone.

Assumption 1. Because a drone can fly off the actual road edge, the drone flight network is larger than the actual road network G. Define the set of drone flight edges $\mathcal{E}^{\text{DR}}$. Assume that $\mathcal{E}^{\text{DR}}$ consists of all available paths between any two nodes $i, j \in \mathcal{N}$ in regards to the drone's flight regulations. The distance of drone flight is calculated as the horizontal distance for the sake of simplicity.

Assumption 2. The drone has a maximum flight range because of its limited battery capacity. The drone must fly back to a truck before the battery runs out. After the drone lands on the truck, the driver replaces a fully charged backup battery and makes sure the drone gets prepared for the next trip.

Assumption 3. The times for the drone to launch and land are neglected. The time to replace the battery is also neglected.

Figure 1 illustrates an example of an arc solution. It comprises truck and drone routes which are the sequence of traversing the arcs. In the example, the truck route is $\{(1,6),(6,9),(9,10),(10,6),(6,7),(7,3),(3,2),(2,1)\}$ and the drone route is $\{(1,5),(5,8),(8,9),(9,10),(10,6),(6,7),(7,4),(4,3),(3,2),(2,1)\}$. The overlap between the two routes means the drone gets aboard the truck.

One of the major considerations in the formulation for DT-ARP is how to calculate the accurate arrival times at each node when the objective function is to minimize the makespan, not the total cost. If we directly study the arc routing problem, it is hard to find a way to formulate MIP. Because multiple arcs are connected to one node, one node can be visited by uncertain multiple times. For example in Figure 1, there are two arrival times at Node 6. In order to overcome this challenge, we use arc-to-node transformation to deal with the multiple visits at the same node. The arc-to-node transformation enables one node to be split into side nodes. For example in Figure 1, Node 6 is connected with two edges, and thus Node 6 has two side nodes, s_{69} and $s_{6,10}$. The arrival times at node s_{69} and $s_{6,10}$ represent the first and the second arrival time at node 6, respectively.

The formulation for DT-ARP is made in two stages: (1) transforming the arc routing problem into the corresponding node routing problem; (2) formulating a mixed integer program. The final truck route and the drone route can be obtained by transforming back the node routing solutions to the original arc routing problem.

3.1 Transforming to node routing problems

We apply two arc-to-node transformation methods proposed by Pearn et al. (1987) and Longo et al. (2006). Pearn et al. (1987) replaced each required edge with three nodes while Longo et al. (2006) used two nodes and achieved the same objective with additional constraints.

3.1.1 The 3-Node Transformation

Pearn et al. (1987) transformed arc routing problems into node routing problems by replacing each required edge $(i,j) \in \mathcal{R}$ by two side nodes s_{ij} and s_{ji} along with a middle node m_{ij}. We let $m_{ij} = m_{ji}$ denote the same middle node. The corresponding node routing problem is defined on the set of nodes

$$\mathcal{V} = \bigcup_{(i,j)\in\mathcal{R}} \{s_{ij}, m_{ij}, s_{ji}\} \cup \{1\},$$

and all undirected edges between the nodes to construct an undirected *complete* graph. The purpose of the middle node m_{ij} is to ensure that the shortest path between two side nodes s_{ij} and s_{ji} is always $s_{ij} \to m_{ij} \to s_{ji}$ or $s_{ji} \to m_{ij} \to s_{ij}$ in sequence. Passing the middle node m_{ij} fulfills traversing the required edge. Therefore, we do not need the set of required edges in the transformed node routing problem. The resulting graph will have $3|\mathcal{R}| + 1$ nodes. An example is shown in Figure 2, where three nodes are introduced for each required edge. The original nodes 2, 3, and 4 are not part of the new transformed graph, but the depot node 1.

The length of each edge in the transformed graph is calculated as follows:

$$d(s_{ij}, s_{kl}) = \begin{cases} 0 & \text{if } (i,j) = (k,l) \\ c_{ij} & \text{if } (i,j) = (l,k) \\ \text{dist}(i,k) & \text{if } (i,j) \neq (k,l), (i,j) \neq (l,k) \end{cases}$$

$$d(1, s_{ij}) = \text{dist}(1,i)$$

$$d(m_{ij}, u) = \begin{cases} \frac{1}{2}c_{ij} & \text{if } u = s_{ij} \text{ or } s_{ji} \\ \infty & \text{otherwise} \end{cases}$$

where $\text{dist}(i,j)$ is the shortest path length between node $i \in \mathcal{N}$ and $j \in \mathcal{N}$ in the original graph

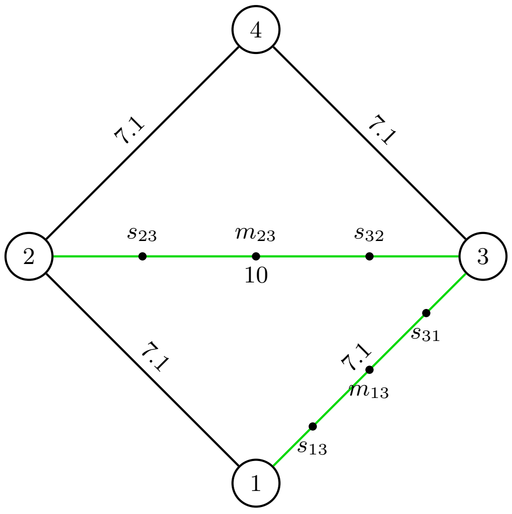


Figure 2: A 3-node transformation example with new nodes introduced by Pearn et al. (1987) (original arc distance is shown next to each arc; green lines are the required edges)

based on the edge length c_{ij}. Since the graph topology for the truck and the graph topology for the drone are different, a length function $d(\cdot,\cdot)$ is created for the truck and the drone separately: $d^{\text{TR}}(\cdot,\cdot)$ and $d^{\text{DR}}(\cdot,\cdot)$, respectively.

Note that the above distance calculation rule is different from the original form in Pearn et al. (1987), but similar to the 2-node transformation of Longo et al. (2006). This modification will enable us to calculate the arrival times and the makespan accurately later in our MIP formulation.

3.1.2 The 2-Node Transformation

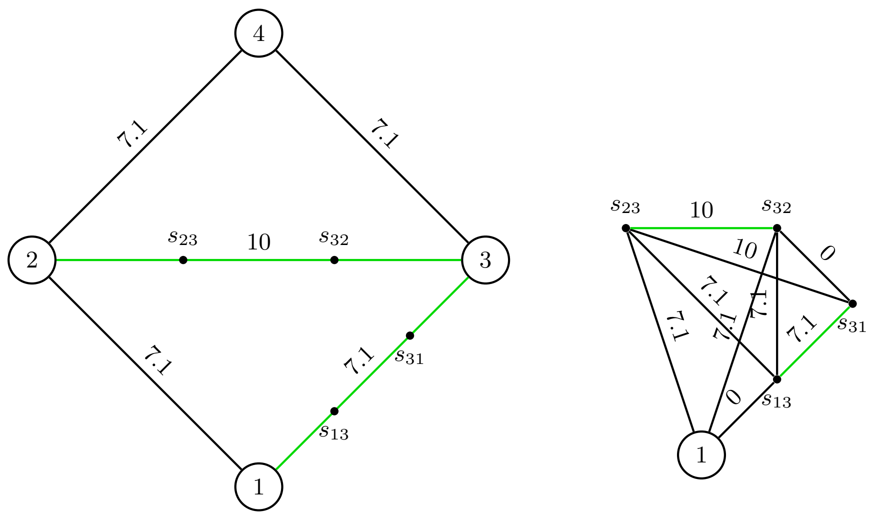
Longo et al. (2006) replaced each required edge $(i,j) \in \mathcal{R}$ with two nodes s_{ij} and s_{ji}. The set of the undirected required edges $\mathcal{Q}$ is then defined as follows:

$$\mathcal{Q} = \{(s_{ij}, s_{ji}) | (i,j) \in \mathcal{R}\}$$

In the 2-node transformation, traversing the required edge (i,j) is the same as visiting two nodes s_{ij} and s_{ji} in sequence ($s_{ij} \to s_{ji}$ or $s_{ji} \to s_{ij}$), which we need to enforce in an optimization problem. While the 2-node transformation reduces the size of the transformed graph, it requires additional constraints. The transformed node routing problem is defined on the complete undirected graph with the node set:

$$\mathcal{V} = \bigcup_{(i,j)\in\mathcal{R}} \{s_{ij}, s_{ji}\} \cup \{1\},$$

and it has $2 \times |\mathcal{R}| + 1$ nodes.



(a) Original ARP graph G

(b) Corresponding VRP complete graph H

Figure 3: Longo et al. (2006) Transformation from ARP to VRP (Green lines show the required edges. The values above edges are the distance between internodes) in Figure 3b.

The distances between internodes are calculated as follows (Longo et al., 2006):

$$d(s_{ij}, s_{kl}) = \begin{cases} 0 & \text{if } (i,j) = (k,l) \\ c_{ij} & \text{if } (i,j) = (l,k) \\ \text{dist}(i,k) & \text{if } (i,j) \neq (k,l), (i,j) \neq (l,k) \end{cases}$$

$$d(1, s_{ij}) = \text{dist}(1, i)$$

where $\text{dist}(i,j)$ is the shortest path distance between node i and j. As in the 3-node transformation case, $d^{\text{TR}}(\cdot,\cdot)$ and $d^{\text{DR}}(\cdot,\cdot)$ are created for the truck and the drone, respectively.

During the transformation from arc routing problem to node routing problem, the solution changes from arc solution to node solution. The node solution is the sequence of visiting nodes. We can get the node-solution by solving the MIP formulation in Section 3.2. The arc solution can be obtained from the equivalent node solution by applying the arc-to-node transformation method in the reverse direction. In the reverse transformation, firstly the side node is transformed back to its original node. In the next step, two adjacent nodes are connected by the shortest path based on the original network.

For example in Figure 1, the truck route in node form is obtain as $\{1, s_{69}, s_{96}, s_{10,6}, s_{6,10}, s_{21}, s_{12}, 1\}$. Transform side node into original node: $\{1, 6, 9, 10, 6, 2, 1\}$; Then connect any two adjacent nodes with the shortest path: the truck route in arc form is {(1,6),(6,9),(9,10),(10,6),(6,7), (7,3),(3,2),(2,1)}. The drone route in node form is obtained as $\{1, s_{15}, s_{51}, s_{89}, s_{98}, s_{10,6}, s_{6,10}, s_{43}, s_{34}, s_{21}, s_{12}, 1\}$. The side nodes are converted into the original node: $\{1, 5, 8, 9, 10, 6, 4, 3, 2, 1\}$. The drone route in arc form is {(1,5),(5,8),(8,9),(9,10),(10,6),(6,7),(7,4),(4,3),(3,2),(2,1)}.

