A MULTI-UAV TRAJECTORY OPTIMIZATION METHODOLOGY FOR COMPLEX ENCLOSED ENVIRONMENTS

A Dissertation
Presented to
The Academic Faculty

By

Sarah Barlow

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Aerospace Engineering

Georgia Institute of Technology

May 2019

A MULTI-UAV TRAJECTORY OPTIMIZATION METHODOLOGY FOR COMPLEX ENCLOSED ENVIRONMENTS

Approved by:

Dr. Dimitri Mavris, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Simon Briceno
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Youngjun Choi
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: March 25, 2019

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Unmanned Aerial Systems (UAS) have become remarkably more popular over the past decade and demonstrate a continuous upward market trend. They are currently being used in various military and consumer applications and have recently gained recognition and potential for commercial purposes as well. As UAS become more accessible and advanced, they are able to be incorporated into a broader range of applications and provide substantial operational benefits.

An area that has great potential for UAS involvement are manufacturing and warehouse environments, as these typically occupy vast spaces. Warehouse logistics and operations are very complex and could significantly benefit from the integration of UAS. Many companies are already exploring using UAS for performing inventory audits to reduce labor costs and time, and improve accuracy and safety. To achieve the maximum benefit from this technology in these environments, multiple vehicles would be essential.

The purpose of this thesis is to optimize the operations of multiple UAS in complex confined environments such as a warehouse. There are added complexities when working with multiple vehicles; for example, ensuring that there are no collisions between vehicles. A great deal of research has been done on vehicle routing and trajectory optimization, but very little has been done with UAS optimization or path planning through these types of environments. This thesis further develops these algorithms and focuses in on the impact UAS involvement could have on warehouse-like operations. The proposed improvements from the current methods will help uncover the most optimal results by changing the process for finding solutions, the criteria under which solutions are ranked, and the operational/experimental setup.

The existing process begins by finding an optimal set of flyable routes for each UAS and then takes that set and offsets each UAS deployment time to ensure there are no collisions; this approach often results in sub-optimal solutions. The updated process that is presented

here calculates the offset deployment time for every set of flyable trajectories and only then does it choose the optimal routing. This new method will resolve the sub-optimality issues from the existing approach. Secondly, the current methods rank the optimal solution by minimizing the total flight time of all the UAS; this value is not the best representation of the actual overall mission time. Therefore, the highest-ranking solution may not actually correspond to the shortest overall mission time. The proposed method will seek to minimize the longest UAV flight time, as this is a better representation of the actual mission time. Lastly, the current methods have an experimental setup where each vehicle deploys from and returns to the same depot location. This approach requires additional vehicle setup time for each vehicle in the solution, prolonging the overall mission time. It also increases the chances of UAVs being in close proximity to one another, which in turn escalates the risk of collisions. The proposed method explores the benefits of multiple deployment spots to significantly improve mission times and simplify operations.

These proposed improvements will be assessed based on their degree of impact on the overall mission time compared to the current methods. They will also be assessed in comparison to one another and in combination with one another to better understand the effectiveness and sensitivities of the presented changes. The best combination will be further analyzed through a design of experiments by varying several inputs and examining the resulting fleet size, computation time, and overall mission time.

The main contributions of this thesis are:

- A new trajectory optimization algorithm which can be applied to multi-UAV use cases in confined and complex environments
- A method for reducing the dimensionality and complexity of a warehouse model which simplifies the computational expense of larger-scale problems
- A diverse set of parameter initialization combinations and corresponding response trends and feasibility

CHAPTER 1
INTRODUCTION

The concept of Unmanned Aerial Systems (UAS) has been around since the late 1800s and since then, UAS have continuously gained significant popularity and technological advancements, especially in the last few decades [1]. Unmanned Aerial Vehicles (UAV) were, at first, just a notion thought up by Nikola Tesla. He believed in the possibilities of engineering vehicles which could be controlled from afar and potentially used in combat situations. He constructed a prototype of a radio-controlled boat in 1898 and stated that the same technology could be applied to other vehicle types, such as unmanned aircraft. While working on this invention, Tesla additionally uncovered the possibilities of controlling several vehicles simultaneously and thus, envisioned an entire fleet of ”flying machines” [2]. Nikola Tesla's visions opened the floodgates for UAS development.

1.1 Background

Over the past century, UAVs have been used for numerous military applications and have recently reached consumer and commercial markets. The total UAS market size is currently around $20.71 billion and is expected to reach $52.30 billion by 2025 [3]. Within this, the commercial UAS market is around $0.882 billion and is expected to grow to $2.034 billion by 2022 [4]. This market growth, which can be seen in Figure 1.1, can be attributed to many factors, including new UAS regulations which became effective in August 2016, a decrease in component costs, allowing UAS to become more affordable, and advancements in technology, enabling integration into a wider range of applications. On the other hand, this market growth could be hindered by safety concerns surrounding the operation of UAS, security or privacy issues, strict government regulations, and the scarcity of air traffic management systems [4].

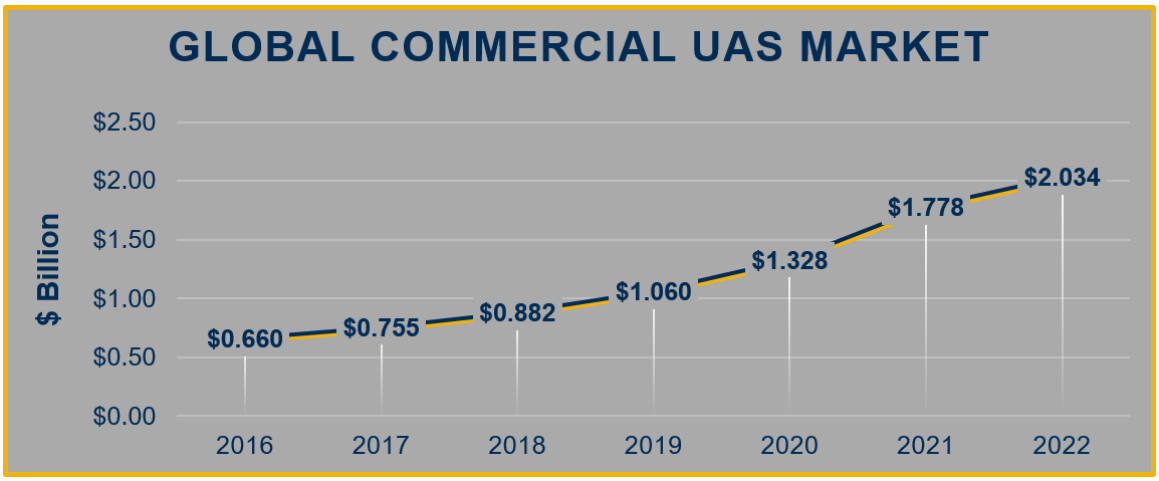

Figure 1.1: Commercial UAS Market Growth [4]

UAVs range in size from small consumer products to large military defense vehicles. Significant enhancements have been made to UAS technology that enables these vehicles to fly faster, longer, have more stability and control, and even carry payloads. Improvements on batteries have equipped UAS with higher endurance. This extended battery life allows vehicles to fly longer ranges before needing to recharge. UAS are remote piloted to deploy, fly a mission, and land within their endurance range [1]. As technology continues to advance and UAS become more accessible, the feasibility and benefits of the integration of these vehicles will drastically expand, as they will be enabled to support a plethora of new operations [5]. Some of the operations currently being considered and explored are site surveying, terrain mapping, natural disaster monitoring, package delivery, wildlife preservation, search and rescue missions, traffic management, and manufacturing/warehouse inventory tracking [3]. With improved sensing technology that has become more readily available, the smaller UAS are safely able to reach new areas and can even begin venturing indoors [1]. Indoor use of UAVs evades the barriers of government/FAA regulations, which allows for more design freedom. Typically, the use cases surrounding UAS occur in outdoor settings, but manufacturing and warehouse environments pose an interesting indoor use case due to their vast sizes. Inside the massive warehouse and manufacturing spaces there exists very complex logistics and operations, where UAVs could

prove to be advantageous.

Large manufacturing and warehouse environments routinely and frequently perform inventory audits to track the products, supplies, and equipment they have on hand. It is important to keep an accurate count because much of the inventory is expensive or may be needed to fulfill customer orders. The current typical process of performing one of these audits consists of workers manually scanning each and every item. This proves to be a very time consuming and labor intense practice which is extremely prone to human error due to the constant repetition [6] [7]. In one study, performing an inventory audit of a 26m by 24m section of a warehouse takes workers approximately 40 hours to complete with only about 90% accuracy [8]. Due to the extensive time this process takes, it is impossible to maintain real-time or even remotely close to real-time inventory data. There are also several hazards which pose safety risks to the workers involved in the current inventory tracking method [9]. Warehouse and manufacturing environments typically house dangerous heavy machinery, minimal air conditioning due to their immense size, very loud surroundings, and items stacked high and stored on very tall shelves [6]. Growing demands to stay competitive, improve customer satisfaction, improve warehouse safety, and increase cost and time savings create a need for more efficient and accurate processes [10] [11].

Some companies with large warehouses, such as Amazon, Walmart, and Ikea, have begun exploring the benefits of incorporating UAS into their business models [10]. UAS platforms have also begun exploring the possibilities of expanding and accommodating to these types of environments by packaging together the necessary technologies that a UAV may need to be fit for such a task. Some of these integrated systems include Infinium Scan and EyeSee, which are UAS platforms designed specifically for warehouse environments and inventory tracking [12]. These platforms do not incorporate the path planning and mission planning that are necessary for this technology to be as successful and beneficial as possible [13]. The rising interest in assimilating UAVs in manufacturing and warehouse environments is the motivation to further understand the operations of the two components

when coupled together.

1.2 Motivation

As discussed, warehouse environments are prone to significant challenges, such as dangerous working conditions, inaccuracies in inventory counts, and lengthy time-consuming processes. Assumptions can be made that the assimilation of UAVs into the inventory audit mission will decrease the need for employees to continuously climb to high shelves and reduce their time spent in these laborious conditions with dangerous machinery. Additional assumptions can be made that inventory accuracy will be strengthened with UAV involvement as the process will no longer be vulnerable to human error. The final metric to explore is the reduction of inventory audit mission time with the incorporation of UAS technology. As the easiest metric to quantify, mission time will be used to assess the benefit of UAVs within warehouse-like enviornments. Overall, the integration of UAS in these environments could provide substantial labor cost and time savings, while also improving employee safety and count accuracy with closer to real-time data and fewer missed items.

A typical consumer quadcopter UAV has an endurance ranging from 10 to 30 minutes [1] and the average Walmart warehouse ranges between 93,000 to 149,000 square meters with shelves reaching 11 meters high [14]. The floor area of a 93,000 square meter warehouse could fit approximately 17 football fields. These vast manufacturing and warehouse spaces, coupled with short endurance vehicles, supports the need for a multi-UAV scenario. Multiple UAVs would be able to cover larger areas and reduce mission times even further than a single vehicle. Adding in additional vehicles creates other complexities that need to be addressed, which motivates the need to figure out how to safely and efficiently deploy and operate multiple vehicles at the same time within these confined environments. It is essential to figure out the best coordination and operations of UAVs to tackle the entire warehouse layout to achieve the maximum benefit from this technology.

Due to the complexities of warehouse areas, the solution to this problem could be ap-

plied to other confined or enclosed environments with few modifications, helping to fill the gap in UAV operations in such environments.

All of this information leads to the following overall goal:

Research Objective: Improve/optimize the total mission time of multiple UAVs performing a warehouse inventory for safety, cost, and efficiency.

The proposed approach for accomplishing this is through a multi-vehicle trajectory optimization algorithm [15] [16] [17].

CHAPTER 2
PROBLEM FORMULATION

2.1 Current Methods

There are many trajectory generation techniques that exist to help solve coverage path planning problems. Several methods were studied to determine the best method for handling the complex enclosed environment UAV use case [18] [19]. First, cellular decomposition, which consists of breaking an area down into smaller subsections (cells) around obstacles using a variety of techniques, such as triangular, trapezoidal, morse, or boustrophedon decomposition. Each of these smaller areas become much simpler to solve and can then be routed to other adjacent cells to complete the coverage path planning problem [20] [21]. Figure 2.1 depicts the cellular decomposition method. Next, a wavefront algorithm was investigated. This method involves locating a starting node and an ending node and then propagating a wave outwards, labeling each of the nodes based on distance. Using these labels, a path can be created through the entire area; this is shown in Figure 2.2. Lastly, the technique of vehicle routing was examined. This method consisted of a depot and a set of waypoints, which all needed to be visited to complete the entire area [22]. A depiction of vehicle routing can be seen in Figure 2.3. The vehicle routing technique is used for the warehouse inventory tracking problem, as this method can easily handle the three dimensional environment as well as the specific waypoints that need to be reached to scan shelves/products.

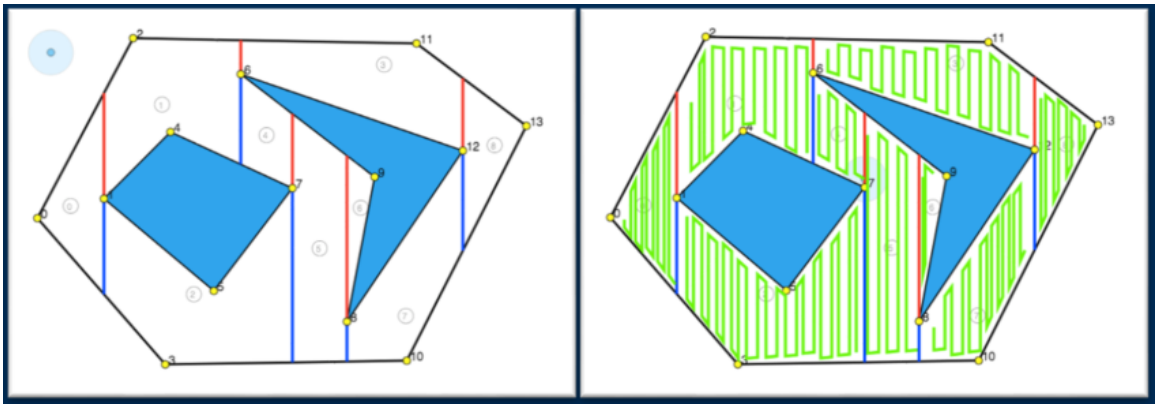

Figure 2.1: Cellular Decomposition [23]

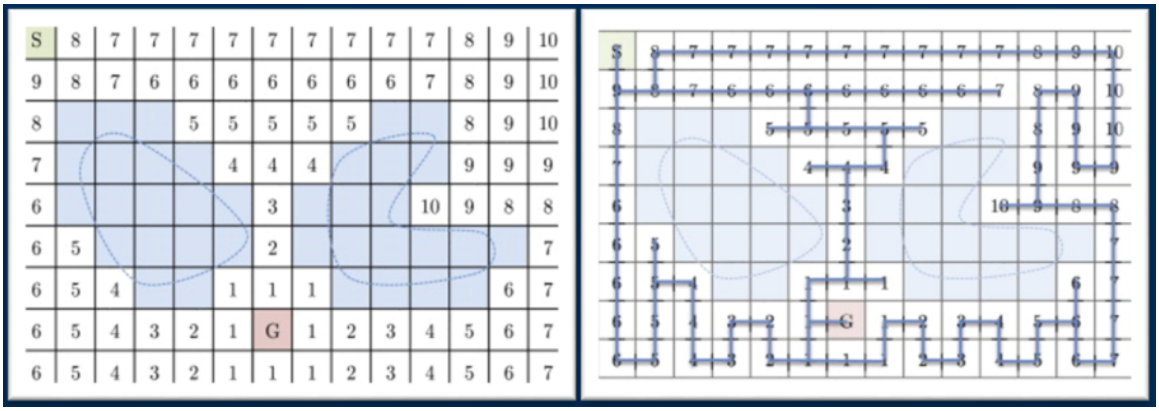

Figure 2.2: Wavefront Algorithm [21]

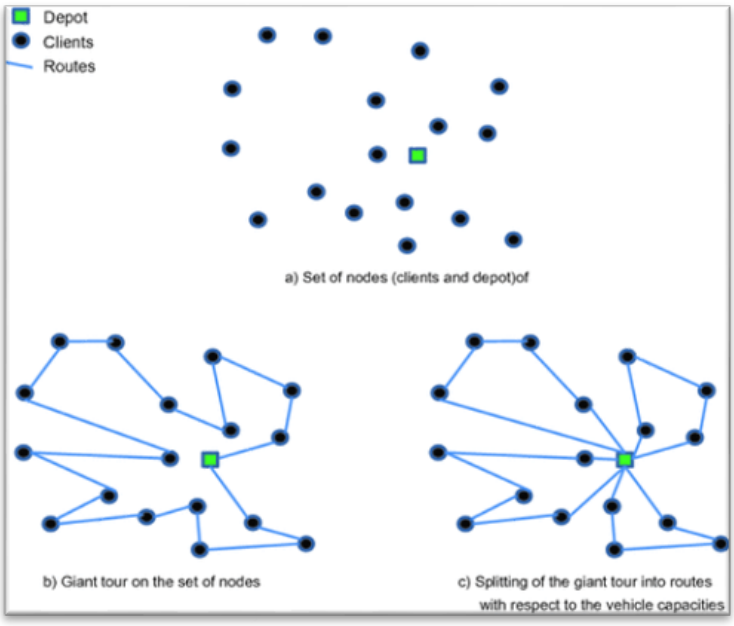

Figure 2.3: Vehicle Routing [22]

Extensive research has been done on vehicle routing problems, the most famous being the Traveling Salesman Problem, discussed in Section 2.2. Several algorithms have been created for a wide variety of routing problems [24]. Research involving multiple vehicle path planning is reasonably available, but of that research, the prevalence of UAS is scarce [25]. Additionally, investigations surrounding single or multi-UAV operations within warehouse/manufacturing or other similar environments is virtually nonexistent. There exist several trajectory optimization algorithms which can be modified or enhanced to fit the needs of UAS technology and warehouse/manufacturing organization.

These algorithms share some very important foundations. Each of the constructed models form a Hamiltonian Cycle. Within graph theory, this is accomplished when each node in a graph is passed through exactly once, thereby connecting the entire graph. An example of a Hamiltonian Cycle is depicted in Figure 2.4. These models can be either directed (asymmetric) or undirected (symmetric) graphs [26]. In a directed graph, an edge connecting two nodes can only travel in one direction, or the path from the one node to another is not equivalent if the order visited was reversed. For undirected graphs, the opposite is

true, the order in which nodes are visited does not matter as the values are the same in either direction [27]. The formulation of undirected graphs is the focus for the UAS trajectory problem. Hamiltonian Cycles are classified as NP-complete because once a solution is found, it is verifiable that all nodes have been visited exactly once. On the other hand, the algorithms created for optimized vehicle routing are classified as NP-hard because, as the number of nodes in a graph increases, the solution's optimality cannot be verified in a finite amount of time [28]. These problems become very computationally expensive as the size increases [29].

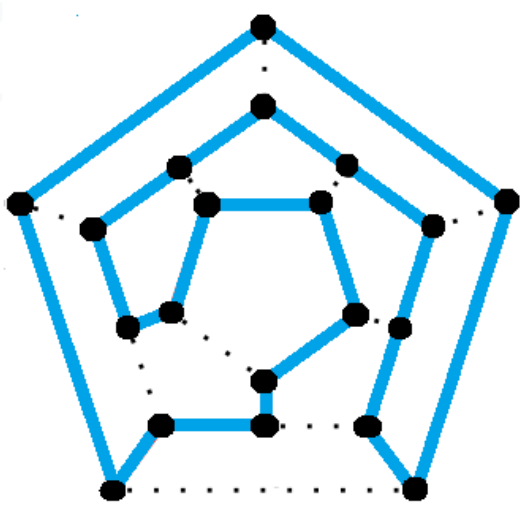

Figure 2.4: Hamiltonian Cycle [30]

The current methods explored here are all defined by complete undirected graphs, $G = (\mathcal{N}, \mathcal{A})$. Where $\mathcal{N}$ is the set of vertices, or nodes, in the graph, $\mathcal{N} = \{0, 1, 2, \ldots, n\}$, and $\mathcal{A}$ is the set of edges, or arcs, that connect the nodes, $\mathcal{A} = \{(i, j) : i, j \in \mathcal{N}, i \neq j\}$. There is then an associated cost set analogous to the edge set, $\mathcal{C} = (c_{i,j})$, this cost can be defined by several different parameters depending on the problem, such as distance, time, labor/operation cost, etc.. In an undirected/symmetric graph, $c_{i,j} = c_{j,i}$ [31]. These graphs are created through each optimization algorithm.

Optimization algorithms consist of three main components: decision variables, constraints and an objective/cost function. The three in combination define an algorithm that

assigns the best values to the design variables based on the goal of the objective function, typically minimizing or maximizing, while adhering to the constraints of the problem.

2.2 Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a widely discussed phenomenon in combinatorial optimization, a subsection of operations research [27]. The TSP is so popular because its concept can easily be applied in many different capacities, such as the production of integrated circuits (ICs) and printed circuit boards (PCBs), shift scheduling, computer wiring, genome mapping, package delivery routes, and many more [31] [32]. The goal of the TSP is to find the shortest path, or tour, that a salesman should take through a given set of cities, where each city is visited exactly once and the salesman returns back to the originating point, or depot, upon completion of the tour. Therefore, the objective function of this problem is to minimize cost, where the cost is equivalent to the length of the tour [31] [32] [33].

2.2.1 Model Formulation

The following are the variables used in the Dantzig et al. TSP model [34]:

$x_{i,j}$: Binary decision variable, where:

$$x_{i,j} = \begin{cases} 1, \text{ if an edge from } i \text{ to } j \text{ appears in the optimal tour} \\ 0, \text{ otherwise} \end{cases}$$

$c_{i,j}$: Cost associated with traveling from city i to city j

The TSP is setup as follows:

$$Minimize \sum_{i<j} c_{i,j} x_{i,j} \tag{2.1}$$

Subject to the following constraints:

$$\sum_{i<k} x_{i,k} + \sum_{j>k} x_{k,j} = 2 \qquad (k \in \mathcal{N}) \tag{2.2}$$

$$\sum_{i,j \in S} x_{i,j} \leq |S| - 1 \qquad (S \subset \mathcal{N}, 3 \leq |S| \leq n-3) \tag{2.3}$$

$$x_{i,j} = 0 \text{ or } 1 \qquad ((i,j) \in \mathcal{A}) \tag{2.4}$$

The objective function, equation (2.1), seeks to minimize the cost of the tour. Equations (2.2), (2.3), and (2.4) impose the degree, subtour elimination, and integrality constraints, respectively [31]. The subtour elimination constraint ensures that the tour created is fully connected, such that the salesman cannot just "appear" in a city, a visual representation of this can be seen in Figure 2.5.

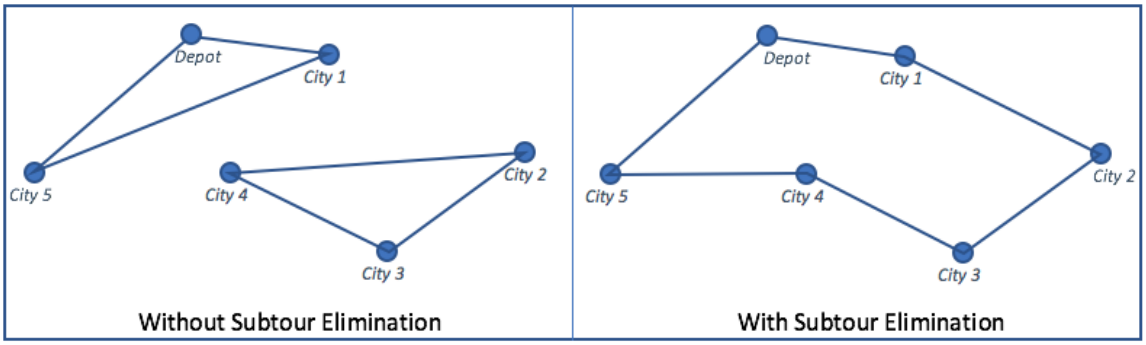

Figure 2.5: Subtour Elimination Constraint

The solved TSP model has assigned values to the decision variable, $x_{i,j}$, corresponding to the optimal tour route. An example of a solved TSP model is shown in Figure 2.6.

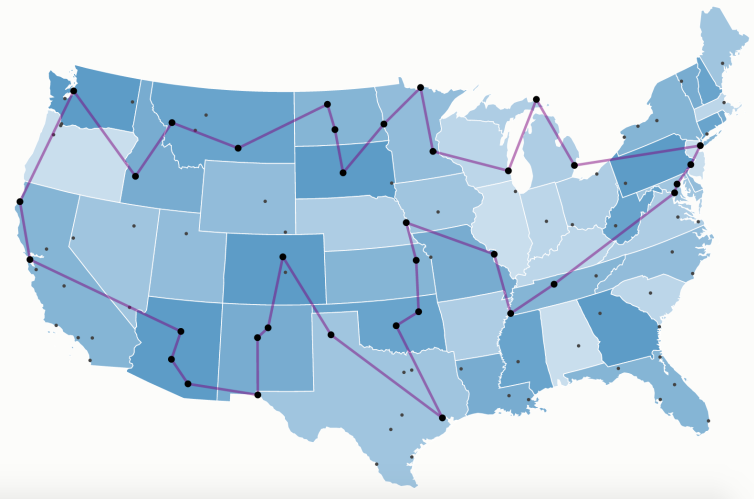

Figure 2.6: Possible Solution to TSP [35]

While the TSP is a very extensive and powerful model, it is computationally expensive and would need several modifications in order to be applied to the warehouse inventory problem [36] [37]. The main issue is that the TSP only formulates one tour for the one salesman that is traveling, whereas the warehouse problem requires a fleet of UAS, so each vehicle would need its own path. The TSP is a great baseline for understanding the generalities of a UAS trajectory optimization.

2.3 Vehicle Routing Problem

The Vehicle Routing Problem (VRP) generalizes the Traveling Salesman Problem. Similar to the TSP, the objective is to minimize the cost of the tours [38] [39]. The VRP has numerous variations, which can change the variable setup, the applied constraints, or the objective function [40] [41]. Some of the variants include [42] [43] [44]:

- Capacitated Vehicle Routing Problem (CVRP): Vehicle load cannot exceed its capacity
- Vehicle Routing Problem with Time Windows: Nodes must be visited within a specified time frame

- Distance-Constrained Vehicle Routing Problem (DVRP): Vehicle cannot exceed its maximum range
- Multiple Vehicle Routing Problem (mVRP): Multiple tours are created, one for each vehicle

The variations above are all subject to each node being visited exactly once by only one vehicle, and each vehicle starting and ending its tour at the depot [43] [45]. The most relevant deviations to model the UAS inventory audit problem are the DVRP and the mVRP.

