OPTIMIZATION OF LAST MILE DELIVERY WITH UNMANNED AERIAL VEHICLE ASSISTANCE

by

Ben Remer

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering

Summer 2019

© 2019 Ben Remer
All Rights Reserved

OPTIMIZATION OF LAST MILE DELIVERY

WITH UNMANNED AERIAL VEHICLE ASSISTANCE

by

Ben Remer

Approved: _____

Andreas Malikopoulos, Ph.D.
Professor in charge of thesis on behalf of the Committee

Approved: _____

Ajay Prasad, Ph.D.
Chair of the Department of Mechanical Engineering

Approved: _____

Levi T. Thompson, Ph.D.
Dean of the College of Engineering

Approved: _____

Douglas J. Doren, Ph.D.
Interim Vice Provost for Graduate and Professional Education

ACKNOWLEDGMENTS

I would like to thank my advisor, Andreas Malikopoulos for guiding me these past couple of years, my labmates Logan Beaver and Behdad Chalaki for struggling with me, and my friends and family for supporting me as I've gone through this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In this thesis, we present an approach to optimizing the last-mile delivery route of a truck using coordination with unmanned aerial vehicles (UAVs). First, a traveling salesman problem is formulated to determine the truck's route. Then, a scheduling problem is formulated to determine the routes for the UAVs. A genetic algorithm is used to solve these problems, and simulated results are presented. The algorithm's complexity is analyzed. In preparation for future research, a small scale testbed platform is discussed and used to verify research on applying machine learning for traffic control. It demonstrates the first successful zero-shot transfer of an autonomous driving policy directly from simulator to a scaled autonomous vehicle under stochastic disturbances.

Chapter 1

LAST MILE DELIVERY IN THE LITERATURE

In a rapidly urbanizing world, we need to make fundamental transformations in how we use and access transportation. We are currently witnessing an increasing integration of our energy, transportation, and cyber networks, which, coupled with the human interactions, is giving rise to a new level of complexity in transportation [1]. As we move to increasingly complex transportation systems [2], new control approaches are needed to optimize the impact on system behavior of the interaction between vehicles at different applications.

Unmanned Aerial Vehicles (UAVs) are becoming increasingly available to be applied in civilian life, from drone racing to filming of events. In the past few years, applying UAVs to perform better delivery services has become a research topic, partially spurred by Amazon's "Air Amazon" delivery concept announced in 2013. Potentially, a delivery truck with a team of UAVs could increase both the time and energy efficiency of a last mile delivery service (that is, the portion of the delivery going from the distribution center to the final destination)[3].

1.1 Last Mile Delivery

Last mile delivery is a significant research area, considered the least efficient part of the delivery. Two major reasons for this are complicated and inefficient routes, and missed deliveries. Both of these issues exacerbate cost and environmental concerns. Additionally, with the rise of e-commerce, last mile delivery has become more and more important in our lives.

There are many different approaches in the literature, such as crowd sourcing of deliveries [4] in which the goal is to increase efficiency by decreasing missed deliveries.

Figure 1.1: An Amazon Drone Delivering a Package

Another approach in the literature are *Green Vehicle Routing Problems* [5][6]. These problems may sometimes try to increase efficiency of last mile delivery by reducing energy usage. Other novel concepts include delivery to trunks [7] which can significantly decrease the total distance traveled.

1.2 Last Mile Delivery with UAV Assistance

Interest in UAVs, which in this thesis is a term taken to be interchangeable with quadrotors and drones, has increased dramatically over recent years. This is in part due to manufacturing advances which have increased the capacity of lithium-ion batteries, a popular choice for UAVs, as well as the dropping price of carbon fiber, a popular choice of material for UAVs. Commercially, applications have arisen in aerial drone photography, drone racing, inspection, 3D Mapping, and more in a diverse set of industries including construction, real estate, agriculture as well as military and police applications. The drone market is rapidly growing and is expecting to grow past $12 billion by 2021 [8].

Recently, the concept of applying UAVs for deliveries has caught on, partially spurned on by Amazon's announcement in 2013 that they would begin making such deliveries [9]. Beyond Amazon, other companies such as DHL have also expressed interest in this solution [10].

Significant research efforts have focused on optimizing the UAV [11], including some relevant efforts into the control schemes of UAVs in conjunction with changing physical parameters due to the package delivery [12]. Additional relevant research includes landing a UAV on a moving target. This task is considered extremely difficult but has been successfully completed before [13]. The significant amount of research allocated to this problem suggests that it would be reasonable to considered the possibility while route planning if coordinating with a delivery truck.

On the operational research end, we can divide the research into 3 subareas: (1) Direct from depot, (2) holistic UAV-Truck delivery, and (3) non-holistic UAV-truck delivery. We define the direct from depot approach as any problem in which there is no coordination between a truck and UAV. In the holistic problem, there is coordination and the both the truck and UAV routes are determined, usually consecutively in that order. In non-holistic problems, there is coordination but the truck route is predetermined.

In the literature, direct from depot concepts are often but not always introduced along holistic approaches. Dorling et al. [11] introduce the Drone Delivery Problem as a MultiTrip Vehicle Routing Problem that allows the drones to be reused and takes into account changing weights as the drone makes deliveries. Mathew, Smith, and Waslander [14] cover in addition to a holistic approach, a Multiple Warehouse Delivery Problem, which while having some similarities to a holistic approach we consider it a direct from depot approach due to the lack of coordination. In the Multiple Warehouse Delivery problem, the UAV makes all deliveries - however there are additional warehouse nodes it can stop at, allowing for it to pick up packages or change batteries.

The breadth of the literature covering holistic approaches is the largest of our three categories. Early versions of this problem were introduced in 2015 [15],[14]. In

their approach, the authors make the assumption that the UAV and the truck will only meet up and depart at some truck delivery node – albeit Murray and Chu [15] permit the UAV to leave and enter the depot separately from the truck, Mathew, Smith, and Waslander [14] do not. This assumption was later relaxed in [16], permitting the UAV and the truck to rendezvous along any edge the truck is traveling. However, there is still the assumption that the truck is parked during this action. This limitation is unnecessary, as landing a quadrotor on a moving target has been explored with some promising results [13]. In this paper, we relax this constraint to explore the potential benefits. To the best of our knowledge this problem has not been addressed in this way in the literature to date.

Finally, there are some non-holistic approaches in the literature. In Boysen et al. [17] they take the case of having a fixed truck route and planning UAV deliveries around it. While this approach may seem lacking in comparison to holistic approaches, there are some advantages. Generally speaking, determining the truck route alone is an NP-hard problem, so by taking it to be known a large amount of complexity is removed from the problem. Furthermore, Boysen et al. note that "special planning requirements in specific logistics branches may require that the truck route is indeed already fixed when having to solve the drone subproblem...for attended home deliveries, time windows have to be agreed where customers are at home to personally receive their shipments from a human delivery person." [17]. In other words, real world restraints may drive the truck route to the extent where it is worthwhile to separate the truck route from the UAV sub-problem, one such driving factor may be predetermined delivery times that fix the truck route.

In several research efforts reported in literature, generally a variant of the Traveling Salesman Problem (TSP) has been formulated. In [14] the problem was addressed as a *heterogeneous delivery problem*, while in [15] the problem was addressed as the *flying sidekick traveling salesman problem*. In the TSP, there is a collection of cities, each of which must be visited exactly once by a salesman, and a depot that the salesman starts and ends at. The goal of the problem is to find the optimal route between these

cities. These classes of problems are NP-Hard, and as such the solutions are calculated offline. In the literature, it is shown that our problem is unique from the standard TSP, as we have multiple agents. In addition, in some formulations, and this is the case in this paper, we have delivery points (or cities) that can only be visited by a delivery truck since the packages might be too heavy for a UAV, and some that can be visited by either the delivery truck or the UAV [3],[15]. In addition, we formulate the problem such that the UAV is energy and range constrained. Additionally, we expand upon the literature by formulating the problem to allow n number of UAVs.

1.3 Contributions

In this thesis our contributions are formulating a unique problem, that permits a great amount of coordination between a truck and UAVs. Unlike similar efforts in the available literature, we coordinate with multiple UAVs and allow the UAVs to land on the truck while in motion. Finally, we submit a two level genetic algorithm solution to solve this formulation by dividing it up into a truck route and UAV route problem.

Chapter 2

UNIVERSITY OF DELAWARE SCALED SMART CITY TESTBED

Scaled testing has numerous advantages. There are many physical problems that can arise that simulation may ignore such as communication delay, system dynamic limits, and more. It is valuable to learn about and experience these limitations. On the other hand, full scales tests can be extremely expensive, difficult from a regulatory perspective, and often must be handled with extra caution for safety reasons. Additionally, full scale tests may be impractical due to the need of speed for successive tests or restrictions in scheduling because of weather or other external factors. A small scale testbed alleviates many of these problems, and can do a good job of balancing the advantages of simulations with the reality of full scale tests.

Over the past two years, the University of Delaware Scaled Smart City (UDSSC) testbed has been constructed and used for many publications. It has been constructed to be able to handle decentralized and centralized control of traffic [18], and has implemented the ability to handle interaction between cars and drones by using crazyflie drones and integrating crazyswarm's [19] API into the greater UDSSC architecture.