3.2 MIP Formulation

For the transformed node routing problem, we build a MIP formulation, extended from the formulation provided by Roberti and Ruthmair (2021) for TSP-D. We make the following modifications and extensions from Roberti and Ruthmair (2021) to accommodate the unique characteristics of DT-ARP. First, they set the arrival time at one node based on the assumption that the drone's speed is greater than the truck's speed. We do not need this assumption. We modify the arrival time restriction as in Constraint (17) so that the time through one arc depends on the truck's speed when the drone gets aboard the truck ($x_{ij}^{\mathrm{TR}} = x_{ij}^{\mathrm{DR}} = 1$). Second, their subtour elimination constraints (arrival times restriction) become invalid here, because some distances between internodes are zero in our problem, such as $d_{1,s_{13}} = d_{s_{31},s_{32}} = 0$ shown in Figure 3b. Therefore, we introduce the variables n^{TR} and n^{DR} to denote the ordered sequence of visiting nodes. Additional constraints, (13) and (14), are added to eliminate subtours. Third, they restricted the drone is allowed to serving only one node during a single flight trip, while our formulation allows it to serve multiple nodes during a single flight trip.

In what follows, the vertex set $\mathcal{V}'$ is defined as $\mathcal{V}' = \mathcal{V} \cup \{V+1\}$, where node $V+1$ represents depot-enter node. Define $\mathcal{V}_1 = \mathcal{V} \setminus \{1\}$ as the set of nodes excluding the depot. The undirected edges are extended as *directed arcs* $\mathcal{A} = \{(i,j) | i,j \in \mathcal{V} : i \neq j\} \cup \{(i, V+1) | i \in \mathcal{V} : i \neq 1\}$. The mathematical formulation is described as follows.

The Objective Function We aim to find routes of a truck and a drone to minimize the total completion time, i.e., the makespan. The value of the makespan is calculated between the times when the truck leaves and returns to the depot as follows:

$$\text{minimize} \quad a_{V+1} \tag{1}$$

Required Edges Each undirected required edge must be traversed at least once by the truck or the drone, as in Constraint (2). It is noted that Constraint (2) is only needed for the 2-node formulation. In the 2-node transformation, visiting s_{ij} and s_{ji} in sequence ensures traversing the edge (i,j). This ensures that all originally required edges are traversed when all side vertices are visited. Constraint (3) restricts that truck routing decision x^{TR} and drone routing decision x^{DR} are binary variables.

$$x_{ij}^{\mathrm{TR}} + x_{ji}^{\mathrm{TR}} + x_{ij}^{\mathrm{DR}} + x_{ji}^{\mathrm{DR}} \geq 1 \qquad \forall (i,j) \in \mathcal{Q} \tag{2}$$

$$x_{ij}^{\mathrm{TR}}, x_{ij}^{\mathrm{DR}} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A} \tag{3}$$

Flow for Truck Route The truck leaves and returns to the depot exactly once. Outflow at the depot-leave node 1 is 1 and inflow at depot-enter node $V+1$ is also 1 (Constraint (4)). Constraint (5) restricts the flow balance for other nodes.

$$\sum_{(1,j)\in\mathcal{A}} x_{1j}^{\mathrm{TR}} = \sum_{(i,V+1)\in\mathcal{A}} x_{i,V+1}^{\mathrm{TR}} = 1 \tag{4}$$

$$\sum_{(i,j)\in\mathcal{A}} x_{ij}^{\mathrm{TR}} - \sum_{(j,i)\in\mathcal{A}} x_{ji}^{\mathrm{TR}} = 0 \qquad \forall i \in \mathcal{V}_1 \tag{5}$$

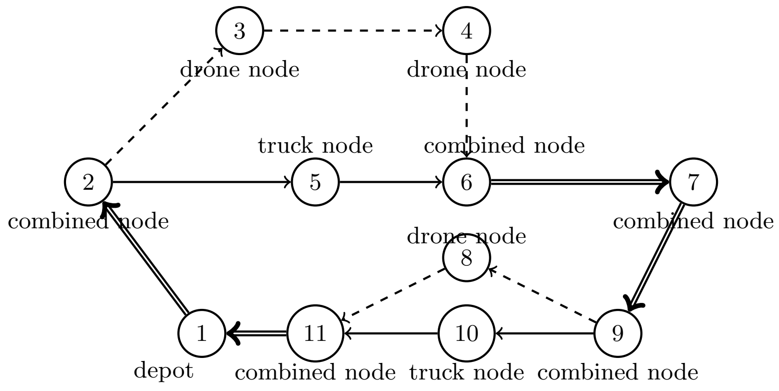


Figure 4: An example for node category in one flight trip (solid lines are the truck route; dashed lines are the drone route; double lines are route of the truck carrying the drone)

Flow for Drone Route Constraints (6) and (7) restrict the flow balance for the drone route.

$$\sum_{(1,j)\in\mathcal{A}} x_{1j}^{\mathrm{DR}} = \sum_{(i,V+1)\in\mathcal{A}} x_{i,V+1}^{\mathrm{DR}} = 1 \tag{6}$$

$$\sum_{(i,j)\in\mathcal{A}} x_{ij}^{\mathrm{DR}} - \sum_{(j,i)\in\mathcal{A}} x_{ji}^{\mathrm{DR}} = 0 \qquad \forall i \in \mathcal{V}_1 \tag{7}$$

Node Category Let y_i^{TR} be a binary variable equal to 1 if node $i \in \mathcal{V}_1$ is only visited by the truck, called a truck node. y_i^{DR} is a binary variable that is equal to 1 if node $i \in \mathcal{V}_1$ is only visited by the drone, called drone node. Let $y_i^{\mathrm{CB}} \in \{0,1\}$ be equal to 1 if node $i \in \mathcal{V}_1$ is visited by both the truck and the drone, called the combined node.

In the example in Figure 4, all nodes are divided into three categories excluding the depot. The drone traverses through combined and drone nodes, and the truck traverses through combined and truck nodes. It is noted that the drone launches or lands at the combined nodes but not all combined nodes are chosen for launching or landing. In the example, Node 7 is visited by the truck and the drone onboard, and is not chosen for launching or landing. Each arc in the drone route can connect with the combined or the drone node, and however it can not connect with the truck node.

Constraint (8) ensures that each node must be one of three categories of nodes. Constraint (9) ensures each arc in the drone route is not connected with a truck node. Constraints (10) and (11) link x_{ij}^{TR} and x_{ij}^{DR} with y and ensure that along the truck (drone) route, the node is either truck (drone) node or combined node. Constraint (12) ensures variable y.

$$y_i^{\mathrm{TR}} + y_i^{\mathrm{DR}} + y_i^{\mathrm{CB}} = 1 \qquad \forall i \in \mathcal{V}_1 \tag{8}$$

$$x_{ij}^{\mathrm{DR}} + x_{ji}^{\mathrm{DR}} \le (1 - y_i^{\mathrm{TR}}) + (1 - y_j^{\mathrm{TR}}) \qquad \forall (i,j) \in \mathcal{A} : i,j \notin \{1, V+1\} \tag{9}$$

$$\sum_{(i,j)\in\mathcal{A}} x_{ij}^{\mathrm{TR}} = y_i^{\mathrm{TR}} + y_i^{\mathrm{CB}} \qquad \forall i \in \mathcal{V}_1 \tag{10}$$

$$\sum_{(i,j)\in\mathcal{A}} x_{ij}^{\mathrm{DR}} = y_i^{\mathrm{DR}} + y_i^{\mathrm{CB}} \qquad \forall i \in \mathcal{V}_1 \tag{11}$$

$$y_i^{\mathrm{TR}}, y_i^{\mathrm{DR}}, y_i^{\mathrm{CB}} \in \{0,1\} \qquad \forall i \in \mathcal{V}_1 \tag{12}$$

Subtour Elimination The subtour elimination constraints are added to avoid causing subtours for the truck (13) and for the drone (14). Let n_i^{TR} and n_i^{DR} denote the sequence order of visiting nodes for the truck and the drone, respectively. Let $V = |\mathcal{V}|$.

$$n_j^{\text{TR}} \geq n_i^{\text{TR}} + Vx_{ij}^{\text{TR}} - (V-1) \qquad \forall (i,j) \in \mathcal{A} \tag{13}$$

$$n_j^{\text{DR}} \geq n_i^{\text{DR}} + Vx_{ij}^{\text{DR}} - (V-1) \qquad \forall (i,j) \in \mathcal{A} \tag{14}$$

$$n_i^{\text{TR}}, n_i^{\text{DR}} \in \mathbb{Z}_+ \qquad \forall i \in \mathcal{V}_1 \tag{15}$$

where $\mathbb{Z}_+$ denotes the set of nonnegative integers.

Arrival Time Let $a_i \in \mathbb{R}_+$ be the arrival time of the truck or the drone at node $i \in \mathcal{V}$. Define t_{ij}^{TR} and t_{ij}^{DR} be the time of traversing arc (i,j) for the truck and the drone, respectively:

$$t_{ij}^{\text{TR}} = \frac{d^{\text{TR}}(i,j)}{v^{\text{TR}}} \quad \text{and} \quad t_{ij}^{\text{DR}} = \frac{d^{\text{DR}}(i,j)}{v^{\text{DR}}}$$

where $d^{\text{TR}}(i,j)$ and $d^{\text{DR}}(i,j)$ are calculated using the 3-node or 2-node transformation function. Constraints (16) and (17) set the arrival times of the truck and the drone at the node. It is noted that when the drone gets aboard the truck ($x_{ij}^{\text{DR}} = 1, x_{ij}^{\text{TR}} = 1$), the arrival time at node j only depends on the truck's traverse time. Constraints (18) and (19) show that the total completion time cannot be lower than the summation of the traverse time by the truck or by the drone. Constraint (20) restricts that the arrival time at each node is a nonnegative continuous variable.

$$a_j \geq a_i + t_{ij}^{\text{TR}} - M(1 - x_{ij}^{\text{TR}}) \qquad \forall (i,j) \in \mathcal{A} \tag{16}$$

$$a_j \geq a_i + t_{ij}^{\text{DR}} - M(1 - x_{ij}^{\text{DR}}) - Mx_{ij}^{\text{TR}} \qquad \forall (i,j) \in \mathcal{A} \tag{17}$$

$$\sum_{(i,j)\in\mathcal{A}} t_{ij}^{\text{TR}} x_{ij}^{\text{TR}} \leq a_{V+1} \tag{18}$$

$$\sum_{(i,j)\in\mathcal{A}} t_{ij}^{\text{DR}} x_{ij}^{\text{DR}} \leq a_{V+1} \tag{19}$$

$$a_i \geq 0 \qquad \forall i \in \mathcal{V}' \tag{20}$$

Drone Flight Range The drone has a maximum flight range because of the limited battery. Let e be the maximum consecutive flight time. Constraint (21) guarantees that the drone can not traverse an arc whose flight time exceeds e unless the drone gets aboard the truck. A variable f_i is introduced to track the flight time in a trip. Constraint (22) sets the tracking flight time f_i. The flight time of a trip must be not greater than e, as in Constraint (23).

