UNMANNED AERIAL VEHICLES: TRAJECTORY PLANNING AND ROUTING IN THE ERA OF ADVANCED AIR MOBILITY

A Dissertation
Presented to
The Academic Faculty

By

Fanruiqi Zeng

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
Daniel Guggenheim School of Aerospace Engineering

Georgia Institute of Technology

August 2022

UNMANNED AERIAL VEHICLES: TRAJECTORY PLANNING AND ROUTING IN THE ERA OF ADVANCED AIR MOBILITY

Thesis committee:

Dr. John-Paul Clarke
Daniel Guggenheim School of Aerospace Engineering & H. Milton Stewart School of Industrial and System Engineering
Georgia Institute of Technology

Dr. David Goldsman
H. Milton Stewart School of Industrial and System Engineering
Georgia Institute of Technology

Dr. Brian J German
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Dr. Graeme James Kennedy
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Dr. Husni R. Idris
Aviation Systems Division
NASA Ames Research Center

Date approved: July 20, 2022

“To secure ourselves against defeat lies in our own hands, but the opportunity of defeating the enemy is provided by the enemy himself.”

-Sun Tzu, The Art of War

This thesis is dedicated to my parents and little brother for their everlasting love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Advanced air mobility (AAM) is a revolutionary concept that enables on-demand air mobility, cargo delivery, and emergency services via an integrated and connected multi-modal transportation network. In the era of AAM, unmanned aerial vehicles are envisioned as the primary tool for transporting people and cargo from point A to point B. This thesis focuses on the development of a core decision-making engine for strategic vehicle routing and trajectory planning of autonomous vehicles (AVs) with the goal of enhancing the system-wide safety, efficiency, and scalability.

Part I of the thesis addresses the routing and coordination of a drone-truck pairing, where the drone travels to multiple locations to perform specified observation tasks and rendezvous periodically with the truck to swap its batteries. Drones, as an alternative mode of transportation, have advantages in terms of lower costs, better service, or the potential to provide new services that were previously not possible. Typically, those services involve routing a fleet of drones to meet specific demands. Despite the potential benefits, the drone has a natural limitation on the flight range due to its battery capacity. As a result, enabling the combination of a drone with a ground vehicle, which can serve as a mobile charging platform for the drone, is an important opportunity for practical impact and research challenges. We first propose a Mixed Integer Quadratically-constrained Programming driven by critical operational constraints. Given the NP-hard nature of the so called Nested-VRP, we analyze the complexity of the MIQCP model and propose both enhanced exact approach and efficient heuristic for solving the Nested-VRP model. We envision that this framework will facilitate the planning and operations of combined drone-truck missions and further improve the scalability and efficiency of the AAM system.

Part II of the thesis focuses on the survivability reasoning and trajectory planning of UAVs under uncertainty. Maintaining the survivability of an UAV requires that it precisely perceives and transitions between safe states in the airspace. We first propose a methodol-

ogy to construct a survivability map for an UAV as a function of the vehicle's maneuverability, remaining lifetime, availability of landing sites, and the volume of air traffic. The issue of trajectory planning under uncertainty has received a lot of attention in the robotics and control communities. Traditional trajectory planning approaches rely primarily on the premise that the uncertainty of dynamic obstacles is either bounded or can be statistically modeled. This is not the case in the urban environment, where the sources of uncertainty are diverse, and their uncertain behavior is typically unpredictable, making precise modeling impossible. Motivated by this, we present a receding horizon control method with innovative trajectory planning policies that enable dynamic updating of planned trajectories in the presence of partially known and unknown uncertainty. The findings of this study have significant implications for achieving safe aviation autonomy within the AAM system.

CHAPTER 1
INTRODUCTION

1.1 Unmanned Aerial Vehicles

An unmanned aerial vehicle (UAV) is an air vehicle that, after it has been turned on, requires no human input to carry out its mission. Within its programmed constraints, a UAV may monitor and assess its status as well as control assets on-board the vehicle [1]. This new technology has the potential to revolutionize both civilian and military aviation. In fact, commercial applications of UAVs have increased dramatically over the last 10 years. Taking into account trends in registrations, the evolution of the regulatory environment, and the underlying demand for unmanned aircraft systems, the FAA predicts that the commercial UAV fleet will be around 835,000 units by 2025, which is 1.7 times the current number of commercial UAVs fleet in 2021, according to [2].