2.1 Construction and Implementation

The UDSSC covers a 20 foot by 20 foot area for testing at a 25 to 1 scale. Fig. 2.1 shows a birdseye view of the testbed. A mainframe computer can connect to over 30 cars via a wifi connection, giving the cars detailed instructions as to how to behave. Cars can be manually driven, be assigned a velocity curve, follow an intelligent driver model (IDM), or use plug-ins such as neural network trained function to determine how to drive. The mainframe has access to complete information about the UDSSC road network which is defined by 150 nodes, 102 edges, and 118 arcs, more clearly shown

Figure 2.1: The University of Delaware Scaled Smart City Lab

Figure 2.2: A typical Crazyflie Drone used in the UDSSC

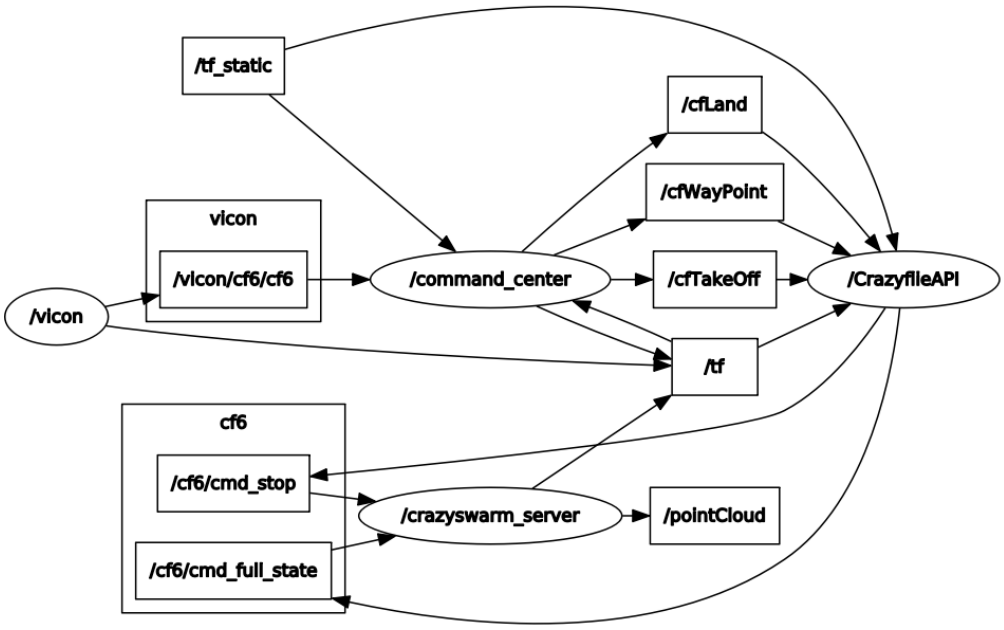

Figure 2.3: Graph of the ROS architecture with one active drone. Square blocks are topics and ovals are nodes.

in the appendix. With this information, algorithms such as A*, Dijkstra's, and others can be run to help determine paths for vehicles on the fly. Vehicle paths can also be predetermined by a user if required. Vehicle control at intersections can be defined by relevant functions for stop signs, traffic lights, or more advance control techniques. The UDSSC has been used to successfully test optimal control techniques for multiple scenarios.

A second computer connects to the Crazyflie drones via a radio connection. Currently the system communicates to up to 5 drones but theoretically can handle a significantly larger number [19]. A typical drone is pictured in Fig. 2.2. By using Robotic Operating System (ROS), the two computers are connected over an Ethernet connection and the drones can be controlled via Crazyswarm's API on the Mainframe computer. Drones have the ability to repeatedly take off, land, follow a trajectory in 2D space, or go to a waypoint in 3D space.

The cars, drones, and physical map location are all tracked using a Vicon camera system consisting of 8 Vantage cameras. This system does the necessary motion tracking to capture the vehicle frames in 3D space. Additionally, the Vicon Camera system can track markers physically attached to the map allowing calculations to be done independent of the calibration process. The Vicon system is connected to the mainframe computer via an ethernet connection. Vicon Bridge, a ROS node that uses Vicon's API, is used to integrate the camera system. Fig. 2.4 visualizes the data connections of the lab system. In addition to the vehicles and map, the Vicon camera system can be expanded to track any particular object of interest. Fig. 2.3 shows a visualization of the ROS architecture for the UDSSC, with one active drone-for each active object there is an additional topic under Vicon. For each additional active drone there is another topic located between the CrazyflieAPI and Crazyswarm Server nodes. The Command Center and vicon nodes are on the Mainframe computer. The CrazyflieAPI and Crazyswarm server nodes are located on the drone computer.

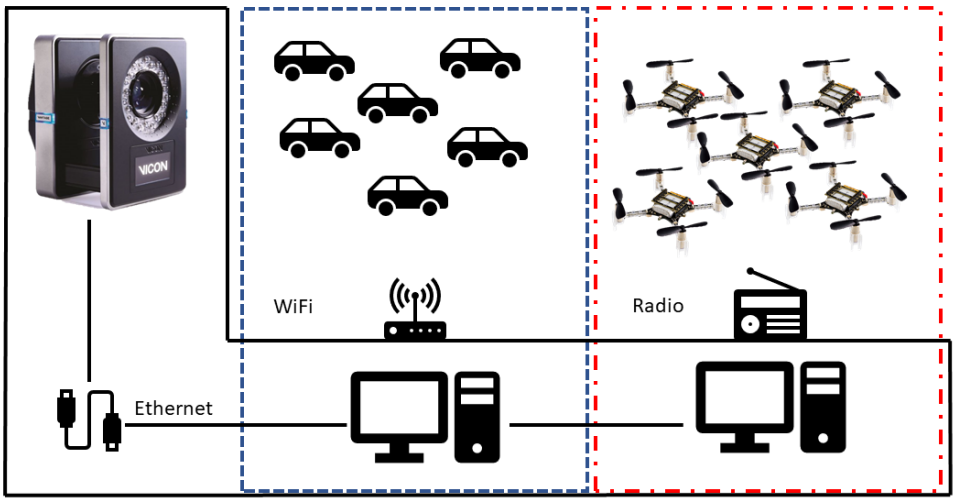

Figure 2.4: Data Connection of Laboratory Set Up

2.2 Testing of Reinforcement Learning

One of the several times the UDSSC has been used for small scale testing has been for experiments involving applying Reinforcement Learning (RL) to control traffic flow through a roundabout scenario to optimize traffic flow throughput. This work was done in partnership with the University of California Berkeley.

2.2.1 Background

RL is an approach to Machine Learning in which an agent learns how to maximize its reward by taking actions with respect to its observed state. In combination with simulation techniques, control of highly complex systems can be achieved. However, one drawback is that it is difficult to achieve "zero-shot" policy transfer (a transfer of the policy function from simulation to reality without additional refinements of the RL controller). In this work, we explored these issues and showed that with proper training techniques, zero-shot policy transfer can be achieved.

In RL, an agent iteratively receives sensory information describing its environment in the form of a set of states $\mathcal{S}$. Based on $\mathcal{S}$, the agent decides on what actions $\mathcal{A}$ to take to advance to the following state s'. These actions are chosen from a stochastic policy $\Pi : \mathcal{S} \rightarrow \mathcal{A}$. The goal of RL is to learn an optimal policy $\Pi^* : \mathcal{S} \rightarrow \mathcal{A}$ by maximizing $R = \mathbf{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$, where r_t is the reward received at time t. This process learns the best actions to take from any given state to maximize the expected cumulative reward.

There are a number of algorithms that exist for deriving an optimal RL policy Π^*. In this study, Π^* is learned via either *Proximal Policy Optimization* (PPO) [20] or *Trust Region Policy Optimization* (TRPO) [21], both policy gradient algorithms.

For the training of the RL policies, we used *Flow* [22], an open-source framework for interfacing RL libraries such as RLlib [23] and rllab [24] with traffic simulators such as SUMO [25] or Aimsun. *Flow* enables the ability to design and implement RL solutions for a flexible, wide variety of traffic-oriented scenarios. RL environments built using *Flow* are compatible with OpenAI Gym [26] and support training with most RL algorithms. *Flow* also supports large-scale, distributed computing solutions via AWS EC2 [1].

2.2.2 Training and Problem Formulation

For training of the various RL policies, a SUMO environment was created, modeled after the southeast corner of the UDSSC map. Two platoons enter the roundabout, one from the north and one from the west. The platoons take the routes through the roundabout show in Fig. 2.6. For the simulations, two groups of a random amount of vehicles are generated and released with a random offset into the roundabout. One group is released from the north (the north group) and one from the west (the west group). The groups are always led by an RL vehicle during training, in the baseline

[1] For further information on Flow, we refer readers to view the *Flow* Github page, website, or article, respectively listed here. Github: `https://github.com/flow-project/flow`, website: `https://flow-project.github.io/`, paper: [22].

scenario the RL vehicle is replaced by a standard IDM model. The RL vehicles are free to take any acceleration as their action.