$$x_{ij}^{\text{DR}} \leq x_{ij}^{\text{TR}} \qquad \forall (i,j) \in \mathcal{A} : t_{ij}^{\text{DR}} > e \tag{21}$$

$$f_j \geq f_i + t_{ij}^{\text{DR}} - M(1 - x_{ij}^{\text{DR}}) - Mx_{ij}^{\text{TR}} \qquad \forall (i,j) \in \mathcal{A} \tag{22}$$

$$0 \leq f_i \leq e \qquad \forall i \in \mathcal{V}' \tag{23}$$

Final Formulation The formulation for the Drone-Truck VRP is summarized as follows:

$$\text{minimize} \quad a_{V+1} \tag{24}$$

$$\begin{aligned}
\text{subject to}\quad & x_{ij}^{\text{TR}} + x_{ji}^{\text{TR}} + x_{ij}^{\text{DR}} + x_{ji}^{\text{DR}} \geq 1 && \forall (i,j) \in \mathcal{Q} && (25)\\
& \sum_{(1,j)\in\mathcal{A}} x_{1j}^{\text{TR}} = \sum_{(i,V+1)\in\mathcal{A}} x_{i,V+1}^{\text{TR}} = 1 && && (26)\\
& \sum_{(i,j)\in\mathcal{A}} x_{ij}^{\text{TR}} - \sum_{(j,i)\in\mathcal{A}} x_{ji}^{\text{TR}} = 0 && \forall i \in \mathcal{V}_1 && (27)\\
& \sum_{(1,j)\in\mathcal{A}} x_{1j}^{\text{DR}} = \sum_{(i,V+1)\in\mathcal{A}} x_{i,V+1}^{\text{DR}} = 1 && && (28)\\
& \sum_{(i,j)\in\mathcal{A}} x_{ij}^{\text{DR}} - \sum_{(j,i)\in\mathcal{A}} x_{ji}^{\text{DR}} = 0 && \forall i \in \mathcal{V}_1 && (29)\\
& y_i^{\text{TR}} + y_i^{\text{DR}} + y_i^{\text{CB}} = 1 && \forall i \in \mathcal{V}_1 && (30)\\
& x_{ij}^{\text{DR}} + x_{ji}^{\text{DR}} \leq y_i^{\text{CB}} + y_i^{\text{DR}} + y_j^{\text{CB}} + y_j^{\text{DR}} && \forall (i,j) \in \mathcal{A} : i,j \notin \{1, V+1\} && (31)\\
& \sum_{(i,j)\in\mathcal{A}} x_{ij}^{\text{TR}} = y_i^{\text{TR}} + y_i^{\text{CB}} && \forall i \in \mathcal{V}_1 && (32)\\
& \sum_{(i,j)\in\mathcal{A}} x_{ij}^{\text{DR}} = y_i^{\text{DR}} + y_i^{\text{CB}} && \forall i \in \mathcal{V}_1 && (33)\\
& n_j^{\text{TR}} \geq n_i^{\text{TR}} + V x_{ij}^{\text{TR}} - (V-1) && \forall (i,j) \in \mathcal{A} && (34)\\
& n_j^{\text{DR}} \geq n_i^{\text{DR}} + V x_{ij}^{\text{DR}} - (V-1) && \forall (i,j) \in \mathcal{A} && (35)\\
& a_j \geq a_i + t_{ij}^{\text{TR}} - M(1 - x_{ij}^{\text{TR}}) && \forall (i,j) \in \mathcal{A} && (36)\\
& a_j \geq a_i + t_{ij}^{\text{DR}} - M(1 - x_{ij}^{\text{DR}}) - M x_{ij}^{\text{TR}} && \forall (i,j) \in \mathcal{A} && (37)\\
& \sum_{(i,j)\in\mathcal{A}} t_{ij}^{\text{TR}} x_{ij}^{\text{TR}} \leq a_{V+1} && && (38)\\
& \sum_{(i,j)\in\mathcal{A}} t_{ij}^{\text{DR}} x_{ij}^{\text{DR}} \leq a_{V+1} && && (39)\\
& x_{ij}^{\text{DR}} \leq x_{ij}^{\text{TR}} && \forall (i,j) \in \mathcal{A} : t_{ij}^{\text{DR}} > e && (40)\\
& f_j \geq f_i + t_{ij}^{\text{DR}} - M(1 - x_{ij}^{\text{DR}}) - M x_{ij}^{\text{TR}} && \forall (i,j) \in \mathcal{A} && (41)\\
& 0 \leq f_i \leq e && \forall i \in \mathcal{V}' && (42)\\
& x_{ij}^{\text{TR}}, x_{ij}^{\text{DR}} \in \{0,1\} && \forall (i,j) \in \mathcal{A} && (43)\\
& y_i^{\text{TR}}, y_i^{\text{DR}}, y_i^{\text{CB}} \in \{0,1\} && \forall i \in \mathcal{V}_1 && (44)\\
& n_i^{\text{TR}}, n_i^{\text{DR}} \in \mathbb{Z}_+ && \forall i \in \mathcal{V}_1 && (45)\\
& a_i \geq 0 && \forall i \in \mathcal{V}' && (46)
\end{aligned}$$

Again, Constraint (25) is only required for the 2-node formulation but not for the 3-node formulation.

4 Adaptive Large Neighborhood Search

In this section, we develop an Adaptive Large Neighborhood Search algorithm for the Drone-Truck Arc Routing Problem. ALNS was first applied to the pickup and delivery problem with time windows (Ropke and Pisinger, 2006). Laporte et al. (2010) first applied ALNS to solve the arc routing problem to minimize the total cost. ALNS is a well-known popular iterative algorithm

```
Algorithm 1: Pseudocode for ALNS
   Input: G, $\mathcal{R}$, DM, RM, $N_{\max}, Z_{\max}$
   Output: $X_{\text{best}}, Y_{\text{best}}$
 1 Initialize the truck required edges route $X_{r0}$ and the drone required edges routes $Y_{r0}$
     (Sec 4.2);
 2 Initialize destroy methods probability $\mathbf{P}_D^0$ and repair methods probability $\mathbf{P}_R^0$ (Sec 4.5);
 3 $X_{r\text{best}} \leftarrow X_{r\text{current}} \leftarrow X_{r0}$, $Y_{r\text{best}} \leftarrow Y_{r\text{current}} \leftarrow Y_{r0}$;
 4 Encode the required edges route into the complete route
     $X_{\text{best}}, Y_{\text{best}} \leftarrow \text{encode}(X_{r\text{best}}, Y_{r\text{best}})$ (Sec 4.1);
 5 Calculate the makespan of current best solution $t_{\text{best}} \leftarrow f(X_{\text{best}}, Y_{\text{best}})$;
 6 $N \leftarrow 1, Z \leftarrow 0$;
 7 while $N \leq N_{\max}$, $Z \leq Z_{\max}$ do
 8     Select a destroy method $d \in$ DM with probability $P_D^N$;
 9     Select a repair method $r \in$ RM with probability $P_R^N$;
10     Let $X_{\text{rnew}}$ and $Y_{\text{rnew}}$ be the new required edges solution obtained by appling destroy
         $d$ and repair $r$ on $X_{\text{rcurrent}}, Y_{\text{rcurrent}}$;
11     Obtain the complete truck route and drone route $X_{\text{new}}, Y_{\text{new}} \leftarrow \text{encode}(X_{\text{rnew}}, Y_{\text{rnew}})$;
12     if $f(X_{\text{new}}, Y_{\text{new}}) < t_{\text{best}}$ then
13         $X_{\text{best}} \leftarrow X_{\text{new}}$, $Y_{\text{best}} \leftarrow Y_{\text{new}}$, $t_{\text{best}} \leftarrow f(X_{\text{new}}, Y_{\text{new}})$, $X_{\text{rbest}} \leftarrow X_{\text{rnew}}$,
             $Y_{\text{rbest}} \leftarrow Y_{\text{rnew}}$, $Z \leftarrow 0$;
14     else
15         $Z \leftarrow Z + 1$;
16         $v = e^{-(f(X_{\text{new}}, Y_{\text{new}}) - f(X_{\text{current}}, Y_{\text{current}}))/T}$;
17         Generate a random number $\epsilon \in [0, 1]$ ;
18         if $\epsilon < v$ then
19             $X_{\text{rcurrent}} \leftarrow X_{\text{rnew}}$, $Y_{\text{rcurrent}} \leftarrow Y_{\text{rnew}}$;
20     $T \leftarrow hT$;
21     Update $\mathbf{P}_D^N$ and $\mathbf{P}_R^N$ (Sec 4.5);
22     $N \leftarrow N + 1$;
```

to solve various vehicle routing problems. The idea of ALNS is to search in a neighborhood by destroying an incumbent solution and repairing it to construct a new solution in each iteration. The adaptivity is achieved by determining the choices of several destroy and repair methods on their previous successes.

The procedure of the proposed ALNS is shown in Algorithm 1. We let DM and RM denote the set of the destroy and repair methods, respectively. The solution has two decisions: the truck route and the drone route. The key part is how to determine the sequence of traversing the required edges. Thus, it is vital to create X_r and Y_r, which represent the sequence of required arcs traversed by the truck and the drone. In each iteration, the new X_r and Y_r in the neighborhood are produced by applying to destroy and repair. The destroy process removes some edges from the truck-required edges route X_r and drone-required edges route Y_r. Next, the repair process can partially reconstruct the required edges route X_r and Y_r. Then, the complete truck route X and drone route Y are obtained by encoding from X_r and Y_r, described in Section 4.1. The destroy and repair methods are chosen by using the roulette-wheel selection principle based on their probabilities. When the method creates a better solution, the probability of the corresponding method increases, as described in section 4.5. The new solution is accepted by the simulated annealing acceptance criterion. The new solution is accepted if its objective value is better than the current best solution; otherwise, the new solution is accepted by the probability of $\epsilon < \exp\big(-(f(X_{\text{new}}, Y_{\text{new}}) - f(X_{\text{current}}, Y_{\text{current}}))/T\big)$, where ϵ is random number in the interval [0,1]. We let T denote the value of the temperature and gradually decrease at each iteration by a rate $h \in [0, 1]$. The stop criteria are the maximum iterations $N_{\max}$ and non-improving iteration $Z_{\max}$.

4.1 Encoding and Decoding

The feasible solution in ALNS has two decisions: the truck route X and the drone route Y. The route is encoded by the string of arcs that represent the sequence of traversing the arcs. Let X_r and Y_r denote the sequence of required edges traversed by the truck and the drone, respectively. It is noted that the edges in X_r and Y_r do not have a direction.

The decoding rule turns X_r and Y_r into the complete route solution X and Y. The decoding rule is made in two steps: (1) construct the complete truck route X; (2) connect the edges in Y_r to the truck route X.

Step 1: Any two required edges in X_r are connected with the shortest path. Define (v_a, v_{a+1}) is the first edge in Xr. Starting from the depot 1, calculate the shortest distance $\text{dist}(1, v_a)$ and $\text{dist}(1, v_{a+1})$ and choose the shortest path to connect. If $\text{dist}(1, v_a) < \text{dist}(1, v_{a+1})$, add $\{(1, v_a), (v_a, v_{a+1})\}$ in X; otherwise, add $\{(1, v_{a+1}), (v_{a+1}, v_a)\}$. The next unassigned edge (v_b, v_{b+1}) in X_r will connect with the last node v in the current partial X. Choose the shortest path $v \to v_b$ or $v \to v_{b+1}$ to append to the end in X. Complete the truck route until all required edges in X_r are done.