UAVs are often classified by their weight, size, endurance, maximum altitude, and degree of autonomy. Commercial UAVs typically fall into one of four categories: (i) fixed-wing aircraft, which have a long flight endurance and a high cruising speed, but require a runway to take off and land; (ii) rotary-wing aircraft, which have the capability to hover and are associated with a high degree of maneuverability; and (iii) a blimp, or an airship that is lighter than air, and thus usually flies at low speeds and has a long endurance; (v) multi-rotor, also known as a drone, which has three or more propellers and can hover or fly in any direction. In this thesis, we will put emphasis on the most common vehicle type—the drone.

1.2 Coordinated Vehicle Routing

There has been a growing body of academic research on optimization for a variety of UAV routing problems in sectors such as aerial reconnaissance, traffic monitoring, meteorological sampling and disaster assessment. The *vehicle routing problem* (VRP) refers to a problem of determining the optimal routes of delivery or collection from one or several depots to a number of cities or customers while satisfying a number of operational constraints. UAVs, as an alternative mode of transportation, have the advantages in terms of lower cost, better service, or the ability to offer new service that are not previously possible. In particular, the employment of UAVs minimizes the cost of recruiting human operators, lowering human resource costs while also improving workplace safety.

The UAV-related VRP problem involves unique challenges coming from one or a combination of the following four aspects: (i) a wide spectrum of drone capabilities for both existing UAVs and those likely to be produced in the future to consider; (ii) demanding constraints on UAV performance and operations; (iii) diverse objectives for different UAV services; and (v) required methodological advances in supporting novel UAV applications. Among all of that, extensive research is needed to address the intrinsic limitation of applying a UAV in most operational scenarios: the limited flight range due to the UAV's battery capacity. Therefore, an important opportunity for practical impact and research challenges is the combination of a drone with a ground vehicle that can serve as a mobile charging platform for the drone which gives rise to a new class of VRPs - Coordinated Vehicle Routing Problem.

Coordinated routing of trucks and drones typically involves sophisticated synchronizations of the operations of multiple vehicles, which requires careful mathematical modeling of spatial and temporal constraints. It is not uncommon that the performance of a model may degrade as the size of the problem increases. Therefore, the development of algorithms and solution techniques that are well-equipped to optimize coordinated VRPs requires care-

ful investigation into the nature of the problem structure which differs significantly with a slight change in the capabilities of drones, operational constraints, and designated objective functions.

In part I of this thesis, we focus on the application of a special type of UAV, known as drone, and address the routing and coordination of a drone-truck pairing where the drone travels to multiple locations to perform observation tasks, named Nested Vehicle Routing Problem (Nested-VRP). To this end:

- In chapter 2, we propose a novel Mixed Integer Quadratically-constrained Programming formulation that incorporates realistic operational constraints. The compactness of the proposed model is compared to compactness of the state-of-the-art model.
- In chapter 3, we develop an effective and efficient neighborhood search heuristic to solve the Nested-VRP. In particular, we first assess the intrinsic complexity of the Nested-VRP model conditional on the prior knowledge on the drone routing. To improve model accuracy, we investigate techniques that enhance the model performance by strengthening constraints. In the end, we conduct extensive computational experiments and provide valuable insights for practitioners.

1.3 Trajectory Planning in Uncertainty

A fundamental need of an UAV is to safely move from one location to a desired location to fulfil a designed mission (i.e., deliver packages and provide rescue and surveillance service). One of the important capabilities of an UAV is to convert high-level mission specifications into a series of motions or actions while avoiding potential conflicts with other active participants in the environment. The term *trajectory planning* usually refers to the problem of determining both a path and corresponding velocity profile along the path from a given initial state to a destination state, while avoiding collisions with obstacles whose dynamics could be known, partially known or even unknown.

The inherent complexity in solving the trajectory planning problem stems from the high dimensionality of the search space when considering the dynamic model of the UAV, the geometric and kinematic characteristics of the obstacles, and the intended objective to be optimized. Furthermore, the UAV is anticipated to function reliably and safely in uncertain and dynamic environments, which is especially difficult in urban contexts where the uncertainties from weather, operational restrictions, human activities, and air traffic are continually increasing.