2.3.1 Model Formulation

Developed by Kara [46], the following is a formulation of an arc based Multi-Vehicle Distance-Constrained Vehicle Routing Problem (mDVRP), beginning with the model's variables:

$x_{i,j}$: Binary decision variable, where:

$$x_{i,j} = \begin{cases} 1, \text{ if an edge from node} i \text{ to node } j \text{ appears in the optimal tour} \\ 0, \text{ otherwise} \end{cases}$$

$y_{i,j}$: flow variable, where:

$$y_{i,j} = \begin{cases} \text{total distance traveled from the depot to node } j \text{ through } i, \text{ if } x_{i,j} = 1 \\ 0, \text{ otherwise} \end{cases}$$

$d_{i,j}$: Distance associated with traveling from node i to node j

m: Number of identical vehicles

D: Maximum distance each vehicle can travel

Objective Function:

$$Minimize \sum_{i=0}^{n} \sum_{j=0}^{n} d_{i,j} x_{i,j} \tag{2.5}$$

Subject to the following constraints:

$$\sum_{i=1}^{n} x_{0,i} = m \tag{2.6}$$

$$\sum_{i=1}^{n} x_{i,0} = m \tag{2.7}$$

$$\sum_{i=0}^{n} x_{i,j} = 1 \qquad (j = 1, 2, \ldots, n) \tag{2.8}$$

$$\sum_{j=0}^{n} x_{i,j} = 1 \qquad (i = 1, 2, \ldots, n) \tag{2.9}$$

$$x_{i,j} \in \{0, 1\} \qquad (\forall (i, j)) \tag{2.10}$$

$$\sum_{j=0, j\neq i}^{n} y_{i,j} - \sum_{j=0, j\neq i}^{n} y_{j,i} - \sum_{j=0}^{n} d_{i,j} x_{i,j} = 0 \qquad (i = 1, 2, \ldots, n) \tag{2.11}$$

$$y_{0,i} = d_{0,i} x_{0,i} \qquad (i = 1, 2, \ldots, n) \tag{2.12}$$

$$y_{i,j} \leq (D - d_{j,0}) x_{i,j} \qquad (j \neq 0, (i, j) \in \mathcal{A}) \tag{2.13}$$

$$y_{i,0} \leq D x_{i,0} \qquad (i = 1, 2, \ldots, n) \tag{2.14}$$

$$y_{i,j} \geq (d_{0,i} + d_{i,j}) x_{i,j} \qquad (i \neq 0, (i, j) \in \mathcal{A}) \tag{2.15}$$

Equations (2.6) and (2.7) ensure that m vehicles leave and return to the depot node, node 0. Equations (2.8) and (2.9) impose the degree constraints so that each node is visited once by only one vehicle. Equation (2.10) is the integrality constraint. Subtours are eliminated with equation (2.11); this is enforced on each vehicle's individual tour. Equations (2.12), (2.13), (2.14), and (2.15) introduce the distance constraints, ensuring that $0 \leq y_{i,j} \leq D$ [46]. Once solved, Figure 2.7 shows an example of how these tours may appear graphically.

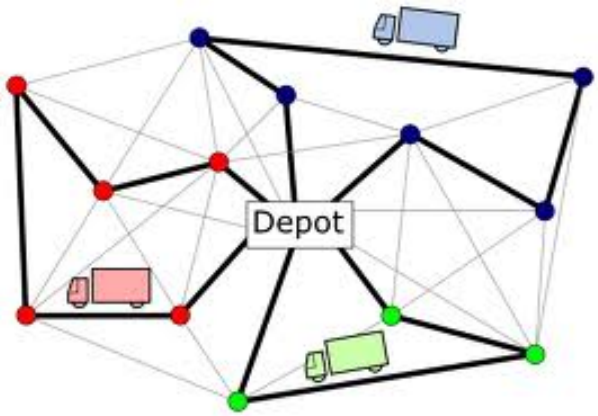

Figure 2.7: Possible Solution to VRP [47]

The VRP modeled by Kara [46], will be the basis of the formulated algorithm. This method encompasses the multi-vehicle aspect as well as the required distance constraints. On the other hand, this method needs modifications to deal with the complexities of UAV operations and warehouse layouts. The VRP discussed here does not consider potential collisions between vehicles, which is essential in creating operational UAV trajectories for confined environments.

2.4 UAS-Based Inventory Tracking Solution

One current model exists that aims to optimize UAS trajectories in warehouse/manufacturing environments. Formulated by Choi et al. [6], this method consists of two separate optimization problems, forming a two-stage process. At a high level, the first stage finds the optimum set of flyable trajectories and the minimum number of UAVs needed for a given network to minimize the overall mission time. The results of this are then fed into the second stage, which computes the optimal offset deployment time of each UAV flying within their respective trajectories, ensuring that any potential collisions between vehicles are avoided, while total mission time is still minimized [6].

This first stage of this algorithm creates an undirected or symmetrical graph using Mixed-Integer Linear Programming (MILP) techniques [46].

When looking at the logic behind the two-stage process of this algorithm, a few questions arise:

- Why not perform these stages simultaneously?
- Does the optimal solution from stage 1 always correspond to the optimum solution from stage 2?

These questions lead to the first main research question:

> **Research Question 1:** How can the existing algorithm be modified to streamline the optimization process and does this modification improve the mission time?

2.4.1 Model Formulation

The algorithm defined by Choi et al. [6] for the first stage of this process, is based on the Kara et al. [46] VRP discussed earlier and is modified to the following:

$x_{i,j,k}$: Binary decision variable, where:

$$x_{i,j,k} = \begin{cases} 1, \text{ if an edge from node } i \text{ to node } j \text{ appears in the optimal tour for vehicle } k \\ 0, \text{ otherwise} \end{cases}$$

$y_{i,j,k}$: Flow variable, where:

$$y_{i,j,k} = \begin{cases} \text{total flight time from the depot to node } j \text{ through } i \text{ by vehicle } k, \text{ if } x_{i,j,k} = 1 \\ 0, \text{ otherwise} \end{cases}$$

$T_{i,j}$: Flight time associated with traveling from node i to node j)

V: Set of identical vehicles

E: Maximum endurance time of each UAS

t_s: Setup time for each UAS

To optimize the multi-UAV trajectories, this algorithm formulates the following objective function that minimizes total mission time including total flight time and setup time of each UAV:

Objective Function:

$$Minimize \sum_{k\in\mathcal{V}}\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} T_{i,j}x_{i,j,k} + \sum_{k\in\mathcal{V}} t_s x_{0,h,k} \qquad (h \neq 0, h \in \mathcal{N}) \qquad (2.16)$$

Subject to the following constraints:

$$\sum_{k\in\mathcal{V}}\sum_{j\in\mathcal{N}} x_{i,j,k} = 1 \qquad (i \neq 0, i \in \mathcal{N}) \qquad (2.17)$$

$$\sum_{j\in\mathcal{N}} x_{0,j,k} = 1 \qquad (k \in \mathcal{V}) \qquad (2.18)$$

$$\sum_{i\in\mathcal{N}} x_{i,0,k} = 1 \qquad (k \in \mathcal{V}) \qquad (2.19)$$

$$\sum_{i\in\mathcal{N}} x_{i,h,k} - \sum_{j\in\mathcal{N}} x_{h,j,k} = 0 \qquad (h \neq 0, h \in \mathcal{N}, k \in \mathcal{V}) \qquad (2.20)$$

$$\sum_{j\in\mathcal{N}} y_{i,j,k} - \sum_{j\in\mathcal{N}} y_{j,i,k} - \sum_{j\in\mathcal{N}} T_{i,j}x_{i,j,k} = 0 \qquad (i \in \mathcal{N}, k \in \mathcal{V}) \qquad (2.21)$$

$$y_{0,j,k} = T_{0,j}x_{0,j,k} \qquad (j \neq 0, j \in \mathcal{N}, k \in \mathcal{V}) \qquad (2.22)$$

$$y_{i,j,k} \leq (E - T_{j,0})x_{i,j,k} \qquad (j \neq 0, (i,j) \in \mathcal{A}, k \in \mathcal{V}) \qquad (2.23)$$

$$y_{i,0,k} \leq E x_{i,0,k} \qquad (i \neq 0, i \in \mathcal{N}, k \in \mathcal{V}) \qquad (2.24)$$

$$y_{i,j,k} \geq (T_{0,i} + T_{i,j})x_{i,j,k} \qquad (i \neq 0, (i,j) \in \mathcal{A}, k \in \mathcal{V}) \qquad (2.25)$$

In this model, equation (2.17) ensures each node is visited only once by only one vehicle. Equations (2.18) and (2.19) guarantee that each UAS deploys and lands at the depot, node 0. Equation (2.20) forces every vehicle to leave each node that it has entered. Elimination of subtours is accomplished with equation (2.21) and the endurance constraints are defined by equations (2.22), (2.23), (2.24), and (2.25) [6].

The objective function defined in this model (2.16) sums the mission time (flight time + setup time) of each vehicle and then seeks to minimize this value. It appears that this function does not represent the actual mission time of the whole process as the vehicles will have overlaps in their flying times. Therefore, this objective function poses the following concern:

- Does the current objective function produce the optimum results for minimizing the overall mission time?

This uncertainty leads to the formation of the second research question:

Research Question 2: What modifications can be made to the existing objective function in order to represent the overall mission time of a task performed by multiple vehicles?

The first stage of the algorithm results in the optimum set of feasible trajectories, which are then taken over to the second stage in the process. Currently, the UAVs deploy every t_s seconds, meaning the 5^{th} UAV will deploy after $5t_s$ seconds. In a perfect world, these would be the most optimal times for the vehicles to deploy because the setup time defines the minimum deployment offset between each UAS. However, in a warehouse environment further investigation needs to be done to ensure the UAVs will not crash into one another throughout their flights.

Stage 2 of this algorithm [6] calculates the position of each vehicle at very small intervals of time and ensures there is always a minimum distance between any two UAVs. If a potential collision is detected, meaning any two vehicles are too close, then one of those vehicles delays its deployment by a set amount of time. Then, the process repeats, continually delaying vehicles, until there are no risks of collisions. Once this is complete, the process stores the vehicle deployment and offset times and begins again, this time changing the

sequence in which each trajectory is flown. In the end, the algorithm determines the optimal trajectory sequence and each UAV deployment time ensuring the shortest collision-free mission.

This algorithm was tested on a 3D model of a section of a warehouse, shown in Figure 2.8. Assuming the following initialization:

- UAS Platform: DJI Phantom 4
- t_s = 120 seconds
- E = 25 minutes
- Scan speed = 0.3 m/s
- Cruise speed = 1 m/s
- Relative Tolerance = 0.05
- Min. Vehicle Distance = 3 meters

The cruise speed is used whenever the vehicle is traveling between shelves and the scan speed is used when the UAV is flying along the length of a shelf to track the inventory.

The relative tolerance defines the value at which to suspend the optimization activities and is calculated as follows:

$$Rel.\,Tol. = \left| \frac{Incumbent\ Sol. - Lower\ Bound}{Incumbent\ Sol.} \right| \tag{2.26}$$

The results of the first stage optimization produced a set of five trajectories which would be assigned to a fleet of five UAVs, these trajectories can be seen in Figure 2.9. This first stage optimization algorithm has an approximate computation time of one day (86,400 seconds). Table 2.1 shows the flight time required to complete each of the five trajectories and each one satisfies the endurance constraint. It is important to note that the times in Table 2.1

correspond only to the UAV flight time required to complete a certain route and does not include the setup or deployment offset time for that vehicle, as that is not a factor when verifying the endurance constraint. However, they are factors when looking at the total mission time.

Results of stage 2 can be seen in Figure 2.10. This graph shows the whole mission, specifically the time when each UAS takes-off and lands. The last UAS lands 79 minutes and 13 seconds after the mission begins, defining the overall mission time [6]. Using these deployment times, a non-collision event is guaranteed; shown by the graph in Figure 2.11. This graph shows the smallest distance between two vehicles at every given point in time throughout the duration of the mission when more than one vehicle is in the air at the same time. It also shows that those vehicle separations never cross the potential collision constraint of 3 meters. All of these factors lead to the result that an inventory audit performed by UAVs in the modeled warehouse would take about 80 minutes to complete safely based on the current algorithm [6].

After looking at the results of this model, understanding the computational iterations that it involves, and seeing the significant offsets required to avoid collisions, the final research question is posed:

Research Question 3: How can the operational setup be transformed to simplify and improve the algorithm and results?

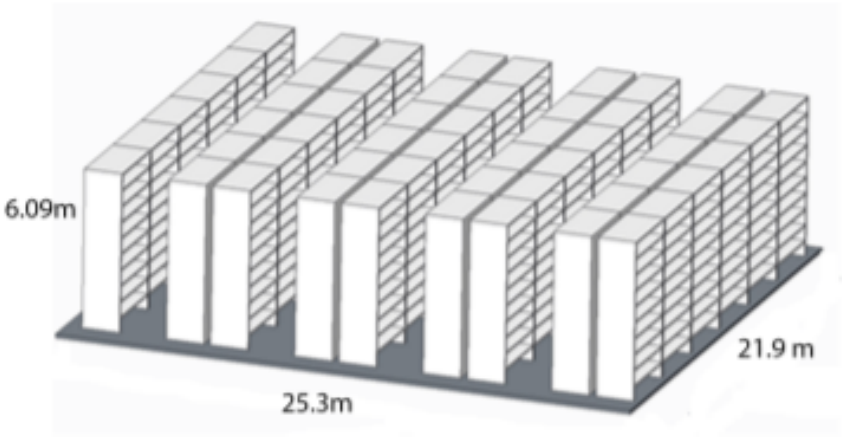

Figure 2.8: 3D Model of Warehouse Section [6]

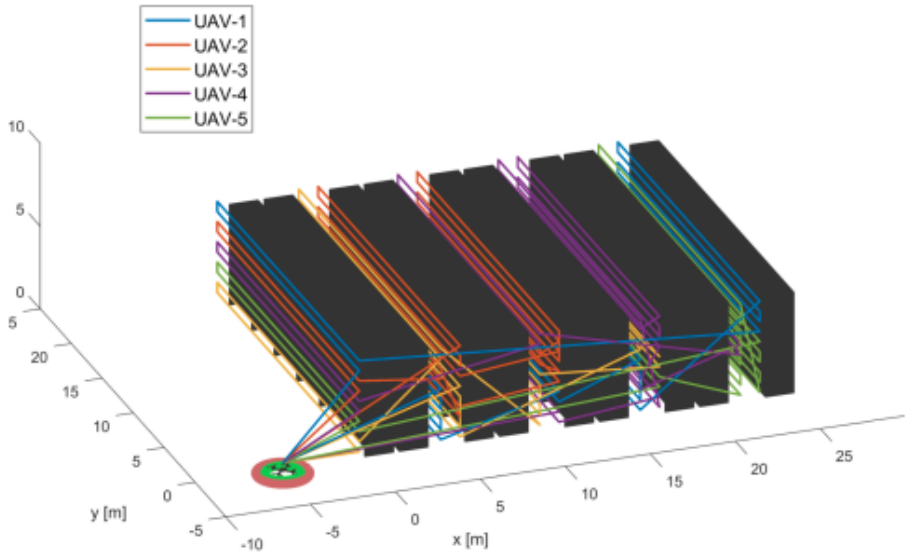

Figure 2.9: Optimal UAS Trajectories [6]

Table 2.1: UAV Trajectory Flight Times [6]

UAV #	Required Flight Time
1	24 min 12 sec
2	23 min 34 sec
3	24 min 4 sec
4	24 min 12 sec
5	24 min 12 sec

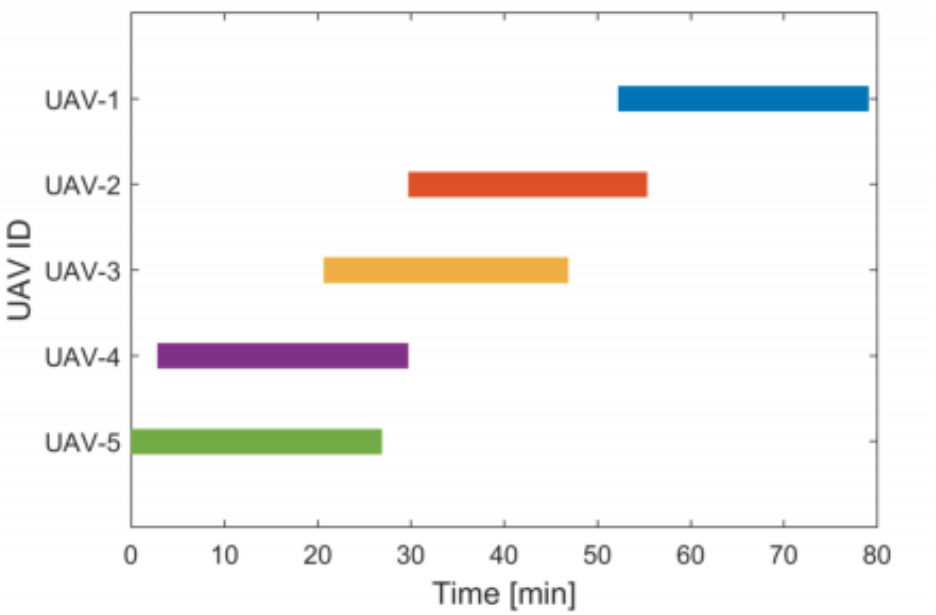

Figure 2.10: Optimum UAS Deployment Times for Non-Collision Event [6]

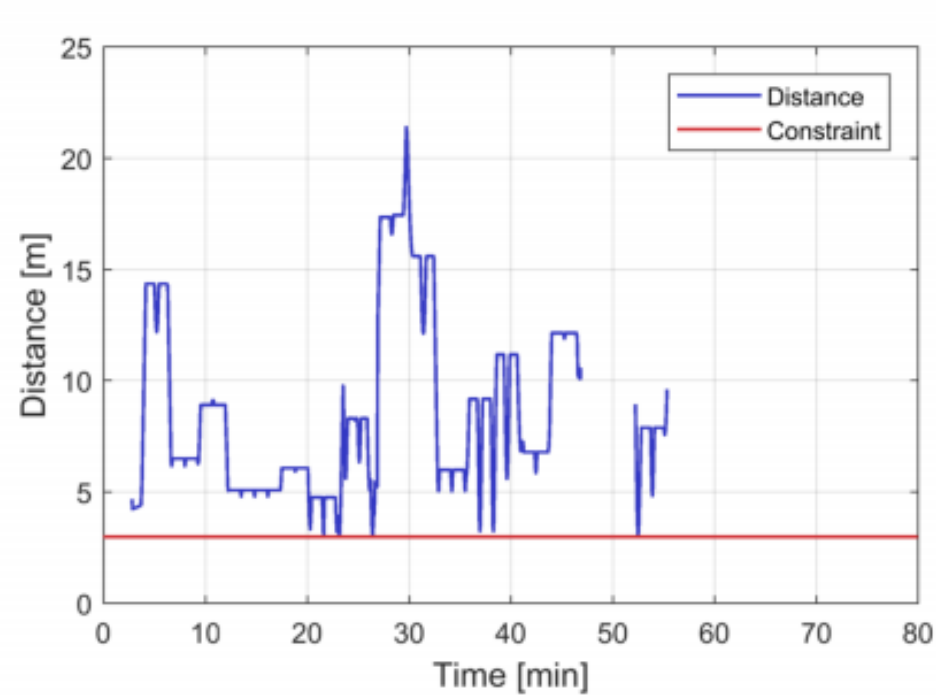

Figure 2.11: Collision Constraint Graph [6]

CHAPTER 3
TECHNICAL APPROACH

3.1 Proposed Methodologies

To improve the sub-optimal results and address the research questions raised from the multi-UAV inventory tracking scenario conveyed by Choi et al. [6], three possible improvements are introduced. First, a new algorithm logic which seeks to minimize the total mission time once collision avoidance has been ensured rather than having separate processes. A single stage algorithm logic will be explored to streamline the optimization process. Next, an objective function that seeks to minimize the maximum UAV mission time rather than minimizing the total of all UAV mission times [48] [49] [50]. Last, an operational setup consisting of multiple deployment locations instead of only one, which enables larger overlaps in flight times and initial vehicle separations [41] [49] [51] [52] [53]. Beginning as conceptual ideas, these methodologies are suggested, formulated, and tested in order to tackle and resolve the gaps within the current methods and further improve the mission time required for a fleet of UAVs performing an inventory count. They are compared against the existing model as the baseline.

The baseline, as well as the three proposed methods, and collision avoidance algorithm are mathematically modeled and programmed with Python. These models use Gurobi as the optimization solver.

In order to test the proposed modifications, a subscale experimental setup is created to simplify computation time and analysis of the initial results. This sample inventory tracking environment contains only two rows of shelves each with only two levels to store products. The shelves are 26 meters long, 1 meter wide, 3 meters tall, and spaced 3 meters apart from each other. The following are additional model parameters:

- Vehicle Endurance: 20 minutes
- Setup Time: 2 minutes
- Cruise Speed: 0.2 m/s
- Min. Vehicle Distance: 3 meters

For simplification, a scanning speed is not used for the initial experiments. No relative tolerance is needed to cut off the optimization solution for a problem of this size as it solves within reasonable time (approximately $<$30 seconds).

As nodes are placed in this subscale model, a flight time matrix, $T_{i,j}$, is created using the distance between nodes i and j and the vehicle's cruise speed. An exception to these calculations occurs if the path from i to j cuts through one of the shelves, as this is unrealistic. Therefore, the corresponding connection i, j is removed as a possibility for the model.

Following these subscale experiments, hypotheses will be formed on the three methodologies and then tested on a realistic warehouse model. Due to the large and complex nature of the warehouse model, an approach for reducing the dimensionality of the problem is explored to simplify computation. As the proposed methods delve into separate aspects of the optimization problem, they do not conflict, and therefore will also be studied in combination with one another during the full model analysis to see if further mission time improvements surface. The optimization computation times for each method will also be analyzed in the larger warehouse simulations.

The best resulting method will then be run through a sensitivity analysis with a varying range of input parameters to determine the robustness and versatility of the algorithm.

These experiments aim to show improvements upon the existing method and answer the research questions that were raised during the initial investigation into this problem.

3.1.1 Overall Methodology

The following methodology, shown in Figure 3.1, is an high-level overview of the steps preformed for this thesis.

3.2 Baseline

For later comparison and evaluation of the proposed modifications, the existing algorithm [6] is solved as the baseline, using the newly defined experimental setup. The baseline is necessary to understand the impact of suggested methods.

3.2.1 Model Formulation

The subscale baseline follows the two stage logic defined by Choi et al. [6]. The first stage minimizes the total mission time of all vehicles with objective function, (2.16), and is subject to (2.17), (2.18), (2.19), (2.20), (2.21), (2.22), (2.23), (2.24), and (2.25).

The formulation of the second stage or collision avoidance algorithm, defines variables as the following:

$\mathcal{Q}$: Set of trajectory sequences

T_q: Mission time for trajectory sequence q

$dist_{min}$: Minimum distance required between vehicles

$dist_{(k,m,t)}$: Distance between vehicles k and m at time t

This algorithm utilizes the following objective function which seeks to find the trajectory sequence with the minimum mission time once a non-collision event has been guaranteed:

Objective Function:

$$Minimize\, T_q \qquad (\forall(q \in \mathcal{Q})) \tag{3.1}$$

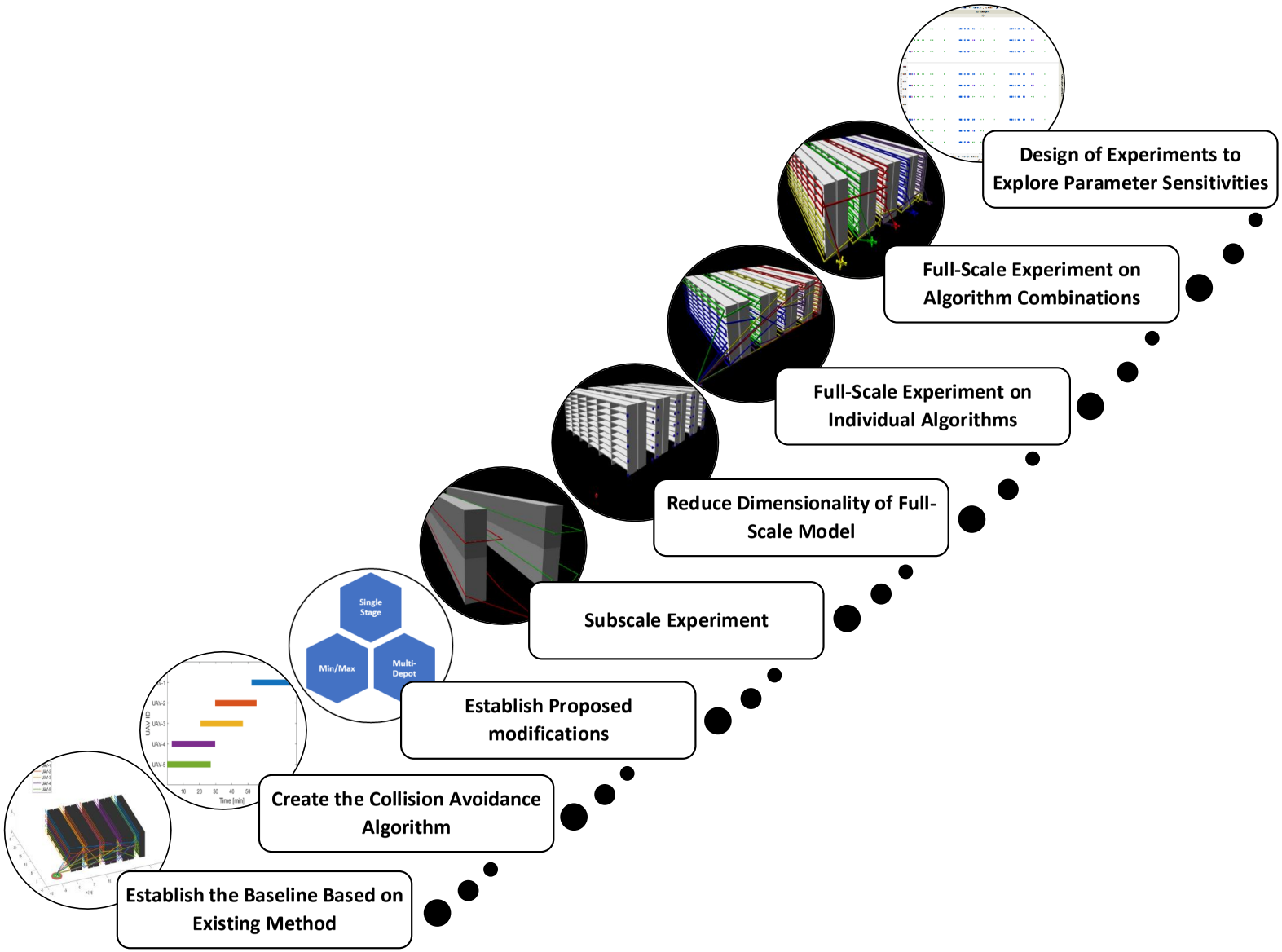

Figure 3.1: Overall Methodology

Subject to the following constraints:

$$dist_{k,m,t} \geq dist_{min} \qquad (k, m \in \mathcal{V}, k \neq m) \tag{3.2}$$

The second stage of this algorithm calculates the position of each vehicle at very small intervals of time and ensures there is always a minimum distance between any two UAVs with constraint (3.2). The algorithm determines the optimal trajectory sequence and each UAV deployment times to ensure the shortest collision-free mission defined by (3.1).