Step 2: Connect drone-required edges Y_r to the truck route X by a greedy rule. First, extract all nodes from the truck route X as $V_x = \{1, v_1, v_2, ...\}$. Initialize the drone truck $Y = \{(1, v_1), (v_1, v_2), ..., (v_i, v_{i+1}), ...\}$. Try to insert a drone edge $\forall(a, b) \in Y_r$ in all possible locations between any two nodes $\forall v_i, v_j \in V_x$. Choose the direction of drone edges with the smallest distance between $\text{dist}(v_i, a) + \text{dist}(v_j, b)$ and $\text{dist}(v_i, b) + \text{dist}(v_j, a)$. Calculate the increased value in the objective value before and after inserting the edge. Choose the location with the least increased objective value to insert the edge (a, b). Let v_i and v_j represent the best insertion location. The section between node v_i and v_j in the current Y is replaced with the drone edge (a, b). Y becomes $\{..., (v_{i-1}, v_i), (a, b), (v_j, v_{j+1}), ...\}$. Next, remove the nodes between node v_i and v_j, and put the vertices a, b in $V_x = \{..., v_i, a, b, v_j, ...\}$. Repeat the above procedure until all required edges in Y_r are done.

Two constraints must be satisfied to construct the drone route:

1. The drone-required edges can not insert between two nodes belonging to an arc in Y_r. For example, the drone-required edge can not insert between a and b.

2. The flight distance must be less than or equal to the maximum drone flight range.

4.2 Initial Solution

The initial X_r and Y_r are created by the Nearest Neighborhood Search. First, the required edges are randomly assigned to the truck set $\mathcal{S}_{xr}$ and the drone set $\mathcal{S}_{yr}$. Next, the required edges route is constructed progressively by adding the nearest edge. The procedure is shown in Algorithm 2.

The distance between any two edges (a, b) and (c, d) shows spatial closeness and is defined as the following equation.

$$\mathrm{dist_e}\Big((a,b),(c,d)\Big) = \frac{1}{4}\Big(\mathrm{dist}(a,c) + \mathrm{dist}(b,d) + \mathrm{dist}(a,d) + \mathrm{dist}(b,c)\Big) \tag{47}$$

where $\mathrm{dist}(i, j)$ is the shortest path distance between node i and j in the graph G.

Algorithm 2: Pseudocode for initialization required edges route

```
  Input: The set of the required edges for the truck or the drone $\mathcal{S}_r$
  Output: The required edges route $r$
1 $r \leftarrow \{(a,b)\}$ with $(a,b) = \arg\min\{\mathrm{dist_e}\big((1,1),(a,b)\big), \forall (a,b) \in \mathcal{S}_r\}$;
2 while $\mathcal{S}_r \neq \emptyset$ do
3     $(c,d) \leftarrow$ the last edge in $r$;
4     Add the nearest edge $(u,v)$ to the end of $r$ with
       $(u,v) = \arg\min\{\mathrm{dist_e}\big((c,d),(u,v)\big), \forall (u,v) \in \mathcal{S}_r\}$;
5     $\mathcal{S}_r \leftarrow \mathcal{S}_r \setminus (u,v)$;
```

4.3 Destroy Methods

Three methods (Random Removal, Worst Route Removal, and Cluster Removal) are applied to destroy a feasible solution.

4.3.1 Random Removal

A certain percentage $q\%$ of edges from the truck and the drone required edges route X_r and Y_r are randomly removed.

4.3.2 Worst Route Removal

Given a solution (X_r, Y_r), the cost for the required edge e is defined as the difference value in the objective function before and after removing edge e from the current solution. It is expressed as

$$\mathrm{cost}(e, X_r, Y_r) = \max\Big\{f\big(\mathrm{encode}(X_r, Y_r)\big) - f_{-e}\big(\mathrm{encode}(X_r, Y_r)\big), 0\Big\} \tag{48}$$

when $f_{-e}(\cdot)$ is the objective function value before and after removing edge e from current solution. Sort all costs for required edge $e \in \mathcal{R}$ in the descending order. Remove the first $\lfloor q\% \times |\mathcal{R}| \rfloor$ edges with the largest costs from X_r and Y_r.

4.3.3 Cluster Removal

The idea of cluster removal is to avoid generating a similar new solution and try to jump into a farther neighborhood to get a solution with the largest change. The relatedness between two edges (u, v) and (i, j) is measured by considering two factors: distance and time. The distance represents the spatial closeness of these two edges. It is defined as the average of the distance of start points and endpoints of two edges (u, v) and (i, j) and calculated as Equation (47).

The time between two edges (u, v) and (i, j) represents temporal closeness and is defined as the average of the arrival times at start points and leave times at endpoints.

$$t\Big((u,v),(i,j)\Big) = \frac{1}{4}\Big(|a_u - a_i| + |a_v - a_j| + |a_u - a_j| + |a_v - a_i|\Big)$$

For any edge $(u, v) \in \mathcal{R}$, the measure of relatedness is defined as the following equations.

$$\begin{aligned} R((u,v),(i,j)) = {} & w_1 \frac{\mathrm{dist_e}((u,v),(i,j))}{\max\{\mathrm{dist_e}((u,v),(k,l)), \forall (k,l) \in \mathcal{R}\}} \\ & + w_2 \frac{t((u,v),(i,j))}{\max\{t((u,v),(k,l)), \forall (k,l) \in \mathcal{R}\} - \min\{t((u,v),(k,l)), \forall (k,l) \in \mathcal{R}\}} \end{aligned} \tag{49}$$

where w_1 and w_2 are weights with the sum of 1.

The smaller $R((u, v),(i, j))$ is, the more related the two edges are. Following steps are followed: randomly select a required edge $(u, v) \in \mathcal{R}$ and calculate $R((u, v),(i, j)), \forall (i, j) \neq (u, v) \in \mathcal{R}$. Sort all $R((u, v),(i, j))$ in descending order. Remove edges with first $\lfloor q\% \times |\mathcal{R}| \rfloor$ in the sequence.

4.4 Repair Methods

There are three repair methods to reconstruct the partial truck and drone required edges route X_r and Y_r. The repair methods are Random Insertion, Greedy Insertion, and Regret Insertion.

4.4.1 Random Insertion

Given partial truck required edges route X_r and drone required edges route Y_r, randomly insert the undecided required edges into them.

4.4.2 Greedy Insertion

The idea of the greedy insertion heuristic is to find the best insertion. A concept of Insertion Cost $I(e, p, X_r, Y_r)$ is introduced to denote the change in the objective value when inserting the edge e into Xr or Y_r at position p. It is expressed as

$$I(e, p, X_r, Y_r) = \Delta f(e, p, X_r, Y_r). \tag{50}$$

Select the position p to insert e with the least insertion cost $I(s, p, X_r, Y_r)$. Repeat the above steps until all required edges are satisfied.

4.4.3 Regret Insertion

The regret insertion is improved by incorporating look-ahead information when selecting the required edge to insert.

For any required edge $e \in \mathcal{R}$, the regret-k cost is defined as

$$R(e, X_r, Y_r) = \sum_{j=1}^{k} \{\Delta f_j(e, X_r, Y_r) - \Delta f_1(e, X_r, Y_r)\}, \tag{51}$$

where $\Delta f(e, X_r, Y_r)$ is the increased value in the objective value after inserting edge e. Sort $\Delta f(e, X_r, Y_r)$ for all possible insertion positions in the increasing order. $\Delta f_k(e, X_r, Y_r)$ means the increased value in the objective for the k-th best insertion position. The best insertion position has the least $\Delta f_1(e, X_r, Y_r)$. The regret insertion is the reconstruction heuristic that chooses to insert the required edge with the maximum $R(e, X_r, Y_r)$ and insert this edge into the position with the least insertion cost. Repeat the above steps until all required edges are inserted.

4.5 Adaptive Probability Update

The adaptivity of ALNS is achieved by selecting the destroy and repair methods based on their previous successes. In each iteration, the methods are chosen by the roulette wheel selection principle based on their probabilities.

The weights are introduced to track the score to measure how well the methods have performed. If the weight is larger, that means the destroy and repair methods create more accepted solutions in previous iterations. The method is chosen by higher probability based on a larger weight. The initial weights are equal to 1. In iteration i, the destroy method d and the repair method r are used. The weight is updated by adding a score of the amount $\sigma_1, \sigma_2, \sigma_3$, or 0. If the methods create a new global best solution, the weight $w_d^{i+1} \leftarrow w_d^i + \sigma_1$ and $w_r^{i+1} \leftarrow w_r^i + \sigma_1$; if the new solution is accepted with a better objective value than the current solution but not the global best one, $w_d^{i+1} \leftarrow w_d^i + \sigma_2$ and $w_r^{i+1} \leftarrow w_r^i + \sigma_2$; if the new solution is accepted with a worse objective value than the current solution, $w_d^{i+1} \leftarrow w_d^i + \sigma_3$ and $w_r^{i+1} \leftarrow w_r^i + \sigma_3$; otherwise, $w_d^{i+1} \leftarrow w_d^i$ and $w_r^{i+1} \leftarrow w_r^i$.

Then, the probability vectors of destroy and repair methods in iteration i are updated as follows:

$$\mathbf{P}_D^i = \left(\frac{w_d^i}{\sum_{d' \in \mathsf{DM}} w_{d'}^i} : d \in \mathsf{DM} \right) \tag{52}$$

$$\mathbf{P}_R^i = \left(\frac{w_r^i}{\sum_{r' \in \mathsf{RM}} w_{r'}^i} : r \in \mathsf{RM} \right). \tag{53}$$

5 Numerical Experiments

The experiments are conducted on the workstation with a 2.2GHz Intel Xeon Processor and 32GB RAM. The MIP formulations in Section 3.2 is solved in Julia 1.6.1 by calling Gurobi v0.9.12. The proposed ALNS is run in Julia v1.6.1. The experiments are implemented on randomly generated data and a set of undirected rural postman problem instances.

The performance of ALNS relies on parameters whose values are tested individually. The values of parameters are tuned as follows. To choose a parameter and test it with the different values from the chosen set. Meanwhile, the other parameters remain unchanged. The parameters in ALNS are calibrated as follows: the maximum iteration $N_{\max}$ is 3000; maximum non-improving iteration $Z_{\max}$ is 300; the initial temperature T_0 is 100; temperature annealing rate h is 0.95; the amounts added to the weights $(\sigma_1, \sigma_2, \sigma_3) = (0.5, 0.3, 0.2)$; the percentage of removal $q\%$ is 30%.

Benchmark We use the multi-start tabu search (MSTS) algorithm of Luo et al. (2021), developed for the multi-visit TSP-D problem, as a benchmark. After the 3-node reformulation, we can convert the drone-truck arc routing problem into a multi-visit TSP-D problem, which can be solved by MSTS. We created our own implementation of MSTS in Julia v1.6.1 to compare the performances.

5.1 One Truck and One Drone

5.1.1 Small-Scale Instances

The small-scale data are randomly generated when the number of nodes $|\mathcal{N}| = 10$ or 15, the number of edges $|\mathcal{E}| = 20$ or 30, the number of required edges $|\mathcal{R}| = 5, 7, 10$. The vertices are randomly distributed in a 100×100 square region. The required edges are randomly chosen from all edges. Each type of randomly generated data has 25 instances. Define the maximum flight time

$$e = \frac{\beta}{v^{\mathrm{DR}}} \frac{\sum_{(i,j)\in\mathcal{E}^{\mathrm{DR}}} d^{\mathrm{DR}}(i,j)}{|\mathcal{E}^{\mathrm{DR}}|}.$$

The parameter β is set as 1,2,3, which determines the flight range. The speed of the truck v^{TR} and the speed of the drone v^{DR} are selected as equal (1,1), slower (1,2), and faster (2,1).