The RL agents can observe the following variables, which together constitute the state of system:

- The positions of the AVs.
- The velocities of the AVs.
- The distances from the roundabout of the 6 closest vehicles to the roundabout for both roundabout entryways.
- The velocities of the 6 closest vehicles to the roundabout for both roundabout entryways.
- Tailway and headway (i.e. distances to the leading and following vehicles) of vehicles from both AVs.
- Length of the number of vehicles waiting to enter the roundabout for both roundabout entryways.
- The distances and velocities of all vehicles in the roundabout.
- Lengths of the incoming inflows.

All elements of the state space are normalized. Note that depending on the training method being used, the RL vehicles may or may not have noise in the state space. In the small scale experiments, noise comes from sensor error and other typical sources of noise.

Finally, there is the reward function the RL vehicles are learning to optimize. The reward function used for all experiments minimizes delay and applies penalties for zero velocities, near-zero velocities, jerky driving, and speeding.

$$r_t = 2 \cdot \frac{\max\left(v_{\max}\sqrt{n} - \sqrt{\sum_{i=1}^{n}(v_{i,t} - v_{\max})^2}, 0\right)}{v_{\max}\sqrt{n}} - p. \tag{2.1}$$

In this reward function, p is defined to be the sum of 4 different penalty functions, or $p = p_s + p_p + p_j + p_v$, where p_s is a penalty for vehicles traveling at zero velocity, designed to discourage standstill; p_p penalizes vehicles traveling below $0.2m/s$, which discourages the algorithm from learning a RL policy which substitutes extremely low velocities to

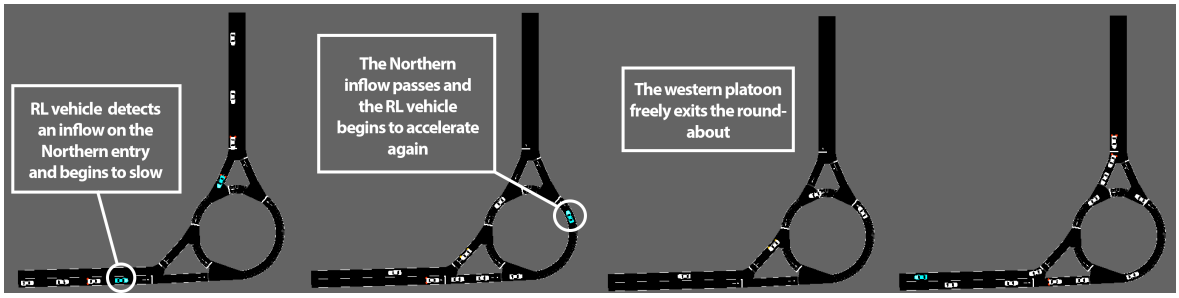

Figure 2.5: Two RL-controlled AVs trained with adversarial multi-agent noise demonstrate ramp metering behavior in this series of images, followed by an image of the baseline where vehicles clash. **First**: RL vehicle slows down in anticipation of a sufficiently short inflow from the north. **Second**: The northern inflow passes through the roundabout at high velocity. **Third**: The western group exits the roundabout at high velocity. **Fourth**: The baseline in which all vehicles are human-driven results in queues and clashing groups.

circumvent the zero-velocity penalty; p_j discourages jerky driving by maintaining a dynamic queue containing the last 10 actions and penalizing the variance of these actions; and p_v penalizes speeding.

Fig. 2.5 shows the vehicles in the sumo environment after some period of training. In general, the vehicles tend to learn a ramp metering behavior, in which the RL vehicle in the western group slows down the vehicle behind it to allow the northern group to pass in front. Seven different policies were trained. They were trained with either no noise, noise in the action produced, noised in the state, or noise in both action and state. Additionally, two different methods of noise creation were used. Adversarial noise, in which another policy is attempting to actively learn to perturb the RL vehicle's state and action in a zero sum game, and a Gaussian noise model in which values were perturbed by a Gaussian model with a standard deviation of 0.1 for the state space and 0.05 for the action space.

2.2.3 Experimental Results and Policy Transfer

The weights of the neural network generated by the the RL were exported in a '.pkl' file. The file was accessible to a python script on the mainframe, which took

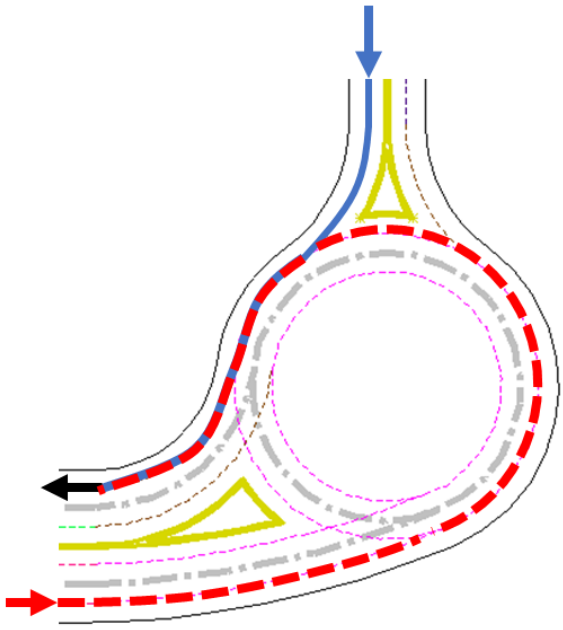

Figure 2.6: The Routes which were used in testing

an input of the state space measured through Vicon, identical to what was previously described, and had an output of the accelerations of the RL vehicles.

Testing was performed using the UDSSC. Vehicle platoons of random sizes were generated and entered the system a random distance away from the roundabout from both the North and Western entrances. Fig. 2.7 shows the two routes used. Additionally, the boxes labeled "N" and "W" shows the range of possible start locations for the first vehicle of the northern group and western groups respectively. After each experiment, a "flush" was performed in which the remaining vehicles passed through the system to allow the creation of extra baseline data (if the flush met conditions of minimum number of vehicles) and repeatable clean experiments. The random distances led to a ± 1.67s entrance time into the roundabout by the different groups of vehicles.

Table 2.1 shows the results of the experiments, with different training environments. For the baseline test, the lead vehicle of each platoon behaved like any non-RL vehicle. We succeeded in achieving improvements with the Action-State noise training

Table 2.1: Experimental results for RL Control of Roundabout Baseline (top), adversarial noise (middle) and Gaussian noise (bottom) cases. Arrows show change relative to baseline.

Noise Type	Mean Time (s)	Mean Speed (m/s)	No. Trials
Baseline	25.7 –	0.29 –	71
Action-State	25.0 ↓	0.25 ↓	33
Action	29.3 ↑	0.30 ↑	16
State	28.8 ↑	0.31 ↑	8
Action-State	30.3 ↑	0.21 ↓	7
State	31.9 ↑	0.30 ↓	7
Action	39.3 ↑	0.27 ↓	8
Noiseless	34.7 ↑	0.29 –	10

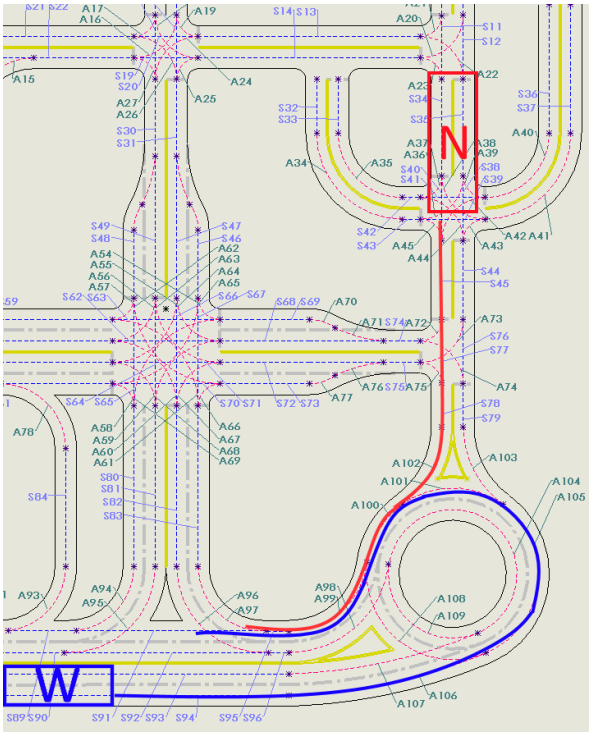

Figure 2.7: The Western and Northern Routes with Varied starts

case with adversarial noise. This result is promising as it represents successful control of a physical system by an RL policy trained solely in simulation. In all experiments, the ramp metering behavior was clearly learned, however in the Gaussian noise case the behavior was deployed too extremely causing reductions in efficiency.

Chapter 3

MATHEMATICAL FORMULATION OF LAST MILE DELIVERY WITH DRONE ASSISTANCE

Before diving into the problem, we review some graph theory. We use $\mathcal{G}(\mathcal{N}, \mathcal{A})$ to describe a graph with a set of arcs (or edges) $\mathcal{A}$ connecting a set $\mathcal{N}$ of N nodes (or vertices), i.e., $\mathcal{N} = \{1, ..., N\}, N \in \mathcal{N}$. Each node is indexed by $i \in \mathcal{N}$. For our problem, we take our graph to be fully connected, and any relevant geographic information, e.g. location of nodes, equations of paths, to be known. This road network of the truck-UAVs system is represented by the graph $\mathcal{G}(\mathcal{N}, \mathcal{A})$.