3.2.2 Validation

The baseline results are shown in Table 3.1, Figure 3.2, and Figure 3.3.

Table 3.1: Subscale Baseline Flight Times

UAV #	Required Flight Time
1	10 min 7 sec
2	10 min 7 sec

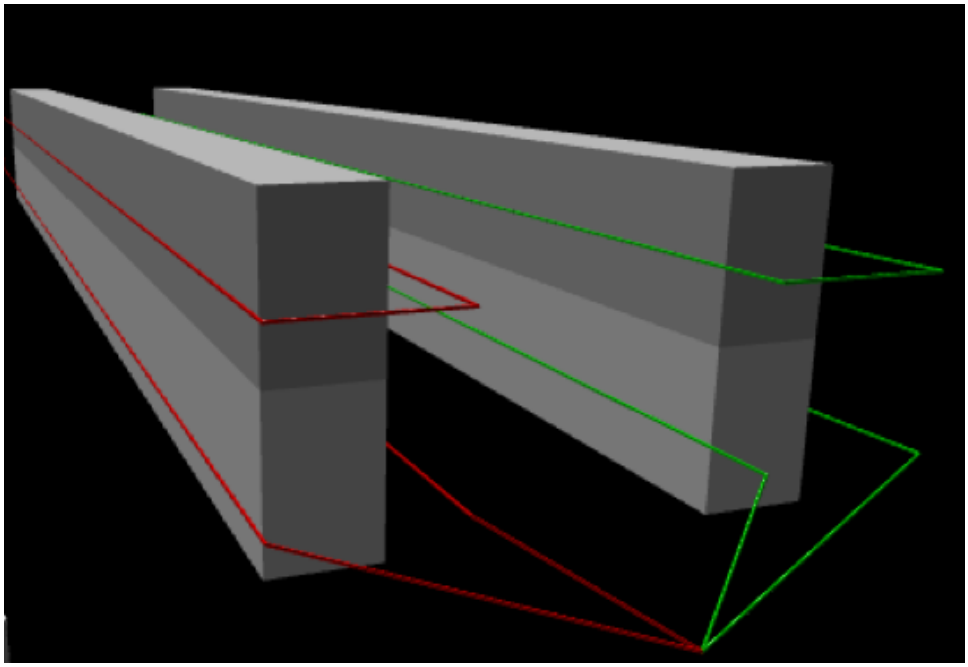

Figure 3.2: Subscale Baseline Optimal Trajectories

Figure 3.3: Subscale Baseline Optimum Deployment Times for Non-Collision Event

3.3 Single Stage Algorithm

To address Research Question 1, a new algorithm logic aims to integrate the two stages of the existing UAV routing algorithm [6] into one coherent system. Recall, the first stage of Choi's et al. method finds the optimum set of feasible trajectories through a warehouse environment for a fleet of UAVs to minimize the mission time. The second stage takes this optimum trajectory set and calculates the minimum offset time of each vehicle to ensure a non-collision event. The new logic transforms this two-stage process into just one. The new

method will intermittently calculate the time offset for every feasible trajectory set, rather than just the optimal solution. Therefore, each feasible trajectory set will be packaged with its associated deployment time offsets to avoid collisions. The algorithm will then choose the optimal trajectory set based on this new package of information to minimize the mission time. Since the existing method does not choose the optimum trajectory set based on actual vehicle deployment times, the shortest mission time cannot be guaranteed.

3.3.1 Model Formulation

The single stage algorithm minimizes the total mission time:

$$Minimize \sum_{k\in\mathcal{V}} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} T_{i,j} x_{i,j,k} + \sum_{k\in\mathcal{V}} t_s x_{0,h,k},\ (h \neq 0, h \in \mathcal{N}) \tag{3.3}$$

Subject to: (2.17), (2.18), (2.19), (2.20), (2.21), (2.22), (2.23), (2.24), and (2.25).

The single stage algorithm uses the same formulation of collision avoidance calculations for each feasible trajectory set with equation (3.1) and constrained by (3.2).

The difference from Choi's et al. algorithm lies within the logic flow of the process; instead of focusing on the optimal solution from stage 1, the new method uses every feasible solution. Within the optimization algorithm, the proposed method uses a callback function, which intermittently routes each feasible trajectory set found by the optimizer, to a function that executes the collision avoidance algorithm in order to obtain the non-collision deployment times and new total mission time. The single stage algorithm seeks to find the minimum mission time once collision-free event is ensured. A high-level visual of this new logic is shown in Figure 3.4.

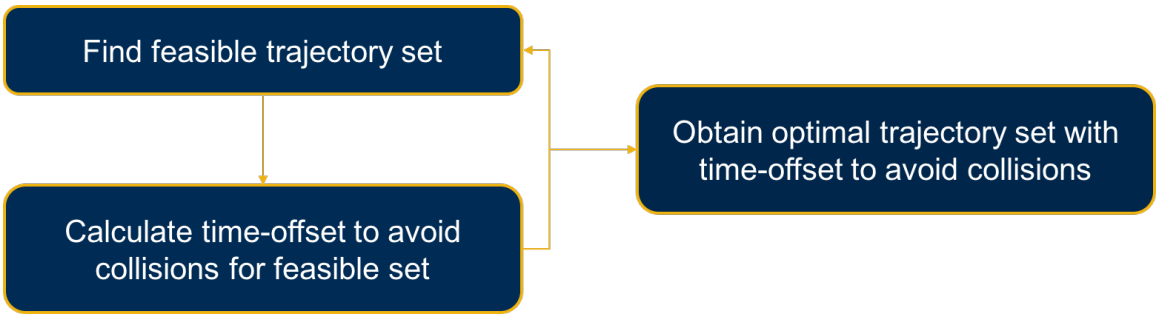

Figure 3.4: Single Stage Algorithm High-Level Logic

3.3.2 Validation

The expected benefit of performing the proposed logic change is the possible improvement of the overall mission time. If the optimal solution from stage 1 of the existing approach [6] happens to correspond to the optimal solution found in the new algorithm, then the results of the two models will be the same, but the latter will provide more confidence having exhausted more options. Alternatively, if the solutions of the two models do not correspond to one another, it is because a different trajectory set was selected by the new algorithm as it proved to be superior once collision avoidance offset times were calculated.

Running the proposed single stage algorithm resulted in Table 3.2, Figure 3.5, and Figure 3.6. When comparing the resulting flight times of the new method in Table 3.2 to the results of the baseline in Table 3.1, the baseline produces a lower cost trajectory set, which is why it was selected as the optimal set. This comparison is done prior to the offset calculations. The results in Figure 3.6 show the overall mission of the trajectory set selected by the single stage algorithm. From this graph, it is shown that the resulting overall mission time from the proposed method is 846 seconds or 14 minutes and 6 seconds, whereas, from Figure 3.3, the baseline overall mission time was 1025 seconds or 17 minutes and 5 seconds. Therefore, the new method's trajectories, depicted in Figure 3.5, provide a significantly reduced overall mission time.

Remember **Research Question 1: How can the existing algorithm be modified to**

streamline the optimization process and does this modification improve the mission time? This question was raised due to the inefficiencies and uncertainties of the two-stage UAS-Based Inventory Tracking Solution [6]. A new algorithm logic is offered to satisfy these concerns:

Hypothesis 1: The use of a callback function, to connect the trajectory optimization and collision avoidance algorithm, calculates the deployment time-offsets, needed to avoid collisions on all feasible trajectory sets. Using the single stage methodology, a multi-vehicle problem established within a confined environment, where collisions are probable, will result in the same or improved total mission time for a full-scale experiment when compared to the two-stage logic flow.

Table 3.2: Subscale Single Stage Flight Times

UAV #	Required Flight Time
1	10 min 11 sec
2	10 min 7 sec

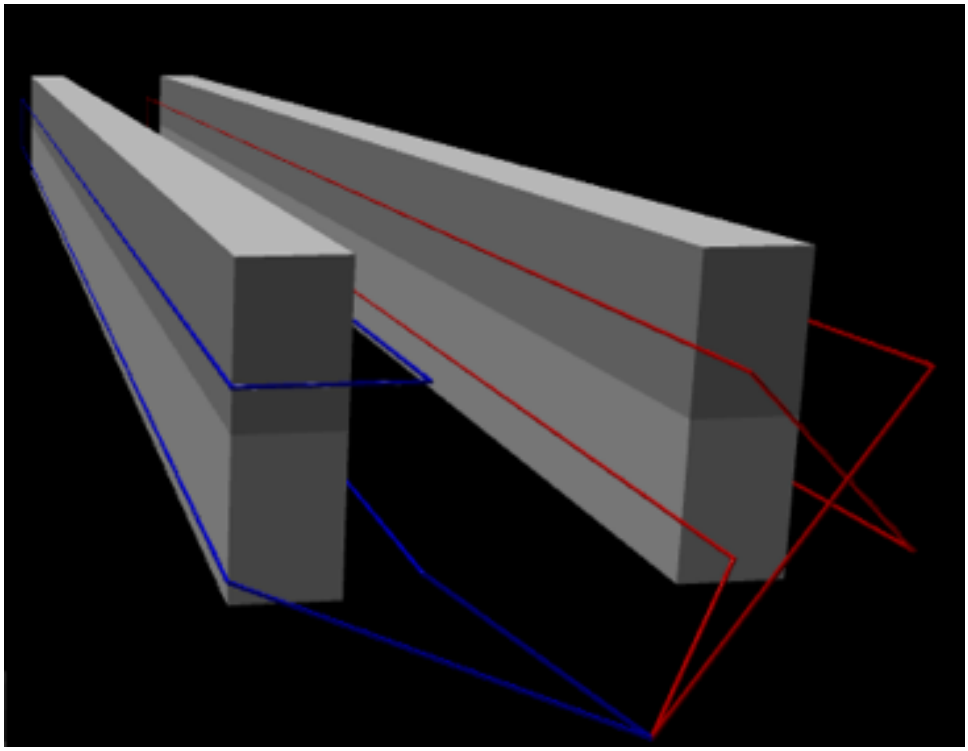

Figure 3.5: Subscale Single Stage Algorithm Optimal Trajectories

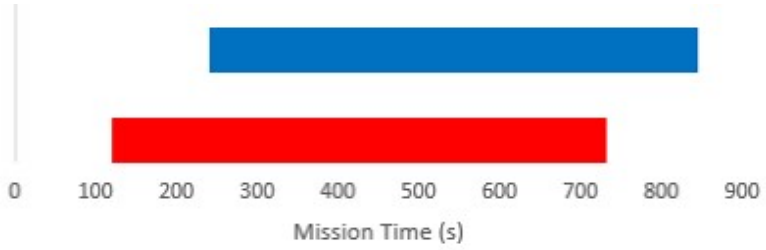

Figure 3.6: Subscale Single Stage Algorithm Optimum Deployment Times for Non-Collision Event

3.4 Min/Max Algorithm

The min/max algorithm aims to change the existing objective function, (2.16), which minimizes the sum total of each vehicles flight time, to a function that represents the total mission time. The proposed objective function will minimize the mission time of the vehicle with the maximum mission time. Since all of the vehicle's mission times start at $t = 0$,

this min/max problem will seek to minimize the mission time of the vehicle that lands last, thus minimizing the entire mission time. The vehicle with the maximum mission time is representative of the entire mission time because when that vehicle lands, the mission is complete. The goal of incorporating a min/max objective function is to have a better representation of the overall mission time. Often this method is used when seeking to evenly distribute the path lengths of each route, for example balancing the assignments of truck delivery drivers [49] [54].

3.4.1 Model Formulation

In addition to adding the min/max capability, a new variable needs to be introduced to ensure the optimum fleet size is selected. The method for doing this in the existing model will no longer be reliable in this new setup. The existing approach minimizes the fleet size through the objective function, (2.16), by adding in the setup times for each vehicle. Therefore, the more vehicles in the fleet, the greater the sum of all the vehicle flight times and setup times. Since the goal is to minimize that value, the smallest allowable fleet size will be used.

In the new formulation, the objective function will not be summing all of the vehicle mission times, but rather looking for the largest one in the set to minimize. With this, it means that increasing the fleet size would potentially result in a shorter overall mission time. In some cases, this may be beneficial, therefore, instead of minimizing fleet size, this new formulation optimizes it. The new model does so by creating a fleet acquisition cost. Therefore, each vehicle added to the system results in a greater fleet cost, or penalty to the system. For this model, the fleet acquisition cost increases by f seconds per vehicle, $F = fv$. The acquisition cost is mapped to an time value in order to correspond with the units of the objective function. The value of f can change to fit the needs of a problem by determining the required amount of time savings which would be worth acquiring an extra vehicle. The optimization, rather than minimization of fleet size, occurs when the

overall mission time involving v vehicles is at least f seconds longer than if it involved $v + 1$ vehicles. Overall, this method tries to minimize fleet size, unless there is a sufficient benefit to increase it, which is also known as a penalty function.

The proposed min/max objective function is formulated as follows:

$$Min(Max(\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} T_{i,j} x_{i,j,k} + t_s + F)) \qquad (k \in \mathcal{V}) \tag{3.4}$$

Subject to: (2.17), (2.18), (2.19), (2.20), (2.21), (2.22), (2.23), (2.24), and (2.25).

To examine the impacts of this new objective function, the two-stage process remains the same with collision avoidance being calculated with equation (3.1) and subject to (3.2).

3.4.2 Validation

The expected results of changing the objective function from equation (2.16) to equation (3.4) is obtaining an accurate representative of the overall mission time. The proposed objective function, (3.4), aims to minimize the maximum mission time from the set of all UAV mission times. The mission time for each UAV is its required setup time and its required flight time.

Looking at the baseline results in Table 3.1, the mission times for the two UAVs, prior to the collision offset calculation, are 12 minutes and 7 seconds and 14 minutes and 7 seconds, respectively, when you add in the associated setup times. Similarly, from the results of the min/max model in Table 3.3, with the setup times added in, the mission times for the two vehicles are 12 minutes and 33 seconds and 14 minutes and 5 seconds. The trajectories associated with these times can be seen in Figure 3.7. Therefore, the baseline would have an actual mission time of 14 minutes and 7 seconds before considering collisions and the new approach's mission time would be 14 minutes and 5 seconds before collision consideration as well. Even though this difference is minimal, the new objective function, (3.4),

still shows an improvement from the previous method. Once the collision-avoidance offset is calculated, the difference between the baseline and the min/max algorithm becomes significantly evident, as can be seen when comparing Figure 3.3 and Figure 3.8.

Recall **Research Question 2: What modifications can be made to the existing objective function in order to represent the overall mission time of a task performed by multiple vehicles?** This question was raised because the current method's objective function for multiple vehicles is not measuring the actual total mission time but rather the mission time of each UAS added together. Therefore, a new objective function was defined to remedy this:

> **Hypothesis 2:** Defining the objective function of the trajectory optimization as a min/max problem will be representative of the overall mission time and therefore, obtain a shorter mission time in the full-scale problem than minimizing the total mission time of all UAVs.

Table 3.3: Subscale Min/Max Algorithm Flight Times

UAV #	Required Flight Time
1	10 min 33 sec
2	10 min 5 sec

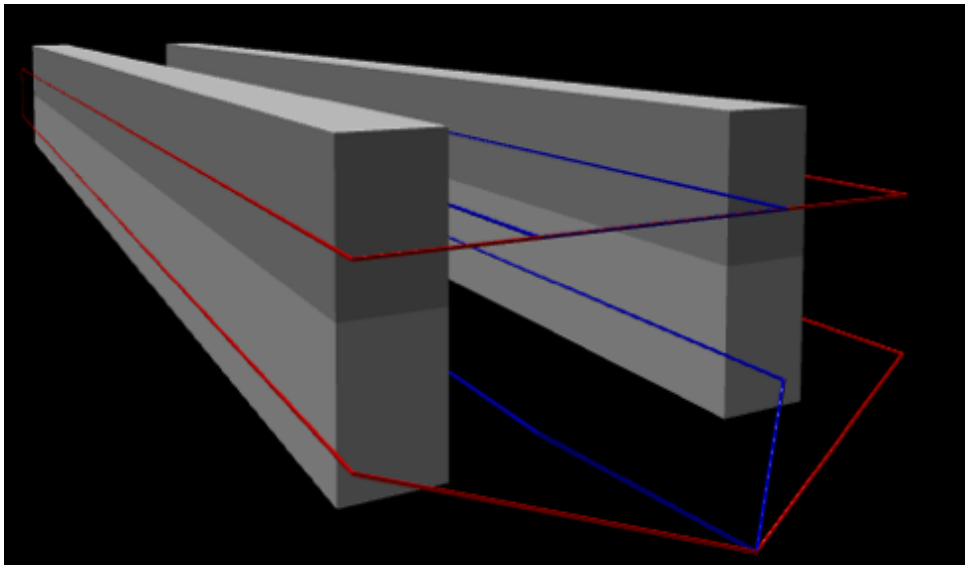

Figure 3.7: Subscale Min/Max Algorithm Optimal Trajectories

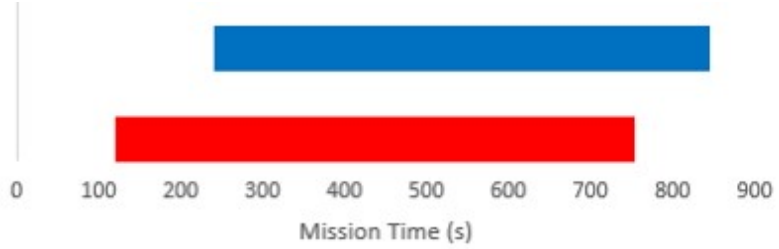

Figure 3.8: Subscale Min/Max Algorithm Optimum Deployment Times for Non-Collision Event

3.5 Multi-Depot Algorithm

A multiple deployment location setup aims to find a better way to handle the operations of this mission. Multiple depots reduce the offset times because each vehicle can begin its setup at $t = 0$; this reduction decreases the overall mission time. Additionally, with each vehicle deploying from separate locations, their trajectories may have fewer collision points since they are initially spread out. Lessening the possibility of accidents allows the vehicles to have larger overlaps in their flight times because fewer deployment offsets will be required. This has two main benefits, further minimization of the overall mission time

and fewer computational iterations to resolve potential collision points.

3.5.1 Model Formulation

The multi-depot method will also require the fleet acquisition cost within the objective function because with each vehicle having the potential to deploy at the same time, there would be no significant penalty for incorporating additional vehicles.

The multi-depot algorithm is defined by the graph, $G = (\mathcal{N}, \mathcal{A})$. Where $\mathcal{N}$ is the set of vertices, or nodes, in the graph. These nodes are broken down into two subsets, $\mathcal{D}$ and $\mathcal{W}$, where $\mathcal{N} = \mathcal{D} \cup \mathcal{W}$. $\mathcal{D}$ represents the set of depot nodes and $\mathcal{W}$ represents the set of waypoint nodes. $\mathcal{D} = \{0, 1, \ldots, d\}$, $\mathcal{W} = \{d+1, d+2, \ldots, w\}$ and $\mathcal{A}$ is the set of edges, or arcs, that connect the nodes, $\mathcal{A} = \{(i,j) : i, j \in \mathcal{N},$ if $i \in \mathcal{D}$ then $j \notin \mathcal{D},$ if $j \in \mathcal{D}$ then $i \notin \mathcal{D}, i \neq j\}$. There is then an associated cost set analogous to the edge set, $\mathcal{C} = (c_{i,j})$, this cost is defined by the flight time between nodes (i, j).

The following sets up the multi-depot optimization model variables:

$x_{i,j,k}$: Binary decision variable, where:

$$x_{i,j,k} = \begin{cases} 1, \text{ if the edge appears in the optimal tour} \\ 0, \text{ otherwise} \end{cases}$$

$y_{i,j,k}$: flow variable, where:

$$y_{i,j,k} = \begin{cases} \text{Time from depot to } j \text{ through } i, \text{ if } x_{i,j,k} = 1 \\ 0, \text{ otherwise} \end{cases}$$

$T_{i,j}$: Flight time from node i to node j

$\mathcal{V}$: Set of identical vehicles

E: Maximum endurance time of each UAV

t_s: Setup time for each UAV

F: Fleet acquisition cost

$\mathcal{D}$: Set of depots

$\mathcal{W}$: Set of waypoints

The multi-depot objective function is formulated as follows, by minimizing the total flight time, setup time of each vehicle, and the fleet acquisition cost:

Objective Function:

$$Minimize \sum_{k\in\mathcal{V}} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} T_{i,j} x_{i,j,k} + \sum_{k\in\mathcal{V}} t_s + F \tag{3.5}$$

Subject to the following constraints:

$$\sum_{k\in\mathcal{V}} \sum_{j\in\mathcal{N}} x_{i,j,k} = 1 \qquad (i \neq j, i \in \mathcal{W}) \tag{3.6}$$

$$\sum_{k\in\mathcal{V}} \sum_{i\in\mathcal{N}} x_{i,j,k} = 1 \qquad (i \neq j, j \in \mathcal{W}) \tag{3.7}$$

$$\sum_{i\in\mathcal{D}} \sum_{j\in\mathcal{W}} x_{i,j,k} = 1 \qquad (k \in \mathcal{V}) \tag{3.8}$$

$$\sum_{j\in\mathcal{D}} \sum_{i\in\mathcal{W}} x_{i,j,k} = 1 \qquad (k \in \mathcal{V}) \tag{3.9}$$

$$\sum_{k\in\mathcal{V}} \sum_{j\in\mathcal{W}} x_{i,j,k} \leq 1 \qquad (i \in \mathcal{D}) \tag{3.10}$$

$$\sum_{k\in\mathcal{V}} \sum_{i\in\mathcal{W}} x_{i,j,k} \leq 1 \qquad (j \in \mathcal{D}) \tag{3.11}$$

$$\sum_{j\in\mathcal{N}} y_{i,j,k} - \sum_{j\in\mathcal{N}} y_{j,i,k} - \sum_{j\in\mathcal{N}} T_{i,j} x_{i,j,k} = 0 \qquad (i \in \mathcal{W}, k \in \mathcal{V}) \tag{3.12}$$

$$y_{i,j,k} = T_{i,j} x_{i,j,k} \qquad (i \in \mathcal{D}, j \in \mathcal{W}, k \in \mathcal{V}) \tag{3.13}$$

$$y_{i,j,k} \leq (E - T_{j,h}) x_{i,j,k} \qquad (i \in \mathcal{N}, h \in \mathcal{D}, j \in \mathcal{W}, k \in \mathcal{V}) \tag{3.14}$$

$$y_{i,j,k} \leq E x_{i,j,k} \qquad (i \in \mathcal{W}, j \in \mathcal{D}, k \in \mathcal{V}) \tag{3.15}$$

$$y_{i,j,k} \geq (T_{h,i} + T_{i,j}) x_{i,j,k} \qquad (h \in \mathcal{D}, i \in \mathcal{W}, j \in \mathcal{W}, k \in \mathcal{V}) \tag{3.16}$$

Equation (3.6) and (3.7) ensure each node is visited once by exactly one vehicle. Equation (3.8) and (3.9) make sure every vehicle starts and ends their route at an unoccupied depot location. While constraints (3.10) and (3.11) ensure that at most only one vehicle can deploy from and land at each depot. Subtours are eliminated with equation (3.12) and the endurance constraints are applied with equations (3.13) through (3.16).

Collision avoidance offsets are determined with equation (3.1) subject to constraint (3.2).

3.5.2 Validation

The results of the multi-depot algorithm can be seen in Table 3.4 and in Figures 3.9 and 3.10. These show significant improvement from the baseline results because the vehicles are able to overlap their entire mission times and initially begin their routes with ample separation to reduce the risk of collisions. Thereby resulting in an overall mission time of 11 minutes and 7 seconds for the new method, compared to the 17 minutes and 5 seconds mission time for the baseline.

After exploring this model's formulation, a drawback is evident in the operational setup. Each vehicle deploys from the same location, thus resulting in initial vehicle offsets, extending the overall mission time, as well as large overlaps in trajectory paths, increasing the vulnerability to potential collisions.

Lastly, **Research Question 3: How can the operational setup be transformed to simplify and improve the algorithm and results?** This question was raised to see if the environmental setup could be altered to achieve better results. Thus, the following operation is proposed:

Hypothesis 3: Deploying the UAVs from multiple depots, rather than a single depot, will significantly reduce the overall mission time in the full-size model because the

vehicles will have greater overlaps in flight times with the ability to deploy simultaneously.