We use two transformation rules to convert ARP into $2 \times |\mathcal{R}| + 1$ and $3 \times |\mathcal{R}| + 1$ VRP. The Single-Drone-Single-Truck results over the randomly generated data when $N = 10$ are shown in Table 3.

Column Obj means the average objective values solved by MIP-3 (Pearn et al., 1987), MIP-2 (Longo et al., 2006), MSTS (Luo et al., 2021) and ALNS. The optimal solution can be obtained by MIP-3 and MIP-2 solved via Gurobi when the number of required edges $|\mathcal{R}|$ is 5 or 7. When $|\mathcal{R}|$ is 10, the instance becomes very large and they cannot be solved to optimality in the limited run time of 3600s. To compare between solutions obtained by MIP formulation and two algorithms, Gap% in terms of objective values is used and defined as follows:

$$\text{Gap\%} = \frac{\text{Objective of Algorithm} - \min\{\text{Objective of MIP-3, Objective of MIP-2}\}}{\min\{\text{Objective of MIP-3, Objective of MIP-2}\}} \times 100 \quad (54)$$

where Algorithm is MSTS or ALNS. Average Gap% is calculated as the average value of Gap% for each type instance with 9 combinations of v^{TR}, v^{DR}, and β.

Compared to the objective values of MIP-3 and MIP2, MSTS and ALNS can not perform better with the gap of 3.81% and 1.73% in the small instances $|\mathcal{R}| = 5$; the gaps are 4.65% and 2.07% when $|\mathcal{R}| = 7$; and the gaps are 4.87% and 2.62% when $|\mathcal{R}| = 10$. ALNS outperforms MSTS in the objective values because of the smaller gap to the optimal solutions. The computational times reveal that MIP-2 runs the fastest when the instances are small and the advantage of the heuristic in run time becomes more evident when the instance size becomes larger. When

Table 3: Single-Drone-Single-Truck Results on Small-Scale Instances with $N = 10$

Instance				Obj						CPU (seconds)			
	v^{TR}	v^{DR}	β	MIP-3[b]	MIP-2[c]	MSTS[d]	Gap%	ALNS	Gap%	MIP-3[b]	MIP-2[c]	MSTS[d]	ALNS
N10E20R5	1	1	1	359.30	359.30	367.23	2.21	363.23	1.09	4.85	**2.96**	11.24	3.64
			2	289.67	289.67	299.32	3.33	290.62	0.33	8.25	**4.90**	10.63	5.72
			3	252.07	252.07	258.62	2.60	254.34	0.90	4.79	**1.28**	9.42	6.32
	1	2	1	358.22	358.22	371.99	3.84	365.32	1.98	4.58	**0.96**	4.63	11.23
			2	256.43	256.43	268.32	4.64	263.23	2.65	8.59	**2.73**	10.42	7.32
			3	196.29	196.29	204.23	4.05	200.42	2.11	9.12	**4.36**	9.98	5.43
	2	1	1	184.26	184.26	191.32	3.83	188.42	2.26	2.81	**1.59**	12.42	7.43
			2	165.14	165.14	173.32	4.96	168.23	1.87	2.22	**2.17**	9.43	5.73
			3	158.54	158.54	166.21	4.84	162.34	2.40	3.23	**1.86**	8.99	6.43
Average				246.66	246.66	255.62	3.81	250.68	**1.73**	5.38	**2.53**	10.42	5.85
N10E20R7	1	1	1	462.89	462.89	483.53	4.46	469.32	1.39	37.42	7.92	**5.35**	16.32
			2	353.17	353.17	372.73	5.54	363.24	2.85	230.80	75.74	33.42	**17.42**
			3	314.13	314.13	326.42	3.91	319.40	1.68	133.66	42.36	79.54	**20.32**
	1	2	1	461.53	461.53	477.32	3.42	470.34	1.91	59.58	**12.92**	43.43	16.23
			2	319.89	319.89	335.43	4.86	326.43	2.04	497.73	126.24	63.43	**34.23**
			3	237.08	237.08	250.32	5.58	242.53	2.30	317.73	180.17	74.52	**43.42**
	2	1	1	235.27	235.27	245.32	4.27	239.43	1.77	13.81	**5.16**	12.43	10.32
			2	200.25	200.25	210.45	5.09	204.32	2.03	15.70	**5.16**	10.53	14.23
			3	193.24	193.24	202.34	4.71	198.43	2.69	18.65	**4.39**	7.34	15.32
Average				308.61	308.61	322.65	4.65	314.83	**2.07**	147.23	51.12	40.00	**20.87**
N10E20R10	1	1	1	594.61[a]	594.61[a]	607.34	2.14	603.24	1.45	1443.86	1066.15	295.34	**125.34**
			2	439.66[a]	434.67[a]	466.34	7.29	452.32	4.06	3593.46	3144.67	222.75	**220.32**
			3	393.13[a]	391.29[a]	436.23	11.48	412.32	5.37	2954.81	2049.52	252.39	**198.52**
	1	2	1	591.06[a]	591.06[a]	603.30	2.07	599.42	1.41	1211.51	938.79	**215.34**	284.55
			2	426.10[a]	394.15[a]	405.63	2.91	400.23	1.54	3600.00	3600.00	**199.23**	204.32
			3	296.07[a]	284.21[a]	295.32	3.91	289.53	1.87	3504.08	2798.71	**189.43**	221.23
	2	1	1	302.16[a]	301.05[a]	308.27	2.40	304.23	1.06	1546.43	918.86	285.35	**198.43**
			2	258.35[a]	256.34[a]	270.43	5.50	263.23	2.69	1795.22	1008.30	263.23	**174.23**
			3	244.98[a]	244.32[a]	259.32	6.14	254.34	4.10	1424.13	910.31	300.11	**199.99**
Average				394.01	387.97	405.80	4.87	397.65	**2.62**	2343.81	1827.58	247.02	**202.99**

[a] Not all 25 instances can be solved to optimality within the limited computational time of 3600s and the objective values are not optimal.
[b] MIP formulation based on the 3-node transformation (Pearn et al., 1987)
[c] MIP formulation based on the 2-node transformation (Longo et al., 2006)
[d] MSTS (Luo et al., 2021) applied after the 3-node transformation

$|\mathcal{R}| = 10$, the average run time of MSTS (247.02s) and ALNS (202.09s) are much less than those of MIP. ALNS also outperforms MSTS in the run time.

Table 4: Single-Drone-Single-Truck Results on Small-Scale Instances with $\mathcal{N} = 15$

Instance				Obj						CPU (seconds)			
	v^{TR}	v^{DR}	β	MIP-3[b]	MIP-2[c]	MSTS[d]	Gap%	ALNS	Gap%	MIP-3[b]	MIP-2[c]	MSTS[d]	ALNS
N15E30R5	1	1	1	446.71	446.71	454.23	1.68	450.43	0.83	3.44	1.50	12.43	6.47
			2	362.33	362.33	371.01	2.40	367.63	1.46	11.74	7.22	10.42	10.34
			3	303.35	303.35	314.42	3.65	309.34	1.98	5.32	4.00	9.45	5.63
	1	2	1	446.42	446.42	457.52	2.49	452.32	1.32	3.50	1.89	7.64	4.63
			2	340.98	340.98	348.32	2.15	344.52	1.04	16.63	10.29	8.63	7.83
			3	253.60	253.60	265.32	4.62	259.32	2.25	12.51	7.51	11.42	6.58
	2	1	1	225.79	225.79	233.43	3.39	228.32	1.12	2.77	1.47	10.24	3.21
			2	205.07	205.07	211.89	3.32	207.53	1.20	4.28	3.19	14.25	7.42
			3	190.00	190.00	200.21	5.37	193.24	1.70	2.91	2.22	11.42	5.43
Average				308.25	308.25	317.37	3.23	312.52	**1.43**	7.01	**4.37**	10.66	6.39
N15E30R7	1	1	1	537.89	537.89	553.42	2.89	549.75	2.21	45.05	9.11	45.33	18.32
			2	424.33[a]	424.33	446.34	5.19	430.23	1.39	637.16	101.28	55.11	22.62
			3	352.69	352.69	375.43	6.45	362.34	2.74	126.60	24.21	34.63	35.23
	1	2	1	533.58	533.58	563.42	5.59	547.63	2.63	53.13	6.95	45.34	19.43
			2	399.35[a]	399.35	420.53	5.30	410.23	2.72	963.13	221.13	39.64	35.64
			3	282.83	282.83	297.34	5.13	289.53	2.37	741.21	125.79	26.71	55.47
	2	1	1	271.76	271.76	290.34	6.84	278.43	2.45	22.17	4.20	36.23	16.43
			2	239.13	239.13	250.43	4.73	244.42	2.21	72.57	16.91	53.53	20.43
			3	225.23	225.23	235.43	4.53	231.52	2.79	57.28	13.49	49.35	19.64
Average				362.98	362.98	381.41	5.18	371.56	**2.39**	302.03	58.12	42.87	**27.02**
N15E30R10	1	1	1	665.25[a]	665.25[a]	683.24	2.70	680.34	2.27	1454.16	1203.22	320.43	290.31
			2	502.19[a]	498.74[a]	520.34	4.33	511.42	2.54	3600.00	3371.43	295.35	300.23
			3	435.41[a]	429.91[a]	450.32	4.75	440.23	2.40	2685.58	2422.05	340.52	243.52
	1	2	1	660.18[a]	660.18[a]	678.42	2.76	675.73	2.35	1305.83	1300.98	299.43	210.24
			2	481.46[a]	481.08[a]	515.62	5.00	510.23	3.90	3555.47	3600.00	287.77	199.53
			3	341.40[a]	337.74[a]	370.53	5.34	359.34	2.16	3402.47	3412.68	310.45	178.46
	2	1	1	335.85[a]	332.41[a]	359.64	6.91	348.53	3.60	1514.19	1341.94	296.34	176.34
			2	278.89[a]	278.72[a]	286.43	2.77	285.47	2.42	1454.84	1520.01	301.53	193.32
			3	262.62[a]	260.58[a]	270.31	3.73	268.34	2.98	1228.25	1015.98	296.43	200.52
Average				440.36	441.40	459.43	4.25	453.29	**2.74**	2263.24	2132.33	305.36	**221.39**

[a] Not all 25 instances can be solved optimally within the limited computational time of 3600s, and the objective values are not optimal.
[b] MIP formulation is based on the VRP transformed from Pearn et al. (1987)
[c] MIP formulation is based on the VRP transformed from Longo et al. (2006)
[d] MSTS (Luo et al., 2021) applied after the 3-node transformation

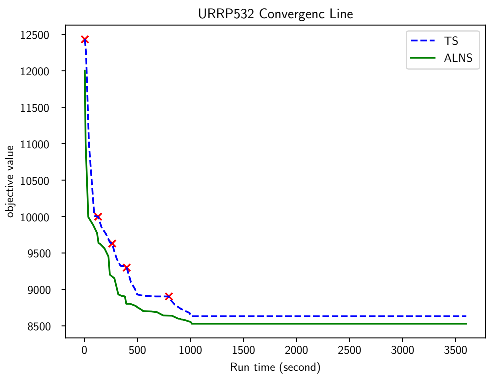
The Single-Drone-Single-Truck results over the randomly generated data when $N = 15$ are shown in Table 4. MIP-3 and MIP-2 solve the small instances well. The average objective value gaps of ALNS 1.43%, 2.39%, and 2.74% show that ALNS can get an acceptable solution within up to 10.34s, 55.47s, and 300.23s for $|\mathcal{R}| = 5, 7, 10$, respectively. Between these two heuristics, ALNS also runs faster and gets better solution than MSTS.