3.1 Modeling the Problem

We reserve $i = 1$ to be the depot node that the truck and UAVs must start and end together. In the example illustrated in Fig. 3.1, the depot node 1 is displayed in dark blue color. We define four disjoint sets $\mathcal{D}, \mathcal{H}, \mathcal{S}$, and $\mathcal{R}$, in which each individual node belongs to. The set $\mathcal{D} \subset \mathcal{N}$ is a set of delivery nodes that can be serviced by the UAVs, although the truck may still service them. In the example illustrated in Fig. 3.1, the set $\mathcal{D}$ has two nodes displayed in orange color. The set $\mathcal{H} \subset \mathcal{N}$ is the set of delivery nodes that must be reached by the truck only. In the example illustrated in Fig. 3.1, the set $\mathcal{H}$ has two nodes displayed in green color. The set $\mathcal{S} \subset \mathcal{N}$ is the set that includes the topology nodes (e.g., street intersections). In the example illustrated in Fig. 3.1, the set $\mathcal{S}$ has six nodes displayed in blue color. Finally, the set $\mathcal{R} \subset \mathcal{N}$ is the set of possible rendezvous and deploy nodes for the UAVs. Initially, this set is empty but an algorithm can identify and locate these nodes.

Fig. 3.1 shows the transformation from the original graph (left) to the graph (right) where the set $\mathcal{R}$ of rendezvous and deploy nodes is included. A second algorithm

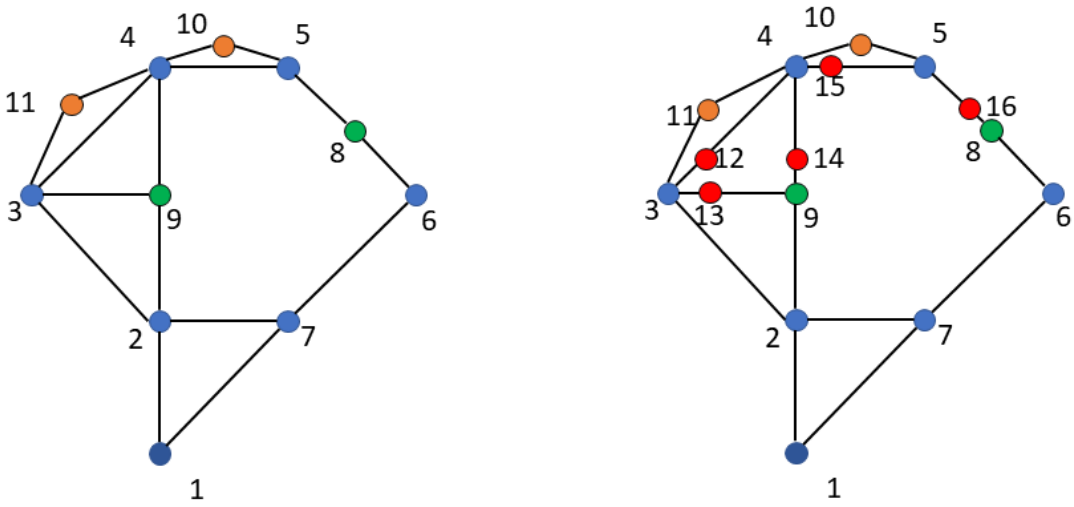

Figure 3.1: Left is the original graph $\mathcal{G}$. Right is the graph modified into $\mathcal{G}'$ by adding rendezvous nodes

can detect possible *sortie*. A sortie is defined as a 3-node route $i - j - k$ that a UAV can take, where $i, k \in \mathcal{R}$ and $j \in \mathcal{D}$. The set of all sorties that can visit a node i is denoted by $\mathcal{F}_i$, while an individual sortie is denoted s_i.

Finally, we find the shortest path between each member of the set $\mathcal{H} \cup \mathcal{D} \cup \mathcal{R} \cup 1$, and use this to create our graph $\mathcal{G}''(\mathcal{N}'', \mathcal{A}'')$ as shown in Fig. 2. As the dynamic element in our problem is the change in mass, the relative lengths between nodes do not change and we can safely make this formulation. This will allows us to constrain our solver to visit each node no more than once, but not prevent us from revisiting street nodes.

In our problem formulation, r denotes our decisions variable, which is an array of nodes that describes the route of the truck. We also denote with x_{ij} the edge that is selected. For instance, if $x_{12} = 1$, then the truck travels from node 1 to node 2. This, of course, occurs when node 1 is followed by node 2 in r. We define r_i to be the ith member of r and r_{-1} to be the last member. Thus, we have

$$x_{ij} = \begin{cases} 1 \text{ if } r_k = j \text{ and } r_{k-1} = i \text{ for some } r_k \in r \\ 0 \text{ otherwise.} \end{cases} \tag{3.1}$$

We are interested in both time and energy of the entire truck-UAVs system, and thus we formulate a multi-objective optimization problem as follows

$$\begin{aligned} &\min \big(\alpha E(r) + (1-\alpha)T(r)\big), \\ &0 \leq \alpha \leq 1, \end{aligned} \tag{3.2}$$

s.t.

$$\sum_{i \neq j} x_{ij} = 1 \quad \forall j \in H, \tag{3.3}$$

$$r_1 = 1, \tag{3.4}$$

$$r_{-1} = 1, \tag{3.5}$$

$$\sum_{i} x_{ij} \leq 1 \quad \forall j, \tag{3.6}$$

$$\sum_{i \in \mathcal{F}_j} \delta(s_i) > 0 \quad \forall j \in \mathcal{D} \notin r, \tag{3.7}$$

where,

$$\delta(s_i) = \begin{cases} 1 \text{ if rendezvous nodes of } s_i \in r, \\ 0 \text{ otherwise.} \end{cases} \tag{3.8}$$

$$\begin{aligned} &\sum_{i=1}^{N} e_{d,total,ij} \cdot x_{ij}^k + \sum_{i=1}^{N} e_{d,total,ji} \cdot x_{ij}^k \leq e_{batt}^k \\ &\forall j \in D \text{ and } \forall k \mid 2 \leq k \leq n+1. \end{aligned} \tag{3.9}$$

In our problem formulation above, α is a weighting parameter, $E(r)$ is the energy costs and $T(r)$ is the total time to complete deliveries. A breakdown of the costs $E(r)$ and $T(r)$ is presented in Section II. The delivery constraint (3.3) ensures truck deliveries are made, while the constraints (3.4) and (3.5) ensure the truck starts and ends at the depot node. The constraint (3.6) implies that we can only visit a node

once; however, recall this constraint applies on the transformed graph, so it is possible to revisit the same street nodes, if needed, due to the connectivity of the original graph. Finally, the constraint (3.7) states that for each delivery being made by the UAVs and not the truck, there is at least one possible sortie. This permits possible solutions for the scheduling portion of the problem. It does so by making use of a function $\delta(s)$ which probes if a sortie's end points are within the route.

Assumption 1: The UAV can fly in a direct route path to its destination node.

Assumption 2: There is no wind velocity and changes of elevations are negligible.

Assumption 3: The effect the package being delivered has on the quadrotor's frontal area and coefficient of drag is negligible.

Assumption 4: The UAVs travel from the truck, to a delivery point, and back to the truck. The UAVs land on top of, and take off from the top of the truck. Additionally, it is assumed that immediately after landing on the truck the UAVs have their batteries swapped and packages loaded by an automated system.

Assumption 5: Outside of initial and final acceleration and braking, the truck moves at the maximum permitted and constant speed along each edge.

3.1.1 Edge Energy Costs of the Truck in Transit

The truck's energy cost can be broken into the energy associated with the acceleration and cruising at a velocity [27, 28, 29, 30]. The mass of the truck will change as deliveries are made, monotonically decreasing throughout, for the TSP portion of our problem as we will ignore the mass fluctuation from the UAV riding the truck. The cost the truck endures from the UAV riding the truck will be taken into account during the scheduling portion of the problem.

$$e_{u,ij} = \int_{accel} \hat{a}_{ij}(c_0 + c_1 v + c_2 v^2)dt, \tag{3.10}$$

where

$$\hat{a}_{ij} = -(1)/(2M_i) \cdot C_d \rho_a A v^2 - \mu g + u_h, \tag{3.11}$$

Subject to:

$$\begin{aligned} u_{brake} \leq u_h \leq u_{acc}, \\ u_{brake} < 0 < u_{acc}, \\ 0 \leq v \leq V_{ij}, \end{aligned} \tag{3.12}$$

where $e_{u,ij}$ is the energy the truck consumes for a given input acceleration u, which for simplicity we will take to be the maximum or minimum value. $\hat{a}$ is the equivalent acceleration, u_{brake} and u_{acc} are vehicle parameters, A is the frontal area of the truck, M_{ij} is the mass of the truck across edge ij, C_d is the coefficient of drag, ρ_a is the density of air, μ is a friction coefficient, and u_h is the acceleration input. This model has the energy usage cut off when decelerating from a high velocity. We need also to include the energy associated with cruising the truck,

$$e_{cruise,ij} = \int_{cruising} b_0 + b_1 V_{ij} + b_2 V_{ij}^2 + b_3 V_{ij}^3 dt \tag{3.13}$$

where b_0, b_1, b_2, and b_3 are vehicle parameters. Therefore, the total energy the truck expends going from node i to node j is

$$e_{t,total,ij} = e_{u,ij} + e_{cruise,ij}. \tag{3.14}$$

This leads to our formulation of the energy costs for the problem.

$$\begin{aligned} E(r) = w_1 \sum_{i \in N, j \in N, i \neq j} e_{t,total,ij}(x_{ij}^1) \\ + w_2 \sum_{k=m-c+1}^{m} \sum_{i \in N, j \in N, i \neq j} e_{d,total,ij}^k(x_{ij}^k) \end{aligned} \tag{3.15}$$

where w_1 and w_2 are weights comparing the relative value of the energy spent of the UAV and the energy spent of the truck. For instance, this could correspond to the cost of gasoline for the truck and electricity for the UAV. Our energy model for the UAV gives power consumption in watt·seconds, but we convert to kWh before proceeding. The energy of the truck is measured in milliliters of fuel. Thus, a reasonable value for

w_1 could be around 0.000747 (based of of 2.87 dollars/gallon of fuel) and a reasonable value for w_2 would be around 0.12 (12 cents/kwh). Note that while $w_2 > w_1$, the truck energy term is expected to be significantly larger than the UAV energy term for a normal range of physical values.