Table 3.4: Subscale Multi-Depot Algorithm Flight Times

UAV #	Required Flight Time
1	9 min 7 sec
2	9 min 7 sec

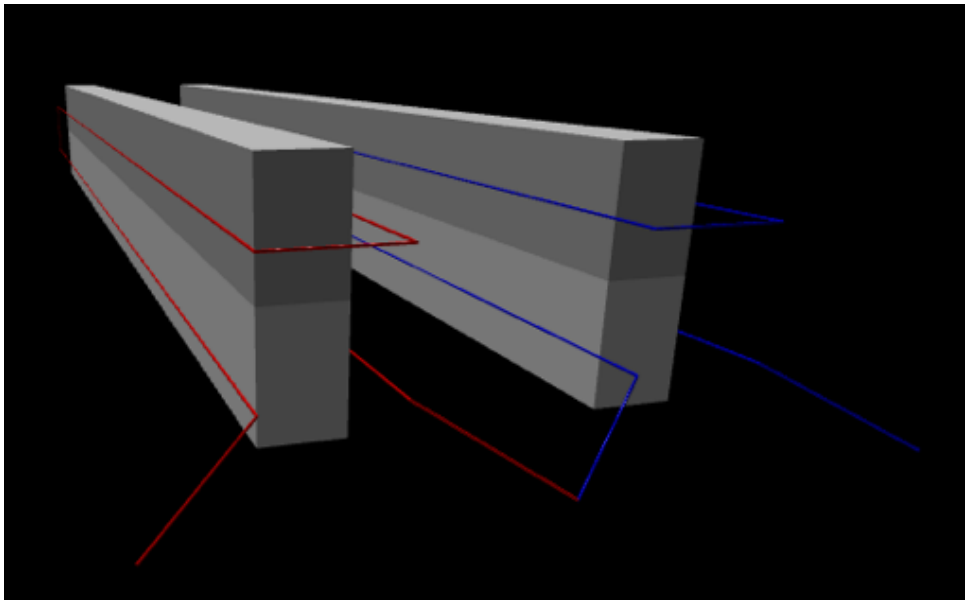

Figure 3.9: Subscale Multi-Depot Algorithm Optimal Trajectories

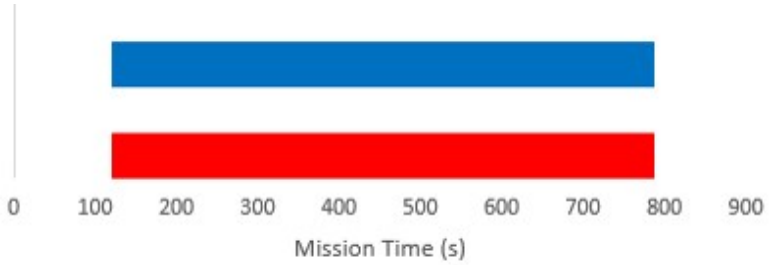

Figure 3.10: Subscale Multi-Depot Algorithm Optimum Deployment Times for Non-Collision Event

3.6 Model Dimensionality Reduction

In addition to the three new methodologies, a model dimensionality reduction is also introduced. This serves to reduce computation time and complexity, as these are exponentially related to the number of variables within a model. Figure 3.11 shows the full-scale warehouse environment and the complete initial node network, which consists of 1 depot node (red) and 180 waypoint nodes (blue and green). A model of this size requires significant computing power as the decision variables are defined as the connections ($x_{i,j,k}$) and flow ($y_{i,j,k}$) between these nodes.

The model is reduced by removing the green nodes from Figure 3.11, resulting in the reduced initial node network shown in Figure 3.12. This new network contains 1 depot node (red) and 45 waypoint nodes (blue). The optimization methodologies will create tours through this new simplified node network and then convert the paths to the complete network representation for collision avoidance. For example, Figure 3.13 shows a possible reduced network connection (yellow path), between node A and node B. Similar to the subscale model, each edge has an associated cost or weight. Previously, this weight corresponded to the flight time between the two nodes. In the new model, an assumption is made that once a waypoint node is visited by a vehicle, that vehicle must travel down the length of that shelf and also return along the shelf directly above. This imaginary path (purple) can be visualized in Figure 3.14. The weight or cost of the path from node A to

node B in the reduced model (yellow path) is equivalent to the flight time of the converted path (purple path). This conversion is used to generate each vehicles full trajectory and is also necessary for the collision avoidance algorithm. It is important to note that this model reduction now creates a directed graph, where $c_{i,j} \neq c_{j,i}$. From Figure 3.14, the path from A to B would have a slightly different flight time than the path from B to A. The reduction of nodes in this model will greatly simplify the trajectory optimization algorithms.

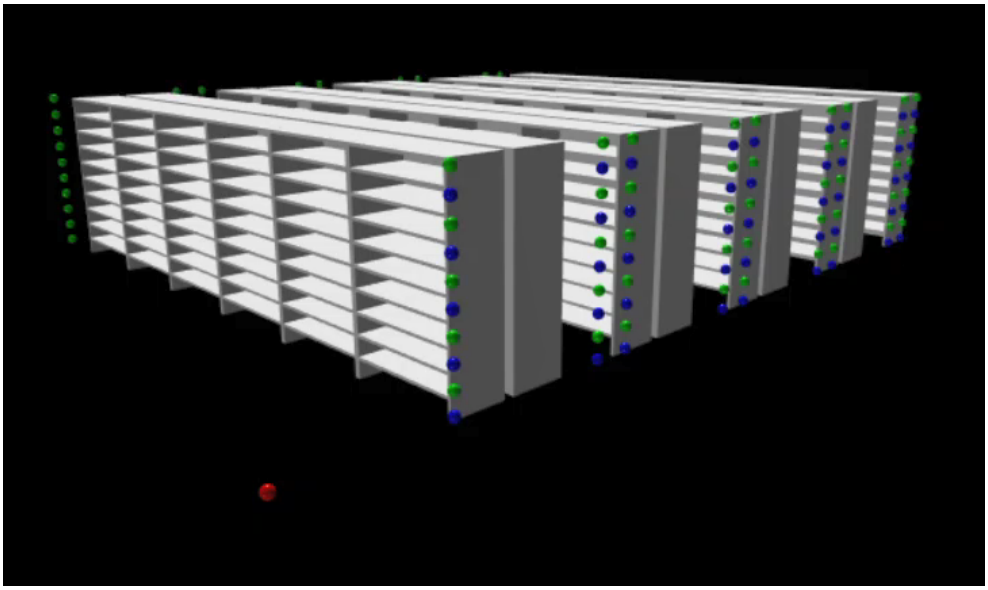

Figure 3.11: Complete Initial Node Network

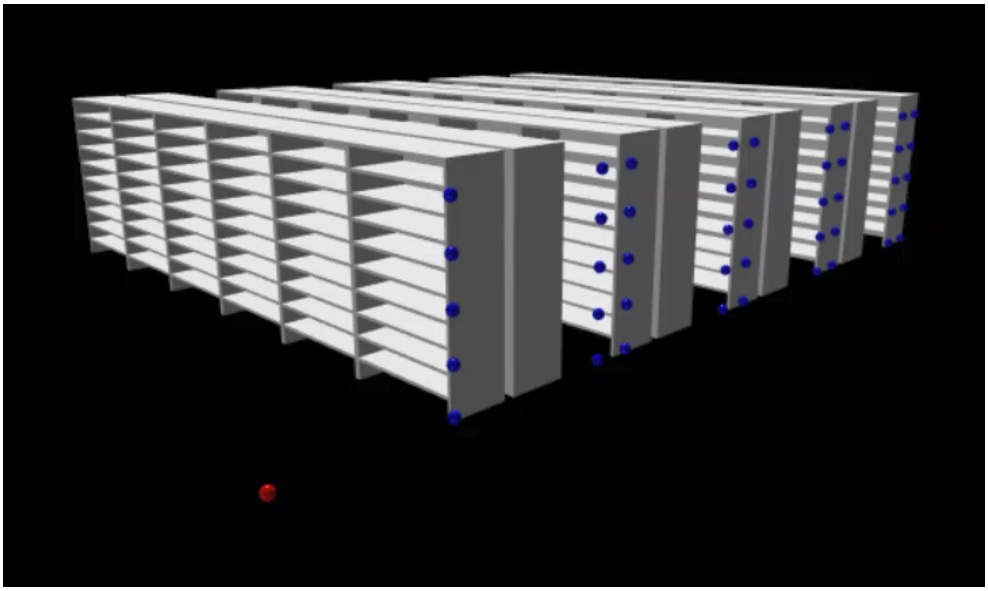

Figure 3.12: Reduced Initial Node Network

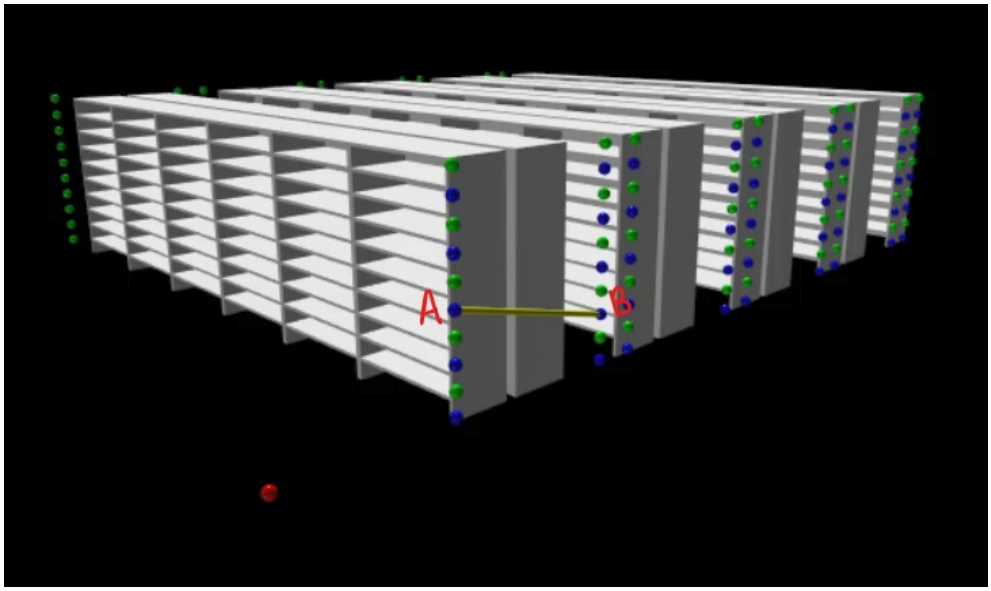

Figure 3.13: Reduced Edge Generation

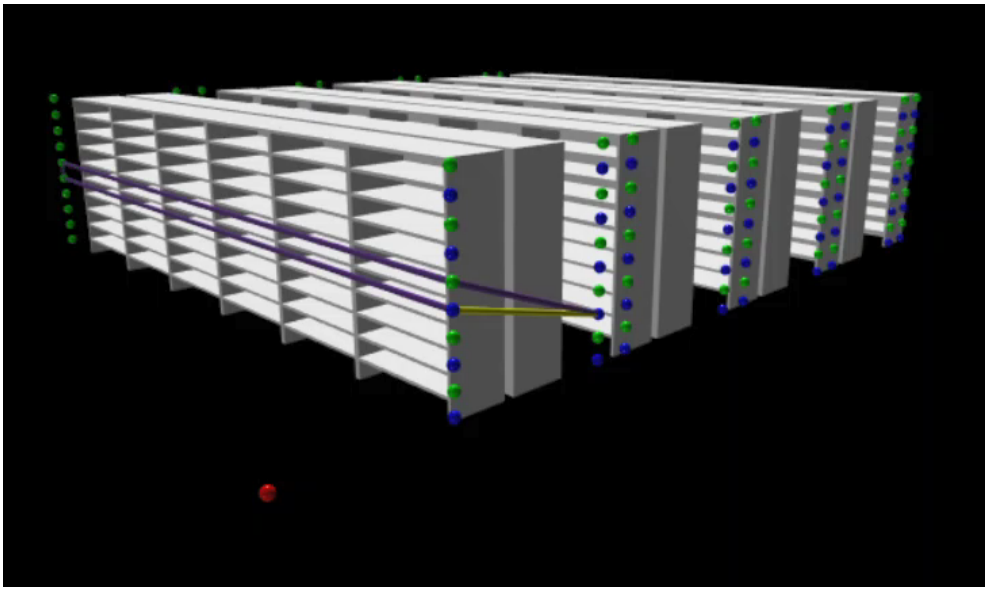

Figure 3.14: Converted Edge Generation

CHAPTER 4
RESULTS ANALYSIS

4.1 Experimental Setup

A warehouse environment, shown in Figure 2.8, is set up to test the capabilities of each proposed model individually and also combined. The experiment utilizes a fleet of homogeneous vehicles, in this case, the DJI Phantom 4. The simulations assume the following information:

- Vehicle Endurance: 25 minutes
- Setup Time: 2 minutes
- Vehicle Acquisition Cost: 3 minutes
- Cruise Speed: 1 m/s
- Scan Speed: 0.3 m/s
- Relative Tolerance: 0.005
- Min. Vehicle Distance: 3 meters

An operational assumption is also made; the UAV's are capturing images of the product tags which contain unique identifiers to process through an image recognition algorithm to obtain an accurate inventory. At the slow scanning speed, the time to capture a clear image is negligible. The above values, with the exception of the vehicle acquisition cost and the relative tolerance, are assumed from the experiment run within the existing method by Choi et al. [6]. These values are used to formulate a point solution to test and evaluate the proposed methods and hypotheses and uncover the best methodology combination to proceed with further testing on a range of input parameters.

4.2 Baseline Results

Similar to the subscale experiments laid out in Chapter 3, the UAS-Based Inventory Tracking Solution formulation developed by Choi et al. [6] is used as the baseline, although this time with the addition of the model dimensionality reduction defined in Section 3.6. These results will be useful for comparison with the new model formulations and results. The first stage of the optimization algorithm results in the five UAV trajectories shown in Figure 4.1, with corresponding flight times shown in Table 4.1. In the second stage, collision avoidance algorithm offsets the deployment of the five vehicles and the results can be seen in Figure 4.2. The baseline has a total mission time of 41 minutes and 7.8 seconds to run an inventory audit in this warehouse section.

Table 4.1: Baseline Flight Times

UAV #	Required Flight Time
1	24 min 2.1 sec
2	23 min 40.5 sec
3	23 min 31 sec
4	23 min 59.4 sec
5	23 min 53.8 sec

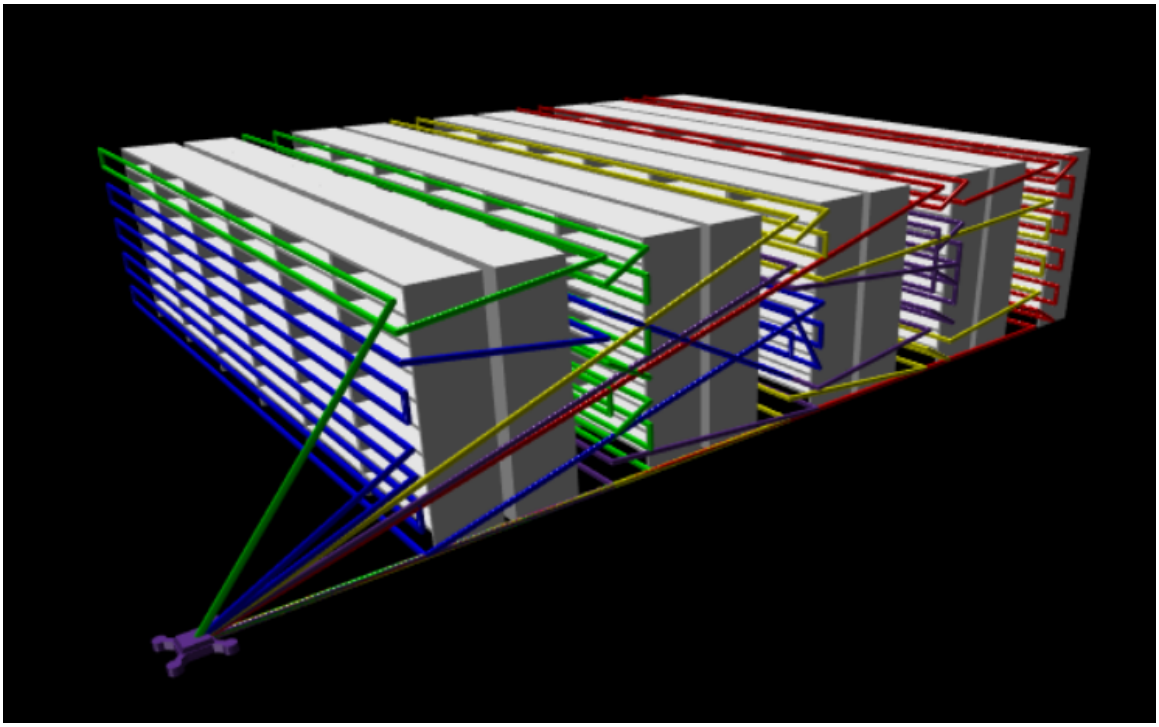

Figure 4.1: Baseline Optimal Trajectories

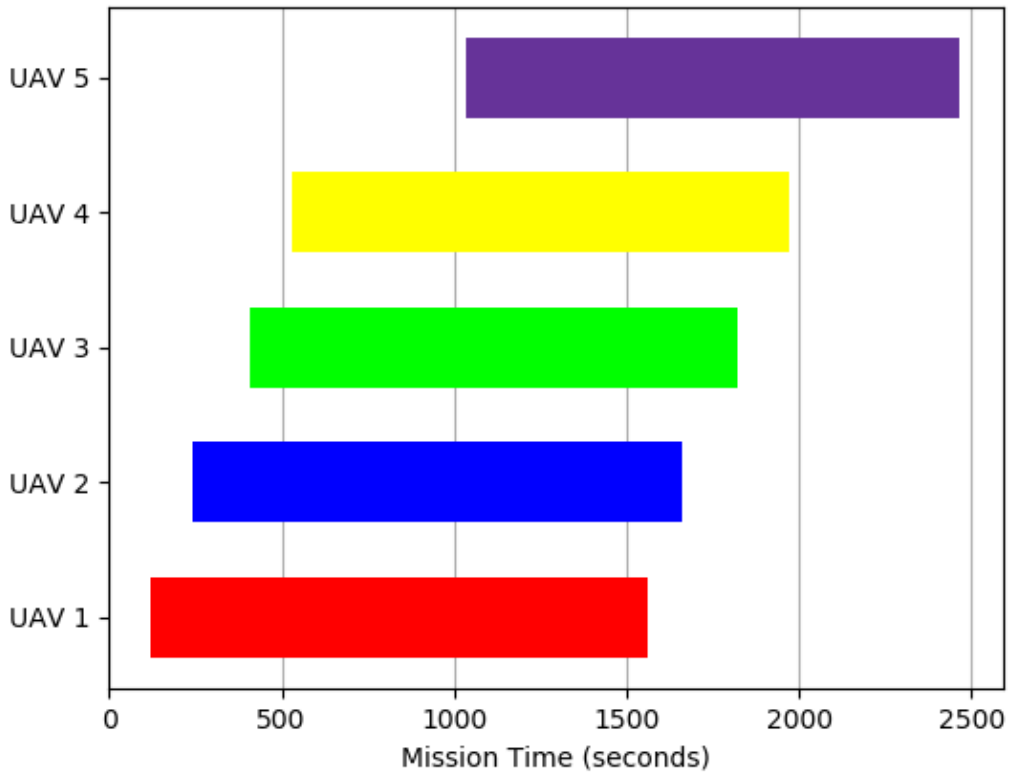

Figure 4.2: Baseline Optimum Deployment Times for Non-Collision Event

4.3 Single Stage Algorithm Results

The results of running the single stage algorithm can be viewed in Table 4.2, Figure 4.3, and Figure 4.4. The resulting mission time for this algorithm was 35 minutes and 32.2 seconds. Compared to the baseline, this shows a 14.6% reduction in the overall mission time to complete an inventory audit with the new methodology. The two-stage logic only calculates the deployment offsets for the optimal trajectory set of stage 1. From these results, it is clear that one of the other feasible (but not necessarily optimal) trajectory sets from the optimization process had fewer collision points and therefore fewer offsets, leading to a significantly reduced mission time. As a reminder, **Hypothesis 1: The use of a callback function, to connect the trajectory optimization and collision avoidance**

algorithm, calculates the deployment time-offsets, needed to avoid collisions on all feasible trajectory sets. Using the single stage methodology, a multi-vehicle problem established within a confined environment, where collisions are probable, will result in the same or improved total mission time for a full-scale experiment when compared to the two-stage logic flow. The results laid out in this section permits the acceptance of Hypothesis 1.

Table 4.2: Single Stage Algorithm Flight Times

UAV #	Required Flight Time
1	24 min 6.2 sec
2	24 min 2.5 sec
3	23 min 41.1 sec
4	24 min 9.2 sec
5	24 min 2.2 sec

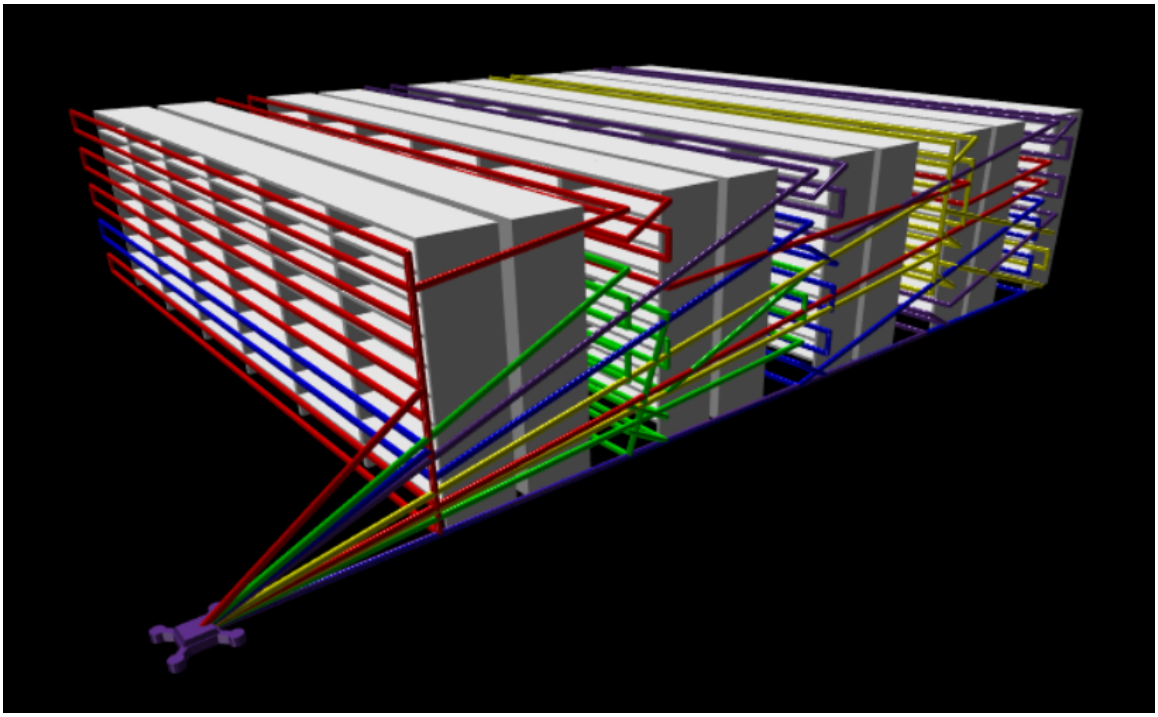

Figure 4.3: Single Stage Algorithm Optimal Trajectories

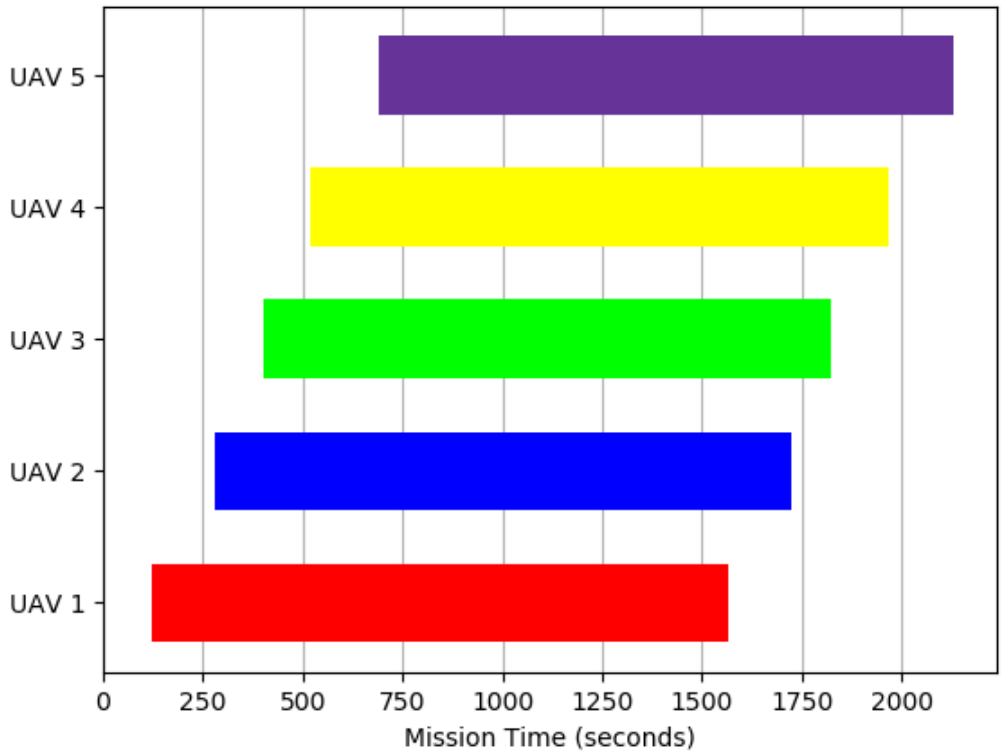

Figure 4.4: Single Stage Algorithm Optimum Deployment Times for Non-Collision Event

4.4 Min/Max Algorithm Results

Vehicle flight times and the associated trajectories for the min/max problem are shown in Table 4.3 and Figure 4.5, respectively. The collision avoidance algorithm produced the deployment offsets seen in Figure 4.6 with a final mission time of 42 minutes and 12.1 seconds. Consequently, incorporating a min/max objective function actually increased the overall mission time compared to the baseline. The argument that the min/max objective function is representative of the overall mission time only holds true before accounting for vehicle offsets and collision avoidance. Once the offset deployment times are calculated, the min/max algorithm trajectories could have more potential for collisions resulting in a longer mission time. Recall, **Hypothesis 2: Defining the objective function of the trajec-**

tory optimization as a min/max problem will be representative of the overall mission time and therefore, obtain a shorter mission time in the full-scale problem than minimizing the total mission time of all UAVs. The collision avoidance considerations and the min/max results depicted prompts the rejection of Hypothesis 2.

