Multiple Drone and Truck Arc Routing Problem

Mohammad Jahidul Alam

A Thesis
in
The Department
of
Mechanical, Industrial and Aerospace Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Applied Science (Industrial Engineering) at
Concordia University
Montréal, Québec, Canada

July 2022

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mohammad Jahidul Alam**

Entitled: **Multiple Drone and Truck Arc Routing Problem**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Industrial Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________________ Chair
Dr. Mingyuan Chen

______________________________ External Examiner
Dr. Hossein Hashemi Doulabi

______________________________ Supervisor
Dr. Ivan Contreras

Approved by ______________________________
Martin Pugh, Chair
Department of Mechanical, Industrial and Aerospace Engineering

____________ 2022

____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Multiple Drone and Truck Arc Routing Problem

Mohammad Jahidul Alam

In this thesis we introduce the Multiple Drone and Truck Arc Routing Problem (MDTARP) which considers a fleet of drones used in synchronization with a ground vehicle to provide service to edges in a network within a given time-horizon. Drones launch from a ground vehicle to perform services on a set of edges and return to recharge batteries ready for its next trip. We formulate a multi-objective mathematical model that maximizes coverage of edges on a network based on given weights while minimizing unnecessary travel by drones and the ground vehicle. We develop an Iterated Local Search heuristic and a Cluster-based Location Search heuristic and assess their performance on several instances representing different scenarios. We compare results with solutions obtained using a commercial solver to showcase the effectiveness of the heuristics and perform computational experiments to provide recommendations for the users.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Dr. Ivan Contreras who believed in me and encouraged me to pursue my research. I am incredibly thankful for his invaluable advice, encouragement, and patience during the completion of my thesis. I would also like to thank Dr. Enrique Dominguez from the University of Malaga for his support and guidance.

My work would not have been possible without my family and friends who's belief in me has kept my spirits and motivation high during this process. This thesis is dedicated to my father who's love, sacrifice and unwavering support has allowed me to make it to this position.

Contents

List of Figures

List of Tables

List of Abbreviations

ACO	Ant Colony Optimization
ARP	Arc Routing Problem
CARP	Capacitated Arc Routing Problem
CPP	Chinese Postman Problem
CVRP	Capacitated Vehicle Routing Problem
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
GV	Ground Vehicle
ILS	Iterated Local Search
MA	Memetic Algorithm
MDTARP	Multiple Drone Truck Arc Routing Problem
ROV	Remote Operated Vehicle
RPP	Rural Postman Problem
SA	Simulated Annealing
TS	Tabu Search
TSP	Travelling Salesman Problem
UAV	Unmanned Aerial Vehicle
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem

Chapter 1

Introduction

Unmanned aerial vehicles (UAVs) or drones are being utilized increasingly in a variety of industries due to their reduced cost, higher speeds and improved safety. Drones do not require the assistance of on-board pilots therefore weigh significantly less than traditional aircrafts and as a result consume less energy. They are not restricted to road networks and can operate in environments that would otherwise be unsafe or inaccessible. PricewaterhouseCoopers [1] estimates the total value of drone powered solutions at over $127bn across industries such as infrastructure, transport, agriculture, telecommunications, security, and media and entertainment.

Amazon announced "Amazon Prime Air" in 2013, a delivery system that aims to get packages to customers in 30 minutes or less using drones [2], carrying out its first delivery in 2016. Finnish postal service, Posti, flew a drone carrying a 3kg parcel a distance of 4km from mainland to the island of Suomenlinna in Helsinki to become the first in Europe to test drone delivery in an inhabited urban environment [3]. Maersk has conducted drone delivery tests to transport spare parts to its workers across its large fleet of tankers estimating savings of $3,000 to $9,000 annually per ship [4]. Zipline, a US-based drone delivery company, regularly delivered Coronavirus test samples in batches from rural regions in Ghana to central medical facilities during the COVID-19 pandemic in 2020 [5]. By the end of 2021, the company successfully distributed more that 200,000 COVID-19 vaccine doses across Ghana [6]. Ford and DJI announced a challenge in 2016 to develop a software that uses drones and vehicles to gather data for a search-and-rescue mission. A drone would have to autonomously take off from a moving vehicle, collect and transmit data, then return and autonomously land on the same moving vehicle [7].

Many of the applications mentioned involve delivery using drones where services have to be performed at a particular point and can be modelled as an extension of the well-known node routing problem (NRP). Applications involving inspection and surveillance can be seen as extensions of the arc-routing problem (ARP) where services have to be performed along linear features. ARPs are a class of problems that involve finding the least cost traversal of a set of edges or arcs on a graph. The classical ARP can be used to model problems such as road maintenance, garbage collection, mail delivery, school bus routing, meter reading, etc. The ARP can be extended to involve drones for applications where services which would otherwise be unsafe, labour intensive, energy intensive or time consuming.

In the context of drone aided-routing, three main areas of application were identified by Singhal et al [8] in a review: civilian, environmental, and defence. Otto et. al [9] highlight promising civil applications including infrastructure, agriculture, transport, security and media. Smith [10] analyses environmental applications including monitoring of deforestation, real-time tracking of animal poachers, early forest fire detection and detection of contamination in sewers and storm drains. Within industries where companies need to operate extensive networks of complex assets, drones are used for investment monitoring, maintenance and inventory. In agriculture, drones are capable of producing precise 3D maps across large areas of land for soil analysis and have been used to achieve uptake rates of 75% and decreased costs by 85% [11]. Drones have been used in major sporting events such as the 2014 World Cup in Brazil to provide real-time data to security teams for crowd monitoring. In the entertainment industry, drones function to provide aerial film and photography capabilities to capture footage for news broadcasting, at sports events and for wildlife documentaries. Drone routing problems can be extended beyond the use of aerial drones and ground vehicles. The Defense Advanced Research Projects Agency (DARPA) in the U.S. carried out a test flight where small drones completed a set of tasks after being deployed and were then recovered by a C-130 aircraft in flight [12]. DARPA envisions using drone swarms that fly into danger zones to conduct missions such as intelligence and surveillance while the mothership remains out of range. The drones can then be recovered and refurbished for another mission. Dubai Electricity and Water Authority (DEWA) use underwater remote operated vehicles (ROV) for montoring and inspection of underwater infrastructure supporting their water desalination systems [13]. Underwater ROVs have been used to search the seabed for signs of the Malaysian Airlines Flight MH370 after its disappearance in 2014 [14]. It is clear that many more applications exist that require continuous service along edges rather than at a single point.

Despite their many advantages, drones suffer from restrictions in battery life which can be a significant shortcoming especially when they need to perform continuous services. As a results they would need regular battery replacements which may limit their movement capabilities. These shortcomings can be avoided if they are used in tandem with land vehicles. Research on routing problems with drones and further variants involving truck-drone combined operations have received increased attention as highlighted in recent surveys [15, 16]. Much of the attention, however, falls on NRPs, specifically problem involving last mile delivery. In this paper we study an ARP involving drones and a ground vehicle that is capable of providing services along linear features represented by edges that can be more beneficial in applications of surveillance and monitoring.

1.1 Thesis Overview

In this thesis we introduce the Multiple Drone and Truck Arc Routing Problem (MDTARP). The MDTARP considers multiple unmanned vehicles used in tandem with a land vehicle to provide service to edges in a network (city streets, transmission lines, railway lines, pipelines, etc.) with the aim of maximising coverage of the network based on the weights given to the edges. In our study, we consider the use of aerial drones and a ground vehicle (GV) to perform surveillance on suburban road networks. The MDTARP can, however, be applied to different industries and environments where unmanned vehicles such as road or water vehicles may be used to improve operation efficiency or safety. In this problem, the GV is available only to transport drones and perform services when needed, such as battery replacements. The drones have a limited battery capacity and may only take-off and land from the GV. The GV begins and ends at a base depot with all drones on board. The drones may be transported between locations by the GV and must return to the GV for battery replacements. The MDTARP is considered an arc routing problem as drones need to perform services on edges based on weights given to the edges.

The main contributions of this thesis are three-fold:

1. We introduce a new arc routing problem that involves a fleet of drones used in tandem with a ground vehicle. We formulate a multi-objective mathematical model that maximizes coverage of edges on a network based on given weights while minimizing unnecessary travel by drones and the ground vehicle.

2. We develop two heuristics to solve the problem with the aim of providing high quality solutions within a relatively short period of time. The problem is decomposed into smaller sub-problems that allow us to achieve improvements in time performance.

3. We evaluate the solutions obtained by the heuristics, comparing them to each other and to results obtained using a commercial solver. We generate several instances to test different scenarios.

We start in Chapter 2 by providing some background on arc routing problems and highlighting some classic heuristic and metahueristic approaches. We look at past and current research done on drone aided routing problems together with solution methodologies employed. In Chapter 3, we formally define the MDTARP and describe the applications that will be considered in this thesis. We describe the objectives of the problem and introduce a Mixed Integer Programming (MIP) formulation. In Chapter 4 we introduce several algorithms to solve the problem providing detailed descriptions of the algorithms and their operations. In Chapter 5 we look at some computational results that were obtained by generating several instances to test different scenarios. We provide details on the process of generating these instances and a description of the instances to help better understand scenarios being tested. We provide solutions obtained using the proposed algorithms and evaluate the results by comparing with results obtained using CPLEX. We look at how certain parameters affect performance of the algorithms and provide suggestions for users. Finally in Chapter 6 we conclude and discuss some future research avenues.

Chapter 2

Background

In this chapter, we provide an overview of the work related to the problem described. In Section 2.1 we look at ARPs, the different variants and some heuristics developed to solve such problems. In Section Section 2.2 we look at some work related to Drone Aided Routing problems, in particular the distinction between Drone Aided Node Routing and Drone Aided Arc Routing problems.

2.1 Arc Routing Problems

ARPs are a class of problems that aim to find the least cost traversal of a set of edges or arcs on a graph subject to some constraints. Such problems are encountered in a variety of practical situations such as road or street maintenance, garbage collection, mail delivery, school bus routing, meter reading, etc.

The Chinese Postman Problem (CPP) consists of determining the least cost closed walk of all edges and arcs of a graph where each edge or arc has an associated cost. The Rural Postman Problem (RPP) is a generalization of the CPP where only a subset of the edges must be traversed. The Capacitated Arc Routing Problem (CARP) allows for more than one vehicle to traverse the edges.

Edmonds and Johnson [17] showed that the CPP is solvable in polynomial time in its original form involving either directed or undirected graphs. There are, however, several variants and side constraints that make the problem $\mathcal{NP}$-hard. Practical applications of this problem can involve both arcs and edges, for example when modeling roads with one-way streets. In this case, choosing the orientation of the edges becomes the main difficulty given the constraints. These problems are known as Mixed Chinese Postman Problems and has been shown by Papadimitriou [18] to be $\mathcal{NP}$-complete. Another variant of the CPP is known as the Windy Chinese Postman Problem where the cost of traversing an edge may vary based on the direction. This can be used for modeling roads on a slope or blowing wind. This problem is $\mathcal{NP}$-hard, as shown by Guan [19], and can be transformed from a Mixed Chinese Postman Problem in polynomial time. Another variant of the CPP considers traversing some edges before others. In practice, this may be useful when modelling snow plowing where main street should be cleared before other streets. Even with just two priority classes, these problems are $\mathcal{NP}$-hard as observed by Dror, Stern, and Trudeau [20] and can be reduced from the $\mathcal{NP}$-hard RPP. Several postmen can be considered in a variant of the CPP where each edge is traversed by at least one of the postmen. Complexity of this problem depends on the objective function chosen. It is polynomial-time solvable if the sum of the tour cost is minimized and $\mathcal{NP}$-hard when maximizing the maximum tour costs.

Most arc-routing problems are modelled as RPPs as there are only a few practical cases where all arcs in a network must be serviced. Unlike the CPP, both the directed and the undirected RPPs are $\mathcal{NP}$-hard, as shown by Lenstra and Kan [21]. The mixed version of the RPP involving both arcs and edges, also known as the Stacker Crane problem, has also been shown to be $\mathcal{NP}$-hard by [22].

The CARP, introduced by Golden and Wong [23], generalizes both the CPP and the RPP. In this problem, each arc in an undirected graph has an associated cost and demand. All arcs with positive demand must be traversed by one vehicle, with a specific capacity, from a fleet housed at the depot. Even if all edges are required to be traversed, the CARP is $\mathcal{NP}$-hard contrasting the polynomial time solvability of the CPP.

2.1.1 Heuristics for Arc Routing Problems

Four heuristics developed to solve ARPs are explored in this section: Construct-Strike, Path Scanning, Augment-Merge and Ulusoy's method.

Construct-Strike was first proposed by Christofides in 1973 [24]. The idea is to make the graph Eulerian. However, because of the capacity constraints, the tasks cannot be included in a single route and therefore simple feasible routes are first extracted successively with their edges erased while keeping the graph connected. When this is no longer possible, new edges are added to avoid isolating the depot and make all node-degrees even. A minimum cost perfect matching problem can be solved to make the graph Eulerian to continue to extract routes.

Path-Scanning was proposed by Golden, DeArmon and Baker in 1983 [25]. Routes are built starting at the depot and extended by adding a required edge that is compatible with the vehicle capacity and not yet serviced. If several required edges appear, the next edge is selected based on a set of rules. If all required edges incident to the last vertex on the route have already been served or the vehicle edges no longer fit the vehicles capacity, the construction of the route stops and the vehicle returns to the depot via a shortest path. Routes are built this way until all required edges are served. In 1992, Evans and Minieka [26] proposed continuing to extend the route to nearest vertex having incident required edges instead of stopping when if no the last vertex has no incident edges. This approach has been adopted by most authors.

Augment-Merge was first proposed in 1981 by Golden and Wong [23] taking inspiration from the Clark and Wright heuristic [27] for the Capacitated Vehicle Routing Problem (CVRP). The heuristic builds one direct trip for each required edge then merges two routes based on the savings encountered. Since routes can be reversed, merges can be made in four different ways and the route with the largest savings in selected. These merges are repeated while vehicle capacity is not violated. The Augment phase involves sorting the initial route in non-increasing order of cost. Each route, staring from the longest, is compared with each of the shorter routes such that the sum of their loads does not exceed vehicle capacity. If the edge served in the shorter route lies within the longer route, that edge can be added to the longer route. This reduce the number of routes and the total cost before the Merge phase is executed.

Ulusoy's route-first cluster-second method [28] is a derivative of a method developed for the CVRP by Beasley [29] in 1983. Beasley's method first creates a tour covering all vertices and ignoring vehicle capacity by solving a Travelling Salesman Problem (TSP) using any algorithm. In the second phase, the tour is split into feasible CVRP routes. Similarly, Ulusoy's method solves the CPP to obtain a giant tour with all required edges, τ, which is then partitioned into feasible routes. When all edges are not required the giant tour can be found by solving the NP-hard RPP using heuristics. Splitting is done by building an auxiliary graph, H, with $\tau + 1$ nodes. Arcs on H correspond to feasible trips and are weighted by cost of the trip. The optimal splitting is determined by the shortest path from node 0 to node τ on H.

2.1.2 Metaheuristics for Capacitated Arc Routing Problems

Prior to 2000, metaheuristics designed for the CARP were either simulated annealing procedures or tabu search algorithms. Since then, based on publications for vehicle routing in general, there has been more focus on population-based heuristics (such as genetic algorithms and scatter search) and multiagent methods (such as ant colonies) and an increase in algorithms that can provide a good balance between running time and solution quality (such as VNS and GRASP) [30].

Simulated Annealing (SA) is a local search metaheuristic that provides a hill climbing mechanism to escape local optima. The concept of annealing in optimization was first introduced in 1983 by Kirkpatrick et al [31]. The inspiration for this technique comes from the process of physical annealing in solids, where crystalline solids are heated and then cooled in a controlled manner to help achieve a state that is free from defects. In SA, random local neighborhood moves are generated at each iteration. The current solution is compared to a newly generated solution. Improving solutions are always selected while non-improving solutions are selected with a probability depending on a temperature parameter. The temperature parameter is controlled based on a scheme known as the cooling schedule. In general, the temperature parameter and is decreased overtime and therefore, overtime, non-improving solutions are selected less frequently. Sufficient hill climbing moves can result in the solution reaching the global optimum. An SA algorithm was the first metaheuritsic dedicated to the undirected CARP in a winter gritting vehicle routing problem, introduced by Eglese in 1994 [32]. The problem involves multiple depots, limited vehicles capacities and different road priorities. Initial solutions are constructed using the Clark and Wright saving heuristic and SA is used to improve the solution. Neighborhood solutions are generated by removing a node from the current route and linking it to an adjacent node or depot to complete the route.

Tabu search (TS) is another method often used to escape local optima, originally proposed by Glover in 1986 [33]. TS is used as an extension of classical local search. Whenever local optimum is encountered in a local search, non-improving moves are accepted and a tabu list is used to record recent history of movements to prevent cycling back to previously visited solutions. Eglese and Li proposed the first TS for the undirected CARP in 1996 [34]. They only consider moves that changes the route servicing an edge only if the new route traverses one of the end-nodes of that edge. Solutions are accepted but penalised if they violate the vehicle capacity constraint. To improve the solution, a heuristic that solves the RRP is applied after the best non-tabu move at each iteration is determined. Hertz, Laporte and Mittaz published CARPET in 2000 [35], a TS initially designed for the RPP that involves four main procedures (Cut, Drop, Add and Shorten). The metaheuristic creates an initial solution based on a route-first cluster-second heuristic. A tour is created by solving an RPP and a CUT algorithm is then used to partition the initial tour into feasible routes. Neighborhood solutions are obtained by removing the service of edge from a route in the solution using a DROP algorithm, and adding it to another route using a ADD algorithm. A tabu list is used to record recent moves and avoid cycling. The search process is diversified by merging several routes into a giant (possibly infeasible) route. This new route is then shortened using the SHORTEN algorithm and decomposed into feasible routes using the CUT procedure. Several intensification procedures are also described to further improve the solution.

The Variable Neighborhood Search (VNS) is a metaheuristic proposed by Mladenović and Henson in 1997 [36] that systematically changes neighborhoods within a local search algorithm. It works based on the facts that a local optimum for one neighborhood is not necessarily the local optimum for another, a global optimum is the local optimum for all neighborhoods and the empirical observation that the local optima for neighborhoods are often close to one another. Therefore information contained in local optima, for example values of variables, can be exploited as they may be equal or close to those in the global optimum. VNS takes a solution and explores increasingly distant neighborhoods jumping to a new solution only if an improvement is made. The metaheuristic starts with an initial solution and k neighborhoods. At the first iteration, a solution is generated from k^{th} neighborhood of the initial solution and a local search method is applied. If the obtained solution is better than the initial solution, then the search procedures starts from the final solution using $k = 1$, otherwise the search is repeated from the initial solution with $k = k+1$. Polacek et al. [37] designed a VNS for the CARP with intermediate facilities. An initial solution is generated using Ulusoy's construction to split a giant tour into feasible trips. In the shaking step, two random segments from different routes are exchanged and edge directions are kept. The local search considers the reversal of each serviced edge in a particular route and the cost of reconnecting the edge into the tour. Modifications are only made when a cost reduction is obtained.

The Greedy Randomized Adaptive Search Procedure (GRASP) is a iterative metaheuristic where in each iteration a greedy randomized heuristic is used to build one trial solution which is improved using a local search procedure. The first GRASP for the undirected CARP was developed in 2005 by Prins and Wolfler [38]. A randomized version of the Merge heuristics is used to construct the trial solutions where two routes are merged based on a given probability and improved by a local search procedure.