3.1.2 Time Costs

The time costs to transverse an edge can be computed based on the velocity, accelerations, and the distance. In doing so, we apply Assumption 5 that outside of initial acceleration and final deceleration, the truck moves at a constant velocity equal to the maximum velocity permitted along that edge.

$$\begin{aligned} t_{t,ij} = (d_{ij} - v_{ij}^2/(2u_{acc}) + v_{ij}^2/(2u_{brake}))/v_{ij} \\ + v_{ij}/u_{acc} - v_{ij}/u_{brake} \text{ if } i \in \mathcal{H}, j \in \mathcal{H} \end{aligned} \tag{3.16}$$

$$\begin{aligned} t_{t,ij} = (d_{ij} + v_{ij}^2/(2u_{brake}))/v_{ij} \\ - v_{ij}/u_{brake} \text{ if } i \notin \mathcal{H}, j \in \mathcal{H} \end{aligned} \tag{3.17}$$

$$\begin{aligned} t_{t,ij} = (d_{ij} - v_{ij}^2/(2u_{acc})))/v_{ij} \\ + v_{ij}/u_{acc} \text{ if } i \notin \mathcal{H}, j \in \mathcal{H} \end{aligned} \tag{3.18}$$

$$t_{t,ij} = d_{ij}/v_{ij} \text{ if } i \notin \mathcal{H}, j \notin \mathcal{H} \tag{3.19}$$

Hence,

$$T(r) = \sum_{j \in N} \sum_{i \in N, i \neq j} t_{t,ij}(x_{ij}). \tag{3.20}$$

Note that the time for the UAV to cross any edge is not in our final equation. This is because the truck and UAV are constrained to rendezvous and must start and end together.

3.1.3 Edge Energy Costs of the UAV in Flight

From Assumption 1, we can generate a path for the UAV k between nodes. We can compute the energy cost of traveling from a node i to a node j, denoted $e^k_{d,total,ij}$, as a sum of the energy cost associated with ascending to that height, denoted $e^k_{a,ij}$, descending from that height, denoted $e^k_{d,ij}$, flying the distance between the two nodes, denoted $e^k_{t,ij}$, and the costs associated with performing the rendezvous maneuver (if applicable). We use the energy model described in [31] in which the authors reported results within 10% of tested quadrotors. We stress that our future analysis is not dependent on this model, but requires an energy model that is dependent on the changing mass of the quadrotor. Furthermore, we note that there are some particular quadrotor concepts reported in the literature [32] that this model might not be appropriate for as they adopt unusual rotor configurations that are not compatible with assumptions made in the energy model. Finally, we simplify this model further with Assumption 2. The energy cost of ascending with this model is

$$\begin{aligned} e^k_{a,ij} = (z - z_i)/V^k_a[d^k_2(m^k_{ij}g)^{3/2} + k^k_1 m^k_{ij} g \cdot \\ \cdot [V^k_a/2 + \sqrt{(V^k_a/2)^2 + m^k_{ij}g/(k^k_2)^2}]], \end{aligned} \tag{3.21}$$

where k^k_1 and k^k_2 are physical parameters of the UAV k that can be found experimentally, z_i is the height of the node i, g is acceleration due to gravity, m^k_{ij} is the mass of the UAV during the ascent (note: this value changes with the mass of the package being carried), and V^k_a is the velocity during ascent. The descent is formulated identically in 3.22, however V^k_d is the descent velocity and is negative

$$\begin{aligned} e^k_{d,ij} = (z_j - z)/V^k_d[d^k_2(m^k_{ij}g)^{3/2} + k^k_1 m^k_{ij} g \cdot \\ \cdot [V^k_d/2 + \sqrt{(V^k_d/2)^2 + m^k_{ij}g/(k^k_2)^2}]]. \end{aligned} \tag{3.22}$$

Finally, we look at the energy costs associated with the UAV traveling between two nodes. We consider the power loss from drag, the power to hover, and the profile

power.

$$
\begin{aligned}
e^k_{t,ij} = \int_{transverse} (P^k_{hover,ij} + P^k_{par,ij} + P^k_{p,ij})dt \\
P^k_{hover,ij} = d^k_1 (T^k_{ij})^{3/2} \\
P^k_{par,ij} = d^k_4 V^k_{ij} 3 \\
P^k_{p,ij} = d^k_2 (T^k_{ij})^{3/2} + d^k_3 (V^k_{ij}\cos(\alpha))^2 (T^k_{ij})^{1/2} \\
T^k_{ij} = \sqrt{(m^k_{ij} g - d^k_5((V^k_{ij}\cos(\alpha))^2)^2 + (d^k_4 V^{k2}_{ij})^2},
\end{aligned}
\tag{3.23}
$$

where d^k_1, d^k_3, d^k_4, and d^k_5 are physical parameters of the UAV l. α is the angle of attack. Changes in frontal area and drag coefficients with the change in the package being carried are negligible if Assumption 3 holds.

Taking these factors into consideration, the total energy cost for the UAV to transverse an edge ij while flying can be written as:

$$
e^k_{d,total,ij} = e^k_{t,ij} + e^k_{a,ij} + e^k_{d,ij} \text{ if } i \in \mathcal{D} \vee j \in \mathcal{D}. \tag{3.24}
$$

This is a function of the UAV's physical parameters, the time to transverse the edge, and the package weight. We will also have to address the case of the UAV riding along the truck. The energy model discussed for the UAV is only for UAV in flight, in other words one of the ends nodes is a delivery node. For when the UAV is on the truck, we model it as an additional cost of energy with the same parameters of the truck but with the same mass of the UAV.

3.1.4 Updating Mass

As deliveries are made, the mass of the truck and the UAV change. The mass of the truck goes down monotonically as deliveries are made-recall that even while the UAV is docked on the truck we handle the energy spent to move it separately from the truck. The mass of the UAV fluctuates with each delivery.

We denote M^*_0 the starting mass of the truck-UAV-packages system and M^f_0 the final mass of the system at the depot node. For every other node i, we denote M_i to

be the mass at that node, after any deliveries, departures, and rendezvouses are made. Finally, we define m_i to be the mass of the delivery at node i. If no delivery is to be made at node i, $m_i = 0$.

We can define the mass at each node as

$$M_j = \sum_{i \in N} x_{ij} M_i - m_j. \tag{3.25}$$

We separately consider the mass of the UAV k across edge ij to be

$$m_{ij}^k = \begin{cases} M_{UAV,k} + m_j \text{ if } j \in \mathcal{R} \\ M_{UAV,k} \text{ otherwise.} \end{cases} \tag{3.26}$$

3.2 Rendezvous and UAV Departure Points

The first step we will take to solve this problem, is to add plausible rendezvous and departure points to the original graph. From our battery constraint (3.32) we can deduce that any acceptable rendezvous and departure point $j \in \mathcal{R}$ to be paired with a delivery node $i \in \mathcal{D}$ for UAV k has a battery constraint

$$e_{batt}^k > e_{d,total,ij}^k. \tag{3.27}$$

Taking (3.21)-(3.23) and (3.27), the maximum distance the UAV can cover under this constraint can be solved for. Due to Assumptions 1 and 2, this is the same distance in any direction, as there will be no obstacles to avoid and no wind. We call this R_{max}. Finally, for each delivery node D_i we just need to find every edge within this radius. For each edge, we create a node along that edge, the precise location of the node is not defined, just that it is along that edge, and add an edge between that pair of nodes and the accessible delivery nodes. The algorithm is

```
1  for each edge ij
2      Create a Node that belong to set R;
3      for each d in D
4          for each UAV k
5          if ij is within R^k_max of d
6              Add edge between nodes along
   edge ij and delivery point;
7          end
8      end
9      if node on edge ij have no edges
   that belong to a node ∈ D
10         delete node;
11     end
12 end
```

Fig. 3.2 visualizes this problem, in which we would add 3 nodes as there are three edges. Two of the edges are reachable from the delivery node, and an edge is drawn between the rendezvous nodes and the delivery node. The next step after what is shown in the figure would be to delete the final node as it has no rendezvous edges connected to it. After finding the rendezvous nodes, we can then find each set of sorties, by applying the following algorithm