Table 4.3: Min/Max Algorithm Flight Times

UAV #	Required Flight Time
1	24 min 1.4 sec
2	23 min 49 sec
3	24 min 1.5 sec
4	23 min 45.5 sec
5	23 min 54.1 sec

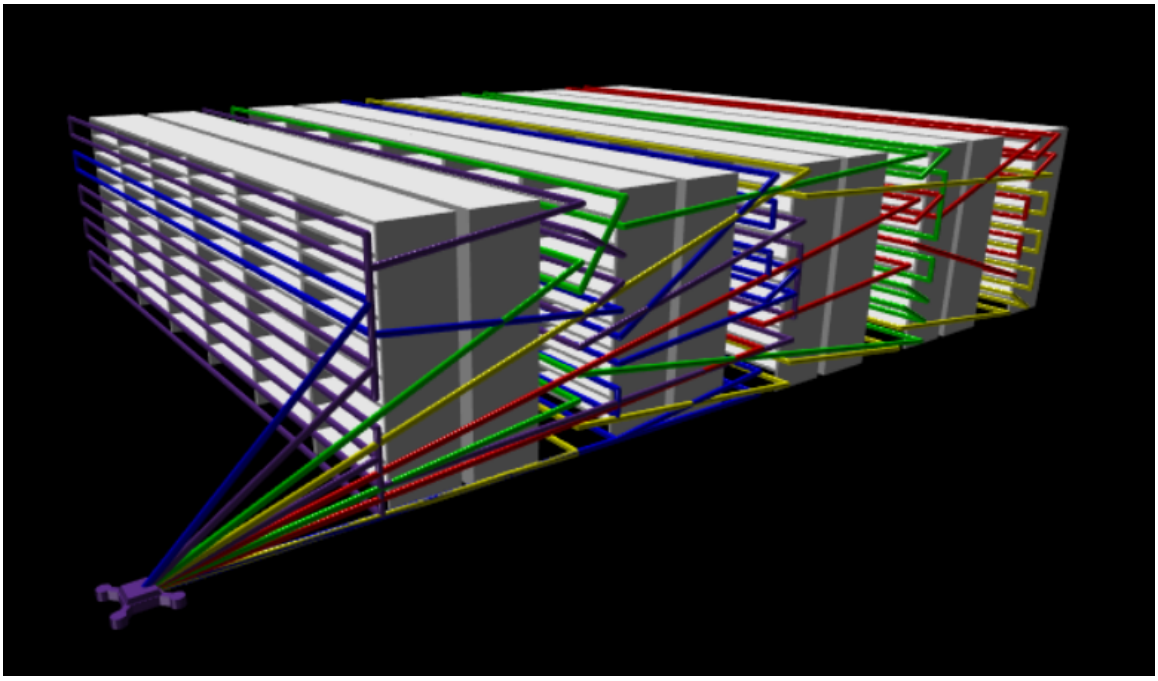

Figure 4.5: Min/Max Algorithm Optimal Trajectories

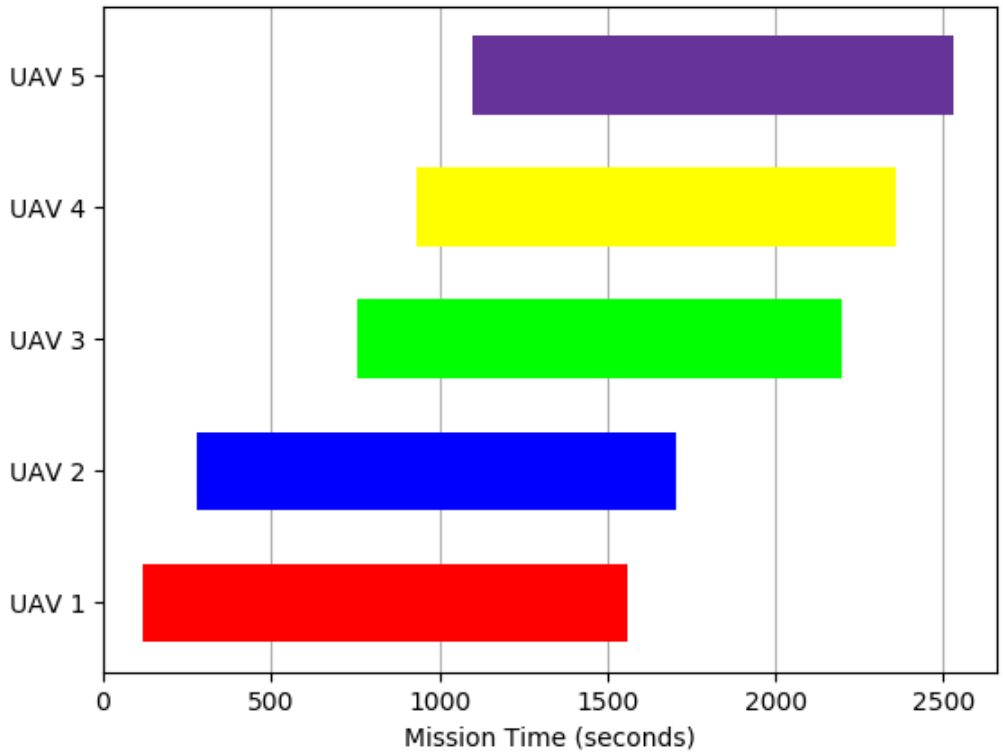

Figure 4.6: Min/Max Algorithm Optimum Deployment Times for Non-Collision Event

4.5 Multi-Depot Algorithm Results

The multi-depot operational setup resulted in shorter vehicle flight times, laid out in Table 4.4. Visualization of these trajectories can be found in Figure 4.7. Collision avoidance algorithm results are shown in Figure 4.8. This methodology had a resulting mission time of 33 minutes and 15.6 seconds, a 21.2% mission time reduction from the baseline. The ability to remove the initial vehicle offsets and allow for a preliminary separation barrier between vehicles has a large impact on decreasing the overall mission time. Remember, **Hypothesis 3: Deploying the UAVs from multiple depots, rather than a single depot, will significantly reduce the overall mission time in the full-size model because the vehicles will have greater overlaps in flight times with the ability to deploy simul-**

taneously. Substantial mission time savings with the multi-depot algorithm enables the acceptance of Hypothesis 3.

Table 4.4: Multi-Depot Algorithm Flight Times

UAV #	Required Flight Time
1	23 min 25.8 sec
2	23 min 29 sec
3	23 min 16.9 sec
4	23 min 21.1 sec
5	23 min 22.6 sec

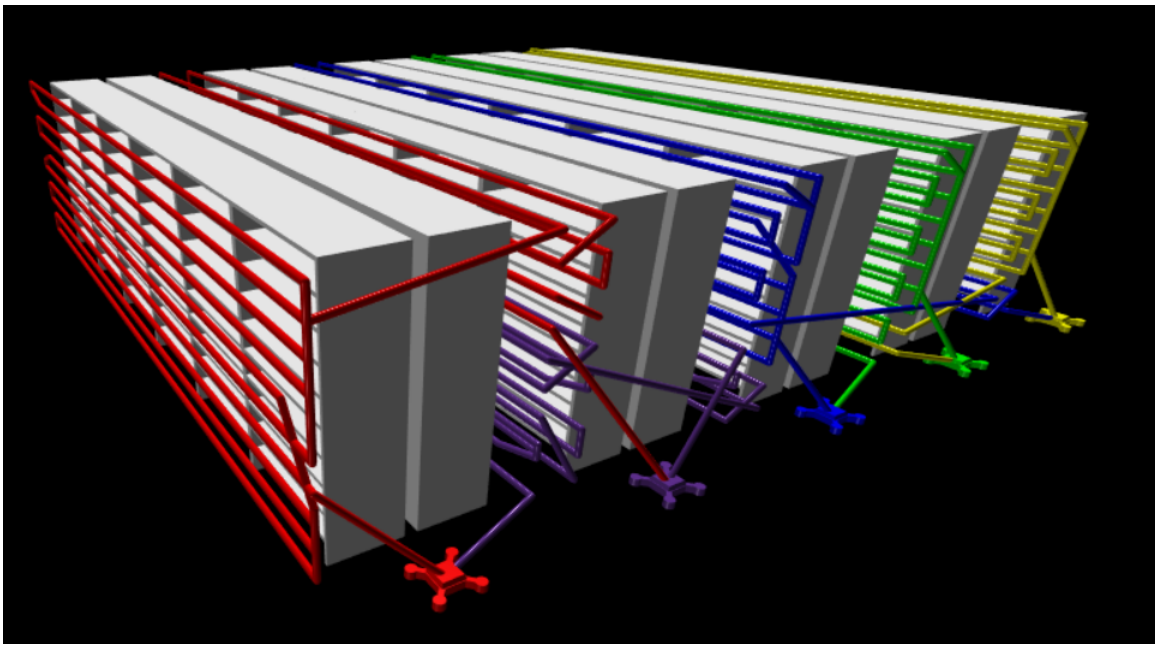

Figure 4.7: Multi-Depot Algorithm Optimal Trajectories

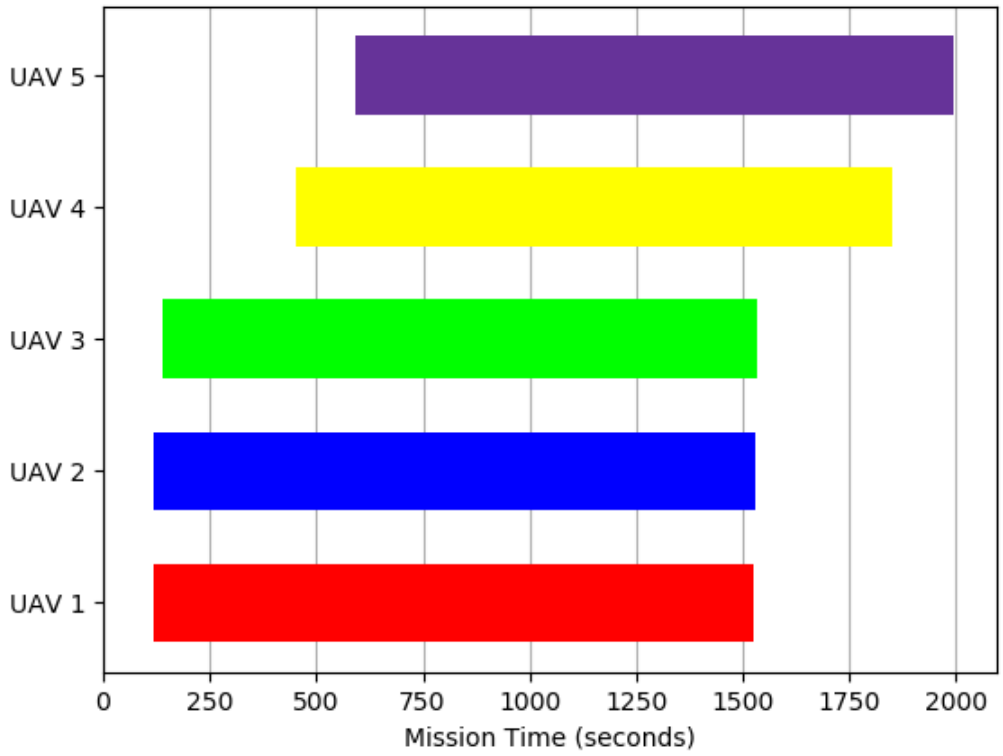

Figure 4.8: Multi-Depot Algorithm Optimum Deployment Times for Non-Collision Event

4.6 Algorithm Combination Results

After running the algorithms individually, they are also simulated in combination with one another to see if further mission time improvements are observed. The results of these combinations are laid out within this section and the analysis and comparison between simulations is discussed in Section 4.7.

4.6.1 Single Stage and Min/Max Algorithm Results

Combining the single stage algorithm with the min/max objective function produced the results shown in Table 4.5, Figure 4.9, and Figure 4.10.

Table 4.5: Single Stage & Min/Max Algorithm Flight Times

UAV #	Required Flight Time
1	23 min 53 sec
2	23 min 53.2 sec
3	24 min 1.8 sec
4	23 min 51.6 sec
5	24 min 1.9 sec

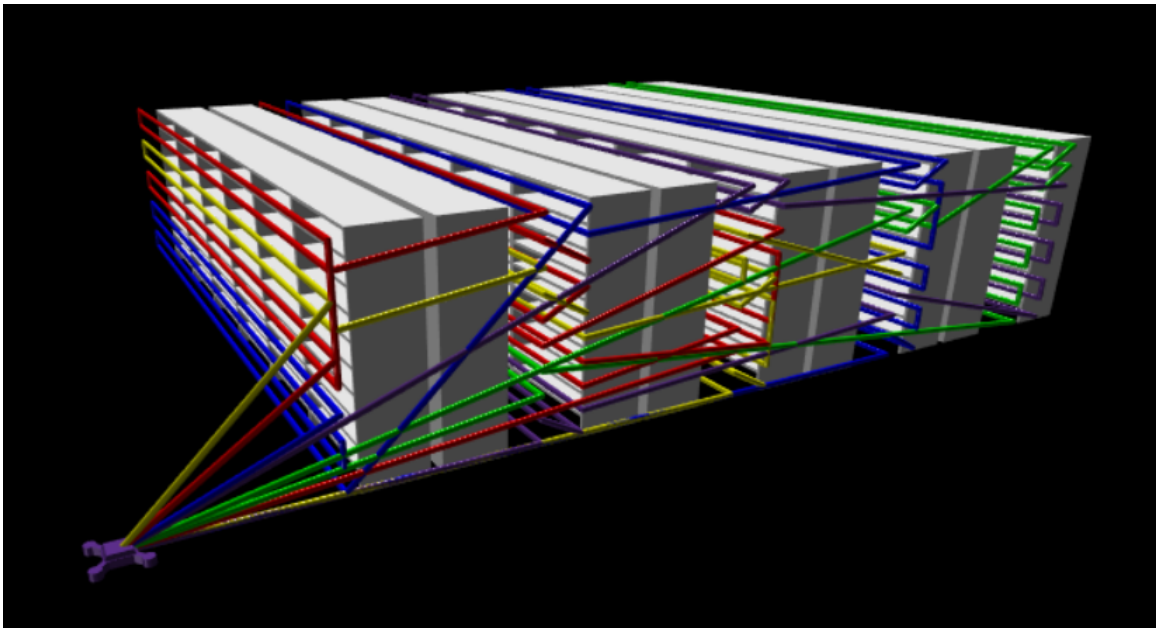

Figure 4.9: Single Stage & Min/Max Algorithm Optimal Trajectories

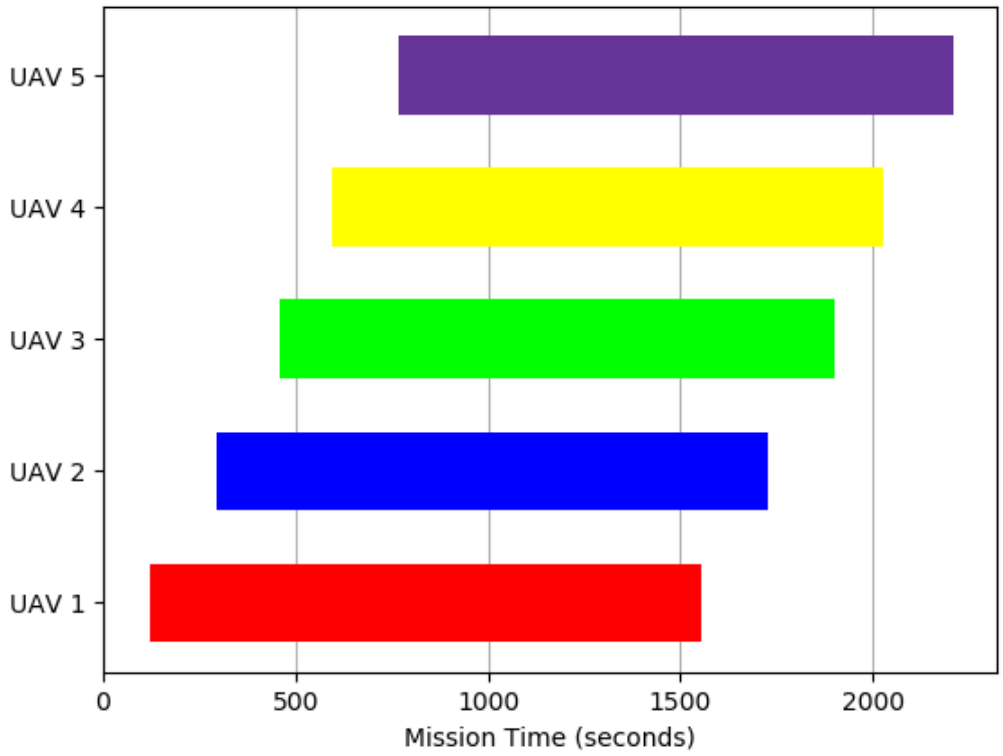

Figure 4.10: Single Stage & Min/Max Algorithm Optimum Deployment Times for Non-Collision Event

4.6.2 Single Stage and Multi-Depot Algorithm Results

Next, the single stage algorithm was packaged with the multi-depot setup and formulated

Table 4.6, Figure 4.11, and Figure 4.12.

Table 4.6: Single Stage & Multi-Depot Algorithm Flight Times

UAV #	Required Flight Time
1	23 min 33 sec
2	23 min 17.6 sec
3	23 min 17.3 sec
4	23 min 35.5 sec
5	23 min 27 sec

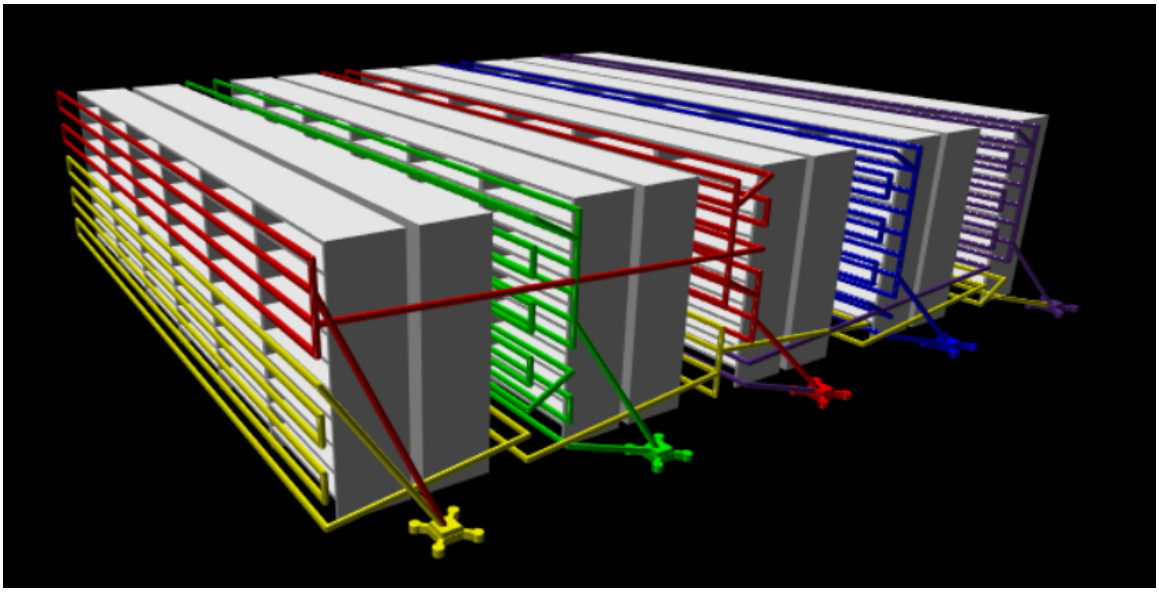

Figure 4.11: Single Stage & Multi-Depot Algorithm Optimal Trajectories

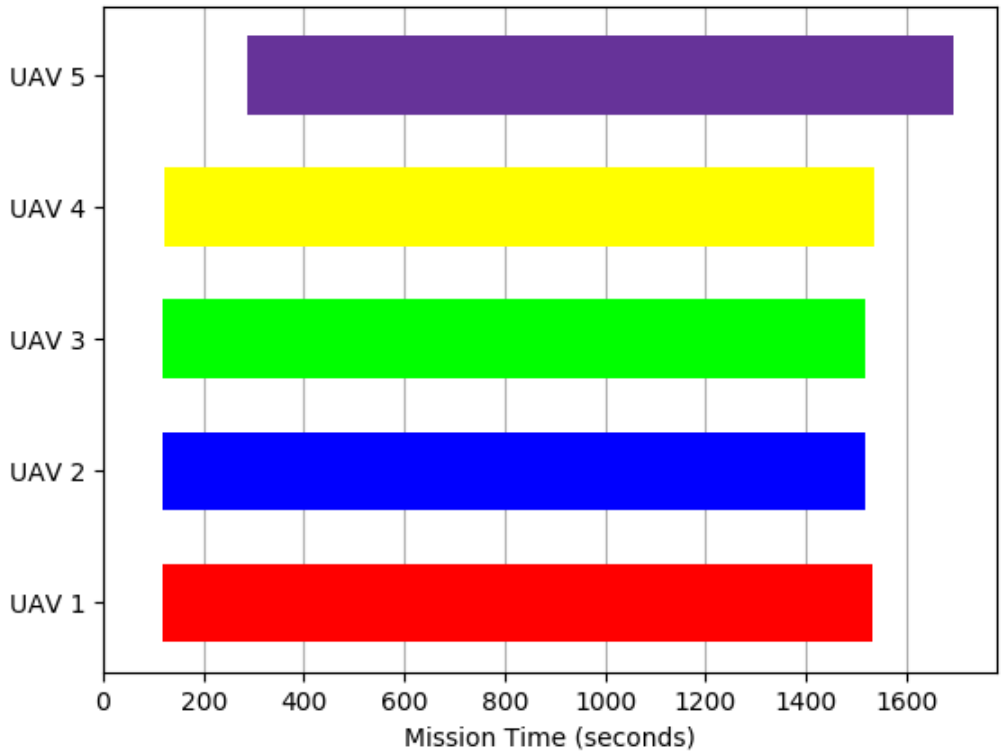

Figure 4.12: Single Stage & Multi-Depot Algorithm Optimum Deployment Times for Non-Collision Event

4.6.3 Min/Max and Multi-Depot Algorithm Results

The min/max and multi-depot algorithms were merged together to create the results in Table 4.7, Figure 4.13, and Figure 4.14.

Table 4.7: Min/Max & Multi-Depot Algorithm Flight Times

UAV #	Required Flight Time
1	23 min 22.9 sec
2	23 min 24.6 sec
3	23 min 18.8 sec
4	23 min 21.7 sec
5	23 min 22.9 sec

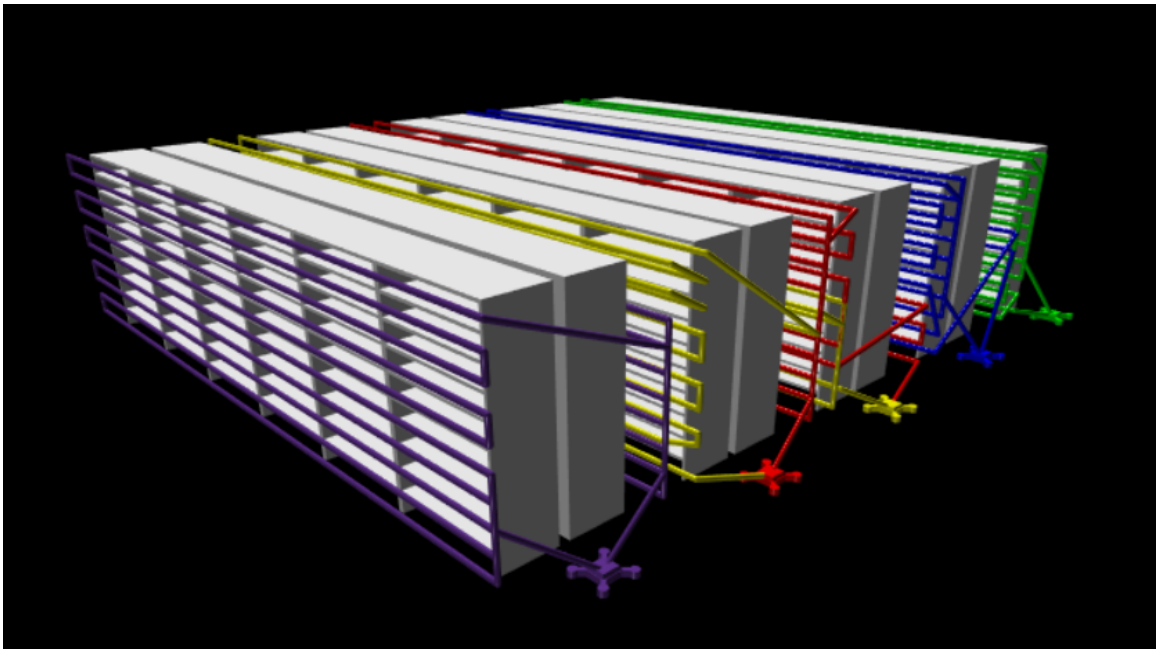

Figure 4.13: Min/Max & Multi-Depot Algorithm Optimal Trajectories

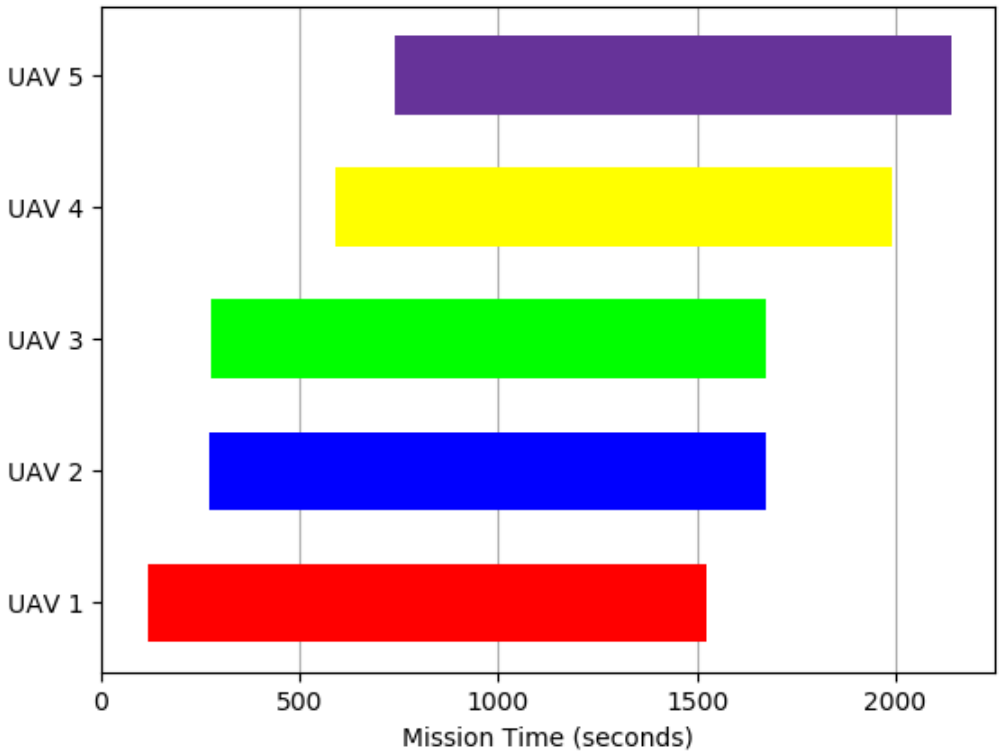

Figure 4.14: Min/Max & Multi-Depot Algorithm Optimum Deployment Times for Non-Collision Event

4.6.4 Single Stage, Min/Max and Multi-Depot Algorithm Results

Finally, all three methods were fused with the simulation results portrayed in Table 4.8, Figure 4.15, and Figure 4.16.