Genetic algorithms (GA) are metaheuristics that take inspiration from natural evolution and exploit the idea of natural selection. GAs guide search with the use of a population of solutions and generate new and potentially better solutions (offspring) by combining two or more solutions (parents) in a crossover operation. In a selection process the population is sampled using a fitness function and parents are chosen to produce the offspring. Variations in the offspring can be introduced by applying mutations, where parts of the solution is changed by doing a random walk. Memetic algorithms (MA) integrate a local search component in GAs to improve children solutions. Lacomme, Prins, and Ramdane-Chérif proposed [39] an MA in 2001 for the undirected CARP. The algorithm uses RPP tours as parent solutions and Ulusoy's splitting procedure to obtain feasible CARP solutions from the children. A local search is applied to individual routes, rather than the RPP tours, in each of the offspring solutions. New parent solutions are created by concatenating resulting routes which then replace existing parents in the population.

Ant colony optimization (ACO), proposed by Dorigo and Di Caro in 1999 [40], is a metaheuristic inspired by the behaviour of ants laying and following pheromones trail as a form of communication. When ant find a food source, they carry some of the food back to the nest and leave a chemical pheromone trail on the ground to guide other ants. Given several paths to the food source, other ants pick a path at random with equal probability. As more ants follow a path, the amount of pheromone on the path increases and therefore the probability an ant chooses that path also increases. Overtime, the amount of pheromone deposited on the shortest path is greater as it is traversed by more ants in that period of time. ACO uses a population of agents (colony of ants) and pheromone trails represented as numerical information used by the ants. Using a given pheromone model, ants construct solutions to the problem in a probabilistic manner at each iteration of the algorithm. A local search may be applied to the solution as an optional step. A part of the solution can then be used to update the pheromone model to be used in the next iteration. Lacomme, Prins and Tanguy presented the first competitive ant colony scheme for the CARP in 2004 [41]. They construct giant tour relaxing the vehicle capacity constraint and split the tour using the Split procedure in [39]. Part of the colony of ants attempt to diversify the solutions while others favor convergence of the algorithm. Pheromones are updated depending on the quality of the solutions and erased after a number of iterations without improvement to avoid being stuck in a local optima.

2.2 Drone Aided Routing

Drones are often used in tandem with trucks in order to overcome their limitations. The use of drones in routing provides benefits due to their high speeds, relatively lower cost and improved safety. They can be used in applications where it may be more dangerous or difficult for larger vehicle to reach. In these cases, because two different vehicles are used, optimization of such routing problems can be particularly challenging. In many applications both vehicles are often required to perform tasks in synchronization, for example replenish items for drone delivery, replace or charge a drone battery, and perform maintenance of drones. In these cases, vehicles must meet at a location at the same time or one vehicle waits for the other before.

Drone routing problems have been extensively studied over the last few years [9], often as an extension to the NRP. In NRPs, service is performed at the vertices and so the real shape of roads or streets, represented by edges on a graph, can be ignored. However, when considering ARPs with drones, service is performed on the edges of the graph (along roads and railways). Drones can travel between any two points on the plane as they do not need to strictly follow edges and may start and end service at any point on an edge. The type of drone used can also affect its flight dynamics. Fixed wing drones, for example, have a minimum turning radius when changing directions. Rotorcraft drones can more easily make sharp turns, however, would require additional time and energy to do so as they need to come to a complete stop first. Smaller drones can be highly susceptible to weather, such a wind speed and direction. In certain cases, the type of service performed can also affect battery usage. Traversing edges while servicing may lead to more usage of battery as opposed to traveling off the network. These factors make Drone ARPs a continuous optimization problem with possibly infinite feasible solution and contribute to making them more difficult to model and solve [42].

2.2.1 Drone Aided Vehicle Routing

In a recent review Macrina at al. [15] identify four classes of drone aided routing problems: 1) the traveling salesman problem with drones (TSP-D), 2) the vehicle routing problem with drones (VRP-D), 3) the drone delivery problem (DDP) and 4) the carrier problem with drones (CVP-D). These classes present variants of the node routing problem where trucks and drones perform operations on a single point.

Two new variants of the TSP combining drones and trucks were introduced by Murray and Chu [43] in 2015 called the flying sidekick travelling salesman problem (FSTSP) and the parallel drone scheduling TSP (PDSTSP). The objective of the FSTSP is to minimize the time taken to serve a set of customers exactly once either with a single truck or a single drone and return both vehicles to the depot. Both vehicles must depart from and return to the depot exactly once either independently or together, in which case the drone is transported by the truck. The drone may take multiple tours over the delivery cycle to serve one customer per tour where it either begin at the depot or at a customer location and returns to the depot or at the location of the truck within its endurance limit. Some customers can only be served by the truck where delivery by drone may be infeasible. A simple, 10-customer problem requires several hours to solve using MILP solvers; thus the authors propose a route and re-assign heuristic. A TSP is solved that assigns the truck to serve all customers, and then customers are removed from the route while drones are added by evaluating the savings achieved. The PDSTSP minimizes the time taken for the truck and a fleet of one or more identical drones to return to the depot having served the customers. In this variant, however, there is no cooperation between the vehicles. Drones can complete multiple trips starting and ending their routes at the depot, serving at most one customer per trip. The heuristic proposed for this problem partitions the customers that can be served by UAV (based on distance and weight capacity of the drones) and by truck. Solutions are constructed by solving a parallel machine scheduling problem to assign eligible customers to UAVs and solving a TSP to determine the truck route through the remaining customers. A local search is then applied to improve the solution.

Salama and Srinivas [44] proposed a variant of the TSP with drones, composing of one truck and multiple drones considering two objective functions: minimizing the total delivery cost and minimizing the completion time. Customers are clustered and drones are carried by the truck to some focal point in each cluster. Deliveries can be made by either the drones or the truck, in which case the customer location becomes a focal point to dispatch drones. The drones then deliver to a single customer and returns to the truck to be taken to the next focal point. The heuristic proposed by the authors uses K-means clustering to find focal points and solves a TSP to determine the optimal route through the clusters.

Moshref-Javadi et al. [45] consider a variant of the travelling repairman problem where the truck makes deliveries itself while serving as a moving hub for multiple UAVs to extend its service range. Drones take off at a truck stop to make a delivery and returns to the truck at a consecutive stop within an allowable period of time. The objective of the problem is to find the route that will minimize the sum of the customer waiting times. The authors developed an algorithm based on the adaptive large neighborhood search (ALNS) metaheuristic. Initial solutions are constructed by randomly assigning all customers to the truck route and improved using a SA algorithm where neighboring solutions are found using 2-Opt and 3-Opt operators. Then customers are removed from the truck route and assigned to UAVs using an elliptical method. An ellipse is fit to the customers using least squares criterion and customers are assigned to the UAVs based on maximum perpendicular distance from the ellipse, feasibility of the solution and savings in customer wait times.

Lou et al. [46] consider a variant of the classic two-echelon TSP, where the truck cruises on the road between rendezvous points carrying a drone that is capable of serving customers. The drone can take-off from the truck at a rendezvous point, visit a set of customers and return to the truck to charge for the next trip. The objective is to minimize the total routing time for the drones. The authors develop two algorithms to construct solutions. The first constructs a complete tour of all the customers using a genetic algorithm and splits this tour into subtours that can cooperate with the route of the truck. The second algorithm, also uses a genetic algorithm to create a complete tour of the rendezvous points and assigns customers to this tour based on the drones limits.

The VRP-D, introduced by Wang et al. [47], generalizes on the TSP-D considering a fleet of trucks equipped with one or more drones aiming to minimize completion time. Packages can be delivered by both trucks and drones, which can deliver a single package at a time and can be picked up and dispatched by the trucks from either the depot or any of the customer locations. This work is extended by Wang and Sheu [48], where drones can perform multiple deliveries and return to a truck different from the one from which it was dispatched.

Kitjacharoenchai et al. [49] describe a two-echelon VRP-D where the first level is the routing of trucks from the depot to the customers and the second level is the routing of the drones from the trucks to the customers. Starting and ending at the main depot, trucks behave like a movable depot for the drones while making its own deliveries. A single drone is dispatched at any customer location, makes several deliveries independently, and returns to the truck from which it took-off. The authors propose two algorithms. The first solves a capacitated VRP problem using the Calrke and Wright Saving Algorithm and apply local searches such as 2-opt, then constructs feasible sub-routes for the drones within the truck routes. The second algorithm is based on Large Neighborhood Search where an initial feasible solution is gradually improved by iteratively being destroyed and repaired.

The DDP is a variant of the VRP where the fleet is composed of drones only and considers characteristics such as battery capacity and flying ranges. The CVP-D combines features from the TSP-D, VRP-D and DDP-D. In this problem, vehicles with complementary capabilities are used in collaboration to carry out deliveries. Small, fast vehicles with limited operational range that can visit a set of customers quickly are carried by large, slow carriers as a base to allow multiple trips. Matthew et al. [50] purpose a variant where a large truck carries packages and a drone that makes a single delivery per trip. Poikonen and Golden [51] consider a variant that allows for the drone to make more than one delivery per trip. The authors extend this problem to include more that one drone in the k-Multi-Visit Drone Routing Problem (k-MVDRP) [52]. Drones are able to launch from the truck carrying one or more packages and return to the truck once the packages have been delivered to recharge and pick up more packages. The problem requires that once drones have launched from a certain location the truck must travel directly to the retrieval location, which may be the same as the launch position, without stopping or launching any additional drones. Therefore in cases whee multiple drone launch at once, they all must return at the same location.

2.2.2 Drone Arc Routing

ARPs with drones have received much less attention than NRPs possibly due to their complexity. Much of the work done on drone aided arc routing considers robots or other unmanned vehicles which may not be much different to regular road vehicles. The interest in using aerial drones comes from the advantage of being able to service a subset of edges on a network and travel directly between points on the network with the Euclidean metric.

Oh et al. [53] present a road-network search route planning problem where multiple drones are used to search road segments on a network. It is modelled as a modified CPP accounting for the unconnected roads, as drones are not restricted to move along roads, and physical constraints of fixed winged drones. Dubins theory is used to model the drones' path and a modified knapsack problem is formulated and solved as a MILP to find optimal solutions minimising flight time. Additionally, a nearest insertion algorithms is introduced to overcome computational burden of the knapsack algorithm.

Li et al. [54] investigate a drone scheduling problem for real-time road traffic monitoring. The problem aims to minimize operating costs while satisfying uncertain monitoring demands and drone capacity constraints. Drones start and end at a depot and are capable of monitoring several arcs and each arc can be monitored by several drones. A MILP model is formulated combining the inventory routing problem with the capacitated arc-routing problem and solved with CPLEX for small instances. A local branching algorithm is developed for larger instances.

Campbell et al. [42] introduces the drone RPP (DRPP) which consists of finding minimum cost tour starting and ending at a depot and traversing a set of edges with associated service cost. In this problem a drone starts and ends its tour at a depot. The drone can travel between vertices without following the edges of the network and the cost of travel is the Euclidean distance. The drone may start and end its service at any point along an edge. An algorithm is proposed that solves several instances of RPP exactly. At each instance of the RPP, intermediate points are added to divide edges into smaller segments providing a discrete optimization problem, Adding more intermediate points brings the problem closer to a continuous problem. The solution to the previous instance is used as an upper bound for each instance. The algorithm is capable of solving instances with up to 92 original lines and 16 connected components.

The authors also introduce an extension to the problem where multiple drones are available and the length of their routes are limited. In the length constrained K-drone rural postman problem (LC K-DRPP), two or more drones can share service of edges. Each drone starts and ends their tour at the depot. A metaheuristic is proposed in [55] where edges are iteratively divided into smaller segments, similar to the algorithm derived for the DRPP. In the first instance, where original edges are used, the algorithm solves an RPP optimally to form a 'giant tour' which is then partitioned into K-routes, one for each drone.

The routes are improved by applying local search procedures. At each subsequent iteration, intermediate points are added to the edges and local search procedures are applied for improvement. The algorithm is tested on instances with up to 137 edges and 2-6 drones.

These papers present variants of ARPs that use drones that do not consider the use of a truck or ground vehicle for assistance. The use of a ground vehicle can aid in overcoming certain limitations faced by drones. Specifically, the limited battery capacity restricts the travel distance of drones making them less effective for certain applications. The following papers explore problem where drones operate with the aid of a ground vehicle as an extension to the ARP.

Amarosi et al. [56] introduces the Mothership and Drone Routing Problem with Graphs (MDRPG). The problem involves one mothership and one drone that visit a set or targets, represented by graphs, in coordination with the objective of minimizing the total distance travelled by both vehicles. The mothership and drone start at a common location, travel together to the first launch point where the drone takes-off from the mothership and flies to a graph to traverse the required egdes and returns to the mothership, possibly at a different location to where it launched, and together return to a final destination. The drones are capable of moving between any point on the edges of each graph and must traverse a percentage of each graph. The authors address versions of the problem where the mothership moves freely on a continuous space and the mothership can move on a network that models a road network. The authors present mixed integer non-linear programming (MINLP) formulations for the problems and a heuristic. The MINLP formulations are solved using CPLEX and Gurobi and compared to solutions obtained from the heuristic. Experiments are run on 5 instances with 10 graphs which increasing number of nodes (from 4 to 10).

The authors also extend the problem to include several homogeneous drones in the All terrain Mothership and Multiple Drones Routing Problem with Graphs (AMMDRPG) [57]. Similar to the MDRPG, the drones and mothership have to coordinate to perform operations on a set of graphs. The drones have a limited flight capacity and must complete the operation and return to the mothership before within a time limit. The mothership moves freely on a continuous space starting and ending together with the drones at common locations. Each graph is visited by one drone which must traverse a given percentage of the graph and must complete its operation before leaving the graph. Drones must launch from the mothership at the same point and must return to the ship at the same point. The vehicles may wait at the rendezvous location for others to arrive therefore need not arrive at the same time. The problem aims to minimize travel distance of the mothership assuming cost of drone trips to be negligible. The authors provide a MINLP formulation to the problem, solved using Gurobi, and a heuristic that is tested on 5 instances of 5 and 10 graphs with varying number of nodes (4 to 10) and drones (1 to 3). The heuristic determines the route of the mothership via a clustering procedure and the entry and exit points of the drone for each graph. The variables are provided to the model as an initial partial solution which then produces a feasible complete solution.

To the best of our knowledge the AMMDRPG is the only problem that considers the use of multiple drones and a mothership in synchronisation as an extension to the ARP. The MDTARP differs from the AMMDRPG in that drones operate on a single graph and is free to traverse edges to maximize coverage based on weights under a given time horizon. Rendezvous locations are predetermined and drones may launch and land at any one of the rendezvous points at any time not having to wait for other drones. Drones may return to edges after a certain time has passed. The MDTARP is better suited for applications where drones must cover a subset of edges under a time constraint, i.e., number of operating hours, and faces restrictions on locations from where drones may launch or land. Given the time restriction it may not be possible for drones to traverse all edges within the time period and therefore drones can cover edges with a higher priority. In areas such as city centers or industrial plants, having designated rendezvous points will help drones avoid obstacles and launch and land in a safe manner. In applications where the state of edges may change over the time horizon, allowing drones to return to edges provides better opportunities to monitor those changes.

Chapter 3

Problem Statement and Mathematical Formulation

This chapter provides a formal definition of the problem, Section 3.1, and we formulate it as a Mixed Integer Programming (MIP) problem, Section 3.2.

3.1 Problem definition

The MDTARP uses drones for applications such as surveillance and monitoring along road networks, rail networks, transmission lines etc. The drones are able to travel in synchronization with a ground vehicle (GV) that is available to perform services, such as replacing batteries, to extend the drones' total tour length. The GV is assumed to be a support vehicle that itself does not service edges, rather exists only to transport and service drones. The drones have limited battery life whereas the GV does not have any such limitations. The speed of the drones and the GV are assumed to be constant and are used to calculate time taken to travel.

The MDTARP is defined using a network that consists of rendezvous nodes, connection nodes and weighted edges. Rendezvous nodes are predefined points on the network where the GV can meet the drones to perform services. The servicing time is considered to be negligible, therefore drones may take-off immediately after landing or may remain on the GV to be transported to another rendezvous point for take-off. Connection nodes signify points on the network such as the end of a block or traffic intersections for road networks. These nodes can be used to define the edges that need to be serviced by the drones. Longer lines, such as for rail networks and transmission lines, can be divided by adding more connection nodes along them. Edges that need to be serviced have an associated weight to signify the importance of traversal. The network is considered fully connected, unless areas exist where drones are unable to fly including restricted zones, high buildings etc. Therefore drones may travel anywhere on the network for purposes of travel between serviceable edges or between rendezvous nodes and serviceable edges.

The drones must complete a tour before the battery is drained and return to the GV. The cost of traversing an edge is the time taken for travel. The GV travels exclusively between rendezvous nodes starting at a base (depot) and returning to the base at the end of the time horizon. Drones can travel anywhere on the network where edges are available. Drones are required to complete traversal of an edge before moving to another edge. The model assumes multi-rotor drones are used for the problem therefore no additional time is required to make sharp turns and change directions. Drones are penalized for traversing edges that are not required to be serviced, i.e., edges on the network that are not weighted. The GV need not use all available rendezvous nodes rather only those that are required to service drones. The GV is also penalised for traveling to service drones, hence minimizing unnecessary travel.

Drones are not required to cover all weighted edges. As the total time available for operation is provided by the user, the drones must maximize coverage within the time-frame. Weights assigned to each edge is proportional to its priority. Therefore in cases where full coverage cannot be achieved, higher priority edges are likely to be covered first. The problem considers scenarios where drones are allowed to return to an edge previously visited after a certain period of time has passed. Hence, over a large period of time drones may decide to return to edges rather than complete coverage of unvisited edges to maximize weight coverage. Users may disallow returns to edges for applications where maximal coverage is preferred. In this case, given sufficient time and/or number of drones, total coverage is possible.

Time is considered a discrete variable in the problem. The time horizon and number of time periods within the horizon is provided by the user. The time taken for the drone or GV to travel can be calculated in terms of time periods depending on the speed of the drones and the GV (also provided by the user). Battery life of the drones is defined in terms of time periods and the battery consumption is proportional to active flight duration.

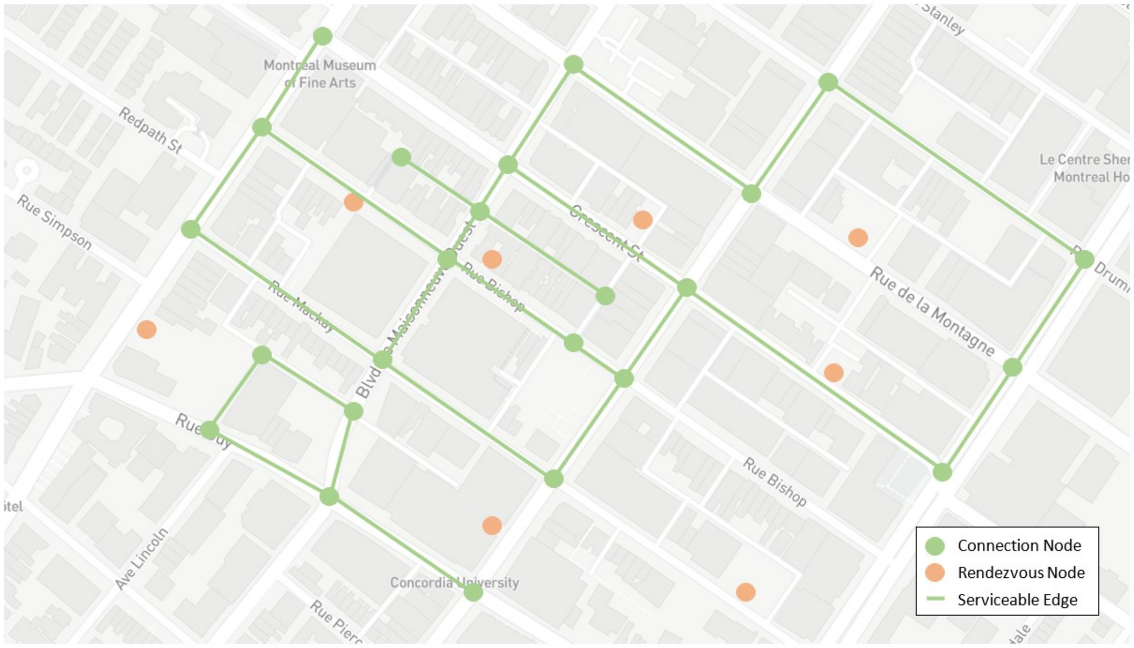


Figure 3.1: Example of road network as a potential application for the problem

For this problem we consider road networks similar to what is shown in Figure 3.1. The connection nodes are defined on traffic intersections while rendezvous points are placed on empty parking lots. The GV travels only between rendezvous points and must travel on the road. Therefore, time taken to travel between nodes can be calculated as number of time periods taken to travel the shortest distance on the road between the nodes rather than the Euclidean distance. As the GV does not perform services along edges the route taken to reach the rendezvous point is irrelevant. Drones can directly fly between connection points to start servicing edges and time for travel is calculated using the Euclidean distance.