Thus, there is an urgent need for research into how UAVs can handle complex missions autonomously and safely in an uncertain environment. In this thesis, we develop a rigorous understanding of the qualitative and quantitative aspects of the survivability of an UAV. Specifically, we first investigate the factors that contribute to a UAV's chance of survival, and then subsequently develop novel trajectory planning algorithms that incorporate our understanding of the sources of uncertainty and enable informed decision-making based on the perceived risks due to these uncertainties. This thesis contributes to the literature by examining trajectory planning problems in environments with extreme uncertainty, where the factors that lead to a trajectory's infeasibility are unknown.

In part II, we focus on the planning of safe trajectories for helicopter-like autonomous vehicles in presence of extreme uncertainty. To this end:

- In chapter 4, we introduce the concepts of robustness and flexibility and explain their role in handling uncertainty within the context of a sequential decision-making framework. Using these key concepts, we establish the theoretic foundation for defining and measuring design and performance metrics for an autonomous vehicle in a flight mission. We further propose a back tracking algorithm and a Monte Carlo simulation to compute critical metrics.

- In chapter 5, we develop a methodology to construct a survivability map for the autonomous vehicle given its remaining lifetime and maneuverability, the topology of

airports, as well as traffic conditions. We investigate the discontinuity of the survivability map of an autonomous vehicle from space and time perspectives. The results have profound impacts on contingency planning for unmanned traffic.

- In chapter 6, we propose four novel trajectory planning policies that guide an autonomous vehicle safely through dynamically changing environment. Most importantly, we conduct extensive computational experiments and demonstrate that by maximizing the trajectory flexibility, the autonomous vehicle gains marginal protection against unknown-unknowns.

1.4 Summary

VRPs and trajectory planning problems are closely related. While the VRPs focus on optimizing the logical order or path to visit a set of locations (e.g., a city or a customer), the trajectory planning problem aims at finding a path with specific timing information along the path. While complexity of solving VRPs originates from the combinatorial nature of the decisions and therefore is exacerbated by the scalability of the problem, the trajectory planning problem tackles the common issues of having incomplete understanding of the environment which is compound by the extreme uncertain phenomenons in it. This thesis provides a unique lens for understanding, analyzing, and addressing crucial issues pertaining to UAVs mobility.

Part I

Coordinated Vehicle Routing

CHAPTER 2
NESTED VEHICLE ROUTING PROBLEM

2.1 Overview

Drones are becoming increasingly popular due to their low cost and high mobility. In this chapter we address the routing and coordination of a drone-truck pairing where the drone travels to multiple locations to perform specified observation tasks and rendezvous periodically with the truck to swap its batteries. In particular, we consider a nested vehicle routing problem (Nested-VRP) where: (a) a single drone is deployed to survey prescribed locations; (b) the locations to be surveyed are distributed across a large geographical area; (c) the duration of the surveillance at each location is unique to the type of survey to be conducted at that location; (d) the drone has limited flight endurance; (e) a single truck with an unlimited supply of fully charged batteries is used to recharge the drone; (f) the drone must rendezvous with the truck before running out of charge; (g) the time required to swap the batteries in the drone is a prescribed positive constant; and (h) perhaps most importantly the time required to complete the sequence of surveys must be minimized. We develops a Mixed Integer Quadratically Constrained Programming (MIQCP) formulation with critical operational constraints, including drone battery capacity and synchronization of both vehicles during scheduled rendezvous.

We seek to answer the following questions: (i) What is the optimal sequence of locations for the drone to visit? (ii) At which of these locations should the drone rendezvous with the truck? and (iii) What is the optimal routing for the truck? Further, because the information must be obtained frequently and in a timely fashion, we must do so via a computationally efficient heuristic algorithm that outperforms previous algorithms. This study contributes to the literature on cooperative vehicle routing in the following ways:

- Although it is not the main contribution of the thesis, to the best of our knowledge, we are the first to provide answers to these questions via a single formulation that incorporates the following real-world considerations: (a) non-zero surveillance times at locations; (b) flight endurance limitations; (c) the requirement that the truck must arrive at the rendezvous location before the drone battery charge has expired; (d) the truck is allowed to perform a battery swap while shipping the drone from one location to the other; and (e) a non-zero battery swapping time.

- A comparison of the proposed Nested-VRP model to the state-of-the-art model regarding model compactness is present. Moreover, we apply linearization and constraint strengthening techniques to further enhance the model performance. We also analyze the complexity of the Nested-VRP model with and without prior information on the drone routing.