Table 4.8: Single Stage, Min/Max & Multi-Depot Algorithm Flight Times

UAV #	Required Flight Time
1	23 min 26.6 sec
2	23 min 24.6 sec
3	23 min 18.8 sec
4	23 min 26 sec
5	23 min 22.9 sec

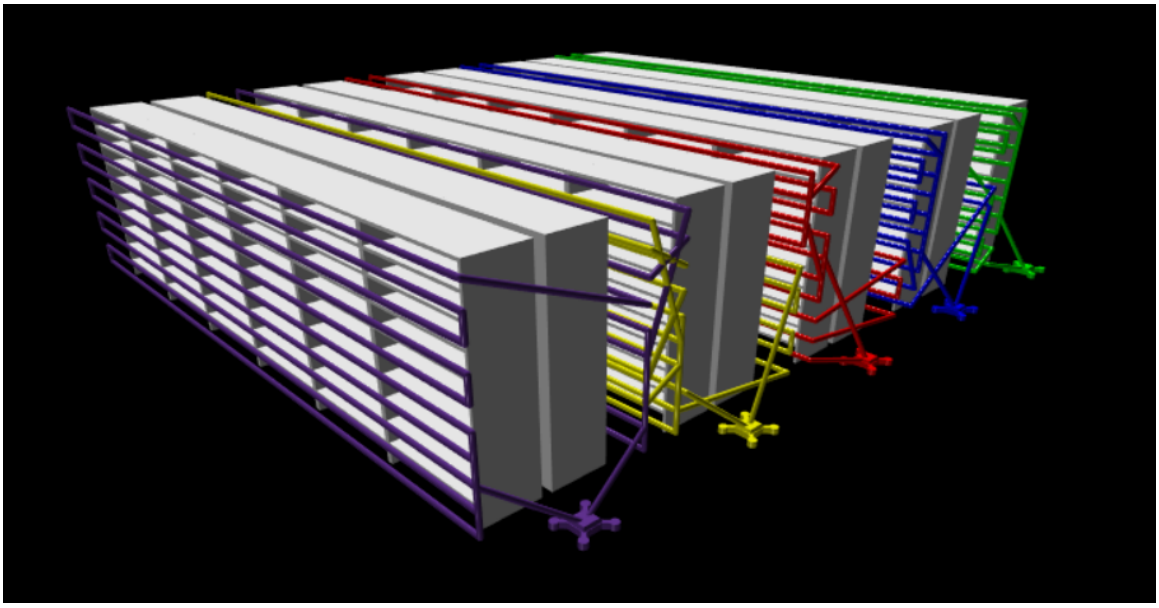

Figure 4.15: Single Stage, Min/Max & Multi-Depot Algorithm Optimal Trajectories

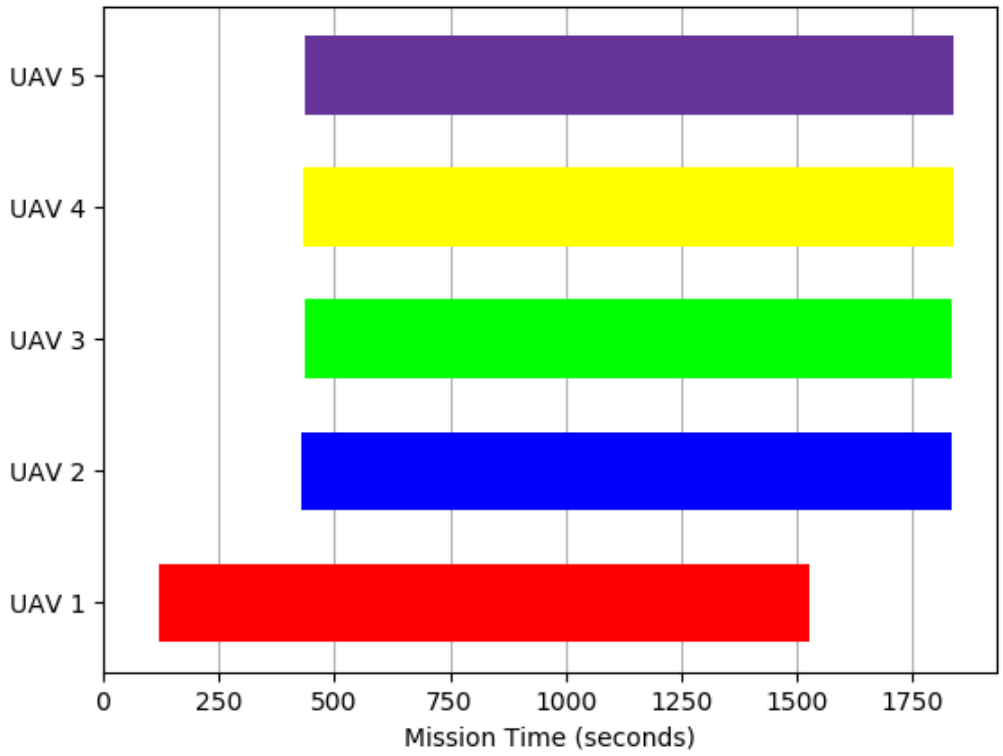

Figure 4.16: Single Stage, Min/Max & Multi-Depot Algorithm Optimum Deployment Times for Non-Collision Event

4.7 Simulation Comparisons

As discussed, each of the considered improvements was simulated individually, as well as in combination with one another, and then compared to the baseline results. The summary of inventory audit mission times and percent difference from the baseline for each experiment is organized in Table 4.9. Figure 4.17 and Figure 4.18 depicts each simulation's adherence to the minimum distance constraint between any two vehicles during the entire mission.

Comparing the results of the individual methods described in sections 3.3, 3.4, and 3.5 for the warehouse inventory test case; the use of multiple deployment locations shows the largest reduction in the overall mission time, -21.2% from the baseline.

The use of the single stage method appeared to significantly improve the results of the baseline, as well as improving the results of the min/max and multi-depot methods. In fact, the best overall mission time of 28 minutes and 16 seconds (-37.1% from the baseline) resulted from using the multi-depot operational setup in combination with the single stage logic. The trajectories and offset times for this combination are shown in Figure 4.11 and 4.12, respectively.

Table 4.10 organizes the optimization computation time for each method. It is important to note that these computation times only correspond to the trajectory generation computation times and do not include the collision avoidance algorithm computation times. The collision avoidance computing time is dependent on the number of feasible trajectory sets that are required to be solved for each methodology. The notable decrease in computation time compared to the previous method is a result of the model dimensionality reduction discussed in Section 3.6. Due to the efficiency of the new algorithm, parallel computing resources were not needed to execute the simulations, as was necessary for the previous method.

Comparisons between the results of the previous method and the new algorithm results are not justified due to initial differences in the models. The main difference was the initial node network setup that involved the dimensionality reduction in the new model. As a result of the increased efficiency of the new model, the optimization process was able to run closer to completion, with a cutoff tolerance of 0.005, rather than the previous model which had a tolerance of 0.05. These differences would not allow for accurate assessments of the proposed changes, which warranted the need for a baseline.

Table 4.9: Results Summary

#	Simulation Description	Mission Time	% Diff
0	UAS-Based Inventory Tracking Solution [6]	79 min 13 sec	N/A
1	Baseline	41 min 7.8 sec	0%
2	Single Stage	35 min 32.2 sec	-14.6%
3	Min//Max	42 min 12.1 sec	+2.6%
4	Multi-Depot	33 min 15.6 sec	-21.2%
5	Single Stage & Min/Max	36 min 50.9 sec	-11.0%
6	Single Stage & Multi-Depot	28 min 16 sec	-37.1%
7	Min/Max & Multi-Depot	35 min 42.9 sec	-14.1%
8	Single Stage, Min/Max & Multi-Depot	30 min 40.2 sec	-29.1%

Table 4.10: Optimization Computation Time

#	Simulation Description	Computation Time
0	UAS-Based Inventory Tracking Solution [6]	$\approx$ 86,400 sec
1	Baseline	828 sec
2	Single Stage	828 sec
3	Min//Max	424 sec
4	Multi-Depot	789 sec
5	Single Stage & Min/Max	424 sec
6	Single Stage & Multi-Depot	789 sec
7	Min/Max & Multi-Depot	7174 sec
8	Single Stage, Min/Max & Multi-Depot	7174 sec

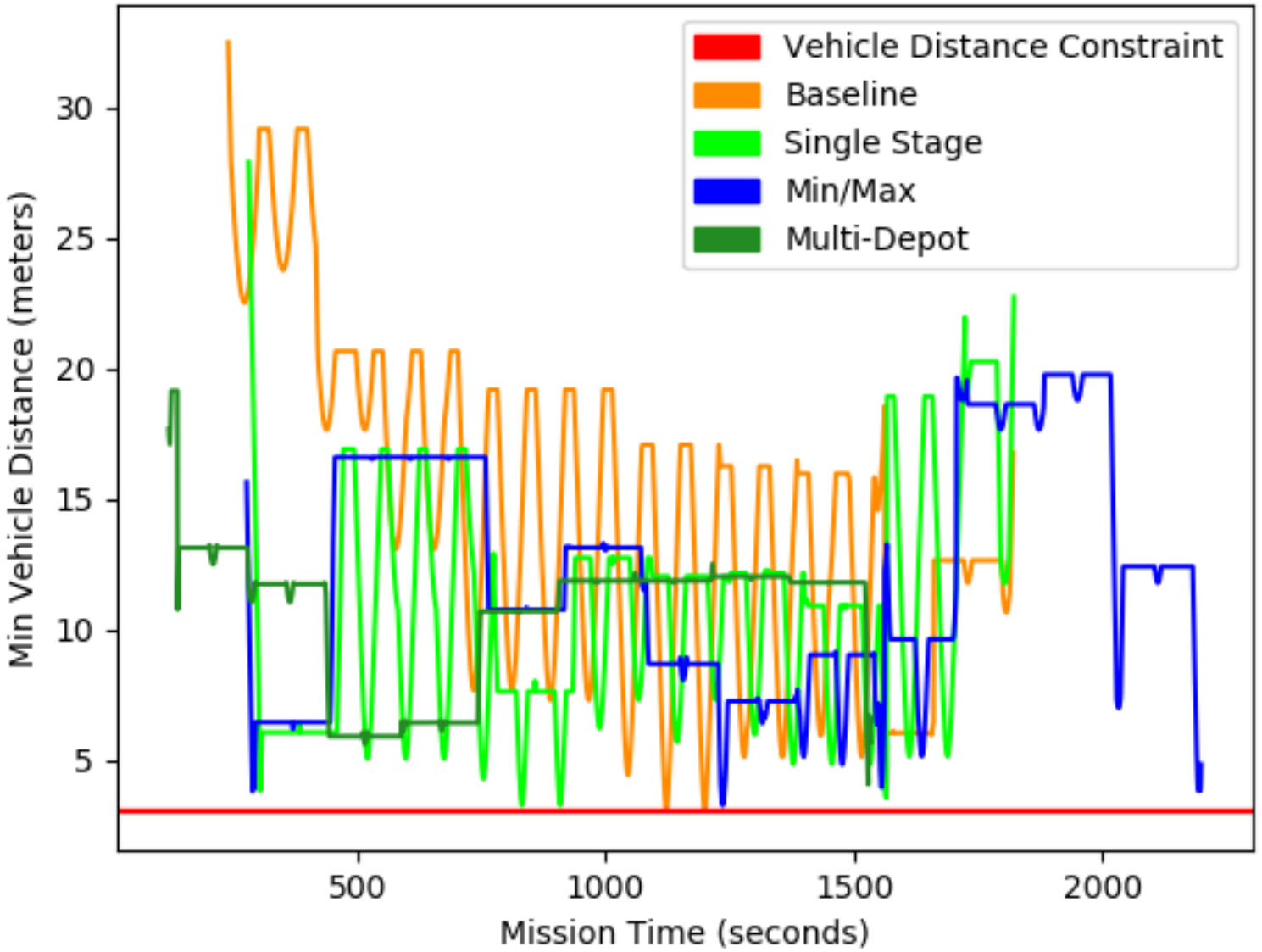

Figure 4.17: Minimum Vehicle Distance Constraint for Individual Algorithms

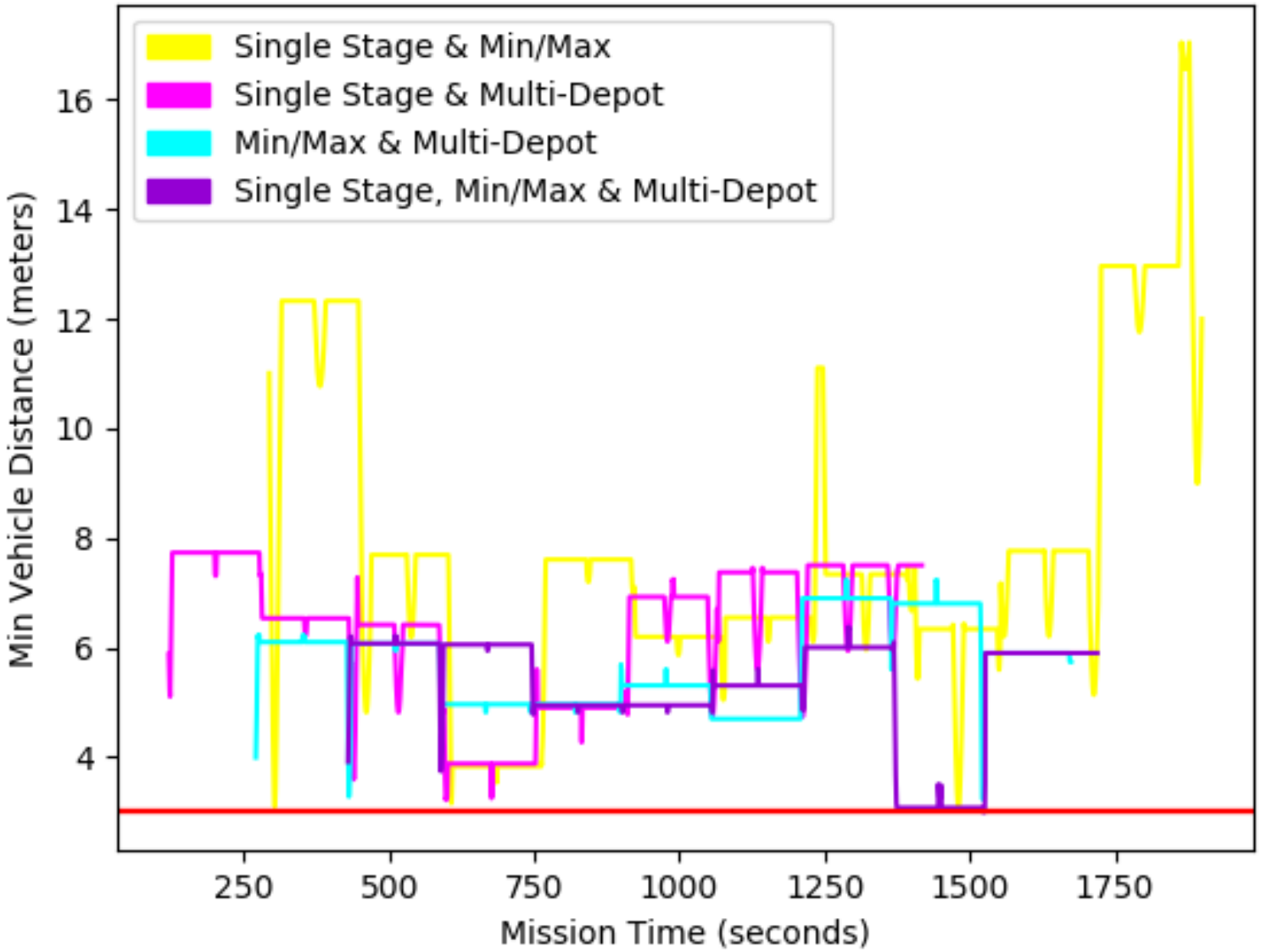

Figure 4.18: Minimum Vehicle Distance Constraint for Combined Algorithms

CHAPTER 5
SENSITIVITY ANALYSIS

The results of the point solution simulations from chapter 4 concluded with the single stage and multi-depot algorithm combination providing the best overall mission time. To further examine this algorithm, a sensitivity analysis is set up with a range of design parameters to test on the single stage/multi-depot combination. This experiment is established to show boundaries of this algorithm and to explore different input combinations, rather than point solutions in order to predict possible outcomes.

This experiment, once again, used a homogeneous fleet of vehicles, although, many vehicle parameters were manipulated throughout, such as vehicle endurance, scan speed, and cruise speed. Varying these parameters will help establish the requirements a vehicle must meet to complete a given configuration. Therefore, the warehouse geometry is also altered to address some different warehouse sizes and to better understand the limitations of an environment. For the collision avoidance algorithm, the minimum vehicle separation distance is varied, as the size of vehicles may require different separations. For the most part, the values for each design parameter were chosen to be greater than and less than the values used for the point solution discussed in chapter 4.

Three responses of the algorithm will be analyzed: fleet size, optimization computation time, and mission time. Fleet size and mission time were chosen as they are the quantities being optimized within the algorithm and will be impacted by the various parameter changes. The computation time is tested because it helps demonstrate the efficiency of the new algorithm.

5.1 Experimental Setup

Fixed Parameters:

- Setup Time: 2 minutes
- Vehicle Acquisition Cost: 3 minutes
- Relative Tolerance: 0.005
- Optimization Computation Time Limit: 1 hour
- Max. Fleet Size: 10 vehicles

Design Parameters:

- Vehicle Endurance: 20, 25, or 30 minutes
- Cruise Speed: 1, 2, or 3 m/s
- Scan Speed: 0.1, 0.2, or 0.3 m/s
- Min. Vehicle Distance: 1, 3, or 5 meters
- Warehouse Volume:
  - Shelf Length: 10, 25, or 40 meters
  - Number of Waypoints:
    - Number of Rows of Shelves: 5, 10, or 15 rows
    - Number of Shelves per Row: 4, 10, or 16 shelves

Design Responses:

- Fleet Size
- Optimization Computation Time
- Mission Time

A full-factorial design of experiments is created with the seven 3-level factors listed above. This resulted in a total of 2,187 iterations. Each iteration terminates when the model is results in a feasible solution or is determined to be infeasible. In addition to a model being physically infeasible, in order to simplify computation time and expense, a model is also declared infeasible if it exceeds the one hour computation time limit without finding a solution or if the fleet size exceeds 10 vehicles.

5.2 Results

The goal of this sensitivity analysis was to examine the effects of varying different input parameters on the overall results and robustness of the algorithm.

In the first graph, Figure 5.2, different combination of design variable inputs were examined to understand the feasible limits of this algorithm. This graph compared vehicle endurance, scan speed, cruise speed, and warehouse volume, which was a function of the shelf length, number of rows, and number of shelves. The only input which is unaccounted for in this graph is the minimum vehicle distance, as this design variable does not play a role in model feasibility since it only impacts the collision avoidance part of the algorithm. Figure 5.1 shows the legend for this first graph. From Figure 5.2, it can be seen that the feasible region grows as the scan speed and vehicle endurance increase. Conversely, the feasible region grows as the warehouse volume decreases and the cruise speed shows minimal effect on the feasibility of the algorithm. This minimal effect is likely due to the significantly smaller amount of mission time spent in the cruise speed regions rather than the scan speed areas. Logically, these results make sense. The largest impacts appear to stem from the scan speed and warehouse volume. Therefore, if using a scan speed of 0.1 m/s, the volume must stay below 2,000 m^3. A scan speed of 0.3 m/s, would require a volume less than 7,000 m^3. Finally, a scan speed of 0.5 m/s, would need a warehouse less than 11,000 m^3 in size to be feasible. For this sensitivity analysis and model, any volumes above 11,000 m^3 returned infeasible.

The graphs shown in Figure 5.3 to Figure 5.7 begin to explore the sensitivity of the fleet size from the varying parameters. When examining the fleet size, the entire design space (feasible and infeasible) is investigated. Table 5.1 concisely describes the relationship between these values. In Figure 5.8, the endurance, scan speed, and volume parameters are combined to help facilitate the prediction of fleet size for given inputs.

Table 5.1: Design Variables vs. Fleet Size

Input	Correlation
Vehicle Endurance	Weak Negative
Scan Speed	Strong Negative
Cruise Speed	None
Volume	Strong Positive
Number of Waypoints	Strong Positive

Next, the optimization computation time is examined with the graphs in Figure 5.9 to Figure 5.14. These graphs have narrowed down the design space to only show the feasible options, as the time limit constraint skews the data trends. Table 5.2 concisely describes the relationship between these values. In Figure 5.15, the number of waypoints, endurance, and scan speed parameters are combined to help facilitate the computation time prediction for specified inputs.

Table 5.2: Design Variables vs. Optimization Computation Time

Input	Correlation
Vehicle Endurance	None
Scan Speed	None
Cruise Speed	None
Volume	None
Number of Waypoints	Strong Positive up to 50 waypoints
Fleet Size	Weak Positive

Finally, the mission time output is studied through the graphs in Figure 5.16 to Figure 5.22. Once again, these graphs only show the feasible design points as the infeasible missions times are non-existent. Table 5.3 concisely describes the relationship between these values. In Figure 5.23, the minimum vehicle distance, endurance, and scan speed variables are combined in a single graph to aid in predicting the overall mission time for given inputs.