3.2 Mathematical Model

3.2.1 Objective Function

The MDTARP can be interpreted as a multi-objective problem. Although weighted edge traversal is the key goal of the problem, it is desirable to restrict unnecessary movement of the drone and truck to preserve battery life and save on cost of fuel. Therefore the following three conflicting objectives are optimized:

1. Maximize traversal of weighted edges by drones based on priority.
2. Minimize drone travel when not traversing weighted edges, i.e., travel between edges and travel to and from the GV.
3. Minimize total travel by the GV.

This leads to a conflict of objectives as optimizing objective 1 would compromise both objectives 2 and 3. Maximizing edge coverage would require more movement by the truck to allow the drone to reach more edges or more traversal of unweighted edges to allow the drone more flexibility to move between weighted edges. In the model defined, the three objective functions have been combined into a single scalar objective function using a weighted sum method. Coefficients α, β and γ have been given to each of the objectives functions 1, 2 and 3, respectively, where $\alpha + \beta + \gamma = 1$. The value of these coefficients represent the importance of different parts of the objective function and impacts the solution to the problem, therefore, proper selection of weight coefficients is important. Ultimately, it is up to the decision maker to select the coefficients for the model depending on the priority of each objective.

3.2.2 MIP Formulation

Let $G = (V, E)$ be an undirected and connected graph. $E = S_1 \cup S_2$ where S_1 is the set of required edges to be serviced and $S_1 \cap S_2 = 0$. Rendezvous nodes, $V_r \subset V$, are points where the GV is able to visit and where drones are able to take-off and land. Connection nodes, $V_c \subset V$, are used by drones to travel between edges and are not accessible by the GV, therefore $V_c \cap V_r = 0$. Normally, G is a fully connected graph, although there are some real applications where the graph is not fully connected due to potential restricted zones (e.g. airports, very tall buildings, etc.). Let U be the set of drones available and $T = \{0, 1, 2, ..., T_m\}$ be the set of discrete time periods where T_m is the time horizon.

Parameters

a_{ij}	number of time periods required for GV to travel from i to j
b_{ij}	number of time periods required for drone to travel from i to j
c_{ij}	cost of GV to travel from i to j
w_{ij}	weight/priority to service edge (i, j)
t_u	battery life of drone u in terms of time periods
t_r	maximum number of time periods before a drone may return to an edge

Decision Variables

x^u_{tij}	1 if drone u starts to service edge (i, j) at time t, 0 otherwise
y^u_{ti}	1 if drone u takes off from vertex i at time t, 0 otherwise
v^u_{tj}	1 if drone u lands at vertex j at time t, 0 otherwise
z_{tij}	1 if GV starts travelling from vertex i to j at time t, 0 otherwise

Model

Maximize

$$\alpha \times \sum_{u\in U}\sum_{t\in T}\sum_{(i,j)\in S_1} w_{ij}x^u_{tij} - \beta \times \sum_{u\in U}\sum_{t\in T}\sum_{(i,j)\in S_2} b_{ij}x^u_{tij} - \gamma \times \sum_{t\in T}\sum_{i\in V_r}\sum_{j\in V_r} c_{ij}a_{ij}z_{tij} \tag{1}$$

Subject to

$$\sum_{j\in V_r} z_{00j} = \sum_{i\in V_r} z_{T_m - a_{i0},i0} = 1 \tag{2}$$

$$\sum_{i\in V}\sum_{j\in V} z_{tij} \leq 1 \qquad \forall t \in T \tag{3}$$

$$\sum_{i\in V_r : t-a_{ij}\geq 0} z_{t-a_{ij},ij} = \sum_{k\in V_r} z_{tjk} \qquad \forall t \in T : t \neq 0, j \in V_r \tag{4}$$

$$y^u_{ti} \leq \sum_{j\in V_r} z_{tij} \qquad \forall u \in U, t \in T, i \in V_r \tag{5}$$

$$\sum_{u\in U} x^u_{tij} \leq 1 \qquad \forall t \in T, (i, j) \in E \tag{6}$$

$$\sum_{(i,j)\in E} x^u_{tij} \leq 1 \qquad \forall u \in U, t \in T \tag{7}$$

$$\sum_{(i,j)\in E} x^u_{t-b_{ij},ij} + y^u_{tj} = \sum_{(j,k)\in E} x^u_{tjk} + v^u_{tj} \qquad \forall u \in U, t \in T : t - b_{ij} \geq 0, j \in V \tag{8}$$

$$y^u_{ti} = v^u_{ti} = 0 \qquad \forall u \in U, t \in T, i \in V : i \notin V_r \tag{9}$$

$$\sum_{(i,j)\in E} \sum_{t'=t}^{min(t+b_{kl},T_m)} x^u_{t'ij} \leq x^u_{tkl} + M(1 - x^u_{tkl}) \qquad \forall u \in U, t \in T, (k,l) \in E \tag{10}$$

$$\sum_{t'=t}^{min(t+t_r,T_m)} \sum_{u\in U} x^u_{t'ij} \leq 1 \qquad \forall t \in T, (i,j) \in S_1 \tag{11}$$

$$v^u_{ti} \leq \sum_{j\in V_r} z_{tij} \qquad \forall u \in U, t \in T, i \in V_r \tag{12}$$

$$\sum_{t'=t}^{min(t+t_u,T_m)} \sum_{i\in V_r} v^u_{t'i} \geq y^u_{tj} \qquad \forall u \in U, t \in T, j \in V_r \tag{13}$$

$$\sum_{t\in T} \sum_{i\in V_r} y^u_{ti} = \sum_{t\in T} \sum_{j\in V_r} v^u_{tj} \qquad \forall u \in U \tag{14}$$

$$x^u_{tij}, y^u_{tk}, v^u_{tk}, z_{tkl} \in \{0,1\} \qquad \forall u \in U, t \in T, (i,j) \in E, k \in V, l \in V \tag{15}$$

Constraint (2) forces the GV to start at the depot at time 0 and end at the depot at the end of the time horizon, T_m. Constraints (3) and (4) ensures GV traverses a single edge at any given time and completes the traversal without any jumps. Constraints (5) ensures drones take off only when GV has reached a rendezvous point. Constraints (6) forces each edge to be watched by exactly one drone at a time. These restrictions guarantee solutions with wider coverage by drones, and helps avoid collisions in the air. Conditions (7)) specify the surveillance capabilities (an edge at a time). Conditions (8) ensures flow conservation while also accommodating for landings, takeoffs and transportation of drones between rendezvous points. Constraints (9) ensures drone can only take off and land at rendezvous points. Constraints (10) assure a correct routing for drones through the edges by not allowing jumps to other edges. Constraints (11) ensures an edge is not traversed by drones more than once before a certain period of time, t_r, has passed. Constraints (12) ensures a GV is available at the rendezvous point to which a drone is returning. Constraints (13) ensures each drone trip is completed before its battery is depleted. Constraints (14) ensures all drones have returned before the end of the time horizon.

Chapter 4

Solution Algorithms

This chapter provides a detailed description of the Iterated Local Search (ILS) heuristic and the Cluster-based Location Search (CLS) heuristic that were developed for this thesis. We start in Section 4.1 by providing the setup of the heuristics describing how the solutions are represented within the algorithms and how the problem is decomposed into sub-problems to aid the solution search. In Section 4.2 and Section 4.3 we provide a detailed description of the operations of the two algorithms.

The heuristics construct tours for the drones and GV separately. GV routes are constructed first in order to ensure synchronisation. Drone routes are then created based on the GV route. Both the heuristics differ in the construction of the GV routes while they use the same method to construct the drone routes. The initial GV route is created randomly for both heuristics. For any given GV route, drone routes are constructed using a rolling horizon approach, described in Section 4.1.3, and a mathematical model is solved using CPLEX to find the best points to take-off and land based on the GV route that maximises the original objective function while keeping in consideration travel constraints. Once the best drones routes are determined, the GV route is optimized so that the GV only visits nodes where the drones take off or land.

ILS generates GV routes by allowing the GV to travel randomly between any of the rendezvous nodes throughout the time-horizon. At each iteration, a new route is generated and a local search procedure takes place. The local search involves iteratively replacing nodes in the route and generating drones routes for each new GV route. A Simulated Annealing (SA) approach is introduced where improving solutions are always accepted and non-improving solutions are accepted based on a probability. This allows the solution to perform hill climbing moves and avoid local optima. After a termination criteria is met, the heuristic starts a new iteration with a new initial GV route. Performing a number of restarts allows for diversification in the solutions being explored. The heuristic ends after a certain number of restarts (specified by the user) have completed.

CLS constructs GV routes such that the GV travels randomly only between a set of available rendezvous nodes. Starting with only the depot, new rendezvous nodes are made available at each iteration. The heuristic terminates after all rendezvous nodes have been made available for the GV to visit. Exploring only a subset of nodes at a time can be effective as it is likely that not all rendezvous nodes are used in the optimal solution. The nodes to be made available at each iteration is selected using a cluster based method to identify nodes closer to weighted edges which are more likely to be used. At each iteration of the heuristic, random GV routes and associated drone routes are generated using the available set of rendezvous nodes for a number of iterations until a certain number of subsequent iterations (specified by the user) results in non-improving solutions.

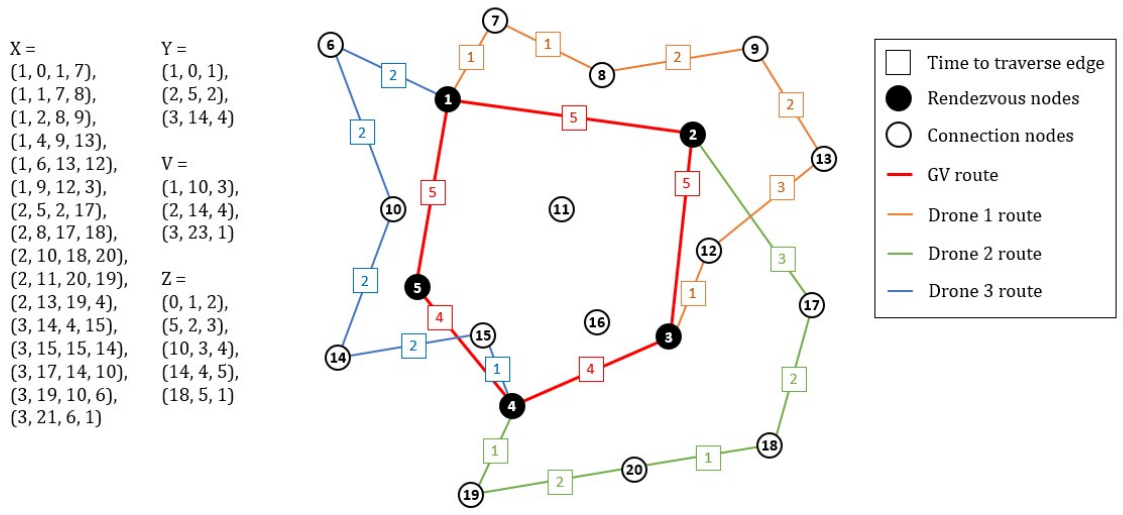
4.1 Heuristic setup

4.1.1 Solution representation

The solution algorithms represent the solutions in a straight forward manner keeping them in the form similar to that defined in the mathematical model. The heuristics output the GV route, drone routes, drone take-offs and drone landings denoted as Z, X, Y and V respectively.

The algorithm directly manipulates the candidate GV route, Z, represented as a ordered sequence of tuples, $(t_1, i_1, i_2), (t_2, i_2, i_3), ..., (t_n, i_n, i_{n+1})$, indicating that the truck travels from i to j at time period t. The length of Z is indicated by n where $1 \leq n \leq |T|$. The length varies depending on the nodes visited and travel times and is equal to $|T|$ in the event where the truck does not move or only travels between nodes that require a travel time of 1 period. All truck routes must start and end at the depot.

The heuristics manipulate a four dimensional binary matrix, denoted as X', of size $|U| \times |T| \times |V| \times |V|$. Element X'_{utij} is 1 if the drone u starts the traversal of edge (i, j) at time t and 0 otherwise. The use of a matrix allows for indexing and slicing which is utilized in the rolling horizon approach described in Section 4.1.3. Once optimal routes are determined, the matrix is converted to a sequence of tuples, X, in the form $\{(u_1, t_1, i_1, i_2), (u_2, t_2, i_2, i_3), ..., (u_n, t_n, i_n, i_{n+1})\}$ for the final solution. Drone takeoffs, Y, and landings, V, are similarly represented in the form $(u_1, t_1, i_1), (u_2, t_2, i_2), ..., (u_n, t_n, i_n)$.


X =
(1, 0, 1, 7),
(1, 1, 7, 8),
(1, 2, 8, 9),
(1, 4, 9, 13),
(1, 6, 13, 12),
(1, 9, 12, 3),
(2, 5, 2, 17),
(2, 8, 17, 18),
(2, 10, 18, 20),
(2, 11, 20, 19),
(2, 13, 19, 4),
(3, 14, 4, 15),
(3, 15, 15, 14),
(3, 17, 14, 10),
(3, 19, 10, 6),
(3, 21, 6, 1)

Y =
(1, 0, 1),
(2, 5, 2),
(3, 14, 4)

V =
(1, 10, 3),
(2, 14, 4),
(3, 23, 1)

Z =
(0, 1, 2),
(5, 2, 3),
(10, 3, 4),
(14, 4, 5),
(18, 5, 1)

Figure 4.1: Simple candidate solution as represented by the algorithms

4.1.2 Truck route

Initial GV routes are generated randomly for both heuristics. Each route is constructed such that the GV starts at the depot, travels randomly between the available rendezvous and returns to the depot before the end of the time-horizon. The GV must respect the time taken to travel between nodes during its travel and may remain at a certain location over a period of time. The TruckRoute algorithm is used to construct initial GV routes. The algorithm takes as input the set of available rendezvous nodes that the GV is allowed to visit and proceeds as follows:

Algorithm 1

```
function TruckRoute ($V_a$)
    Initialize:
      $Z \leftarrow \emptyset$;                    ▷ Empty set for truck route
      $t \leftarrow 0$; $i \leftarrow depot$;
    while $t \leq T_m$ do
        $j \leftarrow$ random node in $V_a$
        if $t + a_{ij} + a_{j,depot} > T_m$ then
            $j \leftarrow$ depot
        end if
        add $(t, i, j)$ to $Z$
        $t \leftarrow t + a_{ij}$
        $i \leftarrow j$
    end while
    return $Z$
end function
```

The GV route is then used in different ways, depending on the heuristic, to construct drone routes. Once the drone routes are determined and the drones' take-off and landing times and locations are known, the GV routes are optimized such that the GV only visits nodes that are used by the drones. The GV visits nodes in the order in which they are used and waits at the location until drones either launch or land before travelling to the next node. The OptimizeTruckRoute algorithm is used to optimize GV routes. The algorithm takes Y and V from the drone routes as input and concatenates them to produce the sequence S. Additionally, a stop is added to S to ensure the GV returns to the depot at the end of the time horizon. S is then sorted in ascending order according to the time variable to ensure the GV visits both take-off and landing locations in order. The algorithm proceeds as follows:

Algorithm 2

```
function OPTIMIZETRUCKROUTE ($Y, V$)
    Initialize:
        $Z \leftarrow \emptyset$;                                  ▷ Empty set for truck route
        $S \leftarrow Y \cup V \cup [(0, T_m, depot)]$ sorted in ascending order by time;
        $t \leftarrow 0$; $c \leftarrow 0$; $i \leftarrow$ depot;
    while $t \leq T_m$ do
        $time \leftarrow$ time at $c^{th}$ stop in $S$
        $j \leftarrow$ node at $c^{th}$ stop in $S$
        add $(t, i, j)$ to $Z$
        $t \leftarrow t + a_{ij}$
        $i \leftarrow j$
        if $t = time$ then
            $c$++
        end if
    end while
    return $Z$
end function
```

4.1.3 Rolling time horizon approach for drone routes

In order to improve solve times for large scale instances, the problem is decomposed and solved in a shorter time interval. We will refer to this time interval as a time window. The time window is shifted across the time horizon for each available drone and a subproblem is solved to find the optimal route for a drone within the time window. Drones must take-off and land within the window or may choose to remain on the truck, therefore the length of each window can only be as long as the battery life of the drone, t_u. DroneRoute algorithm is used to construct drone routes using Model 2. The GV route, provided by the heuristic in concern, is used by the algorithm to determine the best route a particular drone can take within each time-window at every iteration, taking off and landing anywhere the GV has visited.