2.1.1 Related Work

There continues to be growing interest in the coordinated use of drones and trucks to increase the efficiency of surveillance and transportation systems. The theoretical foundation for this work lies in the Traveling Salesman Problem (TSP) and its variant the Vehicle Routing Problem (VRP). Interested readers can consult surveys regarding solution methodologies for the TSP (see, e.g., [3, 4, 5]) and VRP (see, e.g., [6, 7, 8]). The Nested-VRP problem we address, where we seek to optimize the routing for *a single truck* and *a single drone*, can be viewed as an extension of the VRP.

Several algorithms have been developed for the cooredinated operation of a single-truck-single-drone system in a delivery context. Murray and Chu introduced the "Flying Sidekick Traveling Salesman Problem" (FSTSP), where (when appropriate) the delivery drone leaves the truck, completes a single delivery task and returns to the truck when it is at a subsequent customer location [9]. The authors formulated the problem as a Mixed Integer Linear Program (MILP) and proposed two heuristic methodologies whose effectiveness

were assessed and demonstrated via a series of computational experiments. The FSTSP heuristic starts by solving the TSP route[1] for all customers. Then, for each drone-eligible customer, the heuristic will decide whether to assign it to the drone tour or reinsert it into the truck tour at a different position in the TSP route. A similar problem, the Traveling Salesman Problem with Drone (TSP-D) was proposed by Agatz et al. [10].

In work [11], the authors subsequently extended their single-truck-single-drone framework to allow the truck to cooperate with a team of drones. Substantial time savings are achieved at the expense of more-complex coordination between vehicles. Numerous other extensions have been proposed since then: (a) improving the formulation of FSTSP such as [12, 13, 14, 15]; (b) extending the concept to m-truck-m-drone scenarios such as [16] and [17]; and (c) proposing new exact and heuristic methods such as [18, 19, 20]. However, the aforementioned work addresses the limited battery capacity issue by restricting the drone to visit only one intermediate location between leaving and returning to the truck. While this assumption is reasonable in delivery problems, it is quite restrictive in the case of drones performing surveillance tasks.

Research work [21] introduced the "Mothership and Drone Routing Problem (MDRP)" which considers the routing of a mothership and a drone to visit several designated locations. In the infinite-capacity drone routing problem (MDRP-IC) setting, in contrast to the models mentioned in the last paragraph, the drone is allowed to visit multiple targets consecutively before returning to the mothership for refueling. They devised an exact branch-and-bound solution approach and proposed two greedy heuristic approaches that were demonstrated to be competitive in achieving near-optimal solutions. But the model is fundamentally different from the Nested-VRP problem. While the mothership can move freely in 2D continuous space, the route of the truck in our problem is restricted to the road network which is idealized as straight lines between all pairs of locations.

To date, the work most-similar in approach to ours is that of [22]. They address the

[1]The shortest tour for a person to visit a set of locations

truck-drone team logistic (TDTL) problem via an MIP that generates the routes that the drone must follow to visit all the prescribed locations, and assigns rendezvous locations where the drone's batteries are replaced from the truck. Their overall goal is to minimize mission makespan. To deal with the inherent computational complexity, they propose a two-step heuristic approach and demonstrate its performance by comparing it to the exact solution obtained using the Gurobi Optimizer. TDTL departs from the general last mile delivery problem where the drone serves one location per operation. Instead, TDTL allows the drone to serve multiple locations per excursion from the truck. Each excursion involves a set of drone actions including launching from the truck, visiting multiple locations, and returning to the truck. Even though the characteristics of their problem are similar to our problem, their model cannot be adapted to the planning of a surveillance mission. In a typical surveillance mission, the drone battery is being used both when the drone travels between locations and when it executes its observation tasks. In the extreme, one can imagine that if all the observation tasks require a full battery, then the truck would be required to visit every location to refuel the drone — which reduces the surveillance problem to a TSP for the truck.

Separately, efforts have also been made to address the battery limit issue. For example, Dorling et al. derived an energy consumption model that can further be integrated into an MILP seeking to optimize routes of a fleet of drones to complete delivery tasks [23]. Cheng et al. studied a multi-trip drone routing problem, where payload and traveling distance are accounted for in determining the drone's energy consumption [24]. However, they do not consider drones working in collaboration with trucks.

All the work described above motivates the development of a Nested-VRP that takes into account the observation time at each location. Moreover, the Nested-VRP should also penalize the number of recharge stops by incurring a battery swap service time for each swap operation. And, for the sake of improving vehicle safety, the Nested-VRP should also include a restriction that the truck arrives at the rendezvous location before the drone

battery is depleted.