Table 5.3: Design Variables vs. Mission Time

Input	Correlation
Vehicle Endurance	Weak Positive
Scan Speed	Weak Negative
Cruise Speed	None
Min. Vehicle Distance	Strong Positive
Volume	None
Number of Waypoints	None
Fleet Size	Weak Positive

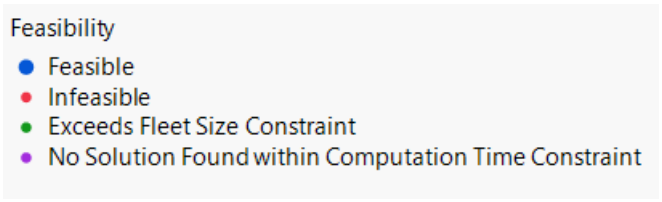

Figure 5.1: Graph Legend for Figure 5.2 and Figure 5.8

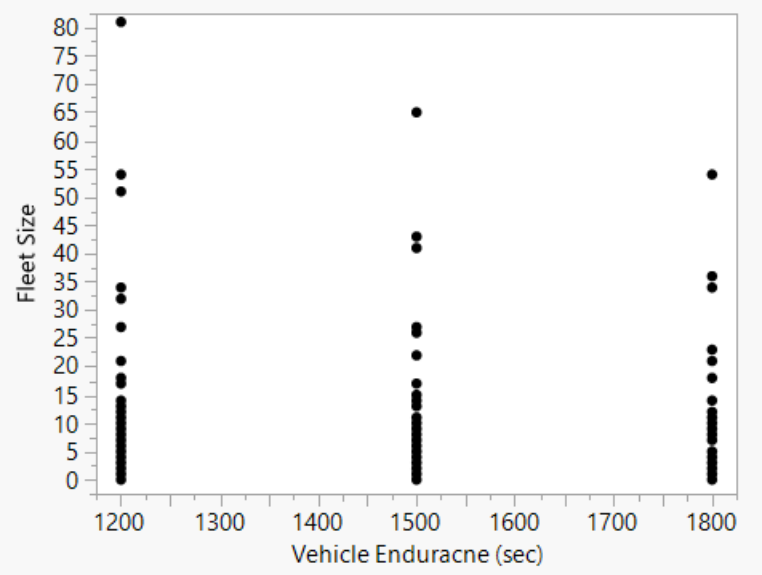

Figure 5.3: Vehicle Endurance vs. Fleet Size

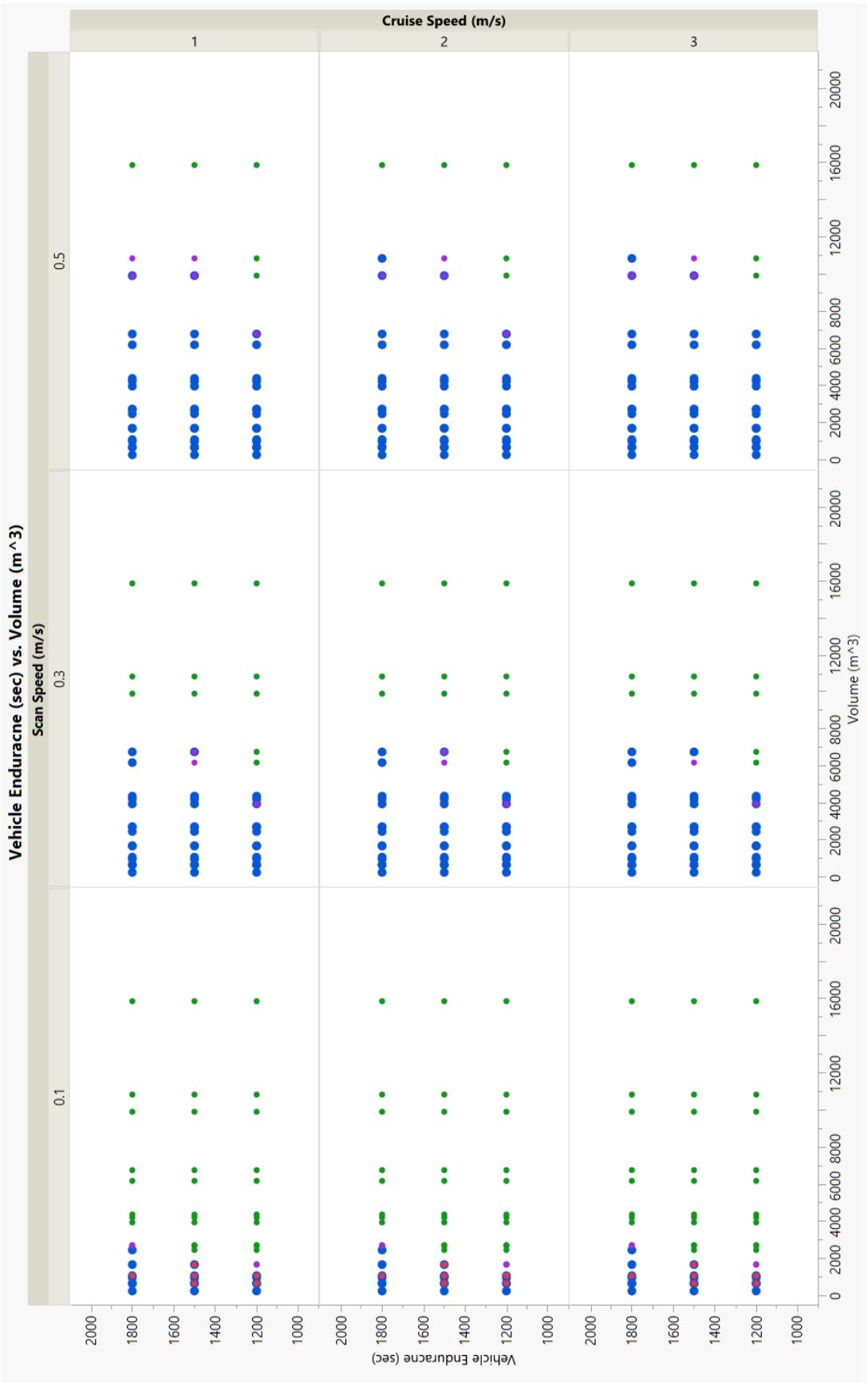

Figure 5.2: Design Variables vs. Feasibility

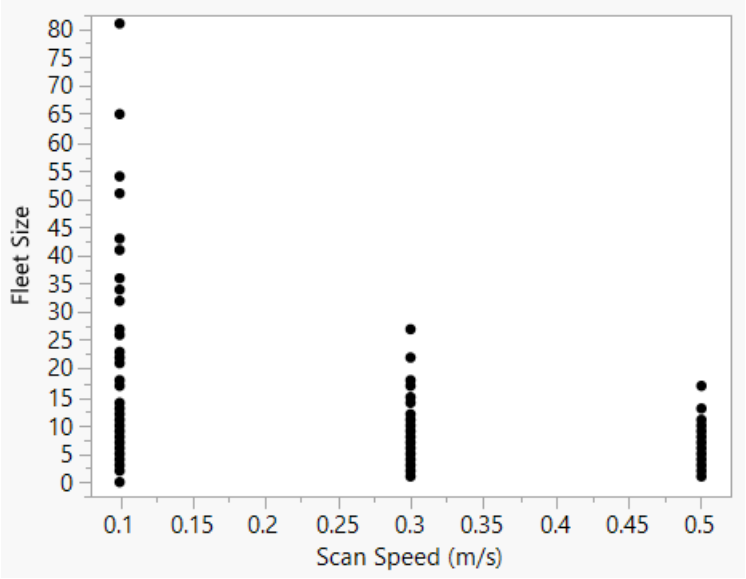

Figure 5.4: Scan Speed vs. Fleet Size

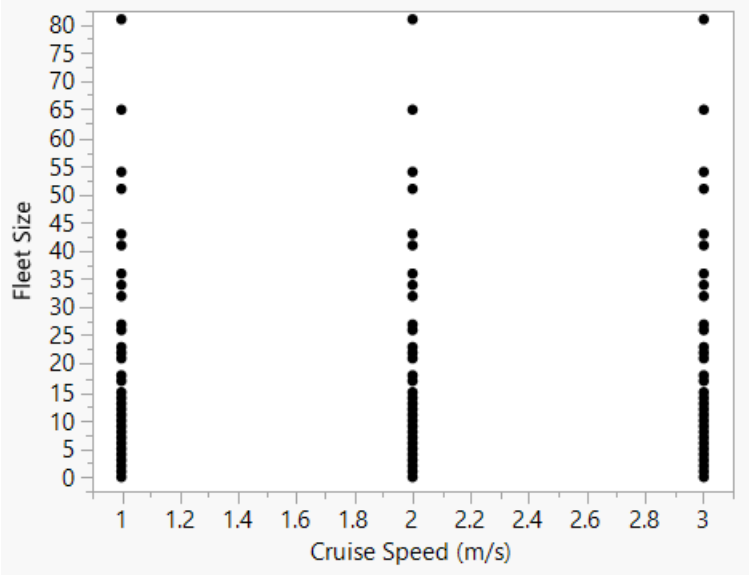

Figure 5.5: Cruise Speed vs. Fleet Size

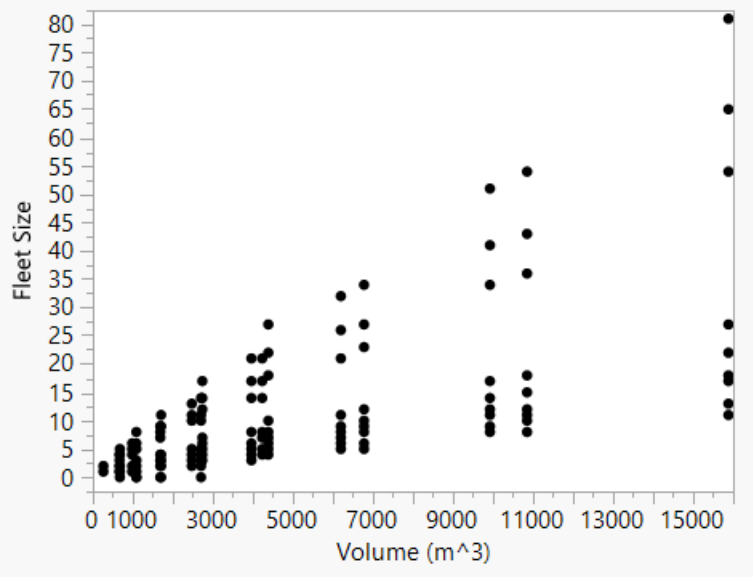

Figure 5.6: Volume vs. Fleet Size

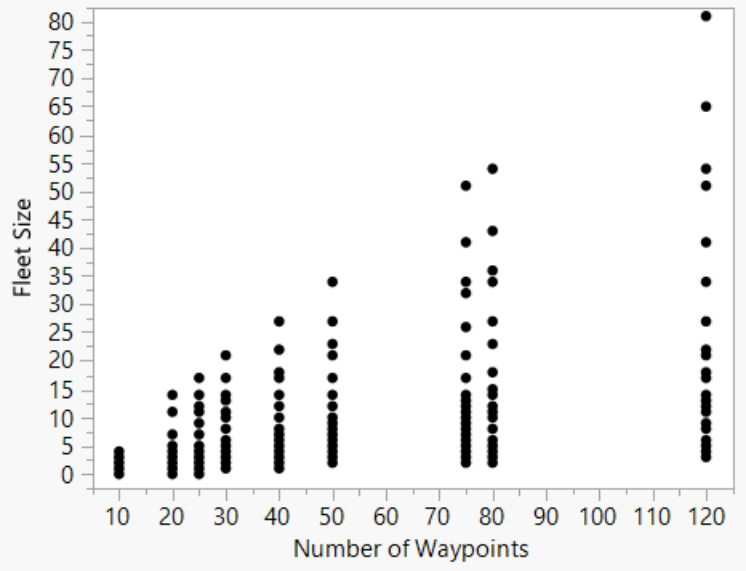

Figure 5.7: Number of Waypoints vs. Fleet Size

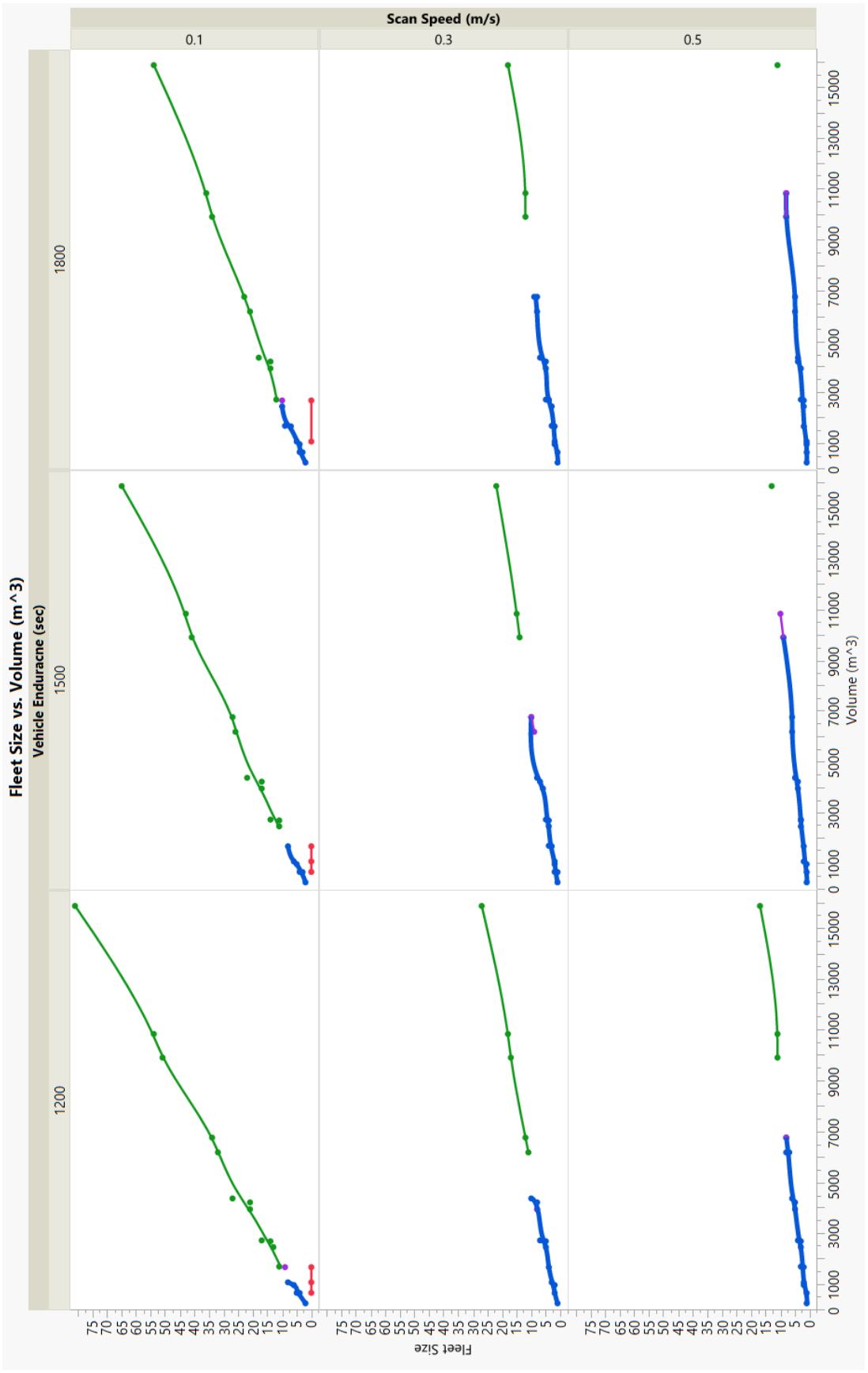

Figure 5.8: Design Variables vs. Fleet Size

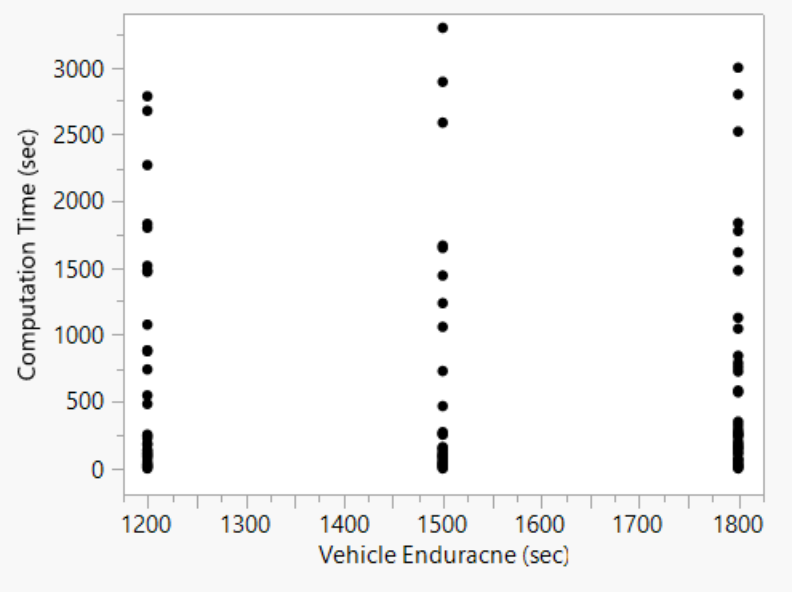

Figure 5.9: Vehicle Endurance vs. Optimization Computation Time

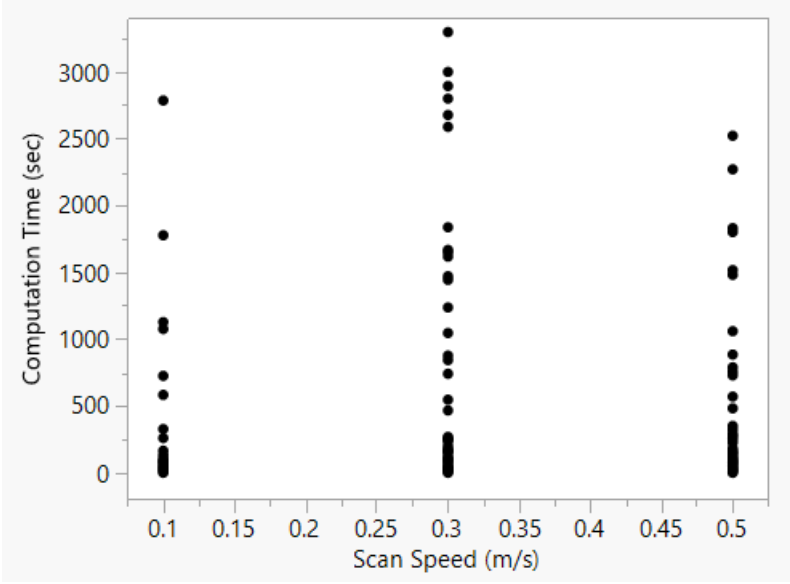

Figure 5.10: Scan Speed vs. Optimization Computation Time

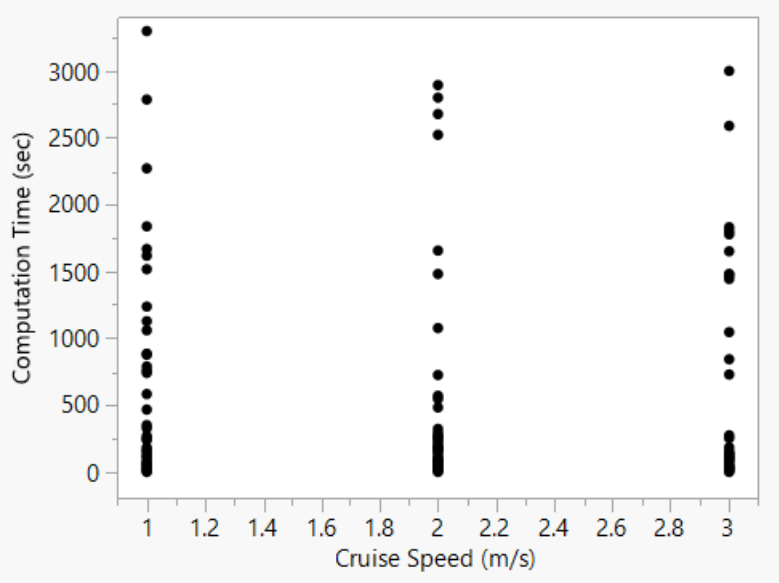

Figure 5.11: Cruise Speed vs. Optimization Computation Time

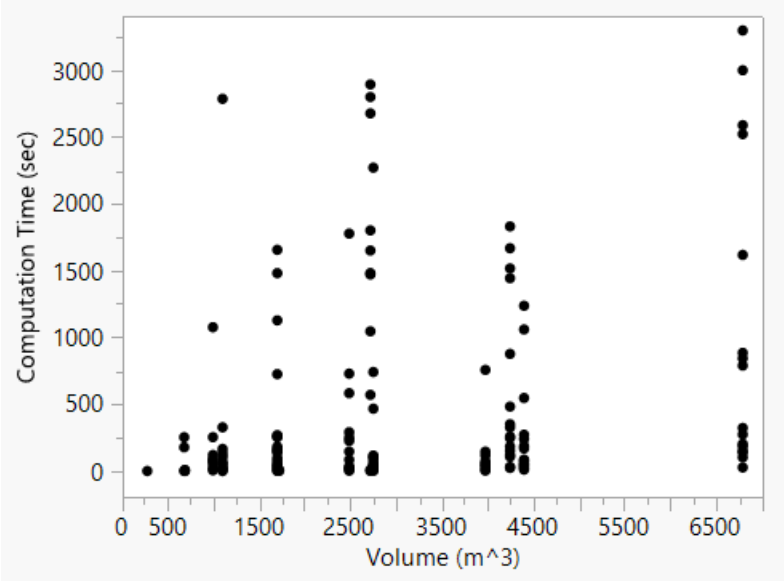

Figure 5.12: Volume vs. Optimization Computation Time

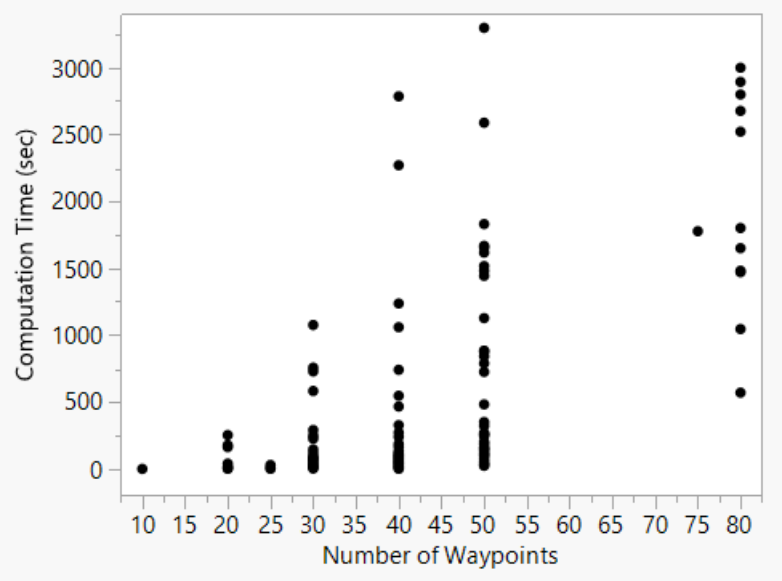

Figure 5.13: Number of Waypoints vs. Optimization Computation Time

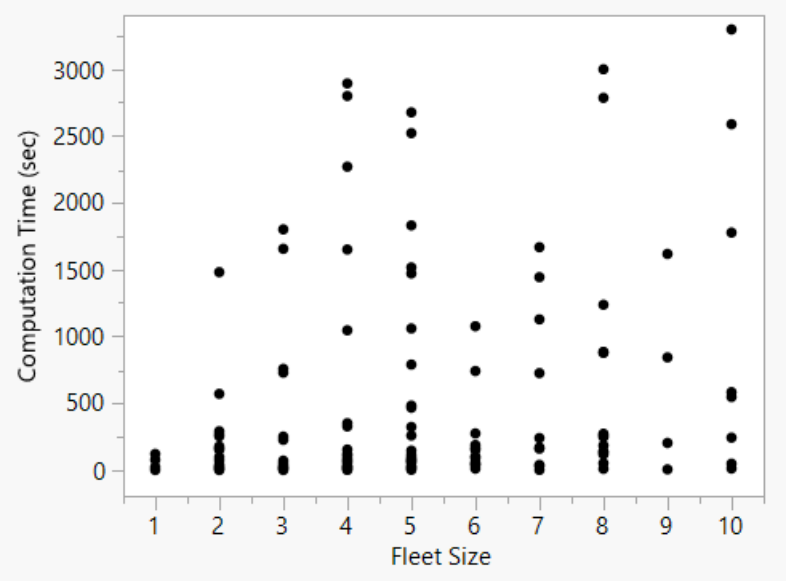

Figure 5.14: Fleet Size vs. Optimization Computation Time

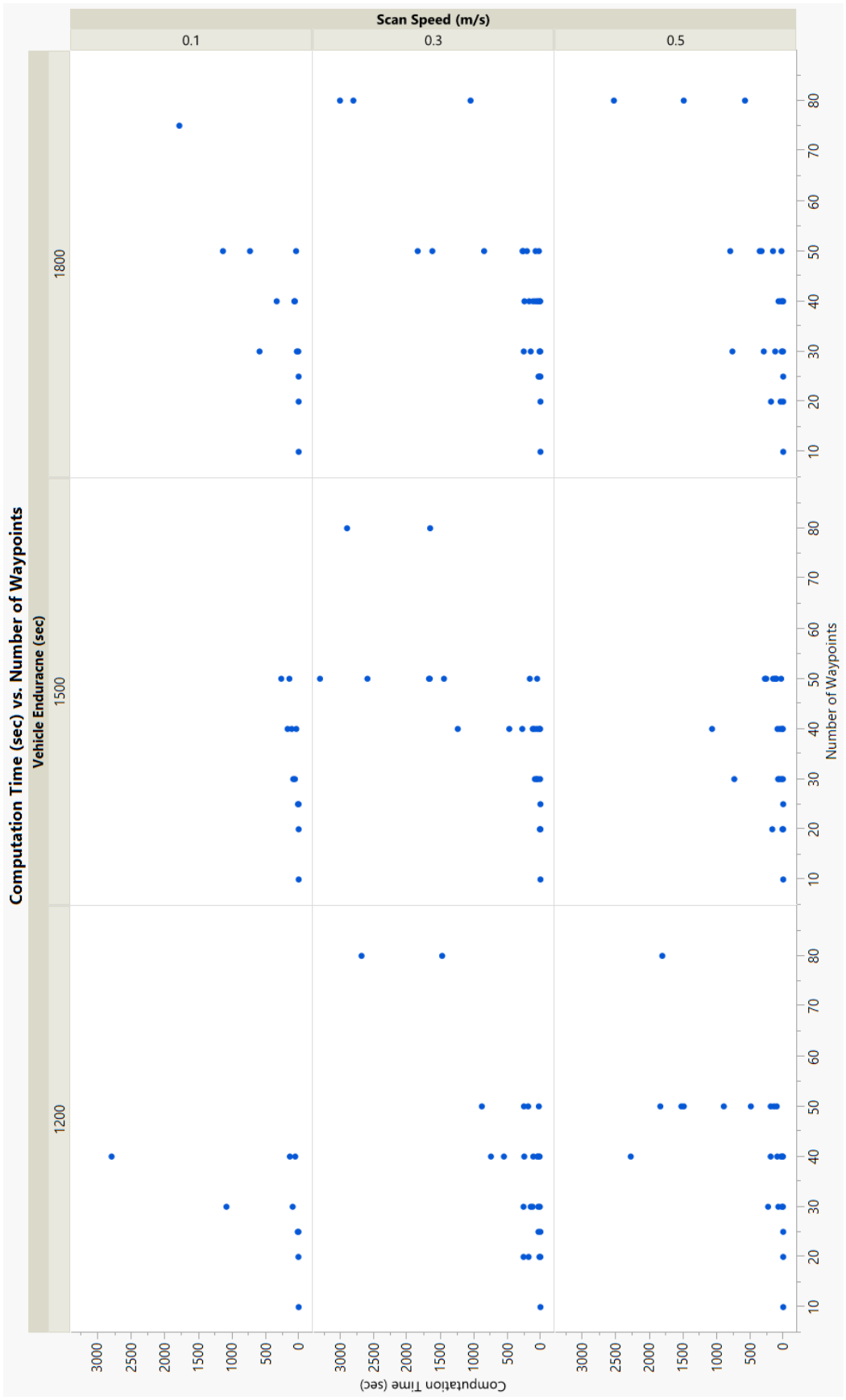

Figure 5.15: Design Variables vs. Optimization Computation Time

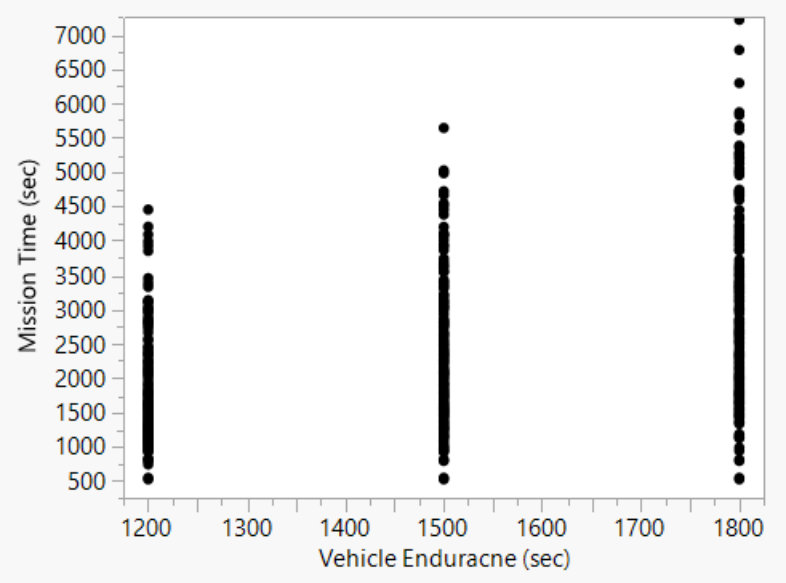

Figure 5.16: Vehicle Endurance vs. Mission Time

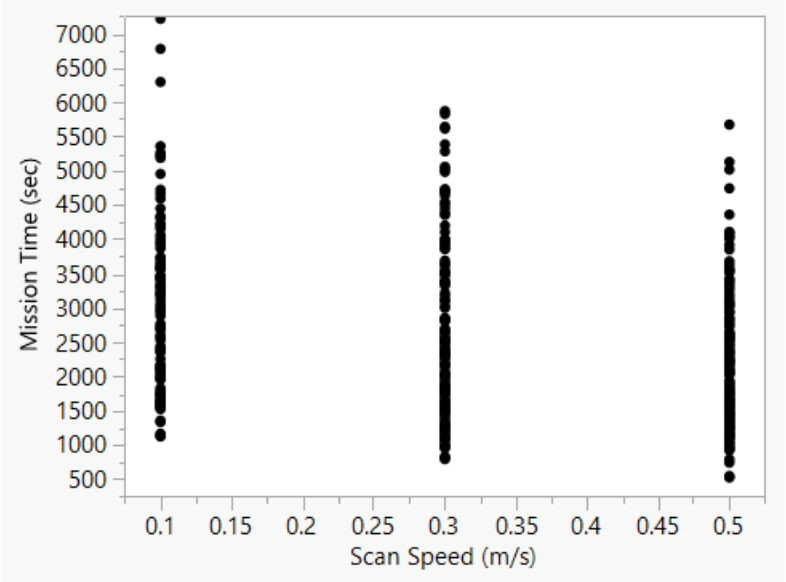

Figure 5.17: Scan Speed vs. Mission Time

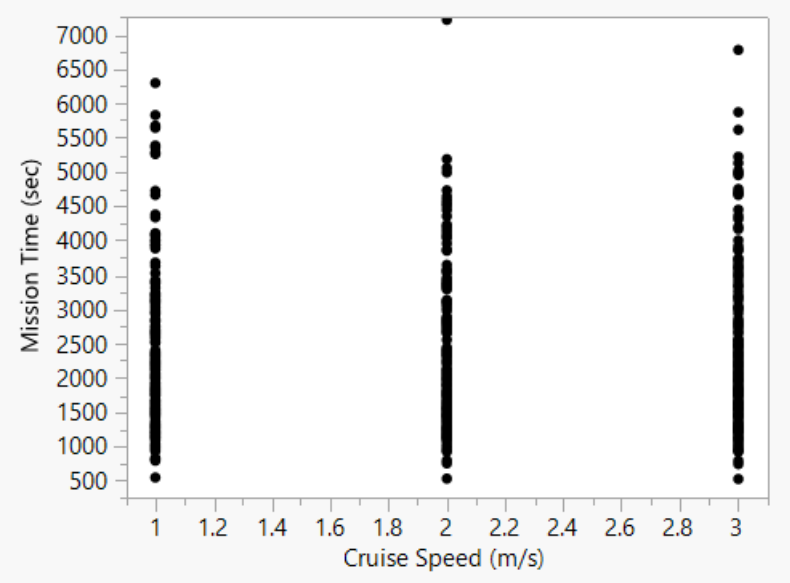

Figure 5.18: Cruise Speed vs. Mission Time

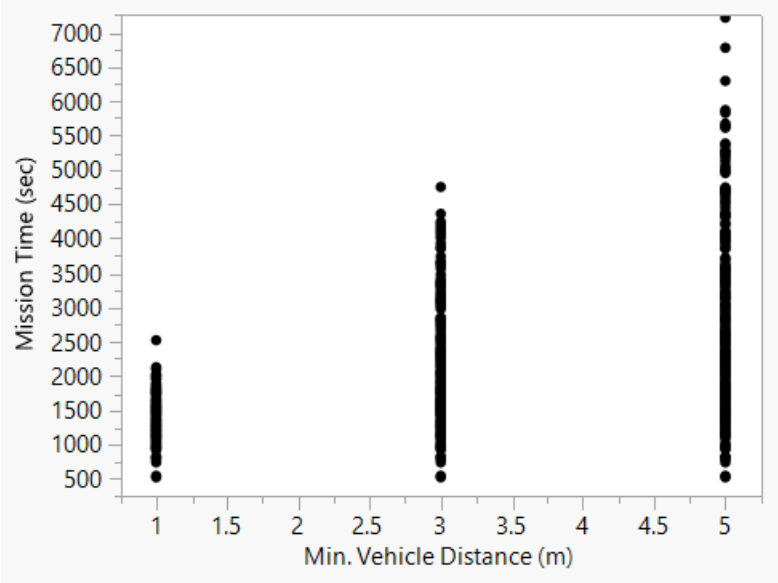

Figure 5.19: Min. Vehicle Distance vs. Mission Time

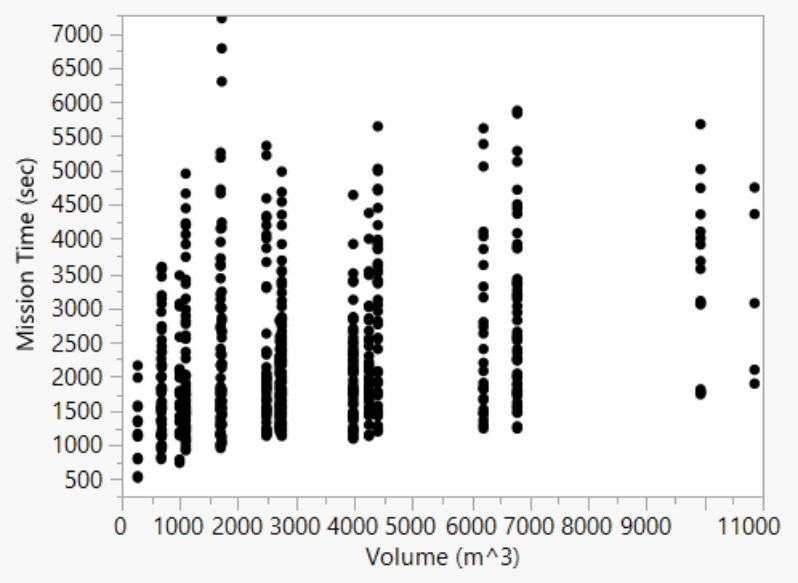

Figure 5.20: Volume vs. Mission Time

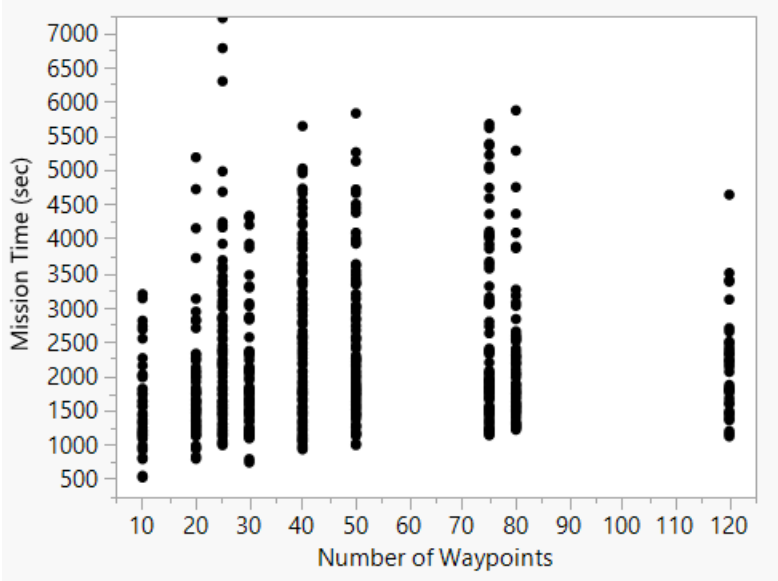

Figure 5.21: Number of Waypoints vs. Mission Time

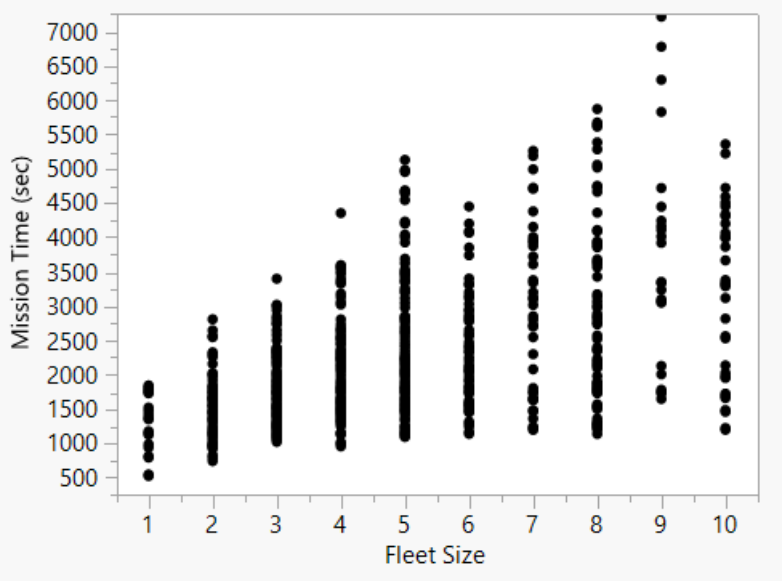

Figure 5.22: Fleet Size vs. Mission Time

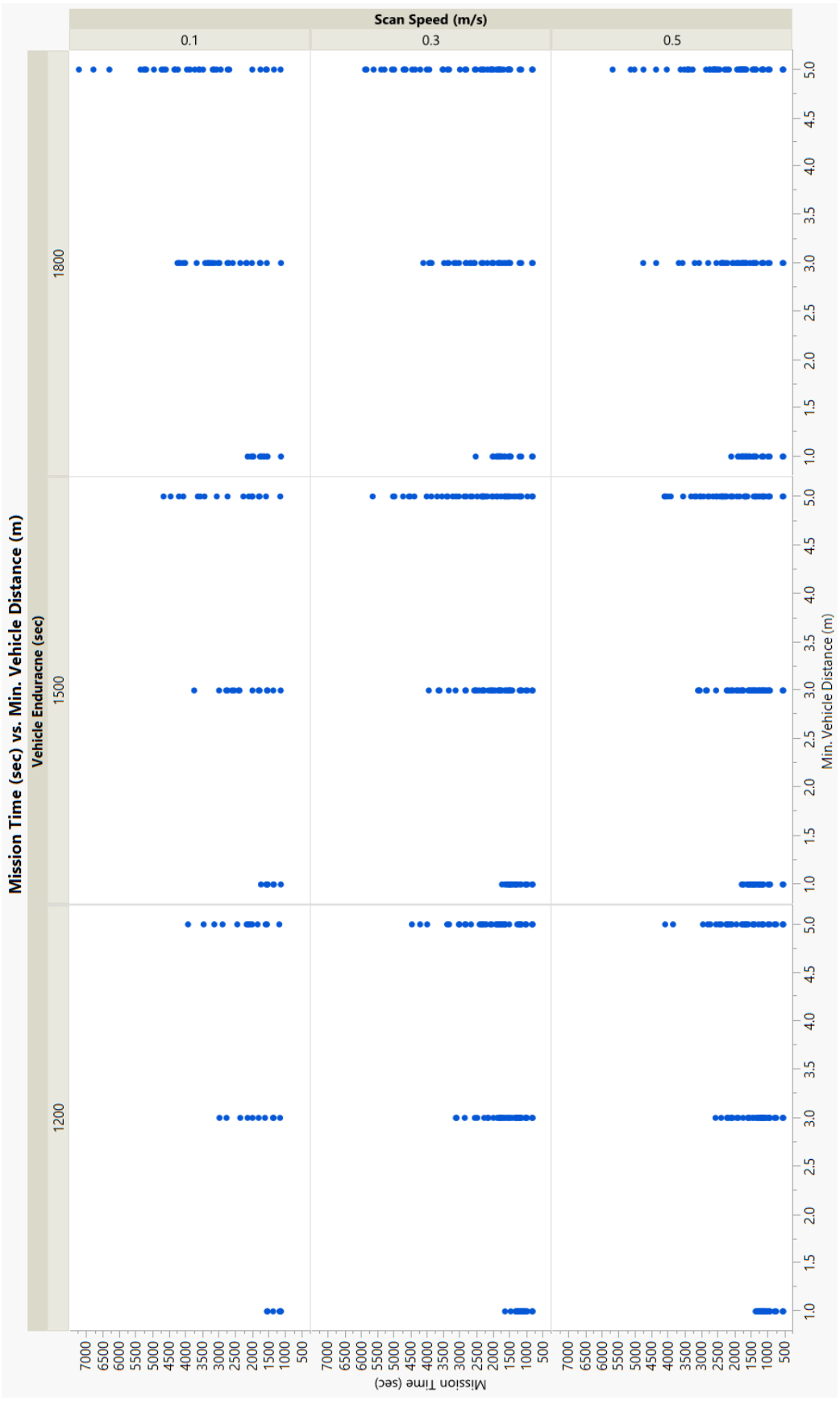

Figure 5.23: Design Variables vs. Mission Time

CHAPTER 6
CONCLUSION

6.1 Future Work

Future work on this problem may include further analysis of these methods through testing on a wide variety of experimental setups or other UAV applications and use cases. Explorations of different warehouse layouts and sizes or other confined environment models, such as search and rescue scenarios, could provide interesting insight to these algorithms. This algorithm would also benefit from having an initial capability to create dynamic or undefined models. This would help account for uncertainties within the environment and versatility of the algorithm [55] [56] [57], thus stepping closer to real-world application.

Full warehouse environments, like those mentioned in section 1.2, would require an additional component to obtain feasible results with the presented algorithm. Due to the massive size of entire warehouses, they will need to be divided into smaller and more manageable areas to accommodate the vehicle endurance and computation time restrictions. Utilizing graph partitioning techniques, such as those discussed in [58] [59] [60], would facilitate this process and allow for optimal integration with the new trajectory optimization.

Additional model reductions could be investigated for even further improvement to the algorithms efficiency. One avenue could be exploring the possibility of having two cameras on each vehicle with the ability to capture and track the products on parallel but opposite shelves in an aisle at the same time. This may not only reduce computational complexity, but also reduce fleet sizes and mission times as well.

6.2 Conclusion

Overall, a new multi-UAV trajectory optimization methodology for UAS operations in confined environment was introduced. As a practical example, a UAS-based inventory tracking problem was used. To optimally and safely operate multi-UAVs, multiple optimization problems using the single stage logic, min/max objective function, and multi-depot operational setup were proposed to improve an existing optimization framework. Numerical simulations were conducted to compare the baseline method and new proposed optimization methods. Significant benefits were observed from the single stage and multi-depot methods, whereas, the min/max objective function was not beneficial for reducing mission time in the warehouse inventory tracking use case. The outcomes from the individual algorithms became further enhanced when the combined algorithm results showed that the single stage logic in combination with the multi-depot operational setup proved to have the largest impact on reducing the overall mission time.

The single stage/multi-depot algorithm combination was analyzed with a range of input parameters to show how the algorithm responds to variance. It was observed that the vehicle's scan speed along with the warehouse volume played the most significant role in determining model feasibility. The fleet size response was most sensitive to the warehouse geometry parameters. The optimization computation time appeared to only be impacted by the number of waypoints and the fleet size. Lastly, the overall mission time saw the greatest influence from the minimum vehicle separation distance to avoid collisions.

The overall research objective to improve and optimize the total mission time of multiple UAVs performing a warehouse inventory audit was addressed with the work completed in this thesis. This work will be presented at the 2019 International Conference on Unmanned Aircraft Systems (ICUAS) [61]. The main contributions of this thesis lies within the following:

- A new trajectory optimization algorithm which can be applied to multi-UAV use

cases in confined and complex environments

- A method for reducing the dimensionality and complexity of a warehouse model which simplifies the computational expense of larger-scale problems
- A diverse set of parameter initialization combinations and corresponding response trends and feasibility

REFERENCES

[1] Y. Choi, Y. Choi, S. Briceno, and D. N. Mavris, "Three-dimensional UAS trajectory optimization for remote sensing in an irregular terrain environment," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2018, pp. 1101–1108.

[2] W. B. Carlson, *Tesla Inventor of the Electrical Age*. Princeton University Press, 2013.

[3] *Global unmanned aerial vehicle (UAV) market 2018-2025 - focus on UAV platforms, UAV payloads, UAV GCS, UAV data links, UAV launch and recovery systems*, 2018.

[4] "Global $2+ billion commercial drones (rotary blade drones, hybrid drones, nano drones fixed wing drones) market 2017-2022," *BioMedReports*, 2017.

[5] D. Zhang, Y. Xu, and X. Yao, "An improved path planning algorithm for unmanned aerial vehicle based on rrt-connect," in *2018 37th Chinese Control Conference (CCC)*, 2018, pp. 4854–4858.

[6] Y. Choi, M. Martel, S. Briceno, and D. Mavris, "Multi-UAV trajectory optimization and deep learning-based imagery analysis for a UAS-based inventory tracking solution," *AIAA SciTech 2019 Forum*, 2019.

[7] Y. Geng, Y. Li, and A. Lim, "A very large-scale neighborhood search approach to capacitated warehouse routing problem," in *17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, 2005, 8 pp.–65.

[8] *Steelcase site survey*, 2018.

[9] *2017 survey of occupational injuries illnesses*, 2018.

[10] *Watch out Amazon: Wal-Mart prepping drone delivery service, seeking fed's approval*, 2015.

[11] *Asset tracking and inventory management solutions market to be worth US $30.59bn by 2026 - tmr*, 2018.