Let $T_{w_n} = \{t_{s_n}, t_{s_n}+1, t_{s_n}+2, ..., min(t_{s_n}+t_u,\ T_m)\}$ be the the n^{th} time-window where t_{s_n} is the start time at each iteration and $Z_{w_n} = \{(t,i,j) \mid t_{s_n} \leq t \leq min(t_{s_n}+t_u,\ T_m)\ \forall\ t,i,j \in Z\}$ be a portion of the GV route, Z, between time periods t_{s_n} and $t_{s_n}+t_u$. Let t_{dy_n} be the take-off time of the drone and t_{dv_n} be the landing time of the drone for the n^{th} time window obtained from the subproblem. The time window is shifted iteratively until the end of the time horizon. The start time of each n^{th} time window depends on the values of $t_{dy_{n-1}}$ and $t_{dv_{n-1}}$. Typically, each time window begins at the end of the drone route of the previous time window, i.e., $t_{s_n} = t_{dv_{n-1}}+1$. However, since the model allows the drone to take-off at anytime but forces the drone to land before the end of time window, it is possible that the best route for the drone is not found. The drone may land in order to satisfy the time constraint rather that continue to traverse more egdes. Therefore, if in the n^{th} time window the drone takes off at anytime after t_{s_n} (i.e., $t_{dy_n} \neq t_{s_n}$), in the next iteration the time window $T_{w_{n+1}}$ shifts such that $t_{s_{n+1}} = t_{dy_n}$ and the model is once again used to determine the best route. In the following iteration the time window $(T_{w_{n+2}})$ shifts so that $t_{s_{n+2}} = t_{dv_{n+1}}+1$. Since the model allows drones to remain on the GV during the time-window, an empty solution can be returned in which case in the following iteration $t_{s_{n+1}} = t_{s_n} + 1$. The time-window continues to shift across the entirety of the time-horizon until $t_{s_n} = T_m$. The entire process repeats for each drone in the set U. The algorithm can therefore be written as follows:

Algorithm 3

```
function DroneRoute ($Z$)
    $X \leftarrow \emptyset$; $Y \leftarrow \emptyset$; $V \leftarrow \emptyset$;                ▷ Empty set for drone route
    for $u = 1$ to $|U|$ do
        $t_s \leftarrow 0$;
        while $t_s < T_m$ do
            $T_w \leftarrow \{t_s,\ t_s + 1,\ t_s + 2,\ ...,\ min(t_s + t_u, T_m)\}$
            $Z_w \leftarrow \{(t, i, j) \mid t_s \leq t \leq min(t_s + t_u,\ T_m)\ \forall\ t, i, j \in Z\}$
            $x', y', v' \leftarrow$ Model 2 solution with $T_w$ and $Z_w$
            $t_{dy} \leftarrow$ time component from $y'$
            $t_{dv} \leftarrow$ time component from $v'$
            if $x'$ is not empty then
                if $t_s = t_{dy}$ then
                    add $x'$ to $X$; add $y'$ to $Y$; add $v'$ to $V$
                    $t_s \leftarrow t_{dv} + 1$
                else
                    $t_s \leftarrow t_{dy}$
                end if
            else
                $t_s$++
            end if
        end while
    end for
    return $X, Y, V$
end function
```

Let $G = (V, E)$ be a connected and undirected graph where V and E are vertices and edges. $E = S_1 \cup S_2$ where S_1 is the set of required edges to be serviced and $S_1 \cap S_2 = 0$. T_w is the sequence of discrete time periods in the time-window and Z_w is a portion of a GV route Z corresponding to the time-window. The parameter X' is created using the drone route, X, from previous iterations in order to pass route information between time windows and other drones. The model used to determine the drone route can be defined as follows:

Parameters

b_{ij}	number of time periods required for drone to travel from i to j
w_{ij}	weight/priority to service edge (i, j)
t_r	maximum number of time periods before a drone may return to an edge
X'_{tij}	1 if a drone has previously traversed edge (i, j) at time t, 0 otherwise

Decision Variables

${x'}_{tij}$	1 if the drone starts to service edge (i,j) at time t, 0 otherwise
${y'}_{ti}$	1 if the drone takes off from vertex i at time t, 0 otherwise
${v'}_{tj}$	1 if the drone lands at vertex j at time t, 0 otherwise

Model 2

Maximize

$$\alpha \times \sum_{t\in T_w} \sum_{(i,j)\in S_1} w_{ij}{x'}_{tij} - \beta \times \sum_{t\in T_w} \sum_{(i,j)\in S_2} b_{ij}{x'}_{tij} \tag{1}$$

Subject to

$$\sum_{t\in T_w} \sum_{i\in V} y'_{ti} \leq 1 \tag{2}$$

$$\sum_{t\in T_w} \sum_{i\in V} v'_{ti} = \sum_{t\in T_w} \sum_{i\in V} y'_{ti} \tag{3}$$

$$\sum_{(i,j)\in E} x'_{t-b_{ij},ij} + y'_{tj} = \sum_{(j,k)\in E} x'_{tjk} + v'_{tj} \qquad \forall t \in T_w : t - b_{ij} \geq 0, j \in V \tag{4}$$

$$y'_{ti} = 0 \qquad \forall t \in T_w, i \in V : (t,i,j) \notin Z_w \ \forall j \in V_r \tag{5}$$

$$v'_{ti} = 0 \qquad \forall t \in T_w, i \in V : (t,i,j) \notin Z_w \ \forall j \in V_r \tag{6}$$

$$\sum_{t'=t_s}^{t} x'_{t'ij} \leq 1 - \lambda \qquad \lambda = \begin{cases} 1, & \text{if } \sum_{t'=min(t-t_r,0)}^{t} X'_{t'ij} > 0 \\ 0, & \text{otherwise} \end{cases} \qquad \forall t \in T_w, (i,j) \in S_1 \tag{7}$$

$$x'_{tij} \leq 1 - \mu \qquad \mu = \begin{cases} 1, & \text{if } X'_{tij} > 0 \\ 0, & \text{otherwise} \end{cases} \qquad \forall t \in T_w, (i,j) \in E \tag{8}$$

$$x'_{tij}, y'_{tk}, , v'_{tk}, \in \{0,1\} \qquad \forall t \in T_w, (i,j) \in E, k \in V \tag{9}$$

The objective function (1) maximises the sum of weights of edges to be serviced and minimises the time spent traversing all other edges. Constraint (2) ensures the drone takes off only once at any given time but also allows the drone to remain on the GV throughout the time period. Constraint (3) ensures the drone only lands if it has taken off. Constraints (4) ensure flow conservation once the drone has taken off and until it has landed. Constraints (5) and (6) ensure the drone takes off and lands only when and where the GV is available to ensure synchronisation of the vehicles. Constraints (7) ensure the drone does not return to an edge before the specified time period has elapsed. X' is used here to pass route information between each time window. This allows the model to take into account edges that have been traversed in previous time windows or by other drones when trying to determining whether a drone can traverse an edge or return to an edge. X'_{tij} is used in the model as a binary parameter that is equal to 1 if any of the drones have previously traversed edge (i, j) in the past t_r time periods. The value for λ can be determined by calculating $\sum_{t'=min(t-t_r,0)}^{t} X_{t'ij}$ for all edges in $S1$ and time periods in T. If the value of summation is greater than 0, edge (i, j) must have been traversed in the past t_r periods and therefore λ is equal 1. Constraints (8) ensure only one drone traverses an edge at a time.

Figure 4.2 illustrates the rolling horizon process to create drone routes on an instance with $T_m = 20$ and $t_u = 5$ involving a single drone. The time window starts at period 1 and has a length of 5 periods ($t_{s_1} = 1$, $T_{w_1} = \{1, 2, 3, 4, 5\}$). A route is found where the drone takes off at period 1 and lands at period 5 ($t_{dy_1} = 1, t_{dv_1} = 5$). Because $t_{dy_1} = t_{s_1}$, the window shifts to period 6 ($t_{s_2} = t_{dv_1} + 1$, $T_{w_2} = \{6, 7, 8, 9, 10\}$) where the drone takes off ($t_{dy_2} = 6$) and lands at period 8 ($t_{dv_2} = 8$). Since the drone does not land at the end of the window, the time window shifts to period 9 ($t_{s_3} = t_{dv_2} + 1$, $T_{w_3} = \{9, 10, 11, 12, 13\}$), after the drone has landed. The drone route found in this window starts at period 10 rather than the start of the window ($t_{dy_3} \neq t_{s_3}$) and lands short of its battery life at the end of the window. Consequently, the window shifts to period 10 ($t_{s_4} = t_{dy_3}$) to determine if a better solution can be found. Once a route is found, window shifts to the end of the route at period 15 ($t_{s_5} = t_{dv_4} + 1$). If no route is found, the window shift to the next period ($t_{s_6} = t_{s_5} + 1$).

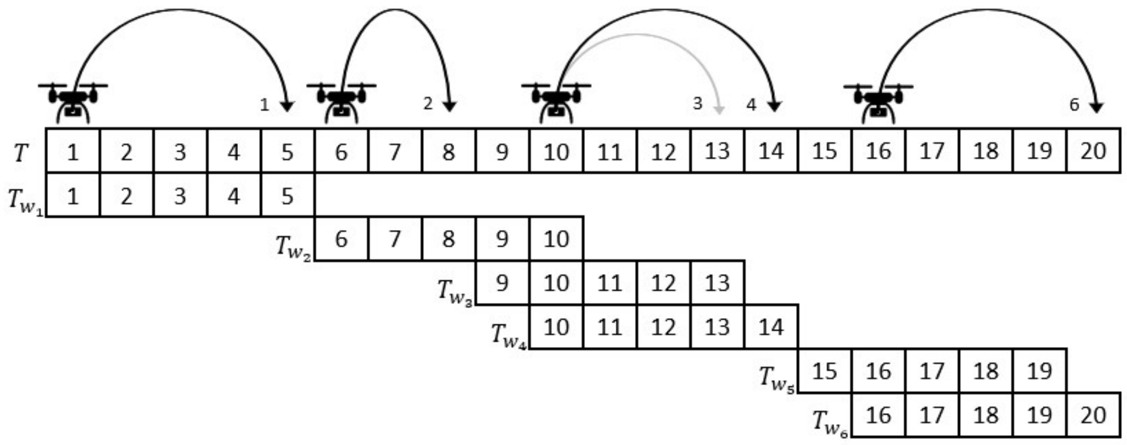


Figure 4.2: Rolling horizon approach

4.2 Iterated Local Search

ILS uses a local search and SA process to transition to neighborhood solutions. The SA process allows for diversification within the solution and helps escape local optima by accepting non-improving solutions. ILS proceeds as follows:

Algorithm 4 Iterated Local Search Heuristic

```
Initialize:
    $V_a \leftarrow V_r$
    $I \leftarrow$ Number of iterations specified by user
for $i = 1$ to $I$ do
    $Z \leftarrow TruckRoute(V_a)$
    while local search termination criteria not met do
        $X, Y, V \leftarrow DroneRoute(Z)$
        $Z_o \leftarrow OptimizeTruckRoute(Y, V)$
        if $f(X_i, Z_i) \leq f(X, Z_o)$ then
            $Z' \leftarrow Z$
            $Z_i \leftarrow Z_o$, $X_i \leftarrow X$, $Y_i \leftarrow Y$, $V_i \leftarrow V$
        else
            $\mathrm{P}(Z' = Z) \leftarrow Pr$
        end if
        $Z \leftarrow NeighborRoute(Z')$
    end while
end for
```

At each iteration of the local search, the updates to the solution are made using a SA procedure. If an improvement is not made, the solution is accepted with a probability, Pr, otherwise the previous GV route is used in the local search. The incumbent is always saved in memory.

4.2.1 Local Search

Local search algorithms explore the neighborhood of the current solution to optimize the objective function. It begins its search from a random feasible solution and iteratively applies a generation mechanism to find better solutions. The current solution is iteratively updated if a better solution is found and the process ends when no improvements can be found. The heuristic performs local search by manipulating the GV route and replacing a random stop on the route by another random point from the set V_r. Because of the time component and synchronization between the GV and drones, altering one point on the GV route can have an affect on all subsequent stops. NeighborRoute is used to generate a GV route in the neighborhood of a given route. The algorithm takes a route Z replaces a stop and adjusts the time component in the subsequent stops to ensure travel times are respected as follows:

Algorithm 5

```
function NeighborRoute($Z$)
    $n \leftarrow$ index of random stop in $Z$
    $i \leftarrow$ current position at $n^{th}$ stop on $Z$
    $j \leftarrow$ random node from set $V_r$
    $t \leftarrow$ time at $n^{th}$ stop on $Z$
    Replace $n^{th}$ stop on $Z$ with $(t, i, j)$
    while $t \leq T_m$ do
        $n$++
        $t \leftarrow t + a_{ij}$
        $i \leftarrow j$
        $j \leftarrow$ current position at $(n+1)^{th}$ stop on $Z$
        if $t + a_{i,j} + a_{j,depot} \geq T_m$ then
            $j \leftarrow$ depot
        end if
        Replace $n^{th}$ stop on $Z$ with $(t, i, j)$
    end while
    return $Z$
end function
```

Figure 4.3 shows possible routes in a problem with a time horizon of 10 periods. The initial truck route, Z_0, is altered by removing the stop at node 2 and replacing it with a stop at node 1. Due to the travel time difference, the total time required to complete the tour changes from 9 periods to 10 periods in Z_1. The next alteration is made by removing node 5 and replacing it with node 7 for Z_2. However this changes the total tour time from 10 periods to 12 periods and is therefore an infeasible solution as the truck must return to the base before the end of the time horizon of 10 periods. This new change is, therefore, not accepted and node 7 is removed altogether from the route resulting in a feasible solution with a tour completed within the time horizon. Each time the route is altered new drones routes are constructed and the GV route is optimized.

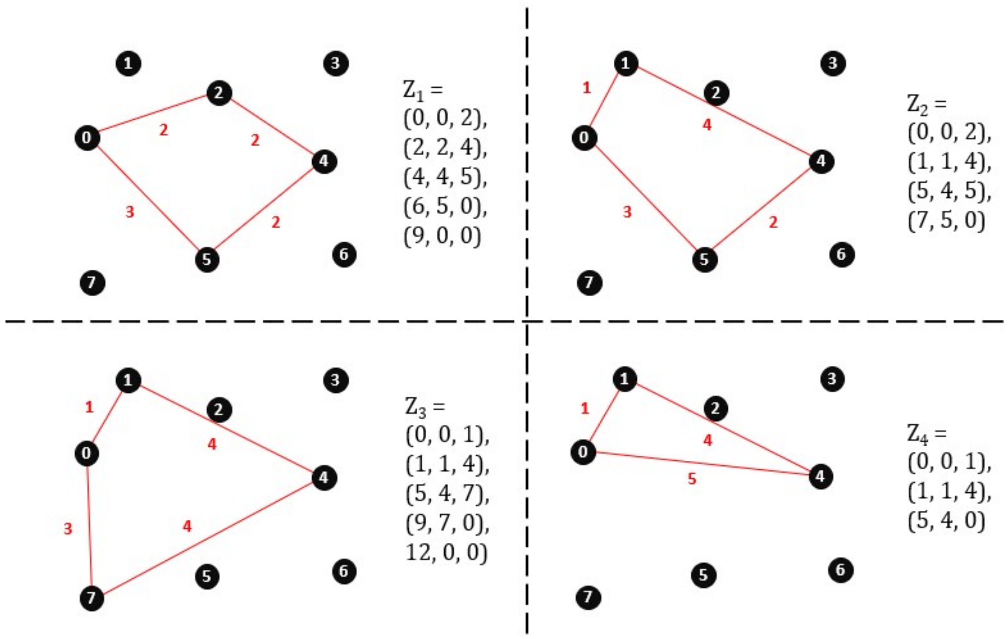


Figure 4.3: Initial GV route (top-left) and candidate neighborhood solutions

4.2.2 Simulated Annealing

The local search often converges to a local optima. Being trapped in a local optima can be avoided if a process exists where non-improving solutions are accepted and updated as the current solution. SA aims to achieve this by accepting non-improving solutions based on the following probability:

$$Pr = \begin{cases} 1, & \text{if } f(j) \leq f(i) \\ e^{\left(-\frac{f(i)-f(j)}{c}\right)}, & \text{otherwise} \end{cases} \tag{1}$$

where $f(i)$ is the current solution, $f(j)$ a neighboring solution in a minimization problem and c is the temperature parameter. New neighborhood solutions are generated using the local search procedure described. The SA process begins with a high value for c which allows for transitions to solutions with high objective degradation and consequently thorough exploration of the neighborhood. With each iteration c is decreased and solutions with lower objective degradation are accepted until c tends to zero where transitions are only made if the solution is an improvement. The manner in which the temperature parameter is decrease is known as the cooling schedule. A geometric cooling schedule is used for the heuristic, where at each iteration $c = c \times k$, where $0 < k < 1$. The SA process runs while $c \leq c_f$ where $c_f > 0$. After the stopping criteria is satisfied, a restart occurs and the process starts again with a new GV route. The values for k and c_f are selected such that the local search is able to sufficiently diversify the solution to escape local optima.

Initial temperature, c_0, is selected to be large enough so that at the first iterations nearly all neighborhood solutions are accepted. This value can be found by using the following equation:

$$c_0 = \frac{-\Delta f_{avg}}{\ln Pr} \tag{2}$$

where Pr can be selected to be close to 1 and $-\Delta f_{avg}$ is the average difference in solutions that strictly improve the value of the objective function. A prior step can be performed where the local search process is run for a number of iterations to obtain set of solution values that can be used to obtain the average increase, $-\Delta f_{avg}$. The initial temperature value obtained from this step can then be used in subsequent iterations of the SA process.

4.3 Cluster-based Locations Search

Not all rendezvous nodes may be necessary for the GV to visit during the course of its trip. Some nodes, possibly those closer to edges in $S1$, may be preferable to visit as they may be used more frequently by drones to take-off and land. Therefore, unlike in Heuristic 1 where all rendezvous nodes are used to create GV routes, CLS iteratively makes nodes available for the GV to travel to. The heuristic begins by creating a trip for the ground vehicle (GV). We remind that $V_a \subseteq V_r$ is a set of rendezvous nodes available for the GV to visit. Initially $V_a = \{0\}$, therefore the GV remains at the depot throughout the time horizon. The heuristic then proceeds as follows:

Algorithm 6 Cluster-based Location Search Heuristic

```
Initialize:
  $V_a \leftarrow \{0\}$
  $I \leftarrow$ Number of iterations specified by user
**while** $V_a \neq V_r$ **do**
    **while** *consecutive non-improving solutions* $\neq I$ **do**
        $Z \leftarrow TruckRoute(V_a)$
        $X, Y, V \leftarrow DroneRoute(Z)$
        $Z_o \leftarrow OptimizeTruckRoute(Y, V)$
        **if** $f(X_i, Z_i) \leq f(X, Z_o)$ **then**
            $Z_i \leftarrow Z_o$, $X_i \leftarrow X$, $Y_i \leftarrow Y$, $V_i \leftarrow V$
        **end if**
    **end while**
    add new node to $V_a$
**end while**
```

The heuristic creates several random initial GV routes for each set of available available nodes allowing for a wider array of take-off and landing times for the drones. Rendezvous nodes are made available based on their proximity to the edges to be serviced as, logically, it is more likely these locations would be used more frequently by drones. The problem of selecting rendezvous nodes based on distance to edges can be identified as a p-median problem which was shown by Kariv and Hakimi to be $\mathcal{NP}$-hard [58]. For this reason, instead of directly solving a p-median problem we use a cluster based method, described in Section 4.3.1, to select rendezvous nodes that will be made available.

4.3.1 Node Selection

The k-means clustering algorithm, first proposed by MacQueen in 1967 [59], is one of the simplest unsupervised learning algorithms that solve the clustering problem. The algorithm is used to classify n observations into k clusters, where k is fixed a priori. This is done through an iterative process where at each step refinements are made to improve clusters. The first step defines k centroids and assigns each of the n points to the nearest centroid to create the first set of clusters. The next step defines k new centroids by calculating the center of each cluster created in the previous step. Each of the n points are then reassigned to their closest centroid. The algorithm iterative moves centroids and reassigns points until no more movements are made. The algorithm aims to minimize the mean squared distance between each of the n points and their closest centroids. This guarantees that the algorithm converges in a finite number of steps as distances only decrease at each step of the process.

There are several techniques for initializing the k centroids. In the most simple approach, the k centroids are initialized randomly i.e., by picking k instances at random and using their locations as the centroids. This approach, however, may converge to a local optima leading to sub-optimal assignments. In order to obtain better results, the algorithm is run several times using newly initialized centroids and the solution with the lowest mean squared distance is used. K-means++ was proposed by David Arthur and Sergei Vassilvitskii in 2006 [60], as an improvement to the K-means algorithm. They introduced an initialization step where centroids are selected so that they are more distant from one another. They showed that this results in the K-means algorithm being less likely to converge to a sub-optimal solution and drastically reduces the number of times the algorithm needs to be run.