5.1.2 Large-Scale Instances

As shown in Tables 3 and 4, Gurobi cannot solve MIP-3 and MIP-2 to optimality in 3600s and formulation can not get a feasible solution when the network is large. The two heuristic methods, MSTS and ALNS, are tested on a set of Undirected Rural Postman Problem instances (Corberán et al., 2021b). The characteristic of the instances UR500 is shown in Table 5.

Table 5: Characteristic of Undirected Rural Postman Problem UR500

	Average	Min	Max
Nodes	446.0	298	499
Edges	1128.9	597	1526
Required Edges	35.3	1	99



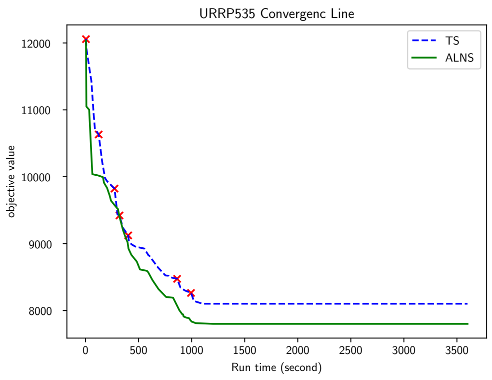


Figure 5: Convergence Lines (The red cross means that the next solution is reconstructed based on a new start in MSTS(Luo et al., 2021))

The comparison between MSTS and ALNS is evaluated in terms of objective value and run time. Gap% in objective value and run-time are defined as follows:

$$\text{Gap\% in Objective Value} = \frac{\text{Objective of MSTS} - \text{Objective of ALNS}}{\text{Objective of ALNS}} \times 100 \tag{55}$$

$$\text{Gap\% in Run Time} = \frac{\text{Run time of MSTS} - \text{Run time of ALNS}}{\text{Run time of ALNS}} \times 100 \tag{56}$$

The average Gap% in objective value and run-time are calculated as the average value of Gap% in objectives and run times for each type instance with 9 combinations of v^{TR}, v^{DR}, and β.

The results of four URPP500 instances are summarized in Table 6. The run time is limited to 1200 seconds for both MSTS and ALNS. ALNS can get a better solution for all instances with an average gap of 1.99%, 4.16%, 1.36%, and 3.54%.

Moreover, convergence lines for URPP532 and URPP535 are illustrated in Figure 5. Tabu search tries to jump into the further neighborhood by generating a new solution based on a new start solution. Because ALNS adopts a further-neighborhood search rule such as worst route removal, it can search for a solution in the further neighborhood, and the search rule of ALNS is more effective. Therefore, ALNS converges faster than MSTS.

5.2 One Truck and Multiple Drones

5.2.1 Small-Scale Instances

Both ALNS and MSTS can solve the case when one truck and multiple drones traverse all required edges jointly. The tests are done on the randomly generated data with $N = 10$ and 15.

Table 6: Single-Drone-Single-Truck Objective Values on Large-Scale Instances

			MSTS	ALNS	Gap%	MSTS	ALNS	Gap%
v^{TR}	v^{DR}	β		UR532			UR535	
1	1	1	10342	10034	3.07	12042	11592	3.88
		2	10225	9987	2.38	11942	11561	3.30
		3	9998	9698	3.09	10093	9899	1.96
1	2	1	8843	8733	1.26	8234	7953	3.53
		2	8632	8529	1.21	8102	7801	3.86
		3	8452	8321	1.57	7842	7504	4.50
2	1	1	7201	7033	2.40	7293	6903	5.65
		2	6992	6843	2.18	7102	6723	5.64
		3	6703	6653	0.75	6983	6643	5.12
	Average		8599	8426	**1.99**	8848	8301	**4.16**
v^{TR}	v^{DR}	β		UR537			UR542	
1	1	1	11023	10932	0.83	11423	11242	1.61
		2	10294	10200	0.92	11232	10923	2.83
		3	10125	10101	0.24	11001	10842	1.47
1	2	1	10023	9994	0.29	10532	10424	1.04
		2	9923	9530	4.12	10423	10211	2.08
		3	9380	9305	0.81	10232	10112	1.19
2	1	1	8990	8942	0.54	10032	9123	9.96
		2	9123	8816	3.48	9834	8942	9.98
		3	8824	8736	1.01	8988	8834	1.74
	Average		9745	9617	**1.36**	10411	10073	**3.54**

Table 7: Two-Drones-One-Truck Results on Small-Scale Instances with $N = 10$

Instance				Obj			CPU (seconds)		
	v^{TR}	v^{DR}	β	MSTS	ALNS	Gap%	MSTS	ALNS	Gap%
N10E20R5	1	1	1	271.26	263.23	3.05	9.87	3.55	178.03
			2	222.32	216.62	2.63	12.01	6.07	97.86
			3	198.62	192.34	3.27	7.4	4.65	59.14
	1	2	1	280.5	275.32	1.88	13.1	4.47	193.06
			2	201.62	195.23	3.27	11.55	6.11	89.03
			3	142.67	138.42	3.07	10.83	5.07	113.61
	2	1	1	140.42	136.42	2.93	12.21	9.36	30.45
			2	114.78	110.54	3.84	8.37	5.12	63.48
			3	105.21	101.32	3.84	9.44	5.11	84.74
Average				186.38	181.05	**3.09**	10.53	5.50	**101.04**
N10E20R7	1	1	1	418.34	409.56	2.14	33.38	17.69	88.69
			2	311.91	308.22	1.20	37.28	14.08	164.77
			3	234.7	229.68	2.19	80.68	15.62	416.52
	1	2	1	381.38	373.75	2.04	45.02	22.11	103.62
			2	259.99	253.81	2.43	64.27	39.26	63.70
			3	170.71	163.95	4.12	75.80	48.2	57.26
	2	1	1	152.8	148.61	2.82	36.51	13.5	170.44
			2	162.5	154.3	5.31	48.65	13.99	247.75
			3	126.74	121.64	4.19	28.56	20.5	39.32
Average				246.56	240.39	**2.94**	50.02	22.77	**150.23**
N10E20R10	1	1	1	517.34	508.74	1.69	311.24	109.74	183.62
			2	488.42	471.12	3.67	235.85	222.72	5.90
			3	436.23	422.74	3.19	237.79	187.22	27.01
	1	2	1	545.42	536.02	1.75	230.34	207.35	11.09
			2	414.23	404.23	2.47	281.53	186.42	51.02
			3	335.32	324.34	3.39	286.63	234.13	22.42
	2	1	1	214.23	205.03	4.49	265.55	183.73	44.53
			2	194.22	182.13	6.64	253.03	178.93	41.41
			3	188.43	178.94	5.30	295.01	196.39	50.22
Average				370.43	359.25	**3.62**	266.33	189.63	**48.58**

Table 8: Two-Drones-One-Truck Results on Small-Scale Instances with $N = 15$

Instance				Obj			CPU (seconds)		
	v^{TR}	v^{DR}	β	MSTS	ALNS	Gap%	MSTS	ALNS	Gap%
N15E30R5	1	1	1	327.83	312.33	4.96	14.93	7.97	87.33
			2	301.23	299.93	0.43	18.12	8.74	107.32
			3	284.32	269.74	5.41	11.85	8.53	38.92
	1	2	1	379.82	351.82	7.96	7.94	4.23	87.71
			2	309.72	299.22	3.51	11.63	6.73	72.81
			3	258.42	254.23	1.65	14.32	6.28	128.03
	2	1	1	248.83	239.32	3.97	12.24	4.61	165.51
			2	201.49	195.93	2.84	15.25	7.82	95.01
			3	189.81	185.14	2.52	11.32	4.33	161.43
Average				277.94	267.52	**3.69**	13.07	6.58	**104.90**
N15E30R7	1	1	1	418.42	402.65	3.92	79.43	16.72	375.06
			2	392.42	385.53	1.79	46.51	24.82	87.39
			3	379.34	375.23	1.10	66.23	34.73	90.70
	1	2	1	413.92	396.23	4.46	55.24	18.93	191.81
			2	394.23	374.11	5.38	50.64	32.94	53.73
			3	372.32	345.34	7.81	65.31	53.67	21.69
	2	1	1	179.24	163.23	9.81	27.23	17.33	57.13
			2	139.53	130.92	6.58	87.63	23.33	275.61
			3	130.23	126.72	2.77	25.75	19.14	34.54
Average				313.29	299.99	**4.85**	56.00	26.85	**131.96**
N15E30R10	1	1	1	524.64	515.44	1.78	335.53	308.53	8.75
			2	396.64	381.52	3.96	314.25	224.12	40.22
			3	343.23	333.13	3.03	323.62	226.12	43.12
	1	2	1	583.62	574.23	1.64	289.13	203.80	41.87
			2	395.23	383.43	3.08	296.37	170.21	74.12
			3	240.32	227.54	5.62	337.65	178.37	89.29
	2	1	1	218.34	213.33	2.35	315.64	225.35	40.07
			2	193.23	182.77	5.72	316.33	236.30	33.87
			3	159.32	145.34	9.62	310.43	222.19	39.71
Average				339.40	328.53	**4.09**	315.44	221.66	**45.67**

The results are summarized in Tables 7 and 8. ALNS gets better objective values with an average gap of 3.09%, 2.94%, and 3.62%. The average run time gaps are 101.04%, 150.23%, and 48.58% when N=10. When N = 15, the average objective gaps are 3.69%, 4.85%, and 4.09%. The average run time gaps are 104.90%, 131.96%, and 45.67%. Since both ALNS and MSTS are limited in running time of 1200s, the gap between them shows that ALNS performs better than MSTS for all randomly generated small-scale data.

5.2.2 Large-Scale Instances

MSTS and ALNS methods also solve four URPP500 instances of large scale in the case when there are two drones and one truck. Both methods use the stop criterion as the maximum iteration number of 5000 and a non-improving iteration number of 1000. The objective values are shown in Table 9. Average gaps are 3.09%, 2.83%, 2.65% and 4.60% for instances UR532, UR535, UR537 and UR542, respectively. ALNS enables us to get a better solution than MSTS in the case of multiple drones and one truck.