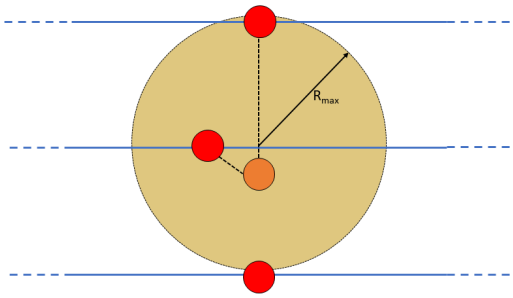

Figure 3.2: In the pictured example, plausible rendezvous nodes have been added in red. Note that the lowest edge is outside of the radius R_{max} and does not an edge connecting the delivery point and the rendezvous nodes.

```
1  for each j in 𝒟
2      Create empty set ℱ_j;
3      for each edge connected to the delivery j
   and node i ∈ ℛ
4          for Each edge connected to the delivery j
   and node k ∈ ℛ
5          if e_ij + e_jk <= battery_max
6              Add sortie i − j − k to set ℱ_j;
7          end
8      end
9  end
```

For each delivery node, we simply check each possible pair of edges between that node and a rendezvous node (note that in this case a "pair" can contain the same edge twice). If the sum of the energy costs of those two edge is less than the battery requirement of the UAV, it is added to set of sorties.

3.3 A Note on the Traveling Salesman Problem

At this point, our model is complete for what we will generally refer to as "The Traveling Salesman Problem" or TSP portion of our problem. In this twist on a classic

problem, the trucks route is generated and optimized, with energy considerations, constrained to visit each truck delivery and enough rendezvous and drone delivery nodes. While more detail is provided in the next section, a genetic algorithm has been employed to find this route. TSP problems are nonlinear, and NP-hard. Thus, unique algorithms designed to find very good solutions are used. While we refer to this solution as optimal, it is not guaranteed to be; it is optimal from the algorithm's standpoint. Genetic Algorithms are stochastic, and are not even guaranteed to find the same solution given the same input.

3.4 Scheduling Problem

After our algorithm has defined a route for the truck, a scheduling problem must be done for the UAVs. The optimal solution is one that minimizes the penalty function

$$J = \sum_{k=1}^{n} J^k, \tag{3.28}$$

where,

$$J^k(T_1^k, \ldots, T_n^k) = \sum_{i}^{n} C^k(T_i), \tag{3.29}$$

where $C^k(T_i)$ is the cost of UAV k doing job i at time T. It is derived from the geometry of the problem, the physical parameters of the truck and UAVs, and the energy models previously discussed. After an optimal solution has been found for the Scheduling problem, we will have to check each previously found unique TSP solution that satisfies the condition

$$TSP < TSP_O + SCHED_O \tag{3.30}$$

as it is possibly a better solution, with TSP_O and $SCHED_O$ being the previously found optimal solutions for the TSP and scheduling problems. We start by discretizing the problem, and for each sortie to perform the action on UAV k a we calculate the total cost and end time T_f^k for a given start time T_s^k. If the cost is greater than the battery constraint (3.32), then that time is excluded. We also ignore sorties that are

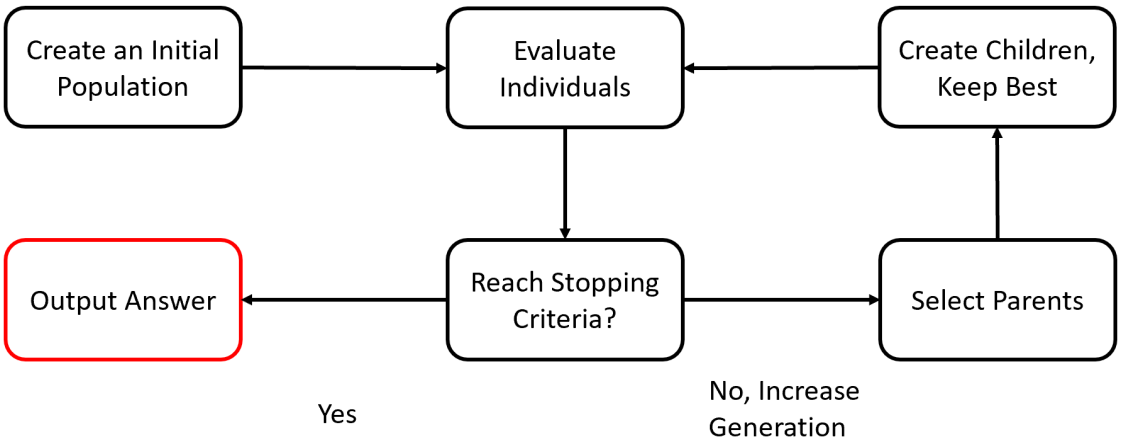

Figure 3.3: Flow chart of a typical Genetic Algorithm (GA)

not plausible due to having an end node not in our truck solution r. We apply the constraints

$$T_{i,f}^k < T_{i+1}^k. \tag{3.31}$$

In other words, a UAV cannot start a job until it has finished the previous one. In addition, we have the previously mentioned battery constraint

$$\begin{gathered}\sum_{i=1}^{N} e_{d,total,ij} \cdot x_{ij}^k + \sum_{i=1}^{N} e_{d,total,ji} \cdot x_{ij}^k \leq e_{batt}^k \\ \forall j \in D \text{ and } \forall k \mid 2 \leq k \leq n+1.\end{gathered} \tag{3.32}$$

We then seek a solution that minimizes (3.29) subject to (3.31) for a given set of N jobs on UAV k $(j_1, j_2, \ldots, j_N)$.

Two popular algorithms for solving similar problems are Simulated Annealing (SA) and Genetic Algorithms (GA). SA in particular has been used to solve problems relating to a truck-UAV tandem making deliveries [14]. In this thesis, we will take the opportunity to explore a genetic algorithm.

First, we will give a brief background on genetic algorithms, but encourage interested readers to investigate further outside of this thesis. Genetic Algorithms,

as the name implies, are inspired by biology. The first step is to create a randomly generated population, and then evaluate each individual member of the population. Some of the individuals are used as parents to create children, which are then evaluated, and the cycle continues until some stopping criteria is met (no change in best evaluation over time time, all constraints met, some sought value met, etc.). Fig. 3.3 has a flow diagram that shows this process. Additionally, it should be noted that GA has a history of being applied to solve TSP problems [33][34].

For our two problem approach, we first run a genetic algorithm for solving the truck problem, saving each solution that meets constraints. We then take the best solution, and run it through the genetic algorithm to solve for the scheduling problem of the UAVs. This increases the total cost, which we then compare to our previous answers. We take the lowest of our previous answers, and if it is less than our newly found complete solution, we test again, otherwise our algorithm is done, as written in the following algorithm

```
1  Run GA for TSP. Return each generation's best;
2  New Best=Run GA for scheduling with best
   value from TSP;
3  while New Best ≠ best value of solutions from TSP
4          Run GA for scheduling with best value from TSP
5          if New value < New Best
6                  New Best = New value;
7          end
8  end
9  return New Best;
```

Note, in this process we have essentially branched and bounded our search, allowing it to be run more efficiently. This end of this process is shown visually in Fig. 6. The orange horizontal line is the current best value, while the blue line shows the results of the genetic algorithm over generations. Each generation below the orange

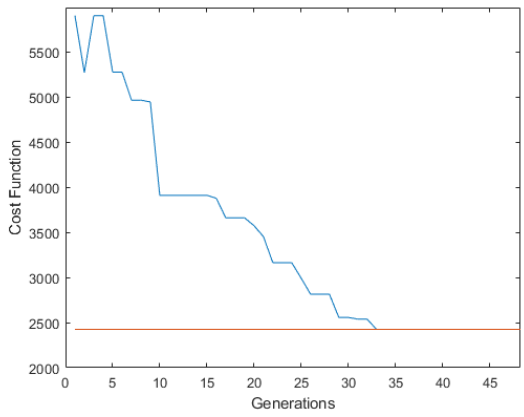

Figure 3.4: A visualization of a near complete result of the genetic algorithm. The horizontal line is the best value, and the orange line is the values the genetic algorithm achieved for the TSP. Any values below the line must be checked for the scheduling problem

line should be checked for the scheduling problem, if it hasn't been checked already.

3.5 Algorithm Analysis

To analyze our algorithm, we take the worst case scenario approach. We will use the variables A for number of arcs in the graph, N for the number of street nodes, D for the number of drone accessible delivery nodes, H for the number of truck delivery nodes, n for the number of drones, p for population in the genetic algorithm and m for the number of generations the genetic algorithm is capped to run at.

First, we consider our rendezvous node creation portion of our algorithm. This is simply a function of the number of arcs, A, and number of drone delivery nodes, D. Each arc is compared with each delivery nodes, or simply put

$$D * A = DA \tag{3.33}$$

in the worst case scenario, every delivery nodes is accessible from every arc. While this is great because it means our drones have incredible range, it means a large amount of rendezvous points, A, are inserted into the graph, doubling the number of arcs. It

should be noted this is a truly worst case scenario, as it becomes a lackluster application of this model. This would mean that every delivery point is accessible from the warehouse for the drones, and coordination with the truck is most likely not necessary. Instead, drone delivery direct from the warehouse should be investigated.