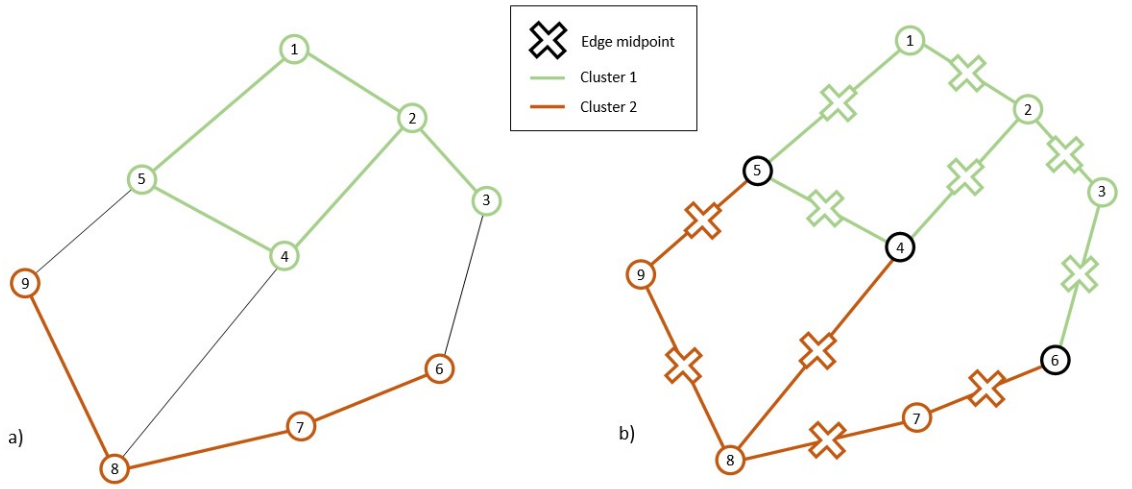


Figure 4.4: Clustering using a) nodes and b) edge midpoints

We use the k-means++ algorithm to cluster edges in $S1$ into k clusters each where $k = |U|$. Figure 4.4 shows two representations of the required edges of a possible graph. The clusters points can be defined in one of two ways: using the nodes at the endpoints of each required edge (Figure 4.4a) or using the midpoints of edges (Figure 4.4b). Using nodes for clustering can lead to ambiguity in how associated edges are clustered. For an edge to be assigned to a certain cluster, nodes at both its endpoints would have to be assigned to the same cluster. If endpoints are at different clusters, it is unclear which cluster the edge belongs to and consequently which drone serves the edge as shown in Figure 4.4a where edge $(5, 9)$, $(4, 8)$ and $(3, 6)$ are left unassigned. Using edge midpoints for clustering alleviates this problem as edges themselves can be assigned to clusters and nodes can be associated with more than one cluster. In Figure 4.4b, nodes 4, 5 and 6 are assigned to both clusters.

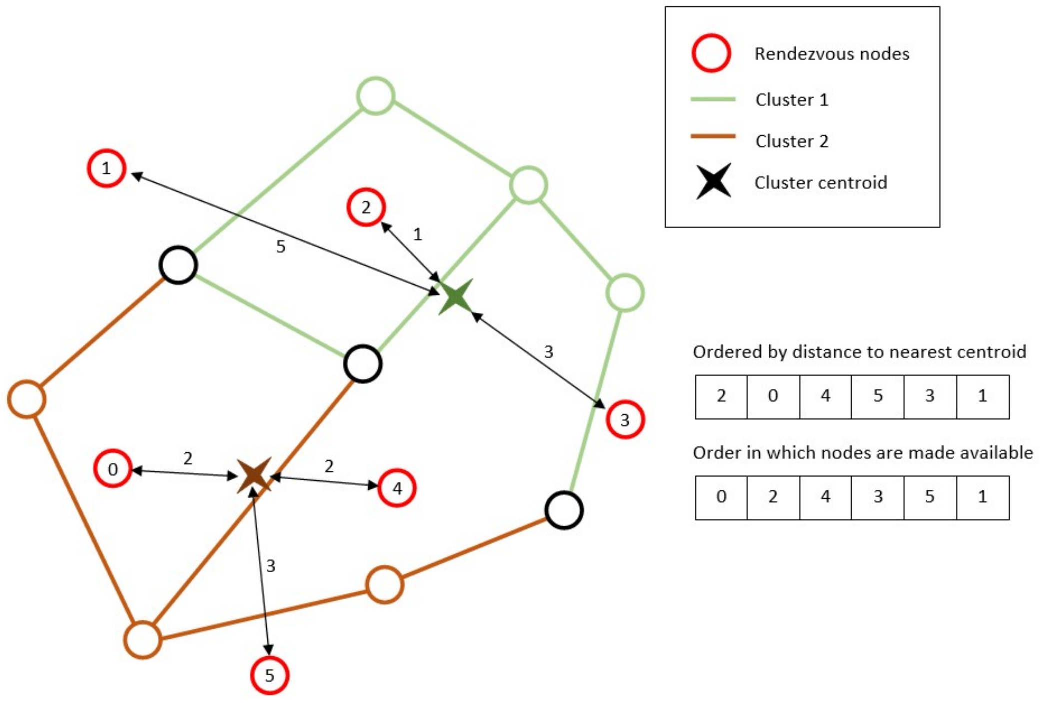


Figure 4.5: Order of rendezvous node availability according to distance from cluster centroids

The K-means clustering algorithm calculates centroids in the process of creating clusters. The centroids from the clusters are used to determine the order in which the rendezvous nodes are made available in each iteration of the heuristic. The euclidean distance between each node and each of the cluster centroids is calculated. The nodes are then ordered from smallest to largest in distance. The depot is always the first available as the GV and all drones must start and end there. Nodes are then added at each iteration in order of shortest distance to a cluster centroid.

Chapter 5

Computational Experiments

In this chapter we present the results of computational experiments to analyze and compare the performance of our proposed formulation solved using CPLEX and the heuristic algorithms. In Section 5.1 we look at how test instances are generated and how time parameters are calculated. Section 5.2 looks at how different weight coefficients affect the objective function and solve times and results obtained from solving different instances. Finally Section 5.3 takes a closer look at each of the heuristics and their performance with different parameters. The algorithms were coded in Python 3.7 language and run on a Dell XPS with an Intel Core i5-9300H CPU processor at 2.40 GHz and 8 GB of RAM under a Windows 11 environment. CPLEX Version 12.10.0.0 is used to solve instances for comparison and to solve subproblems.

5.1 Instance Generation

In order to evaluate the performance of the proposed solution methods, instances were generated to test for different parameters. In order to create instances, we first define the following parameters:

- Size of grid, $n \times m$ in km
- Number of rendezvous (n_r) and connection nodes (n_c)
- Length of time horizon (T_m) in hours
- Number of time periods (n_T)
- Speed of drone (s_d) and speed of ground vehicle (s_g) in km/h

We then proceed as follows:

1. Generate $n_r + n_c$ random coordinates within the grid, where first coordinate is assigned as the depot, the first n_r nodes are assigned as rendezvous nodes and the last n_c nodes are assigned as connection nodes.
2. Define weighted edges manually with each connection node forming at least two serviceable edges. Assign weights by randomly selecting an integer between 1 and 5.

3. Calculate time to travel between each node for drones and GV.

The travel time are calculated based on drone/GV speed, number of time periods and distance between edges. Distance (d_{ij}) is first calculated between each node:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad \forall i, j \in S1 \tag{1}$$

We then compute the number of periods in an hour (hpp) as follows:

$$hpp = \frac{T_m}{n_T} \tag{2}$$

The time taken for GV (a_{ij}) and drones (b_{ij}) is then computed in time periods as:

$$a_{ij} = \begin{cases} 1, & \text{if } i = j \\ \left\lceil \frac{d_{ij}}{s_g \times hpp} \right\rceil, & \text{otherwise} \end{cases} \qquad \forall i \in V_r, j \in V_r \tag{3}$$

$$b_{ij} = \begin{cases} 1, & \text{if } i = j \\ \left\lceil \frac{d_{ij}}{s_d \times hpp} \right\rceil, & \text{otherwise} \end{cases} \qquad \forall (i, j) \in E \tag{4}$$

Finally cost for GV to travel between rendezvous nodes is computed as:

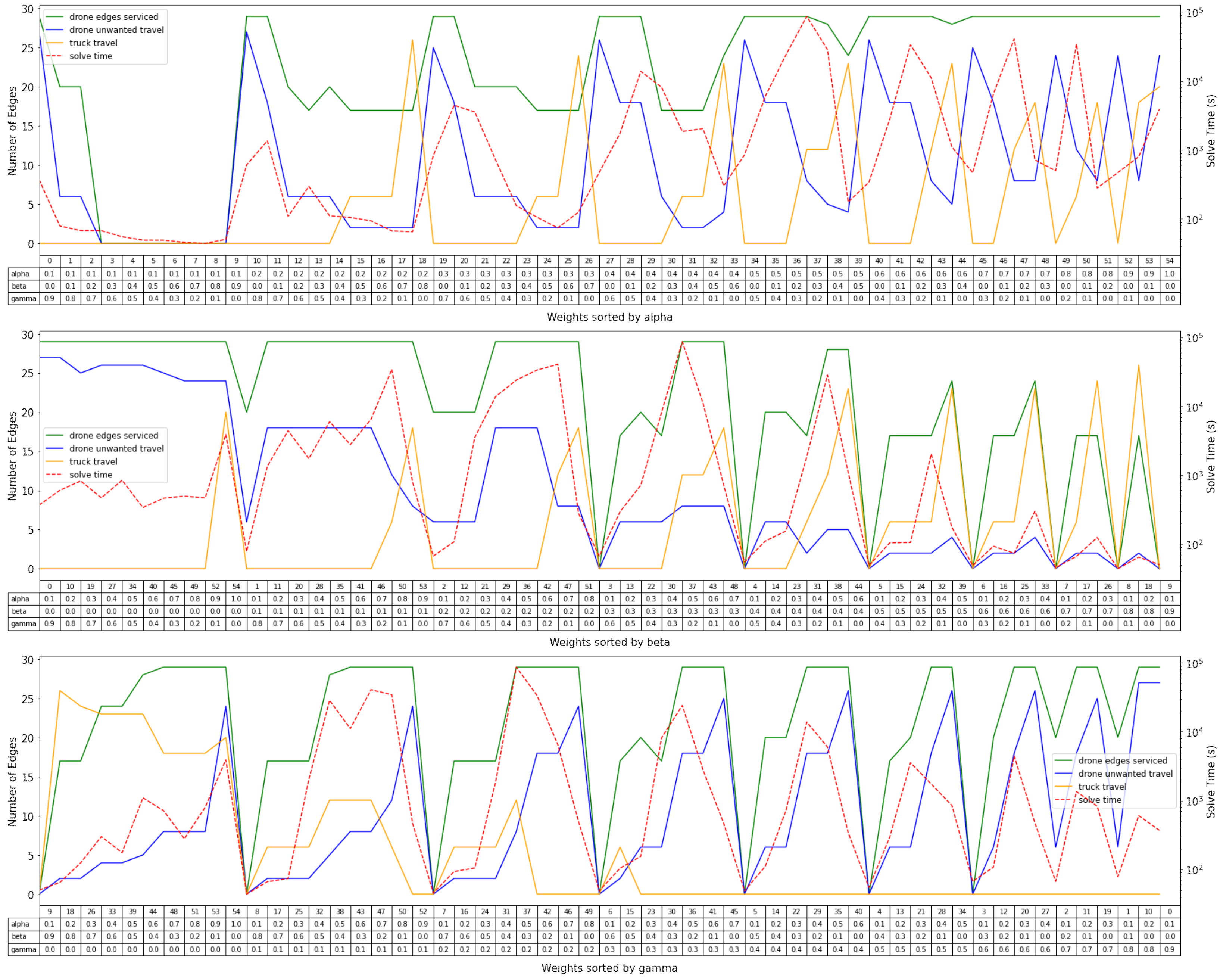
$$c_{ij} = \begin{cases} 0, & \text{if } i = j \\ 1, & \text{otherwise} \end{cases} \qquad \forall i \in V_r, j \in V_r \tag{5}$$

5.2 Computational Results

5.2.1 Weight coefficients

In this section we test how varying weight coefficients for the objective function will affect the final solution. To identify desirable weights, we test instance with varying values for α, β, γ for all $\alpha \in \{0.1, 0.2, ..., 1\}$ and $\beta, \gamma \in \{0, 0.1, 0.2, ..., 1\}$ such that $\alpha + \beta + \gamma = 1$. A total of 55 variations have been tested on instance IN.18 of the dataset detailed in Table 5.2. CPLEX was used to solve the problems for each variation.

Figure 5.1 illustrates the results of the test with further details in Appendix A.1. The number of edges serviced by the drone (objective 1), number of unweighted edges traversed (objective 2) and the number of node visited by the truck (objective 3) is plotted separately on the figure to observe how the different coefficient affect each objective. We can see that solve time is effected significantly by the weights used, varying from 40s up to 1 day of CPU time. For any value of α solve time is minimum when γ is 0 and β takes the maximum value, and solve time is maximum when γ takes the maximum value and β is 0. Increasing γ restricts movement of the truck and therefore hinders the ability for the drone to serve more weighted edges. At $\gamma = 0$, the truck is given no restriction to travel. This will improve edge coverage by the drone but may be expensive in real-world scenarios. Therefore, a relatively low γ value is desirable to improve solve times while still maintaining some restriction to truck travel. At higher values of β, the drone is penalized for traveling between edges or between edges and rendezvous nodes. Therefore, at values of $\beta > 0.4$, servicing of edges is not maximized to avoid penalties for travel to reach those edges.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
alpha	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.4
beta	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.0
gamma	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0	0.6

	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
alpha	0.4	0.4	0.4	0.4	0.4	0.4	0.5	0.5	0.5	0.5	0.5	0.5	0.6	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.7	0.8	0.8	0.8	0.9	0.9	1.0
beta	0.1	0.2	0.3	0.4	0.5	0.6	0.0	0.1	0.2	0.3	0.4	0.5	0.0	0.1	0.2	0.3	0.4	0.0	0.1	0.2	0.3	0.0	0.1	0.2	0.0	0.1	0.0
gamma	0.5	0.4	0.3	0.2	0.1	0.0	0.5	0.4	0.3	0.2	0.1	0.0	0.4	0.3	0.2	0.1	0.0	0.3	0.2	0.1	0.0	0.2	0.1	0.0	0.1	0.0	0.0

Weights sorted by alpha

	0	10	19	27	34	40	45	49	52	54	1	11	20	28	35	41	46	50	53	2	12	21	29	36	42	47	51	3
alpha	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.1
beta	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.3
gamma	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0	0.6

	13	22	30	37	43	48	4	14	23	31	38	44	5	15	24	32	39	6	16	25	33	7	17	26	8	18	9
alpha	0.2	0.3	0.4	0.5	0.6	0.7	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.1	0.2	0.3	0.4	0.1	0.2	0.3	0.1	0.2	0.1
beta	0.3	0.3	0.3	0.3	0.3	0.3	0.4	0.4	0.4	0.4	0.4	0.4	0.5	0.5	0.5	0.5	0.5	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.8	0.8	0.9
gamma	0.5	0.4	0.3	0.2	0.1	0.0	0.5	0.4	0.3	0.2	0.1	0.0	0.4	0.3	0.2	0.1	0.0	0.3	0.2	0.1	0.0	0.2	0.1	0.0	0.1	0.0	0.0

Weights sorted by beta

	9	18	26	33	39	44	48	51	53	54	8	17	25	32	38	43	47	50	52	7	16	24	31	37	42	46	49	6
alpha	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.1
beta	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0	0.6
gamma	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.3

	15	23	30	36	41	45	5	14	22	29	35	40	4	13	21	28	34	3	12	20	27	2	11	19	1	10	0
alpha	0.2	0.3	0.4	0.5	0.6	0.7	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.1	0.2	0.3	0.4	0.1	0.2	0.3	0.1	0.2	0.1
beta	0.5	0.4	0.3	0.2	0.1	0.0	0.5	0.4	0.3	0.2	0.1	0.0	0.4	0.3	0.2	0.1	0.0	0.3	0.2	0.1	0.0	0.2	0.1	0.0	0.1	0.0	0.0
gamma	0.3	0.3	0.3	0.3	0.3	0.3	0.4	0.4	0.4	0.4	0.4	0.4	0.5	0.5	0.5	0.5	0.5	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.8	0.8	0.9

Weights sorted by gamma

Figure 5.1: Affect of weight coefficient on objective function and solve times

5.2.2 Experimental Description

In order to test different scenarios, data is generated with varying parameters. For all instances certain parameters, shown in Table 5.1, are kept constant. GV speed is set to 30km/hr based on common speed limits within city centers and residential areas. Drone speed is set to 20km/hr to ensure better surveillance capabilities at low altitude flight. Objective coefficients are set to $\alpha = 0.6$, $\beta = 0.2$, $\gamma = 0.2$. ILS uses a temperature parameter $c = 20$, and a cooling schedule of $c = c \times 0.9$. The heuristic performs 10 restarts, where at each iteration the SA algorithm terminates when $c \leq 5$. CLS accepts 10 non-improving routes before adding a new rendezvous node to V_a. Each instance is solved using CPLEX with a maximum time limit set to 7,200 seconds (2 hours) of compute time. The instances are run 10 times using each heuristic, and the average solution value is reported in Table 5.2.

Table 5.1: Heuristic parameters used during testing

Parameter	Value
GV speed	30 km/hr
Drone speed	20 km/hr
α	0.6
β	0.2
γ	0.2
Restarts	10
Initial temperature parameter (c)	20
Cooling schedule (x)	0.9
Non-improving solutions	10

It can be observed that CPLEX is unable to provide optimal solutions within 2 hours of computation time even for small instances. Comparatively, the heuristics are capable of providing solutions that are atleast as good within a relatively short period of time. Solution times for all heuristics increase with size of graph, number of time periods and number of drones. We test how each parameter affects solve time and quality of solution obtained by the heuristics and compare then to solutions obtained by CPLEX. The results detailed in Table 5.2 are illustrated in Figure 5.2.

Instances 1-45 keep a constant ratio of time periods per hour at 60/hr, i.e., each time period represents 1 minute. Instances IN.1-IN.15 consider a time horizon and drone battery life at 30 periods and do not allow drones to return to edges. Instances 16-45 increases the time horizon to 1 hour and reduces drone battery life to 20 periods therefore drones will have to return for battery replacements.

Solve times increase with number of connection nodes as shown by instances IN.1-IN.5, that uses a single drone, and IN.16-IN.20, that uses 2 drones, with the same number of rendezvous nodes. This is largely due to the use of the RH algorithm which executes depending on the number of connection nodes, number of drones and time-horizon of the problem. This can be see as the increase in time for instances IN.16-IN.20 is much greater as it uses more drones and a longer time-horizon. As this algorithm is shared amongst all heuristics, its performance governs the overall time performance of each of the heuristics.

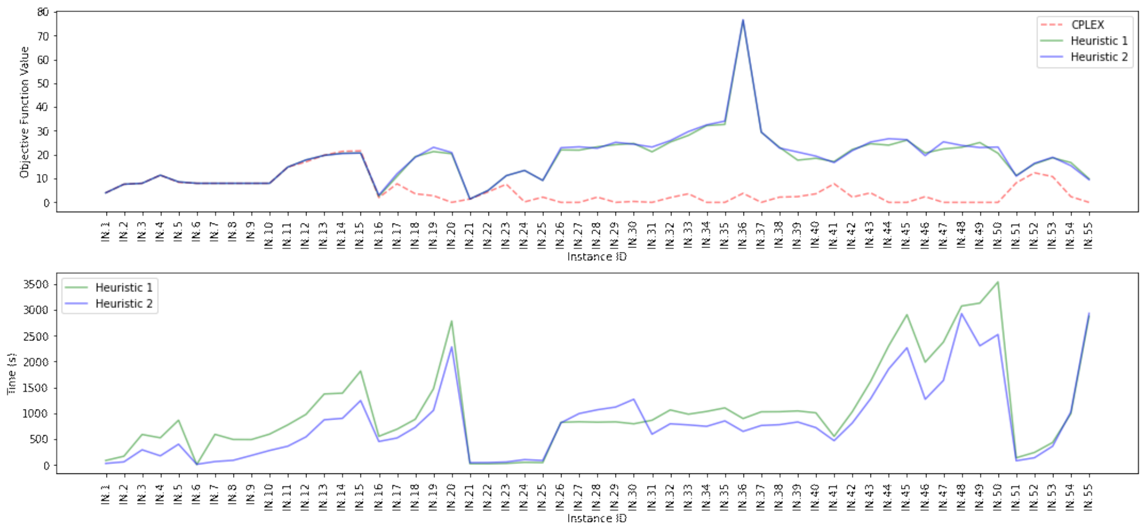


Figure 5.2: Comparison of objective function values (top) and solve times (bottom) between heuristics

Increasing the number of rendezvous nodes while keeping a constant number of connection nodes also increases solve times as shown by instances IN.6-IN.10 and IN.26-IN.30. This is more prominent in CLS as the number of iterations depends directly on the number of rendezvous nodes. Instance IN.21-IN.25 uses the same graphs as IN.16-IN.20 respectively but with only 1 rendezvous node. We can observe that each heuristic provides not only the same solution but also executes in the same amount of time. In such a case, the heuristics become identical as they perform only one iteration of the RH algorithm as no movement of the GV is required. As only a single iteration performed, the results can be obtained in a very short amount of time when compared to the time taken for instances IN.16-IN.20. However, we also note the difference between the objective function values between instances IN.16-IN.20 and IN.21-IN.25. As instances IN.21-IN.25 only include one rendezvous node, they represent problems where a GV is not available. The difference in objective values show the benefits of using a GV to aid drone routing.