[12] T. Jackson, "The flying drones that can scan packages night and day," *BBC News*, 2017.

[13] H. G. Zhu, H. Xin, and C. W. Zheng, "Research on UAV path planning," *Applied Mechanics and Materials*, vol. 58-60, p. 2351, Jun. 2011.

[14] "Walmart Distribution Center Network USA — MWPVL," 2018.

[15] P. Gun, A. Hill, and R. Vujanic, *Multi-vehicle trajectory optimisation on road networks*, 2018.

[16] A. Alves Neto, D. G. Macharet, M. F. Campos, and M., "On the generation of trajectories for multiple UAVs in environments with obstacles," *Journal of Intelligent Robotic Systems*, vol. 57, no. 1-4, pp. 123–141, Jan. 2010.

[17] Y. Chen, J. Yu, X. Su, and G. Luo, "Path planning for multi-UAV formation," *Journal of Intelligent Robotic Systems*, vol. 77, no. 1, pp. 229–246, Jan. 2015.

[18] J. Chen, F. Ye, and T. Jiang, "Path planning under obstacle-avoidance constraints based on ant colony optimization algorithm," in *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, 2017, pp. 1434–1438.

[19] Z. He and L. Zhao, "The comparison of four UAV path planning algorithms based on geometry search algorithm," in *2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 2, 2017, pp. 33–36.

[20] L. D. Seneviratne, K. W-S, and S. W. E. Earles, "Triangulation-based path planning for a mobile robot," *Proceedings of the Institution of Mechanical Engineers*, vol. 211, no. 5, p. 365, 1997.

[21] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258 –1276, 2013.

[22] C. Prodhon and C. Prins, "Metaheuristics for vehicle routing problems," in *Metaheuristics*, P. Siarry, Ed. Cham: Springer International Publishing, 2016, pp. 407–437, ISBN: 978-3-319-45403-0.

[23] *Homework 3. Trapezoidal Cell Decomposition and Coverage.*

[24] A. Becker and A. Paul, *A framework for vehicle routing approximation schemes in trees*, 2019.

[25] A. Sathyan, N. Boone, and K. Cohen, "Comparison of approximate approaches to solving the travelling salesman problem and its application to UAV swarming," *International Journal of Unmanned Systems Engineering.*, vol. 3, no. 1, pp. 1–16, 2015.

[26] P. Toth and D. Vigo, "Models, relaxations and exact approaches for the capacitated vehicle routing problem," *Discrete Applied Mathematics*, vol. 123, no. 1, pp. 487 –512, 2002.

[27] G. Laporte, "A concise guide to the traveling salesman problem," *The Journal of the Operational Research Society*, vol. 61, no. 1, pp. 35–40, Jan. 2010.

[28] J. F. Puget, "No, The TSP Isn't NP Complete," 2013.

[29] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science (pre-1986)*, vol. 6, no. 1, p. 80, Oct. 1959.

[30] "Hamiltonian Cycle: Simple Definition and Example," 2017.

[31] R. Matai, S. Singh, and M. L. Mittal, "Traveling salesman problem: An overview of applications, formulations, and solution approaches," in *Traveling Salesman Problem*, D. Davendra, Ed., Rijeka: IntechOpen, 2010, ch. 1.

[32] F. Nuriyeva and G. Kizilates, "A new heuristic algorithm for multiple traveling salesman problem," *TWMS Journal of Applied and Engineering Mathematics*, vol. 7, no. 1, pp. 101–109, 2017.

[33] F. Lam and A. Newman, "Traveling salesman path problems," *Mathematical Programming*, vol. 113, no. 1, pp. 39–59, May 2008.

[34] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *Journal of the Operations Research Society of America*, vol. 2, no. 4, pp. 393–410, 1954.

[35] "The Traveling Salesman Problem with integer programming and Gurobi," 2018.

[36] C. A. Feinstein, *The computational complexity of the traveling salesman problem*, 2012.

[37] C. H. Papadimitriou and K. Steiglitz, "On the complexity of local search for the traveling salesman problem," *SIAM Journal on Computing*, vol. 6, no. 1, pp. 76–8, Mar. 1977.

[38] G. Laporte, "Fifty years of vehicle routing," *Transportation Science*, vol. 43, no. 4, pp. 408–416, Nov. 2009.

[39] Y. Z. Mehrjerdi, "Multiple-criteria decision-making combined with vrp: A categorized bibliographic study," *International Journal of Supply and Operations Management*, vol. 2, no. 2, pp. 798–820, 2015.

[40] F. Alonso, M. J. Alvarez, and J. E. Beasley, "A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions," *The Journal of the Operational Research Society*, vol. 59, no. 7, pp. 963–976, Jul. 2008.

[41] R. Lahyani, L. C. Coelho, and J. Renaud, "Alternative formulations and improved bounds for the multi-depot fleet size and mix vehicle routing problem," *OR Spectrum*, vol. 40, no. 1, pp. 125–157, Jan. 2018.

[42] Z. Zhang, Y. S. Yao, and J. H. Zhang, "Algorithm evolution from traveling salesman problem to vehicle routing problem," *Applied Mechanics and Materials*, vol. 411-414, p. 1872, Sep. 2013.

[43] S. Almoustafa, "Distance-constrained vehicle routing problem: Exact and approximate solution (mathematical programming)," PhD thesis, 2013, p. 1.

[44] C.-L. Li, D. Simchi-Levi, and M. Desrochers, "On the distance constrained vehicle routing problem," *Operations research*, vol. 40, no. 4, p. 790, 1992.

[45] K. Karagul, "A novel constructive routing algorithm for fleet size and mix vehicle routing problem," *International Journal of Combinatorial Optimization Problems and Informatics*, vol. 5, no. 2, pp. 58–73, 2014.

[46] I. Kara, "Arc based integer programming formulations for the distance constrained vehicle routing problem," in *3rd IEEE International Symposium on Logistics and Industrial Informatics*, 2011, pp. 33–38.

[47] P. Sharma, "Any R packages to solve Vehicle Routing Problem?," 2010.

[48] C. Y. Ren, "Study on improved tabu search algorithm for min-max vehicle routing problem," *Applied Mechanics and Materials*, vol. 87, p. 178, Aug. 2011.

[49] X. Wang, B. Golden, and E. Wasil, "The min-max multi-depot vehicle routing problem: Heuristics and computational results," *The Journal of the Operational Research Society*, vol. 66, no. 9, pp. 1430–1441, Sep. 2015.

[50] C. Y. Ren, "Applying genetic algorithm for min-max vehicle routing problem," *Applied Mechanics and Materials*, vol. 97-98, p. 640, Sep. 2011.

[51] M. Mirabi, N. Shokri, and A. Sadeghieh, "Modeling and solving the multi-depot vehicle routing problem with time window by considering the flexible end depot in each route," *International Journal of Supply and Operations Management*, vol. 3, no. 3, pp. 1373–1390, 2016.

[52] P. Stodola, "Using metaheuristics on the multi-depot vehicle routing problem with modified optimization criterion," *Algorithms*, vol. 11, no. 5, p. 74, 2018.

[53] A. G. Kek, R. L. Cheu, and Q. Meng, "Distance-constrained capacitated vehicle routing problems with flexible assignment of start and end depots," *Mathematical and Computer Modelling*, vol. 47, no. 1, pp. 140 –152, 2008.

[54] D. Applegate, W. Cook, S. Dash, and A. Rohe, "Solution of a min-max vehicle routing problem," *INFORMS Journal on Computing*, vol. 14, no. 2, p. 132, 2002.

[55] M. Yao and M. Zhao, "Unmanned aerial vehicle dynamic path planning in an uncertain environment," *Robotica*, vol. 33, no. 3, pp. 611–621, Mar. 2015.

[56] H. Cicibas, K. A. Demir, and N. Arica, "Comparison of 3d versus 4d path planning for unmanned aerial vehicles," *Defence Science Journal*, vol. 66, no. 6, pp. 651–664, 2016.

[57] H.-m. Zhang and M.-l. Li, "Rapid path planning algorithm for mobile robot in dynamic environment," *Advances in Mechanical Engineering*, vol. 9, no. 12, Dec. 2017.

[58] C. Sakouhi, A. Khaldi, and H. B. Ghezal, *An overview of recent graph partitioning algorithms*, 2018.

[59] C. Xu and G. Hong-mei, *An approximation algorithm for graph k-partitioning*, 2012.

[60] A. S. Muttipati and P. Padmaja, "Analysis of large graph partitioning and frequent subgraph mining on graph data," *International Journal of Advanced Research in Computer Science*, vol. 6, no. 7, Sep. 2015.

[61] S. Barlow, Y. Choi, S. Briceno, and D. N. Mavris, "A multi-UAV trajectory optimization methodology for complex enclosed environments," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019.