Table 9: Two-Drones-One-Truck Objective Values on Large-Scale Instances

			MSTS	ALNS	Gap%	MSTS	ALNS	Gap%
v^{TR}	v^{DR}	β	UR532			UR535		
1	1	1	6894.67	6689.33	3.07	7528.00	7461.33	0.89
		2	6816.67	6658.00	2.38	7961.33	7728.37	3.01
		3	6665.33	6465.33	3.09	6728.67	6599.33	1.96
1	2	1	5171.33	5017.20	3.07	4760.03	4746.15	0.29
		2	5006.67	4889.00	2.41	4471.13	4355.67	2.65
		3	5034.67	4854.33	3.71	4567.00	4500.67	1.47
2	1	1	5712.67	5507.33	3.73	5793.53	5478.19	5.76
		2	5476.33	5265.00	4.01	5384.67	5092.77	5.73
		3	5059.67	4945.47	2.31	5561.65	5365.67	3.65
Average			5759.78	5587.89	**3.09**	5861.78	5703.13	**2.83**
v^{TR}	v^{DR}	β	UR537			UR542		
1	1	1	7348.67	7188.00	2.24	7615.33	7494.67	1.61
		2	6862.67	6789.32	1.08	7488.23	7282.76	2.82
		3	6850.00	6701.22	2.22	7334.16	7228.12	1.47
1	2	1	6315.00	6009.67	5.08	6294.33	6001.11	4.89
		2	6012.15	5836.63	3.01	6298.67	5938.76	6.06
		3	5601.37	5545.32	1.01	6088.33	5757.14	5.75
2	1	1	6725.98	6676.65	0.74	7517.00	6866.34	9.48
		2	6700.01	6668.34	0.47	7138.00	6728.34	6.09
		3	6854.67	6347.10	8.00	6912.65	6697.33	3.21
Average			6585.61	6418.03	**2.65**	6965.19	6666.06	**4.60**

5.3 Analysis on Speed and Drone Range

The numbers of instances solved to optimality are given in Table 10. There are 25 instances of each type. All instances are solved within the limited computational time of 3600 seconds.

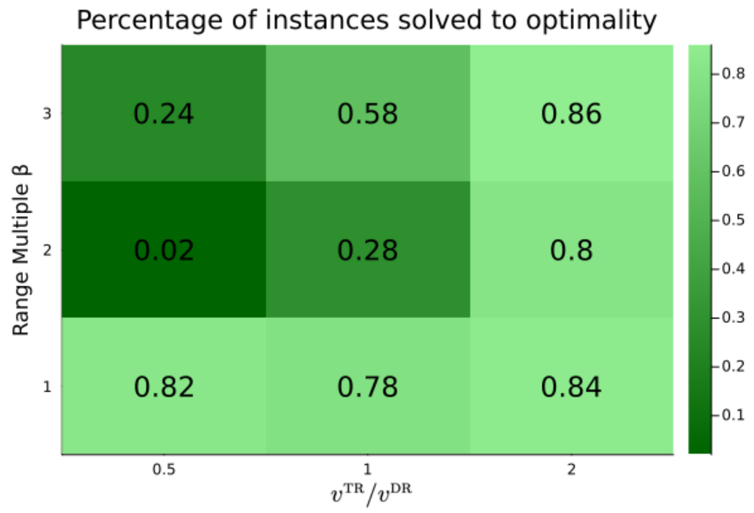


Figure 6: Percentage of Small-Scale Instances with $|\mathcal{R}| = 10$ Solved to Optimality

When $|\mathcal{R}| = 5$, all 25 instances of each type can be solved optimally. When $|\mathcal{R}| = 7$, MIP-3 and MIP-2 can not obtain the optimal solution in some cases when the truck speed $v^{\mathrm{TR}} = 1$, the drone speed $v^{\mathrm{DR}} = 2$, range parameter $\beta = 2$ or 3. As the increase in the number of required edges $|\mathcal{R}| = 10$, the property of the problem becomes obvious. The problem is simpler to solve when the truck is faster than the drone ($v^{\mathrm{TR}} = 2, v^{\mathrm{DR}} = 1$) because the truck tends to service the most edges, and the drone gets onboard the truck in the most time. Then, the problem is harder to solve when the speeds are the same ($v^{\mathrm{TR}} = v^{\mathrm{DR}} = 1$) because it leads to the situation that it is equivalent to a 2-truck ARP with one truck having a length constraint. It is the hardest to solve when the drone is faster than the truck ($v^{\mathrm{TR}} = 1, v^{\mathrm{DR}} = 2$). Because the faster drone can traverse more required edges and benefits in reducing the completion time.

The different maximum drone ranges also affect the complexity of the problem. For different values of $\beta = 1, 2, 3$, the average percentages of instances solved to optimality are 72.67%, 25.33%, 37.33% by MIP-3 and are 82.00%, 36.67% and 56.00% by MIP-2, respectively. The maximum flight range is short when $\beta = 1$, so the drone cannot traverse some edges. As the maximum range increases to $\beta = 2$, there are more feasible solutions where the drone can service some edges. If the maximum range becomes very large, the problem is equivalent to a 2-truck ARP with the trucks having different speeds. That becomes easier to solve.

The percentages of optimal solutions over the randomly generated small-scale instances with 10 required edges are drawn in Figure 6. Figure 7 shows the average run times of ALNS over large-scale instances UR500. The darker the color is, the harder the problem is to solve. Thus, the worst case happens when $v^{\mathrm{TR}}/v^{\mathrm{DR}} = 0.5$ and the range parameter $\beta = 2$. Because the minimum makespan occurs in the situation where there is no or less waiting time at the drone's landing combined node.

5.4 Analysis on Robustness of ALNS versus MSTS

The robustness of a heuristic means how much the solutions vary if being repeated several times in the same instance. The robustness of a method is expressed as the standard deviation. The randomly generated instances N15E30-R5, R7, and R10 are used to evaluate the robustness.

Table 10: Number of Small-Scale Instances Solved to Optimality

				# opt			# opt	
v^{TR}	v^{DR}	β	Instance	MIP-3	MIP-2	Instance	MIP-3	MIP-2
1	1	1	N10E20R5	25/25	25/25	N15E30R5	25/25	25/25
		2		25/25	25/25		25/25	25/25
		3		25/25	25/25		25/25	25/25
1	2	1		25/25	25/25		25/25	25/25
		2		25/25	25/25		25/25	25/25
		3		25/25	25/25		25/25	25/25
2	1	1		25/25	25/25		25/25	25/25
		2		25/25	25/25		25/25	25/25
		3		25/25	25/25		25/25	25/25
1	1	1	N10E20R7	25/25	25/25	N15E30R7	25/25	25/25
		2		25/25	25/25		24/25	25/25
		3		25/25	25/25		25/25	25/25
1	2	1		25/25	25/25		25/25	25/25
		2		24/25	25/25		22/25	25/25
		3		24/25	24/25		25/25	25/25
2	1	1		25/25	25/25		25/25	25/25
		2		25/25	25/25		25/25	25/25
		3		25/25	25/25		25/25	25/25
1	1	1	N10E20R10	18/25	20/25	N15E30R10	18/25	19/25
		2		1/25	6/25		3/25	8/25
		3		6/25	15/25		10/25	14/25
1	2	1		19/25	21/25		20/25	20/25
		2		0/25	0/25		1/25	1/25
		3		2/25	9/25		2/25	3/25
2	1	1		17/25	22/25		17/25	21/25
		2		16/25	20/25		17/25	20/25
		3		17/25	21/25		19/25	22/25

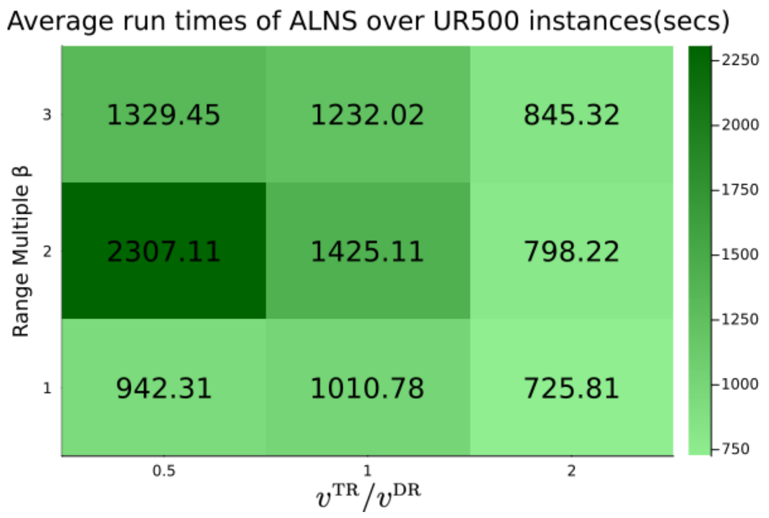


Figure 7: Average Run Time of ALNS over Large-Scale Instances UR500

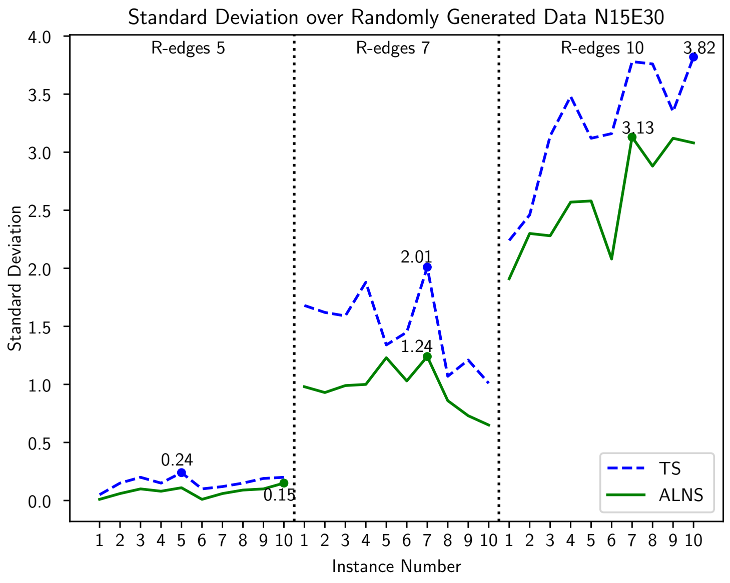


Figure 8: Standard Deviation of Makespan by ALNS and MSTS over Small-Scale Instances

Each instance is repeated 10 times, and the standard deviation is calculated from the 10 repeated solutions. The standard deviations of ALNS and MSTS are illustrated in Figure 8. For all instances, ALNS has less standard deviation than MSTS. With the increasing number of required edges, the values of Std increase. The less the standard deviation is, the more stable the method is. We observe that ALNS has better robustness than MSTS.

6 Concluding remarks

This paper considers the Drone-Truck Arc Routing Problem (DT-ARP). The drone and the truck cooperatively serve all required edges at least once. Since the drone can fly off of the road network, the DT-ARP extends the traditional ARP. With a limited battery capacity, the drone needs to fly from and to vehicles for a replacement of the battery. The key challenge is how to determine the truck and drone route to minimize the completion time. In order to obtain optimal solutions, we transform arc routing problems into node routing problems with two transformation methods and formulate mixed-integer programming models. The experiments reveal that MIP formulations can solve the problem well for the small-size network. However, for larger networks with more than 10 required edges, an efficient and effective heuristic is necessary. A heuristic method based on Adaptive Large Neighborhood Search is proposed to solve the DT-ARP. The performance of ALNS is evaluated using small-size randomly generated ARP instances and large-size undirected rural postman problem instances.