Instances IN.11-IN.15 and IN.41-IN.45 test the affects of increasing the number of drones with same graphs. Although the solve times for each heuristic increases significantly with each additional drone, the increase in objective function value indicates better coverage of edges with more drones.

Instances IN.31-IN.35 increases the number of weighted edges while keeping number of rendezvous nodes and connection nodes constant. This allows for higher objective function values as drones are able to find better routes through weighed edges are reducing unwanted travel. Instances IN.36-IN.40 allows drones to revisit edges after increasing time intervals. Allowing drones to return after shorter periods of time leads to higher objective function values as drones can revisit edges with a higher weight more frequently. However in certain cases, e.g. instance IN.36 where there is no time limit for return, this is impractical as drones will traverse the same edge for the entirety of the time-horizon. We note that increasing the number of weighted edges and altering the return times have little effect on solve times.

Instances IN.46-IN.50 varies the time-horizon from 1 hour to 3 hours at 30 minute increments and consequently varies the number of time periods to keep a 60 periods/hr ratio. Instances IN.51-IN.55 varies the number of periods while keeping the time horizon at 1 hour, i.e., each period represents 6, 4, 2, 1 and 0.5 minutes for each instance respectively. The battery life and return times are adjusted proportionally. Using fewer total number of time periods can improve solve time, but in turn can worsen the quality of the solution as traversal times in terms of periods will not be representative of real world cases. For example, if the problem is defined to with 30 periods and a time horizon of 1 hour, i.e., 1 period = 2 minutes, edges that would realistically take 1 minute to traverse will have to be rounded up to 2 minutes or 1 period to fit with the problem parameters. Therefore a trade-off of solution accuracy or total solve time will have to be considered when selecting the number of periods to be used.

We observe that in general CLS performs better in terms of solution quality. This may be attributed to the fact that, for many instances, not all intersection nodes are utilized by the GV over the time-horizon. ILS generates GV routes randomly allowing it to travel between all node in the set V_r. In cases where, not all nodes are used, this approach leads to wasted time when travelling between unnecessary nodes. As the process is also completely random, finding the right sequence of nodes to visit over the time-horizon become tricky and may only result in highly quality solutions after many interactions are run over a longer period of time. CLS isolates certain nodes before generating routes. Although largely random, the routes generated better represent the optimal solution as the nodes are made available in order of which would be most beneficial to drones. Solve times for the heuristics largely depend on the parameters selected by the user. Although CLS seems to provide solutions quicker, its execution time is proportional to the total number of rendezvous nodes available as more nodes means more total iterations as shown in results from instances 26-30. The maximum number of rendezvous nodes used in the instances tested is 5, therefore CLS performs a maximum of 5 iterations while ILS performs 10 restarts giving CLS a time advantage.

Table 5.2: Comparison of heuristic solutions with CPLEX solutions

	Parameters								CPLEX			Iterated Local Search Heuristic				Cluster-based Location Search Heuristic			
ID	$\|V_r\|$	$\|V_c\|$	|S1|	|U|	t_u	t_r	|T|	T_m	Solution	Time	% GAP	Solution	Max Time	Min Time	Avg. Time	Solution	Max Time	Min Time	Avg. Time
IN.1	5	10	12	1	30	30	30	0.5	4.0	51.94	0.00	4.0	87.19	83.02	85.65	4.0	29.91	28.20	28.84
IN.2	5	15	18	1	30	30	30	0.5	7.6	580.44	0.00	7.6	169.70	166.91	168.13	7.6	61.97	57.24	58.87
IN.3	5	20	21	1	30	30	30	0.5	8.0	3668.58	0.00	8.0	613.75	568.06	589.38	8.0	320.21	257.78	292.44
IN.4	5	25	27	1	30	30	30	0.5	11.4	10389.61	140.79	11.4	584.62	484.18	524.77	11.4	186.78	170.06	175.64
IN.5	5	30	32	1	30	30	30	0.5	8.4	7200.55	215.33	8.6	877.82	858.86	865.32	8.6	483.04	331.38	400.99
IN.6	1	20	21	1	30	30	30	0.5	8.0	758.73	0.00	8.0	7.94	5.96	6.73	8.0	11.83	11.03	11.42
IN.7	2	20	21	1	30	30	30	0.5	8.0	1728.16	0.00	8.0	610.21	577.17	592.17	8.0	69.65	55.28	64.71
IN.8	3	20	21	1	30	30	30	0.5	8.0	1857.44	0.00	8.0	503.87	480.20	492.37	8.0	91.90	82.71	88.83
IN.9	4	20	21	1	30	30	30	0.5	8.0	7200.92	54.07	8.0	503.84	479.20	490.78	8.0	200.62	172.17	182.83
IN.10	5	20	21	1	30	30	30	0.5	8.0	3668.58	0.00	8.0	636.83	554.74	596.34	8.0	305.00	250.59	279.51
IN.11	5	20	21	2	30	30	30	0.5	14.8	7204.20	64.70	14.8	825.76	744.20	775.39	14.8	385.85	322.14	362.27
IN.12	5	20	21	3	30	30	30	0.5	17.0	7227.16	47.29	17.8	1026.36	892.71	978.70	17.8	688.55	442.56	542.85
IN.13	5	20	21	4	30	30	30	0.5	19.8	7218.23	34.59	19.7	1483.53	1260.89	1373.34	19.7	955.13	813.48	873.94
IN.14	5	20	21	5	30	30	30	0.5	21.4	7212.97	30.73	20.6	1452.56	1296.69	1388.47	20.5	1006.26	804.14	901.43
IN.15	5	20	21	6	30	30	30	0.5	21.6	7219.02	32.33	20.7	1897.33	1723.89	1817.60	20.7	1328.20	1143.17	1246.22
IN.16	5	10	12	2	20	60	60	1	2.0	7204.48	534.68	2.6	572.00	517.76	553.61	3.0	639.71	323.07	453.28
IN.17	5	15	18	2	20	60	60	1	7.8	7215.33	209.07	10.9	777.57	596.08	692.18	12.0	560.46	451.80	521.38
IN.18	5	20	21	2	20	60	60	1	3.6	7227.49	914.15	19.2	892.95	871.13	883.98	18.9	799.83	685.31	727.89
IN.19	5	25	27	2	20	60	60	1	2.8	8220.29	1381.21	21.3	1573.20	1296.87	1474.40	23.1	1260.13	814.18	1056.23
IN.20	5	30	32	2	20	60	60	1	0.0	7200.15	-	20.4	3551.70	2392.42	2786.67	20.9	2814.34	1960.16	2284.28
IN.21	1	10	12	2	20	60	60	1	1.4	7200.87	730.45	1.4	21.63	21.01	22.80	1.4	42.20	41.34	42.94
IN.22	1	15	18	2	20	60	60	1	4.4	7204.58	403.05	5.0	21.50	21.04	21.83	5.0	43.32	42.05	45.27
IN.23	1	20	21	2	20	60	60	1	7.6	7215.91	367.84	11.2	27.55	25.74	29.06	11.2	53.76	50.44	56.93
IN.24	1	25	27	2	20	60	60	1	0.2	7211.06	20134.31	13.4	48.97	48.00	50.28	13.4	99.57	98.01	102.50
IN.25	1	30	32	2	20	60	60	1	2.2	7209.87	495336.36	9.2	43.10	42.26	44.39	9.2	85.01	84.26	85.57
IN.26	6	20	32	2	20	60	60	1	0.0	7218.99	-	22.0	835.75	807.09	824.17	22.9	967.17	701.19	815.40
IN.27	7	20	32	2	20	60	60	1	0.0	7214.07	-	21.9	856.01	810.14	835.71	23.3	1056.67	962.56	994.60
IN.28	8	20	32	2	20	60	60	1	2.2	7219.23	1607.39	23.3	852.07	789.29	826.86	22.7	1328.38	932.61	1067.34
IN.29	9	20	32	2	20	60	60	1	0.0	7277.60	-	24.2	876.87	802.07	833.53	25.2	1307.27	938.59	1119.58
IN.30	10	20	32	2	20	60	60	1	0.4	8630.80	9351.61	24.7	831.87	768.47	795.77	24.4	1464.33	1095.68	1271.23
IN.31	5	20	26	2	20	60	60	1	0.0	7216.70	-	21.2	936.90	807.28	864.79	23.2	633.77	546.84	594.80
IN.32	5	20	31	2	20	60	60	1	2.0	7201.57	2772.88	25.3	1110.30	984.74	1064.34	25.9	825.93	772.27	796.26
IN.33	5	20	36	2	20	60	60	1	3.6	7202.98	1763.42	28.1	993.77	975.11	981.86	29.7	981.08	643.17	774.29
IN.34	5	20	41	2	20	60	60	1	0.0	7203.38	-	32.2	1073.36	1005.52	1035.49	32.5	820.50	647.01	746.67
IN.35	5	20	46	2	20	60	60	1	0.0	7203.55	-	32.7	1126.96	1089.25	1103.80	34.1	923.31	737.17	852.09
IN.36	5	20	21	2	20	0	60	1	3.8	7244.33	6486.69	76.3	925.04	879.43	897.46	76.6	735.20	550.26	649.17
IN.37	5	20	21	2	20	15	60	1	0.0	7204.43	-	29.4	1052.98	1012.46	1028.05	29.5	880.86	694.70	763.25
IN.38	5	20	21	2	20	30	60	1	2.2	7229.20	2736.28	23.1	1096.31	973.91	1030.04	22.8	834.65	729.13	778.81
IN.39	5	20	21	2	20	45	60	1	2.4	7215.79	1598.54	17.7	1172.38	971.24	1045.15	21.1	969.21	696.32	831.33
IN.40	5	20	21	2	20	60	60	1	3.6	7226.37	914.15	18.5	1114.09	920.67	1007.86	19.4	875.25	631.63	721.50
IN.41	5	20	21	1	20	30	60	1	7.8	7211.84	663.85	17.1	556.30	545.94	552.70	16.7	744.03	281.46	468.83
IN.42	5	20	21	2	20	30	60	1	2.2	7232.08	2736.28	22.2	1095.25	961.07	1031.10	21.7	854.35	781.55	812.29
IN.43	5	20	21	3	20	30	60	1	4.0	7207.18	304730.00	24.6	1660.15	1552.83	1614.41	25.3	1430.96	1039.07	1279.30
IN.44	5	20	21	4	20	30	60	1	0.0	7209.66	-	24.0	2467.87	2171.96	2306.30	26.7	2484.86	1426.88	1858.86
IN.45	5	20	21	5	20	30	60	1	0.0	7211.07	-	26.1	3011.49	2763.28	2908.20	26.3	2302.57	2227.50	2267.81
IN.46	5	20	21	2	20	60	60	1	2.4	7217.78	1170.42	20.7	2334.48	1667.01	1988.35	19.6	1347.82	1205.55	1271.10
IN.47	5	20	21	2	20	90	90	1.5	0.0	7419.21	-	22.4	2664.60	2182.98	2377.24	25.4	1744.43	1437.89	1634.93
IN.48	5	20	21	2	20	120	120	2	0.0	7240.90	-	23.1	3187.00	2935.12	3075.93	23.9	3109.23	2811.18	2928.53
IN.49	5	20	21	2	20	150	150	2.5	0.0	9643.13	-	25.1	3652.55	2777.73	3132.45	23.0	2712.81	1923.52	2305.12
IN.50	5	20	21	2	20	180	180	3	0.0	7231.97	-	20.6	4007.14	3232.37	3542.57	23.2	2701.30	2394.88	2524.74
IN.51	5	20	21	2	4	10	10	1	8.2	7201.69	58.23	11.3	138.99	135.96	137.66	11.0	84.17	72.84	80.06
IN.52	5	20	21	2	5	15	15	1	12.4	7201.50	136.60	16.1	245.21	235.90	240.94	16.4	171.17	115.74	138.53
IN.53	5	20	21	2	10	30	30	1	10.8	7204.02	184.38	18.7	441.42	425.78	434.68	18.9	390.89	343.10	362.67
IN.54	5	20	21	2	20	60	60	1	2.4	7217.78	1170.42	16.7	1080.79	946.89	991.66	15.4	1203.70	932.84	1025.91
IN.55	5	20	21	2	40	120	120	1	0.0	7216.19	-	9.9	3086.99	2650.01	2875.76	9.5	3452.29	2506.71	2934.01

5.3 Evaluation of Heuristics

In this section we evaluate the performance of each heuristic based on hyperparameters used to control the search process. Section 5.3.1 we look at how the result found using the rolling horizon approach used to construct drone routes for both heuristic compares to optimal results found using CPLEX. In Section 5.3.2 we look at how varying the cooling schedule, temperature parameters and number of restarts affects the performance of ILS and what affects varying the stopping criteria has on the results obtained using CLS.

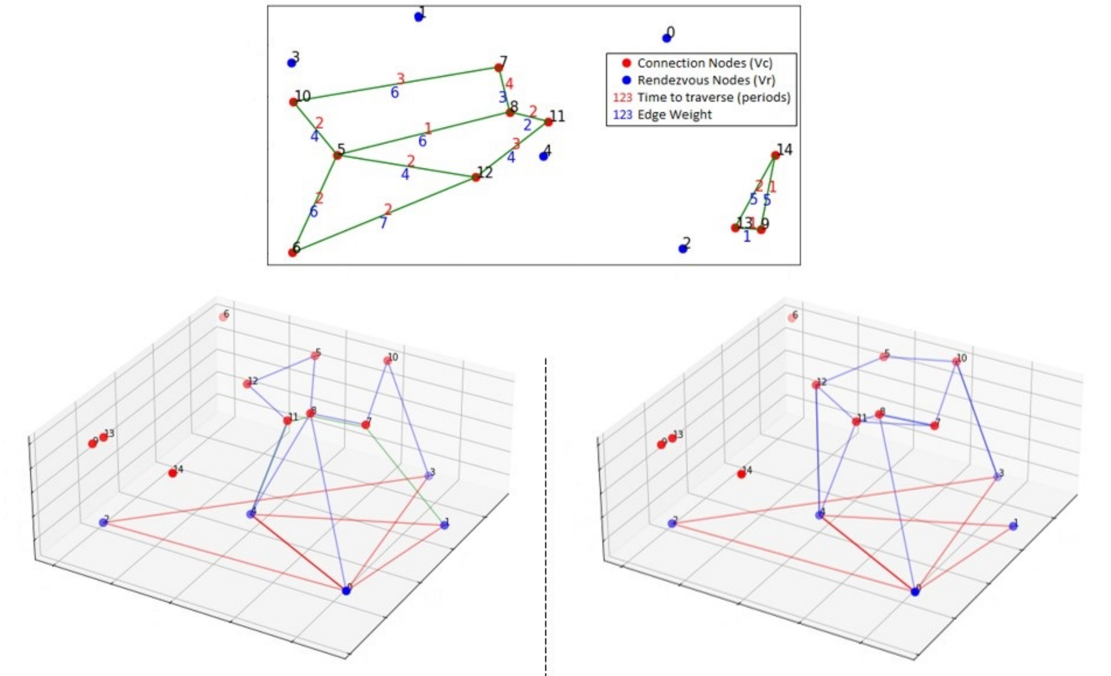
5.3.1 Rolling Horizon Approach

In order to demonstrate the benefits of the rolling horizon (RH) approach, a comparison has been done with solutions obtained using CPLEX. To perform this comparison several instances were used to test different parameters. For each instance a random GV route was created using Algorithm 1 and based on this route, optimal drone routes were created using both methods. In order to support this test, the original formulation was modified so that the GV is no longer considered rather the GV route is directly supplied to the problem. The modified formulation can be found in Appendix A.2. All instances were run with a maximum run-time set at 86,400 seconds (1 day).

Table 5.3: Comparison of rolling approach with CPLEX results

									CPLEX		Rolling	
ID	$\|V_r\|$	$\|V_c\|$	|S1|	|U|	t_u	t_r	|T|	T_m	Solution	Time(s)	Solution	Time(s)
IN.101	5	10	12	1	30	30	30	0.5	4.0	89.05	4.0	0.7
IN.102	5	15	18	1	30	30	30	0.5	7.6	412.38	7.6	1.2
IN.103	5	20	21	1	30	30	30	0.5	8.0	2511.03	8.0	5.4
IN.104	5	25	27	1	30	30	30	0.5	11.2	26586.60	11.2	3.2
IN.105	5	30	32	1	30	30	30	0.5	6.8	6432.67	6.8	2.6
IN.106	5	10	12	1	20	30	60	1	7.4	1784.73	7.0	1.6
IN.107	5	15	18	1	20	30	60	1	5.0*	86410.93	9.6	3.1
IN.108	5	20	21	1	20	30	60	1	17.6*	86406.73	12.8	4.3
IN.109	5	25	27	1	20	30	60	1	16.4*	86426.95	18.8	5.2
IN.110	5	30	32	1	20	30	60	1	10.8*	86444.88	13.0	6.4
IN.111	5	10	12	2	20	30	60	1	7.2	52719.15	7.0	1.8
IN.112	5	15	18	2	20	30	60	1	11.0*	86410.33	11.0	4.4
IN.113	5	20	21	2	20	30	60	1	19.4*	86431.31	20.6	5.1
IN.114	5	25	27	2	20	30	60	1	21.2*	86459.22	24.6	6.8
IN.115	5	30	32	2	20	30	60	1	15.2*	86432.48	17.0	6.3

The results in Table 5.3 show that the RH method is capable of significantly reducing solve times while producing close to optimal results. The RH fails to provide optimal results mainly due to that fact that each sub-horizon and routes for each drone is being optimized independently while only receiving information about previous intervals. Figure 5.3 shows the solution to instance IN.111 where the RH method is unable to obtain the optimal solution to the problem. We observe that the optimal solution utilizes both available drones while the RH method solution only uses one. In both solutions, drones take the same route between periods 0 and 19. In the optimal solution, the drones remain on the GV between periods 19 and 32 while in the RH solution, the drone takes off immediately after landing at period 19. This is because in the RH method, the drone is able to receive more reward in the immediate sub-horizon starting at period 19 therefore will always capitalize on it regardless of whether a better solution is available in the future. Also due to the fact that each drone is managed successively rather than together, the RH method is unable to determine whether a drone would benefit from leaving certain edges for following drones to covers. In the optimal solution, routes for both drones between periods 32 and 51 are intertwined allowing drone to receive more reward while remaining the air for a shorter period of time. In the RH solution, the second drone is not used alongside the first because given the route of the first drone, the second drone would not be able to obtain more reward without negatively impacting the objective function. Due to the shortcomings of the RH method, for certain instances neither heuristic will be able to provide optimal solutions since drone routes may never be optimal even with the optimal GV routes.