After this step, our total number of nodes is $N+D+H+A$ and we will attempt to simplify the graph. If we have a graph with T total nodes, the run time for finding an optimal route between any two given nodes is $O(T^2)$. We are interested in each pair of delivery and rendezvous nodes. The combinatoric equation that describes this can be written in this case as

$$P = (D + H + A)!/(D + H + A - 2)! \tag{3.34}$$

which simplifies to

$$P = (D + H + A)(D + H + A - 1) \tag{3.35}$$

from here, the time to find our simplified graph is

$$P(N + D + H + A)^2 \tag{3.36}$$

or rewritten for convenience as

$$(D + H + A)(D + H + A - 1)/(N + D + H + A)^2 \tag{3.37}$$

At this point, it should be noted the theoretical assumption made in this step, that is *it is assumed that the number of street nodes greatly exceeds the number of deliveries.* The point of this step is to simplify the graph. Thus we take $N >> (D+H)$. However, A is limited for any graph by N^2, thus we can write the complexity at this point as

$$O(A^4). \tag{3.38}$$

This results emphasizes the previous insight into how problematic for the complexity of the problem a large ranges for the drones can be, despite being a clearly positive quality in physical application. If a significantly smaller amount of rendezvous nodes is added, the complexity becomes $O(N^2)$. Next, we move on the TSP genetic algorithm

we have in place. The complexity is a function of the number of generations, the size of the population, and the size of individual members. For each generation the route must be cycled through to evaluate fitness (constraints met), and utility. Using R for the number of rendezvous node, we can write the complexity of the TSP as

$$O(m(3p(D+H+R))). \tag{3.39}$$

Recall in our worst case scenario $R = A$. This result occurs when the drones have a significantly large range. Due to our graph simplification in the previous step, the complexity for the genetic algorithm increases linearly with any variable. Note that without the previous simplification, evaluating fitness would have become non-linear.

Next, we take the drone scheduling portion of our problem. While checking if feasibility conditions are met can take some more intense calculation, the complexity of these calculations are constant, the number of which only increases linearly with the number of deliveries to be scheduled. In this case, we take the max - D deliveries. The complexity of this stage can be written as

$$O(m(3p(D))) \tag{3.40}$$

with the total complexity up to this point being

$$O(m(3p(D+H+A)) + m(3p(D)) + A^4) \tag{3.41}$$

if the range of the drone is sufficiently large, otherwise it is

$$O(m(3p(D+H+R)) + m(3p(D)) + N^2). \tag{3.42}$$

Finally, we must take into account our backtrack procedure. In the worst case scenario, we must check the scheduling problem for every generation. Thus our final complexity is

$$O(m(3p(D+H+R)) + m^2(3p(D)) + A^4) \tag{3.43}$$

or with low drone range

$$O(m(3p(D+H+R)) + m^2(3p(D)) + N^2) \tag{3.44}$$

.

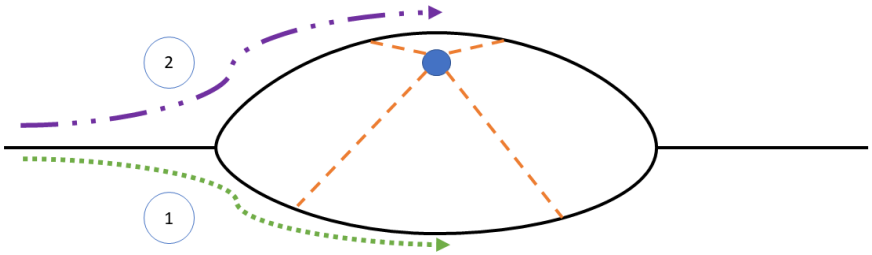

Figure 3.5: An instance in which failing to backtrack will give sub-optimal routes. Distances exaggerated for visibility.

3.5.1 Sub-optimal Heuristics

To improve algorithm speed, we propose a few heuristics that can reduce run time with minimal, although possibly present, effects on optimality.

Previously, to help ensure an optimal solution, we recommended checking all TSP solutions that fell under the condition

$$TSP < TSP_O + SCHED_O \tag{3.45}$$

where TSP_O was the previously found optimal TSP solution and $SCHED_O$ was the previously found scheduling solution. In practice, the TSP portion of the solution should generally dominate the final costs, and often if the solution $TSP_i > TSP_j$ then $TSP_i + SCHED_i > TSP_j + SCHED_j$ is often the case, even if $TSP_i < TSP_j + SCHED_j$.

To prove that a less optimal solution may be found, we take the following example as shown in Fig. 3.5. In this example, the lower route is found to be optimal by the TSP portion of the problem. However, it is only slightly better than the upper route for the TSP. The upper route is however, significantly better for the scheduling portion of the problem. Without backtracking to check both routes, there can be some loss of optimality. This heuristic greatly increases the efficiency of the algorithm in the worst case scenario. Recall that in the worst case scenario, we must run the scheduling problem for *every generation* that the genetic algorithm found a solution for the TSP, which in the worst case scenario is m generations. This led to the step in (3.43) in

which the complexity was multiplied by a factor of m. However, as discussed, as the cost is dominated by the truck, we still get reasonably good results by ignoring this backtracking procedure. Thus this heuristic drops our algorithm complexity to the result after (3.42).

Chapter 4

EXPERIMENTAL RESULTS

Simulations were run to estimate the improvement that could be achieved with drone coordination compared to a baseline scenario of a standard truck delivery route.

4.1 Simulation

Simulations were run in Matlab using the constants shown in Table 4.1. α was set to 0.9, greatly skewing the result towards time optimization. Delivery points were randomly generated, with a rule of no more than one per edge to guarantee a distribution of delivery points throughout the road network.

Table 4.1: Coefficients of Truck and UAV Energy Parameters

Truck	Value	UAV	Value
b_0	0.1569	k_1	0.8443
b_1	2.45×10^{-2}	k_2	$0.3051(kg/m)^{1/2}$
b_2	-7.415×10^{-4}	d_1	$2.8037(m/kg)^{1/2}$
b_3	5.975×10^{-5}	d_4	0.0296 kg/m
c_0	7.224×10^{-2}	d_5	0.0279 Ns/m
c_1	9.681×10^{-2}	-	-
c_2	1.075×10^{-3}	-	-

Fig. 4.1 shows the example of a network graph with delivery points and rendezvous points added into the graph. For each generated problem, a result for a baseline TSP -i.e. no drone assistance- and a result for 2 drones assisting the truck were found.

Table 4.2: Average Values of Cost Function

Assisted	Unassisted	Improvement	% Improvement
372.94	473.02	100.07	20.77%

Table 4.2 shows the average improvement - leading to a roughly 20% improvement. In no result did the drone assisted case ever perform worse than the truck by itself.

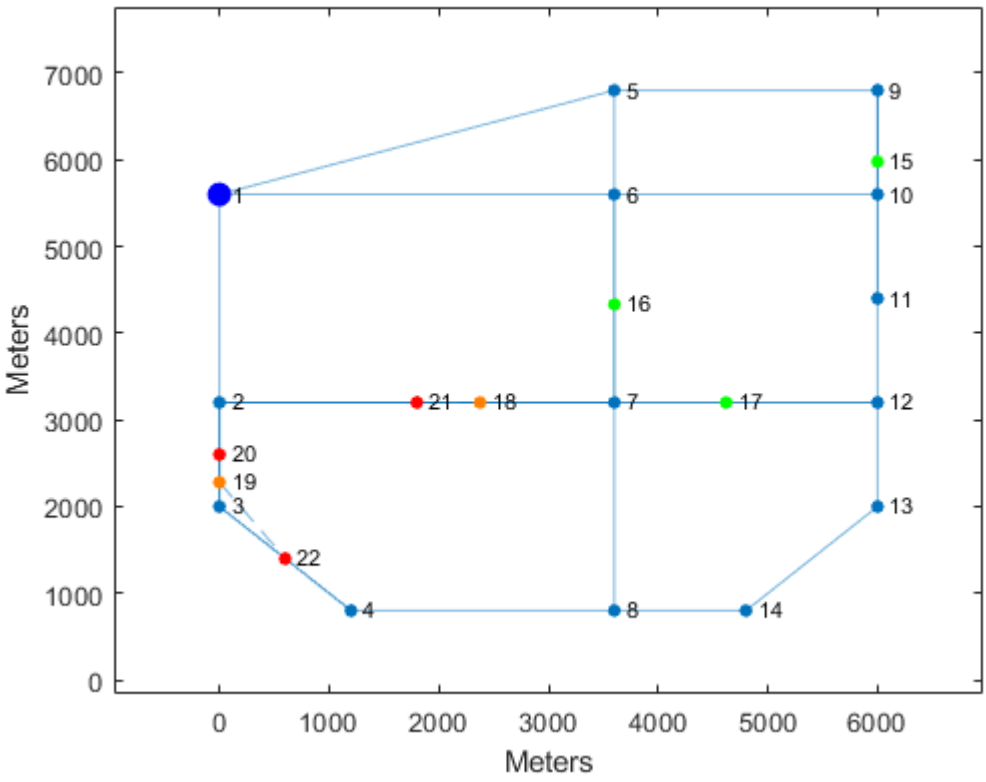

Figure 4.1: Network Graph after Delivery Points are Generated and the Rendezvous and Departure Point Algorithm has been run. Red points are possible rendezvous points, orange are drone-capable delivery points, and green are delivery points that require the truck.