We can also use an existing method for the Traveling Salesman Problem with Drones (TSP-D) to solve DT-ARP. After applying a transformation, we can obtain an equivalent TSP-D with multi-visits, which may be solved by a multi-start tabu-search (MSTS). We found that ALNS outperforms MSTS both in the solution quality and the computational time.

Further analysis of the truck/drone speed and maximum drone flight range shows that the problem is hard to solve when the maximum flight range is double the average edges' distances, and the drone is twice as fast as the truck. The robustness of ALNS is also better than Tabu Search by comparing the standard deviation from the repeated solved solutions.

As for directions of future research, the heuristic method, Adaptive Large Neighborhood Search, may be improved, such as by using strong initialization or by some other destroy and repair methods. The future work could extend to the DT-ARP with multiple trucks and multiple drones onboard per truck.

Acknowledgment The first author's research was supported by the start-up fund from Zhejiang Gongshang University (Project No.1310XJ2323030). The second author's research was supported by the U.S. National Science Foundation (NSF) grant with Award No. 2032460. The third author's research was supported by the Korean government (MSIT) through the National Research Foundation of Korea (NRF) grant RS-2023-00259550.

References

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

Ahirwar, S., R. Swarnkar, S. Bhukya, G. Namwade. 2019. Application of drone in agriculture. *International Journal of Current Microbiology and Applied Sciences* **8**(1) 2500–2505.

Altin, I., A. Sipahioglu. 2024. Drone arc routing problems and metaheuristic solution approach. *Drones* **8**(8) 373.

Amorosi, L., J. Puerto, C. Valverde. 2021. Coordinating drones with mothership vehicles: The mothership and drone routing problem with graphs. *Computers & Operations Research* **136** 105445.

Amorosi, L., J. Puerto, C. Valverde. 2023. A multiple-drone arc routing and mothership coordination problem. *Computers & Operations Research* **159** 106322.

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.

Boysen, N., D. Briskorn, S. Fedtke, S. Schwerdfeger. 2018. Drone delivery from trucks: Drone scheduling for given truck routes. *Networks* **72**(4) 506–527.

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.

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

Chow, J. Y. 2016. Dynamic UAV-based traffic monitoring under uncertainty as a stochastic arc-inventory routing policy. *International Journal of Transportation Science and Technology* **5**(3) 167–185.

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.

Corberán, Á., R. Eglese, G. Hasle, I. Plana, J. M. Sanchis. 2021a. 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, Á., I. Plana, M. Reula, J. M. Sanchis. 2021b. Arc routing problems: Data instances. `https://www.uv.es/corberan/instancias.htm`. Last updated: May 2021.

Corberán, T., I. Plana, J. M. Sanchis. 2025. The min max multi-trip drone location arc routing problem. *Computers & Operations Research* **174** 106894.

De Maio, A., D. Laganà, R. Musmanno, F. Vocaturo. 2021. Arc routing under uncertainty: Introduction and literature review. *Computers & Operations Research* **135** 105442.

Di Puglia Pugliese, L., G. Macrina, F. Guerriero. 2021. Trucks and drones cooperation in the last-mile delivery process. *Networks* **78**(4) 371–399.

Dille, M., S. Singh. 2013. Efficient aerial coverage search in road networks. *AIAA Guidance, Navigation, and Control (GNC) Conference*. 5094.

Eiselt, H. A., M. Gendreau, G. Laporte. 1995a. Arc routing problems, part i: The chinese postman problem. *Operations Research* **43**(2) 231–242.

Eiselt, H. A., M. Gendreau, G. Laporte. 1995b. Arc routing problems, part ii: The rural postman problem. *Operations Research* **43**(3) 399–414.

Engberts, B., E. Gillissen. 2016. Policing from above: Drone use by the police. *The future of drone use*. Springer, 93–113.

Ghiasvand, M. R., D. Rahmani, M. Moshref-Javadi. 2024. Data-driven robust optimization for a multi-trip truck-drone routing problem. *Expert Systems with Applications* **241** 122485.

Golden, B. L., R. T. Wong. 1981. Capacitated arc routing problems. *Networks* **11**(3) 305–315.

Ha, Q. M., Y. Deville, Q. D. Pham, M. H. Hà. 2020. A hybrid genetic algorithm for the traveling salesman problem with drone. *Journal of Heuristics* **26** 219–247.

Jiang, J., Y. Dai, F. Yang, Z. Ma. 2024. A multi-visit flexible-docking vehicle routing problem with drones for simultaneous pickup and delivery services. *European Journal of Operational Research* **312**(1) 125–137.

Kuo, R., S.-H. Lu, P.-Y. Lai, S. T. W. Mara. 2022. Vehicle routing problem with drones considering time windows. *Expert Systems with Applications* **191** 116264.

Laporte, G., R. Musmanno, F. Vocaturo. 2010. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science* **44**(1) 125–135.

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

Li, H., J. Chen, F. Wang, Y. Zhao. 2022. Truck and drone routing problem with synchronization on arcs. *Naval Research Logistics (NRL)* **69**(6) 884–901.

Li, M., L. Zhen, S. Wang, W. Lv, X. Qu. 2018. Unmanned aerial vehicle scheduling problem for traffic monitoring. *Computers & Industrial Engineering* **122** 15–23.

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, Z., R. Gu, M. Poon, Z. Liu, A. Lim. 2022. A last-mile drone-assisted one-to-one pickup and delivery problem with multi-visit drone trips. *Computers & Operations Research* **148** 106015.

Luo, Z., M. Poon, Z. Zhang, Z. Liu, A. Lim. 2021. The multi-visit traveling salesman problem with multi-drones. *Transportation Research Part C: Emerging Technologies* **128** 103172.

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.

Mogili, U. R., B. Deepak. 2018. Review on application of drone systems in precision agriculture. *Procedia Computer Science* **133** 502–509.

Momeni, M., H. Soleimani, S. Shahparvari, B. Afshar-Nadjafi. 2022. Coordinated routing system for fire detection by patrolling trucks with drones. *International Journal of Disaster Risk Reduction* **73** 102859.

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.

Morandi, N., R. Leus, J. Matuschke, H. Yaman. 2023. The traveling salesman problem with drones: The benefits of retraversing the arcs. *Transportation Science* **57**(5) 1340–1358.

Moshref-Javadi, M., A. Hemmati, M. Winkenbach. 2020. A truck and drones model for last-mile delivery: A mathematical model and heuristic approach. *Applied Mathematical Modelling* **80** 290–318.

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.

Oh, H., S. Kim, A. Tsourdos, B. A. White. 2014. Coordinated road-network search route planning by a team of UAVs. *International Journal of Systems Science* **45**(5) 825–840.

Oh, H., H. Shin, A. Tsourdos, B. White, P. Silson. 2011. Coordinated road network search for multiple UAVs using dubins path. *Advances in Aerospace Guidance, Navigation and Control*. Springer, 55–65.

Otto, A., N. Agatz, J. Campbell, B. Golden, E. Pesch. 2018. Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks* **72**(4) 411–458.

Pearn, W.-L., A. Assad, B. L. Golden. 1987. Transforming arc routing into node routing problems. *Computers & Operations Research* **14**(4) 285–288.

Petitprez, E., F. Georges, N. Raballand, S. Bertrand. 2021. Deployment optimization of a fleet of drones for routine inspection of networks of linear infrastructures. *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 303–310.

Pisinger, D., S. Ropke. 2019. *Large Neighborhood Search*. Springer International Publishing, Cham, 99–127. doi: 10.1007/978-3-319-91086-4_4. URL `https://doi.org/10.1007/978-3-319-91086-4_4`.

Poikonen, S., B. Golden. 2020. Multi-visit drone routing problem. *Computers & Operations Research* **113** 104802.

Poikonen, S., B. Golden, E. A. Wasil. 2019. A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing* **31**(2) 335–346.

Rabta, B., C. Wankmüller, G. Reiner. 2018. A drone fleet model for last-mile distribution in disaster relief operations. *International Journal of Disaster Risk Reduction* **28** 107–112.

Rakha, T., A. Gorodetsky. 2018. Review of unmanned aerial system (uas) applications in the built environment: Towards automated building inspection procedures using drones. *Automation in Construction* **93** 252–264.

Rave, A., P. Fontaine, H. Kuhn. 2023. Drone location and vehicle fleet planning with trucks and aerial drones. *European Journal of Operational Research* **308**(1) 113–130.

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

Ropke, S., D. Pisinger. 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science* **40**(4) 455–472.

Sipahioglu, A., G. Kirlik, O. Parlaktuna, A. Yazici. 2010. Energy constrained multi-robot sensor-based coverage path planning using capacitated arc routing approach. *Robotics and Autonomous Systems* **58**(5) 529–538.

Sun, W., Z. Luo, X. Hu, W. Pedrycz, J. Shi. 2024. An improved variable neighborhood search algorithm embedded temporal and spatial synchronization for vehicle and drone cooperative routing problem with pre-reconnaissance. *Swarm and Evolutionary Computation* **91** 101699.

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.

Wang, D., P. Hu, J. Du, P. Zhou, T. Deng, M. Hu. 2019. Routing and scheduling for hybrid truck-drone collaborative parcel delivery with independent and truck-carried drones. *IEEE Internet of Things Journal* **6**(6) 10483–10495.

Wang, Z., J.-B. Sheu. 2019. Vehicle routing problem with drones. *Transportation Research Part B: Methodological* **122** 350–364.

Wu, G., K. Zhao, J. Cheng, M. Ma. 2022. A coordinated vehicle–drone arc routing approach based on improved adaptive large neighborhood search. *Sensors* **22**(10) 3702.

Xia, Y., W. Zeng, C. Zhang, H. Yang. 2023. A branch-and-price-and-cut algorithm for the vehicle routing problem with load-dependent drones. *Transportation Research Part B: Methodological* **171** 80–110.

Xu, B., K. Zhao, Q. Luo, G. Wu, W. Pedrycz. 2023. A gv-drone arc routing approach for urban traffic patrol by coordinating a ground vehicle and multiple drones. *Swarm and Evolutionary Computation* **77** 101246.

Xue, G., Y. Li, Z. Wang. 2023. Vessel-uav collaborative optimization for the offshore oil and gas pipelines inspection. *International Journal of Fuzzy Systems* **25**(1) 382–394.

Yu, L., E. Yang, P. Ren, C. Luo, G. Dobie, D. Gu, X. Yan. 2019. Inspection robots in oil and gas industry: a review of current solutions and future trends. *2019 25th International Conference on Automation and Computing (ICAC)*. IEEE, 1–6.

Zandieh, F., S. F. Ghannadpour, M. M. Mazdeh. 2024. New integrated routing and surveillance model with drones and charging station considerations. *European Journal of Operational Research* **313**(2) 527–547.

Zeng, F., Z. Chen, J.-P. Clarke, D. Goldsman. 2022. Nested vehicle routing problem: Optimizing drone-truck surveillance operations. *Transportation Research Part C: Emerging Technologies* **139** 103645.

Zhou, H., H. Qin, C. Cheng, L.-M. Rousseau. 2023. An exact algorithm for the two-echelon vehicle routing problem with drones. *Transportation Research Part B: Methodological* **168** 124–150.