X = (0, 0, 0, 8), (0, 6, 8, 7), (0, 9, 7, 10), (0, 15, 10, 10), (0, 16, 10, 3),
(0, 32, 4, 8), (0, 35, 8, 5), (0, 41, 5, 12), (0, 45, 12, 11),
(0, 49, 11, 4), (1, 33, 4, 11), (1, 35, 11, 8), (1, 37, 8, 7),(1, 40, 7, 1)
Y = (0, 0, 0), (0, 32, 4), (1, 33, 4)
V = (0, 19, 3), (0, 51, 4), (1, 44, 1)

X = (0, 0, 0, 8), (0, 6, 8, 7), (0, 9, 7, 7), (0, 10, 7, 10), (0, 16, 10, 3),
(0, 19, 3, 10), (0, 22, 10, 5), (0, 26, 5, 12), (0, 30, 12, 4), (0, 33, 4, 11),
(0, 35, 11, 8), (0, 37, 8, 7), (0, 40, 7, 11), (0, 44, 11, 12), (0, 48, 12, 4)
Y = (0, 0, 0), (0, 19, 3), (0, 33, 4)
V = (0, 19, 3), (0, 33, 4), (0, 51, 4)

Figure 5.3: Optimal solution (left) and rolling approach solution (right) for instance IN.111 (top)

From Table 5.3, we observe that as the instances get larger, the time required to solve the problem increases proportionally. This increase in time becomes significant when we consider that drone routes have to be generated using the RH method hundreds of times throughout the course of either of the heuristics. As a result, this method becomes a bottleneck not only in being able to achieving optimal solutions but also in attaining practical run-times.

5.3.2 Heuristic Parameters

Both heuristics involve several parameters that can be adjusted by the user to either prioritize solution quality or solve-time. In order to demonstrate how these parameters affect the final solution, several experiments were carried out on instance IN.18. For each experiment, the heuristic in concern was run 10 times and the average solution value visualized in the figures below. The minimum and maximum values are also included to demonstrate the deviation of results.

ILS performs a number of restarts to widen its search. Figure 5.4 shows how increasing the number of restarts affects the objective function value and total execution time of the heuristic. Increasing the number of restarts performed allows the heuristic to obtain better results but also increases the execution time significantly, in a linear fashion. It can be observed that after a certain number of restarts, the improvement in the solution starts to stagnate while solve-time continues to climb. Hence, in cases where solutions have to be obtained in a short amount of time, performing fewer restarts is still capable of providing high quality solutions.

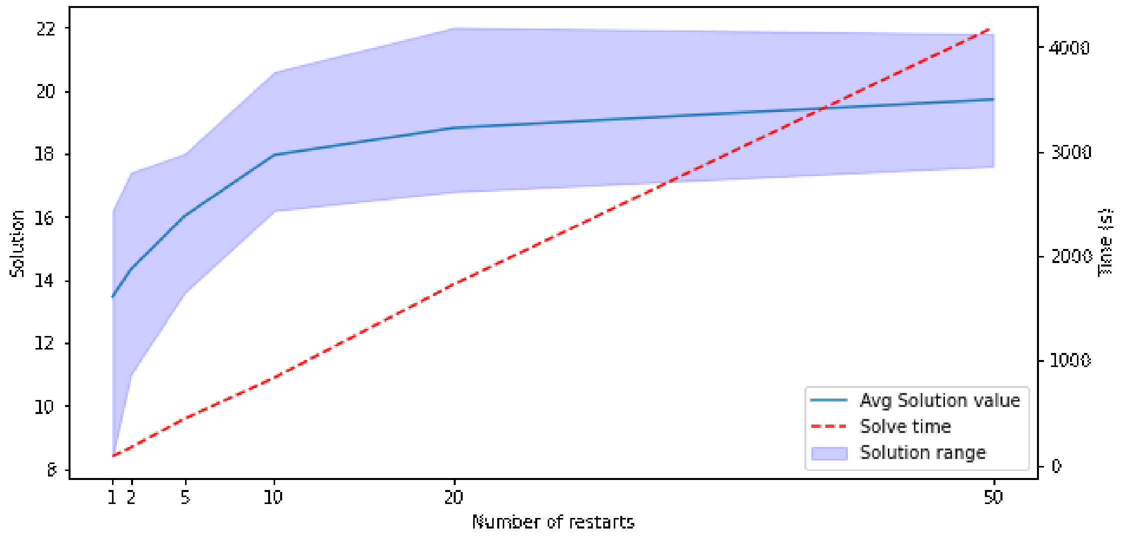


Figure 5.4: Change in solution value and run-time with increasing number of restarts

The cooling schedule in the SA process of ILS determines how the temperature parameter varies over the each iteration. Figure 5.5 shows how the cooling schedule affects the objective function value and total execution time of the heuristic. For lower values of x, the temperature parameter decreases more drastically and as a result the SA algorithm accepts fewer non-improving solutions. This restricts diversification of the solutions but allows the stopping criteria to be met sooner. From the figure we observe that allowing for more diversification results in obtaining better solutions overall.

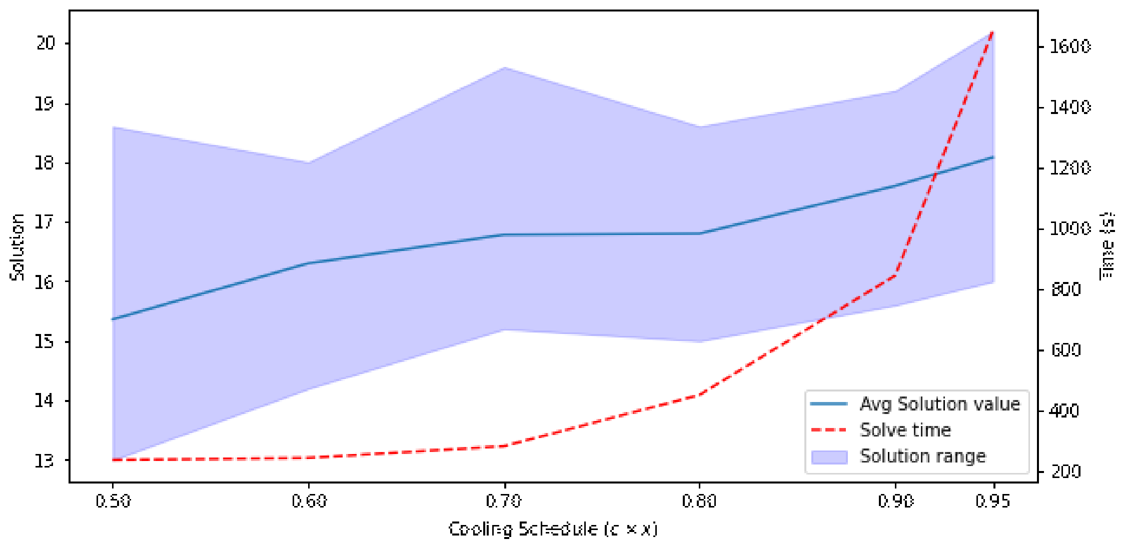


Figure 5.5: Change in solution value and run-time with changing temperature schedules

CLS iteratively generates GV routes for each set of available rendezvous nodes until a number of iterations generate non-improving solutions. Figure 5.6 shows how increasing the number of non-improving results affects the objective function value and total execution time of the heuristic. Similar to ILS, it can be observed that after a number of iterations, the improvement in solution is no longer significant in relation to the total time taken. As a result, allowing for fewer non-improving solution may be beneficial. However, it must be noted that, by looking at the solution ranges, over a similar period of time CLS is capable of obtained better results than ILS.

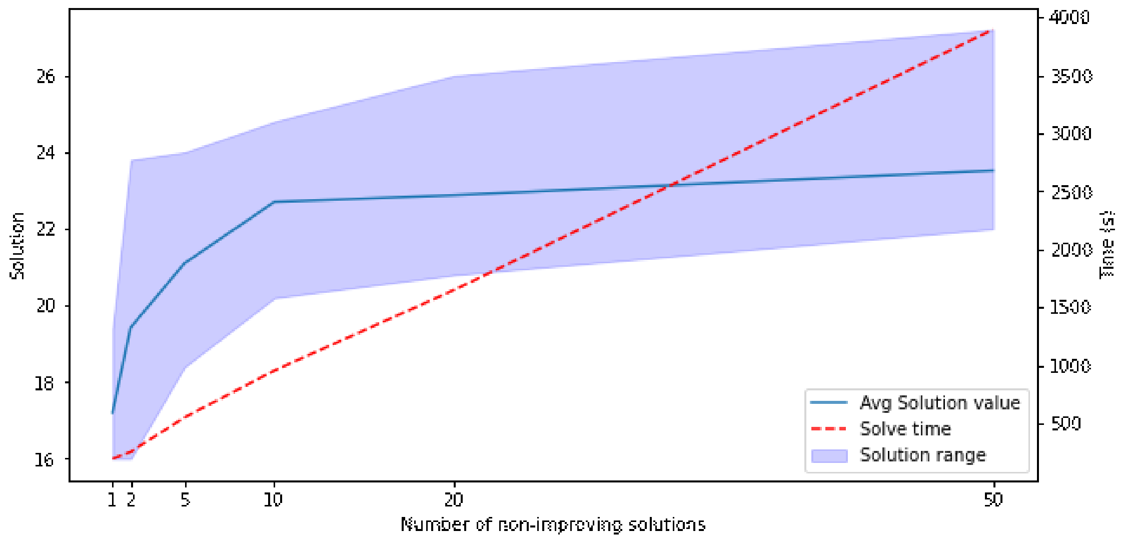


Figure 5.6: Change in solution value and run-time with increasing number of allowed non-improving solutions

5.4 Case Study

In this section we explore a real world application of the problem described in this thesis. In particular we consider surveillance of road networks during a protest. In September 2019, an estimated 500,000 people attended a climate change march in downtown Montreal, the largest in the province's history [61]. In such situations where roads are flooded with large numbers of people, it is important for authorities to monitor the roads in order ensure safety of the participants. The use of drones to perform such tasks can provide a good solution allowing authorities to keep an eye from the air in a manner that is safe for attendants. Figure 5.7 shows a map of downtown Montreal. The edges represent roads that authorities may like to monitor during the protests with priorities given to roads that will be used by attendants during the march. The map includes 80 edges to monitor and 10 rendezvous nodes. The graph is considered to be fully connected. The rendezvous nodes are placed in parking lots and open spaces that are considered safe for the drones to land. Edges are weighted from 1-5 where a weight of 5 is given to edges that encountered the highest volume of attendants. It is assumed that drones are mounted with a camera to transmit or return video footage. Therefore the drone speed is limited to 20 km/hr to ensure consistent and clear footage is available at low altitude flight (<50m). The time required by the GV to travel between rendezvous nodes is the shortest path between the points obtained using Google Maps. Since the GV must follow the road network, speed limits, traffic flow directions etc. the use of Google Maps provides an accurate representation of the problem. Since the drones do not have the same restrictions, the time required by drones to traverse edges is calculated using the speed of the drone and the Euclidean distance between the coordinates of each node on the map. For this case we use the same parameters as mentioned in Section 5.2.2 for each heuristic.

We look at 6 scenarios outlined in Table 5.4. Figure 5.8 illustrates the incumbent solution for each respective instance. RI.1, RI.2 and RI.3 includes 10 rendezvous nodes while RI.4, RI.5 and RI.6 includes only the depot. We observe the benefits of using a GV in synchronisation with drones. In each case, using a GV extends the range of the drone allowing them to reach edges that would otherwise not be possible due to their limited battery life. RI.2 and RI.4 considers the use of 5 drones while the rest use 2 drones. Using more drones makes it possible to cover all required edges in a short period of time. RI.3 and RI.5 allows drones to return to edges after a certain period of time. Although this reduces the total coverage of the graph, drone are able to focus on edges with higher priority. While monitoring events such as protests and marches, this feature can prove to be useful as the state of the event changes overtime and multiple traversals can enable those changes to be better monitored. As observed previously, CLS consistently provides better solutions when compared to ILS. However, in each instance involving 10 rendezvous nodes, CLS runs for about 40% longer. In this particular case, the additional time spent in search may contribute to the improved solution results. As expected, when a single rendezvous node is involved both heuristics provide the same results in the same time as they both reduce to a single iteration of the rolling horizon algorithm.

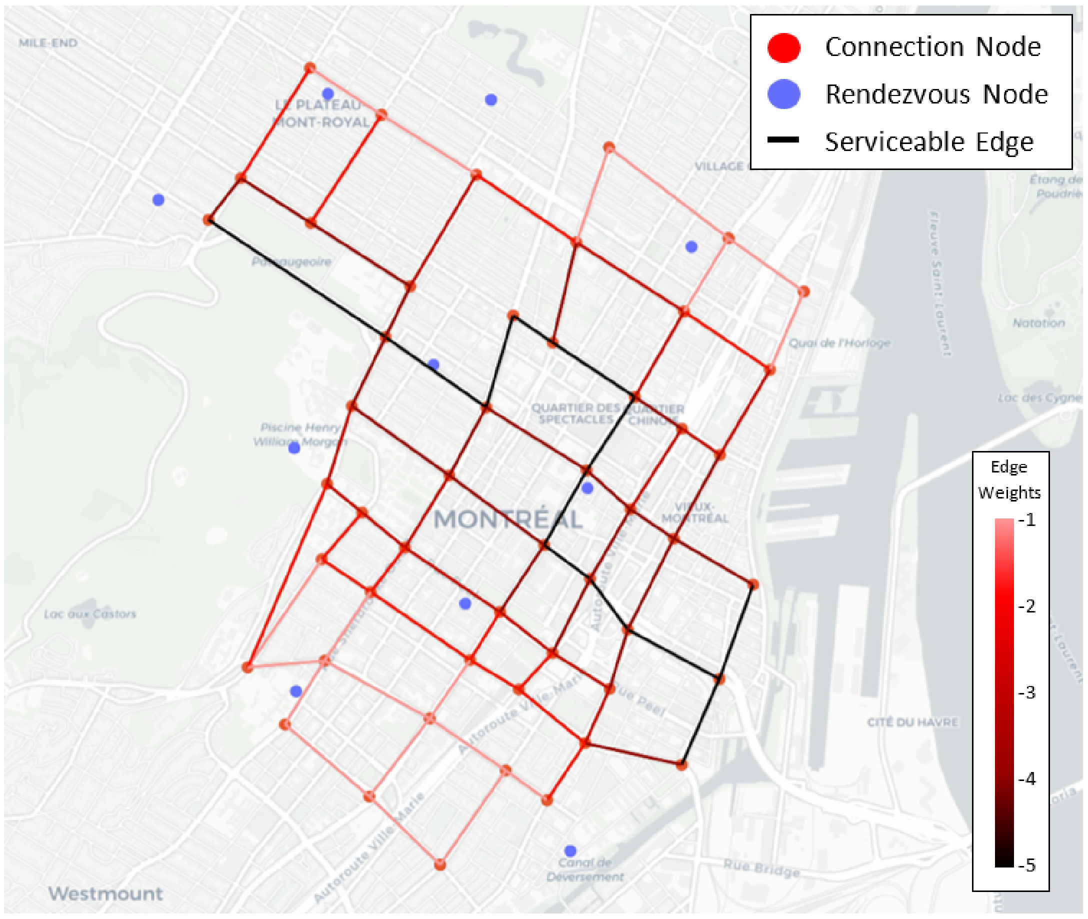


Figure 5.7: Map of downtown Montreal with edges to be monitored based on priority

Table 5.4: Case instances and results obtained using heuristics

	Parameters					ILS Heuristic		CLS Heuristic	
ID	$\lvert V_r\rvert$	$\lvert V_c\rvert$	$\lvert S1\rvert$	$\lvert u\rvert$	t_r	Solution	Time(s)	Solution	Time(s)
RI.1	10	50	80	2	60	95.6	3645.74	104.6	5027.41
RI.2	10	50	80	5	60	113.0	5448.07	117.6	7764.17
RI.3	10	50	80	2	30	117.0	3681.81	123.0	5250.17
RI.4	1	50	80	2	60	89.0	30.37	89.0	34.76
RI.5	1	50	80	5	60	107.6	61.52	107.6	68.29
RI.6	1	50	80	2	30	114.0	41.97	114.0	41.97

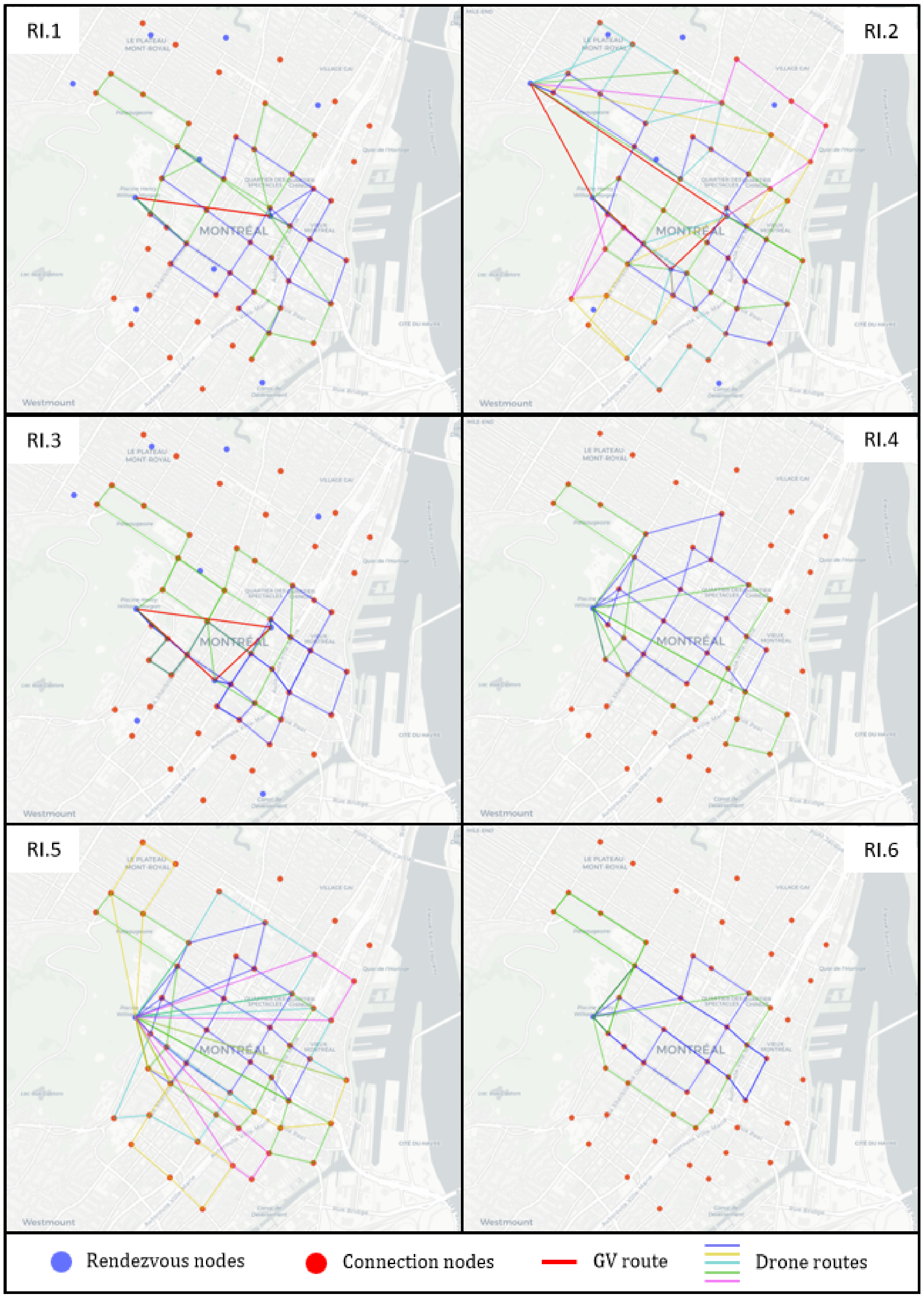


Figure 5.8: Illustration of best solution found for each instance in the case study

Chapter 6

Conclusion

In this thesis we introduced the MDTARP, formulated it as an MILP problem, and developed two heuristics, the Iterated Local Search heuristic and the Cluster-based Location Search, in order to solve the problem within reasonable time. The MDTARP describes the problem of using multiple drones to provide service along edges on a network in tandem with a ground vehicle used to provide service to and transport the drones.