BIBLIOGRAPHY

[1] Andreas A. Malikopoulos. Centralized stochastic optimal control of complex systems. In *Proceedings of the 2015 European Control Conference*, pages 721–726, 2015.

[2] Andreas A Malikopoulos. A duality framework for stochastic optimal control of complex systems. *IEEE Transactions on Automatic Control*, 61(10):2756–2765, 2016.

[3] Andrea Ponza. Optimization of Drone-Assisted Parcel Delivery. page 80, 2016.

[4] Yuan Wang, Dongxiang Zhang, Qing Liu, Fumin Shen, and Loo Hay. Towards enhancing the last-mile delivery : An effective crowd-tasking model with scalable solutions. *Transportation Research Part E*, 93:279–293, 2016.

[5] Canhong Lin, K.L. Choy, G.T.S. Ho, S.H. Chung, and H.Y. Lam. Survey of Green Vehicle Routing Problem : Past and future trends. *Expert Systems with Applications*, 41:1118–1138, 2014.

[6] Enjian Yao, Zhifeng Lang, Yang Yang, and Yongsheng Zhang. Vehicle routing problem solution considering minimising fuel consumption. *IET Intelligent Transport Systems*, 9(5):523–529, 2015.

[7] Damián Reyes, Martin Savelsbergh, and Alejandro Toriello. Vehicle routing with roaming delivery locations. *Transportation Research Part C*, 80:71–91, 2017.

[8] Divya Joshi. Commercial unmanned aerial vehicle (uav) market analysis – industry trends, companies and what you should know. https://www.businessinsider.com/commercial-uav-market-analysis-2017-8, 08-Aug-2017. Online; accessed 2019-03-14.

[9] Matt Hickey. Meet amazon prime air, a delivery-by-aerial-drone project. https://www.forbes.com/sites/matthickey/2013/12/01/meet-amazon-prime-air-amazons-delivery-by-aerial-drone-project/67b61d0779b2, 1-Dec-2013. Online; accessed 2019-03-17.

[10] Michael Franco. Dhl uses completely autonomous system to deliver consumer goods by drone. https://newatlas.com/dhl-drone-delivery/43248/, 10-May-2016. Online; aAccessed 2019-03-17.

[11] Kevin Dorling, Jordan Heinrichs, Geoffrey G. Messier, and Sebastian Magierowski. Vehicle Routing Problems for Drone Delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85, 2017.

[12] Animesh Kumar Shastry, Mahathi T Bhargavapuri, Mangal Kothari, and Soumya Ranjan Sahoo. Quaternion Based Adaptive Control for Package Delivery using Variable-Pitch Quadrotors. In *Indian Control Conference*, 2018.

[13] Lingyun Xu and Haibo Luo. Towards Autonomous Tracking and Landing on Moving Target *. 2016.

[14] Neil Mathew, Stephen L. Smith, and Steven L. Waslander. Planning Paths for Package Delivery in Heterogeneous Multirobot Teams. *IEEE Transactions on Automation Science and Engineering*, 12(4):1298–1308, 2015.

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

[16] Politecnico D I Bari, Leonardo Caggiani, Mario Marinelli, Mauro Dell Orco, and Michele Ottomanelli. En-route truck-drone parcel delivery for optimal vehicle routing strategies. 12:253–261, 2017.

[17] Nils Boysen, Dirk Briskorn, Stefan Fedtke, and Stefan Schwerdfeger. Drone delivery from trucks : Drone scheduling for given truck routes. *Networks*, (June):506–527, 2018.

[18] Adam Stager, Luke Bhan, and Andreas Malikopoulos. A Scaled Smart City for Experimental Validation of Connected and Automated Vehicles. In *IFAC Symposium on Control in Transportation Systems*, 2018.

[19] James A. Preiss*, Wolfgang Hönig*, Gaurav S. Sukhatme, and Nora Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3299–3304. IEEE, 2017. Software available at https://github.com/USC-ACTLab/crazyswarm.

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[21] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[22] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *arXiv preprint arXiv:1710.05465*, 2017.

[23] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Joseph Gonzalez, Ken Goldberg, and Ion Stoica. Ray rllib: A composable and scalable reinforcement learning library. *arXiv preprint arXiv:1712.09381*, 2017.

[24] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[25] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

[26] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[27] Jackeline Rios-Torres and Andreas A Malikopoulos. Automated and Cooperative Vehicle Merging at Highway On-Ramps. *IEEE Transactions on Intelligent Transportation Systems*, 18(4):780–789, 2017.

[28] Jackeline Rios-Torres, Andreas Malikopoulos, and Pierluigi Pisu. Online Optimal Control of Connected Vehicles for Efficient Traffic Flow at Merging Roads. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2015-Octob:2432–2437, 2015.

[29] Abdus Samad Kamal, Masakazu Mukai, Junichi Murata, and Taketoshi Kawabe. Model Predictive Control of Vehicles on Urban Roads for Improved Fuel Economy. 21(3):831–841, 2013.

[30] M. A.S. Kamal, Masakazu Mukai, Junichi Murata, and Taketoshi Kawabe. Ecological vehicle control on roads with up-down slopes. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):783–794, 2011.

[31] Zhilong Liu, Raja Sengupta, and Alex Kurzhanskiy. A Power Consumption Model for Multi-rotor Small Unmanned Aircraft Systems. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 310–315, 2017.

[32] S Driessens and P E I Pounds. Towards a more efficient quadrotor configuration. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1386–1392, 2013.

[33] Michael Manuel Smith and Yun Shiow Chen. A novel evolutionary algorithm for the traveling salesman problem. *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, (2):2515–2517, 2011.

[34] H.-K. Tsai, J.-M. Yang, Y.-F. Tsai, and C.-Y. Kao. An Evolutionary Algorithm for Large Traveling Salesman Problems. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, 34(4):1718–1729, 2004.

Appendix

UDSSC MAP

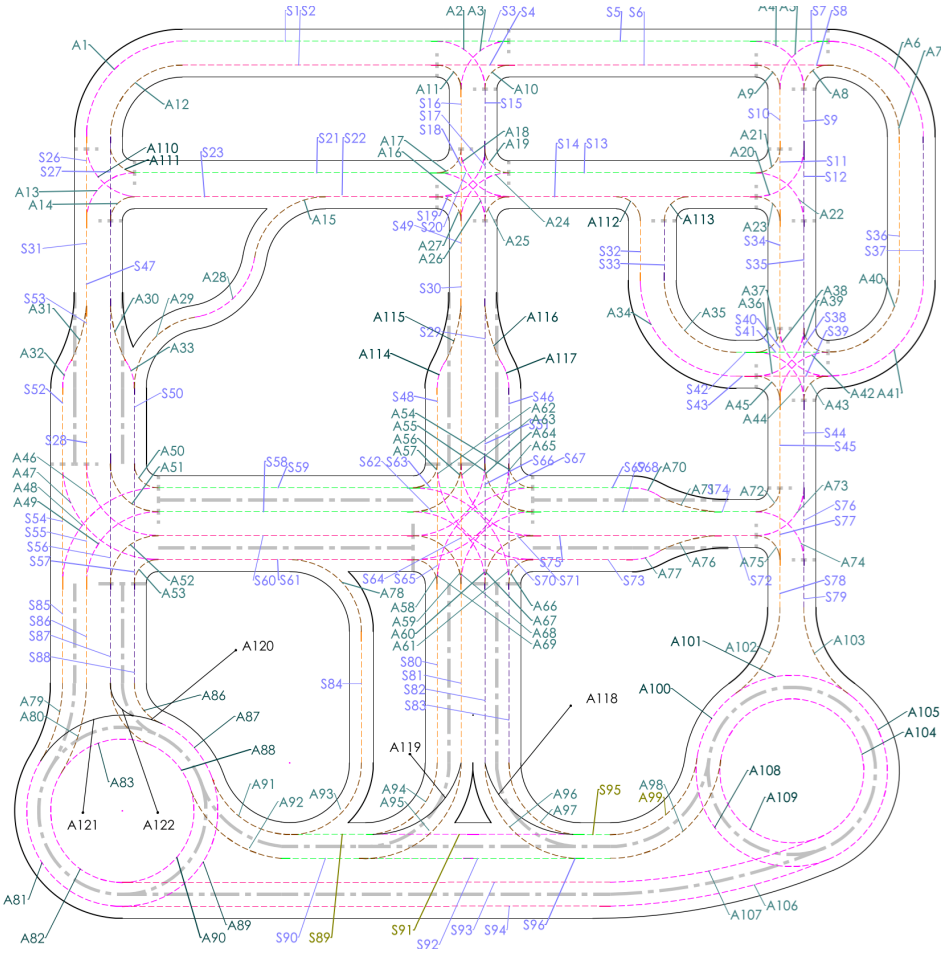

Figure A.1: Map of the UDSSC with Arcs and Edges labeled