The heuristics decompose the problem by constructing drone routes and the GV route separately. GV routes are created first using different methods for each heuristic based on which a rolling horizon method is used in both to construct drone routes. The rolling horizon method solves a routing problem using CPLEX on smaller time horizons and specified take-off and landing locations. ILS uses a local search and SA procedure to escape local optima when constructing solutions and performs several restarts to diversify the search space. CLS uses a clustering method to iteratively make rendezvous points available to visit and constructs several GV routes based on those points.

Solutions obtained from both heuristics are compared to solutions obtained when solving the problem using CPLEX. The results have shown that both heuritics are capable of obtaining solutions that are close to and often better than solutions obtained using CPLEX after 2 hours of CPU time. We explore how the heuristic parameters affect solve time and final results and provide recommendations to users. We also look at how changing the coefficients of the objective function affects the solution to help users make selections based on their needs. We are able to identify that CLS outperforms ILS in terms of objective function value however in some cases, such as instances with a large number of rendezvous points, ILS may provide solutions quicker.

We recognize that there are certain limitations that affect the performance of the heuristics. The rolling horizon approach, although aids in improving run-time performance of the heuristics, in some instances prevents from obtaining optimal solutions. We acknowledge that as problems get larger, the use of a general purpose solver, such as CPLEX, to solve the routing problem when constructing drone routes can become more and more time consuming resulting in overall run-time of the heuristics becoming impractical. In order to alleviate dependency on general purpose solvers and potentially improve overall run-time a separate algorithm can be developed to solve the drone arc routing problem that takes into consideration, edge weights and previously visited edges. Compromises in solution optimality made by the rolling horizon method can also be addressed.

In future research, further studies can be conducted in a number of areas. Extensions to the formulation introduced in this thesis could involve:

- *Partial edge coverage.* The current formulation requires edges to be traversed completely before drones are able to move to a new edge. In certain applications such an requirements is necessary, eg. urban city environments where buildings may hinder travel between roads or residential areas where drones would travel over private property leading to security concerns. However this limits edge coverage capabilities as drones do not start traversal of an edge if it is unable to complete it. Allowing partial also allows several drones to work together to complete traversal of a single edge and takes better advantage of the agility provided by drones.
- *Limiting the number of batteries available and allowing for battery recharge time.* The current formulation assumes an unlimited number of batteries for drones to use during the course of its travel. This may be an impractical assumption when, over a large period of time, drones may need battery replacements several times. Limiting the number of batteries available is a better representations of real-life scenarios and allowing time for batteries to recharge can extend the time drones can remain in the air over the total time-horizon.
- *Drone and GV meet at any point on an edge.* The current formulation requires the GV and drones to meet only on one of the rendezvous points. Allowing meetings to occur anywhere on the network may improve edge coverage as the GV could meet drones during their traversal of an edge and as a result drones would have to make less unnecessary trips for battery replacements.
- *Windy arc routing problem.* The presence of wind can affect performance of drones by either increasing time required to traverse edges, or reducing battery life when speed is maintained. Taking into account wind speed and direction over the time-horizon presents a problem more practical and representative of real world scenarios.
- *Allowing for non-homogeneity in drones.* The current formulation assumes all drones are the same in terms of speed and battery life. Extensions could consider drones travelling at different speeds with differing limitations on battery life.

References

[1] “Clarity from above: Pwc global report on the commercial applications of drone technology,” 2016.

[2] Amazon, “Amazon.com: Prime air,” *Amazon Prime Air*, 2017.

[3] “Finnish post office tests drone for parcel delivery,” *Reuters*, 2015.

[4] “Maersk just used a drone to deliver cookies to a tanker,” *Fortune*, 2016.

[5] “Zipline begins drone delivery of covid-19 test samples in ghana,” *CNBC*, 2020.

[6] “Zipline, pfizer and biontech collaboration paves the way for automated, on-demand delivery of first mrna covid-19 vaccines in ghana,” *Pfizer*, 2021.

[7] DJI, “Dji kicks off 2016 dji developer challenge - dji,” 2016.

[8] G. Singhal, B. Bansod, and L. Mathew, “Unmanned aerial vehicle classification, applications and challenges: A review,” 11 2018.

[9] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch, “Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey,” *Networks*, vol. 72, pp. 411–458, 12 2018.

[10] K. W. Smith, “The use of drones in environmental management,” *World Environmental and Water Resources Congress 2015: Floods, Droughts, and Ecosystems - Proceedings of the 2015 World Environmental and Water Resources Congress*, pp. 1352–1361, 2015.

[11] S. Mikhailova, “Using ex-military drones to plant trees,” *Huffpost Impact*, 2015.

[12] S. Losey, “Darpa nabs gremlin drone in midair for first time,” *DefenseNews*, 2021.

[13] “Dewa uses drones to ensure quality of desalinated water according to highest international standards,” *Government of Dubai Media Office*, 2022.

[14] G. Topham, “Flight mh370: robots search sea bed for signs of malaysia airlines plane | malaysia airlines flight mh370,” *The Guardian*, 2014.

[15] G. Macrina, L. Di, P. Pugliese, F. Guerriero, and G. Laporte, “Drone-aided routing : A literature review,” vol. 120, 2020.

[16] S. H. Chung, B. Sah, and J. Lee, “Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions,” *Computers Operations Research*, vol. 123, p. 105004, 11 2020.

[17] J. Edmonds and E. L. Johnson, “Matching, euler tours and the chinese postman,” *Mathematical Programming*, vol. 5, pp. 88–124, 1973.

[18] C. H. Papadimitriou, “On the complexity of edge traversing,” *Journal of the ACM (JACM)*, vol. 23, pp. 544–554, 7 1976.

[19] M. Guan, “On the windy postman problem,” *Discrete Applied Mathematics*, vol. 9, pp. 41–46, 9 1984.

[20] M. Dror, H. Stern, and P. Trudeau, “Postman tour on a graph with precedence relation on arcs,” *Networks*, vol. 17, pp. 283–294, 1 1987.

[21] J. K. Lenstra and A. H. G. R. Kan, “On general routing problems,” *Networks*, vol. 6, pp. 273–280, 1 1976.

[22] G. N. Frederickson, M. S. Hecht, and C. E. Kim, “Approximation algorithms for some routing problems,” *SIAM Journal on Computing*, vol. 7, pp. 178–193, 5 1978.

[23] B. L. Golden and R. T. Wong, “Capacitated arc routing problems,” *Networks*, vol. 11, pp. 305–315, 23 1981.

[24] N. Christofides, “The optimum traversal of a graph,” *Omega*, vol. 1, pp. 719–732, 12 1973.

[25] B. L. Golden, J. S. Dearmon, and E. K. Baker, “Computational experiments with algorithms for a class of routing problems,” *Computers and Operations Research*, vol. 10, pp. 47–59, 1 1983.

[26] J. R. Evans and E. Minieka, *Optimization algorithms for networks and graphs, Second edition, revised and expanded.* CRC Press, 1 1992.

[27] G. Clarke and J. W. Wright, “Scheduling of vehicles from a central depot to a number of delivery points,” *Operations Research*, vol. 12, pp. 568–581, 1964.

[28] G. Ulusoy, “The fleet size and mix problem for capacitated arc routing,” *European Journal of Operational Research*, vol. 22, pp. 329–337, 12 1985.

[29] J. Beasley, “Route first-cluster second methods for vehicle routing,” *Omega*, vol. 11, pp. 403–408, 1 1983.

[30] C. Prins, *Chapter 7: The Capacitated Arc Routing Problem: Heuristics.* Society for Industrial and Applied Mathematics, 10 2015.

[31] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, pp. 671–680, 5 1983.

[32] R. W. Eglese, “Routeing winter gritting vehicles,” *Discrete Applied Mathematics*, vol. 48, pp. 231–244, 2 1994.

[33] F. Glover, “Future paths for integer programming and links to artificial intelligence,” *Computers Operations Research*, vol. 13, pp. 533–549, 1 1986.

[34] R. W. Eglese and L. Y. O. Li, “A tabu search based heuristic for arc routing with a capacity constraint and time deadline,” *Meta-Heuristics*, pp. 633–649, 1996.

[35] A. Hertz, G. Laporte, and M. Mittaz, "Tabu search heuristic for the capacitated arc routing problem," *Operations Research*, vol. 48, pp. 129–135, 2000.

[36] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers Operations Research*, vol. 24, pp. 1097–1100, 11 1997.

[37] M. Polacek, K. F. Doerner, R. F. Hartl, and V. Maniezzo, "A variable neighborhood search for the capacitated arc routing problem with intermediate facilities," *Journal of Heuristics 2007 14:5*, vol. 14, pp. 405–423, 10 2007.

[38] C. Prins and R. W. Calvo, "A fast grasp with path relinking for the capacitated arc routing problem," 2005.

[39] P. Lacomme, C. Prins, and W. Ramdane-Cherif, "A genetic algorithm for the capacitated arc routing problem and its extensions," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2037, pp. 473–483, 2001.

[40] M. Dorigo and G. D. Caro, "Ant colony optimization: A new meta-heuristic," *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, vol. 2, pp. 1470–1477, 1999.

[41] P. Lacomme, C. Prins, and A. Tanguy, "First competitive ant colony scheme for the carp," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3172 LNCS, pp. 426–427, 2004.

[42] J. F. Campbell, Ángel Corberán, I. Plana, and J. M. Sanchis, "Drone arc routing problems," *Networks*, vol. 72, pp. 543–559, 2018.

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

[44] M. Salama and S. Srinivas, "Joint optimization of customer location clustering and drone-based routing for last-mile deliveries," *Transportation Research Part C: Emerging Technologies*, vol. 114, pp. 620–642, 5 2020.

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

[46] Z. Luo, Z. Liu, and J. Shi, "A two-echelon cooperated routing problem for a ground vehicle and its carried unmanned aerial vehicle," *Sensors (Basel, Switzerland)*, vol. 17, 2017.

[47] X. Wang, S. Poikonen, and B. Golden, "The vehicle routing problem with drones: several worst-case results," *Optimization Letters*, vol. 11, pp. 679–697, 2017.

[48] Z. Wang and J. B. Sheu, "Vehicle routing problem with drones," *Transportation Research Part B: Methodological*, vol. 122, pp. 350–364, 4 2019.

[49] P. Kitjacharoenchai, B. C. Min, and S. Lee, "Two echelon vehicle routing problem with drones in last mile delivery," *International Journal of Production Economics*, vol. 225, 2020.

[50] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning paths for package delivery in heterogeneous multirobot teams," *IEEE Transactions on Automation Science and Engineering*, vol. 12, pp. 1298–1308, 10 2015.

[51] S. Poikonen and B. Golden, "The mothership and drone routing problem," *INFORMS Journal on Computing*, vol. 32, pp. 249–262, 3 2020.

[52] S. Poikonen and B. Golden, "Multi-visit drone routing problem," *Computers Operations Research*, vol. 113, p. 104802, 1 2020.

[53] H. Oh, S. Kim, A. Tsourdos, and B. A. White, "Coordinated road-network search route planning by a team of uavs," *International Journal of Systems Science*, vol. 45, pp. 825–840, 5 2014.

[54] M. Li, L. Zhen, S. Wang, W. Lv, and X. Qu, "Unmanned aerial vehicle scheduling problem for traffic monitoring," *Computers Industrial Engineering*, vol. 122, pp. 15–23, 8 2018.

[55] J. F. Campbell, Ángel Corberán, I. Plana, J. M. Sanchis, and P. Segura, "Solving the length constrained k-drones rural postman problem," *European Journal of Operational Research*, vol. 292, pp. 60–72, 7 2021.

[56] L. Amorosi, J. Puerto, and C. Valverde, "Coordinating drones with mothership vehicles: The mothership and drone routing problem with graphs," *Computers Operations Research*, vol. 136, p. 105445, 12 2021.

[57] L. Amorosi, J. Puerto, and C. Valverde, "Coordinating drones with mothership vehicles: The mothership and multiple drones routing problem with graphs," *Computers and Operations Research*, vol. 136, 9 2021.

[58] O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problems. ii: The p-medians," *SIAM Journal on Applied Mathematics*, vol. 37, pp. 539–560, 1979.

[59] J. Macqueen, "Some methods for classification and analysis of multivariate observations," pp. 281–297, University of California Press, 1967.

[60] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," pp. 1027–1035, 2007.

[61] B. Shingler, "'we are changing the world': Greta thunberg addresses hundreds of thousands at montreal climate march," *CBC News*, 2019.

Appendix A

A.1 Affect of varying coefficient weights on objectives

SN	α	β	γ	Obj 1	Obj 2	Obj 3	Solve Time (s)	Solution
0	0.1	0	0.9	29	27	0	369.21	2.5
1	0.1	0.1	0.8	20	6	0	78.26	1.2
2	0.1	0.2	0.7	20	6	0	67.13	0.6
3	0.1	0.3	0.6	0	0	0	67.01	0
4	0.1	0.4	0.5	0	0	0	54.75	0
5	0.1	0.5	0.4	0	0	0	48.86	0
6	0.1	0.6	0.3	0	0	0	48.88	0
7	0.1	0.7	0.2	0	0	0	45.40	0
8	0.1	0.8	0.1	0	0	0	43.85	0
9	0.1	0.9	0	0	0	0	50.22	0
10	0.2	0	0.8	29	27	0	603.71	5
11	0.2	0.1	0.7	29	18	0	1347.85	3.2
12	0.2	0.2	0.6	20	6	0	108.05	2.4
13	0.2	0.3	0.5	17	6	0	294.38	1.8
14	0.2	0.4	0.4	20	6	0	110.38	1.2
15	0.2	0.5	0.3	17	2	6	104.15	0.8
16	0.2	0.6	0.2	17	2	6	93.18	1.2
17	0.2	0.7	0.1	17	2	6	66.43	1.6
18	0.2	0.8	0	17	2	26	64.75	2
19	0.3	0	0.7	29	25	0	825.36	7.5
20	0.3	0.1	0.6	29	18	0	4432.77	5.7
21	0.3	0.2	0.5	20	6	0	3540.94	4.2
22	0.3	0.3	0.4	20	6	0	714.83	3.6
23	0.3	0.4	0.3	20	6	0	154.65	3
24	0.3	0.5	0.2	17	2	6	105.11	3.2
25	0.3	0.6	0.1	17	2	6	73.64	3.6
26	0.3	0.7	0	17	2	24	124.47	4
27	0.4	0	0.6	29	26	0	468.78	10
28	0.4	0.1	0.5	29	18	0	1737.32	8.2
29	0.4	0.2	0.4	29	18	0	13774.92	6.4
30	0.4	0.3	0.3	17	6	0	8092.35	5.4
31	0.4	0.4	0.2	17	2	6	1847.40	5.2
32	0.4	0.5	0.1	17	2	6	2026.20	5.6
33	0.4	0.6	0	24	4	23	299.89	6.4
34	0.5	0	0.5	29	26	0	845.15	12.5
35	0.5	0.1	0.4	29	18	0	5973.56	10.7
36	0.5	0.2	0.3	29	18	0	23970.55	8.9
37	0.5	0.3	0.2	29	8	12	86406.64	7.7
38	0.5	0.4	0.1	28	5	12	28397.16	8.3
39	0.5	0.5	0	24	4	23	174.65	9
40	0.6	0	0.4	29	26	0	339.22	15
41	0.6	0.1	0.3	29	18	0	2782.58	13.2
42	0.6	0.2	0.2	29	18	0	33493.89	11.4
43	0.6	0.3	0.1	29	8	12	11071.62	11.4
44	0.6	0.4	0	28	5	23	1095.50	11.8
45	0.7	0	0.3	29	25	0	464.08	17.5
46	0.7	0.1	0.2	29	18	0	6481.89	15.7
47	0.7	0.2	0.1	29	8	12	40556.03	14.7
48	0.7	0.3	0	29	8	18	712.03	15.1
49	0.8	0	0.2	29	24	0	495.61	20
50	0.8	0.1	0.1	29	12	6	34259.08	18.2
51	0.8	0.2	0	29	8	18	278.49	18.4
52	0.9	0	0.1	29	24	0	467.78	22.5
53	0.9	0.1	0	29	8	18	794.65	21.7
54	1	0	0	29	24	20	3897.35	25

A.2 Modified Formulation for Drone Route Optimization with Predetermined Truck Route

All parameters and decision variables remain the same as the original problem. A truck route is provided and used to create optimal drone routes. Let $Z = (t_1, i_1), (t_2, i_2), ..., (t_n, i_n)$ be the truck route represented as a sequence of tuples where the truck travels from node i at time t.

Model

Maximize

$$\times \sum_{u\in U}\sum_{t\in T}\sum_{(i,j)\in S_1} w_{ij}x^u_{tij} - \times \sum_{u\in U}\sum_{t\in T}\sum_{(i,j)\in S_2} b_{ij}x^u_{tij} \tag{1}$$

Subject to

$$y^u_{ti} \leq \sum_{(i,j)\in E} x^u_{tij} \qquad \forall u \in U, t \in T, i \in V_r \tag{2}$$

$$y^u_{ti} = 0 \qquad \forall u \in U, t \in T, i \in V : (t, i) \notin Z \tag{3}$$

$$v^u_{ti} = 0 \qquad \forall u \in U, t \in T, i \in V : (t, i) \notin Z \tag{4}$$

$$v^u_{tj} \leq \sum_{(i,j)\in E : t-b_{ij}\geq 0} x^u_{t-b_{ij},ij} \qquad \forall u \in U, t \in T, j \in V \tag{5}$$

$$\sum_{u\in U} x^u_{tij} \leq 1 \qquad \forall t \in T, (i, j) \in E \tag{6}$$

$$\sum_{(i,j)\in E} x^u_{tij} \leq 1 \qquad \forall u \in U, t \in T \tag{7}$$

$$\sum_{(i,j)\in E} x^u_{t-b_{ij},ij} + y^u_{tj} = \sum_{(j,k)\in E} x^u_{tjk} + v^u_{tj} \qquad \forall u \in U, t \in T : t - b_{ij} \geq 0, j \in V \tag{8}$$

$$\sum_{(i,j)\in E} \sum_{t'=t}^{min(t+b_{kl},T_m)} x^u_{t'ij} \leq x^u_{tkl} + M(1 - x^u_{tkl}) \qquad \forall u \in U, t \in T, (k, l) \in E \tag{9}$$

$$\sum_{t'=t}^{min(t+t_w,T_m)} \sum_{u\in U} x^u_{t'ij} \leq 1 \qquad \forall t \in T, (i, j) \in S_1 \tag{10}$$

$$\sum_{t'=t}^{min(t+t_u,T_m)} \sum_{i \in V_r} v_{t'i}^u \geq y_{tj}^u \qquad \forall u \in U, t \in T, j \in V_r \tag{11}$$

$$\sum_{t \in T} \sum_{i \in V_r} y_{ti}^u = \sum_{t \in T} \sum_{j \in V_r} v_{tj}^u \qquad \forall u \in U \tag{12}$$

$$x_{tij}^u, y_{tk}^u, v_{tk}^u \in \{0,1\} \qquad \forall u \in U, t \in T, (i,j) \in E, k \in V \tag{13}$$