2.2 Model Assumptions

In the Nested-VRP, a set of geographically scattered locations is given. Each of these locations has an associated observation task with a prescribed duration. These observation tasks are completed by *a single drone* with a limited battery life supported by *a single truck* with an unlimited supply of fully charged batteries. Due to limitations on the drone's battery capacity, the drone and truck must periodically meet so that an almost-discharged battery can be swapped with a completely-charged battery supplied by the truck. Specifically, the Nested-VRP considers the routing of the drone including traveling to and making an observation at each location. We make the following assumptions regarding the physical properties of the drone and the truck, as well as the principles guiding the collaboration of the two vehicles:

A.1 The drone is equipped with a replaceable battery that is fully charged before the start of the mission. The drone's flight endurance is limited in time due to the battery's limited capacity.

A.2 Moreover, the drone's battery consumption is proportional to the active flight duration (i.e., the travel time between two locations or the observation time at the second location). The primary risk of using an oversimplified drone energy consumption model, according to [11], is that the drone will be unable to reach the designated locations. Such issue raises serious concerns about the drone's safety in the event of an unplanned landing and potentially reduces the mission's efficiency.

A.3 We further assume that the road segments between pairs of locations are straight lines. The drone and the truck move according to Euclidean distance between two locations with constant speeds. Therefore, the travel times of the drone and the truck both satisfy the triangle inequality.

A.4 The drone, by default, moves no slower than the truck.

A.5 The truck has a sufficient battery supply and/or the capability to recharge batteries en route. Thus, there are no constraints on the number of completely-charged batteries that are available when the truck rendezvous with the drone for battery swaps.

A.6 The battery swap process can only occur during segments where the truck is carrying the drone or at survey locations either before or after the drone surveys the location. Suspending this assumption would significantly increase the flexibility for two vehicles to choose rendezvous locations, potentially saving time by reducing the coordination effort required by two vehicles, as shown in Figure 2.1 (a)–(b).

A.7 Each swap operation, including collecting the drone, swapping the batteries, and repositioning the drone for take-off, delays the mission by a predetermined amount of service time.

A.8 Each location can be visited by the drone once and exactly once. This assumption eliminates solutions that contain cyclic operations (i.e., the vehicle starts and ends at the same location). As shown in Figure 2.1, (c) represents a Nested-VRP solution; in contrast, scenario (d) saves mission time by performing cyclic operations around location 2. In this case, location 2 delays the mission by an observation task at location 2 and a sequence of tasks associated with the cycle. To the best of the authors' knowledge, some most-promising research in studying the cyclic operation only consider a unit cycle of length two (see, e.g., [25], [26], [19], [27]) in a delivery context. In our case of a time-sensitive surveillance problem, it is an extremely challenging task to use a Mixed Integer Programming (MIP) model to describe the combinatorial choice of locations on a single cycle and describe the precedence relationship of two locations on two distinct cycles that are connected to the same hub. Further research on considering the cyclic operations in the Nested-VRP model is beyond the scope of this thesis.

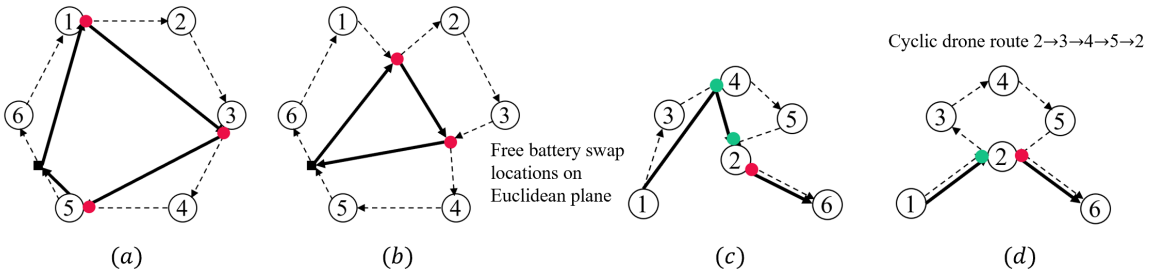


Figure 2.1: Potential improvements can be achieved by suspending some of the assumptions. (a)–(b) Allowing the battery swap locations to be anywhere on the 2D plane could potentially reduce the number of times the truck and drone rendezvous. (c)–(d) Allowing cyclic operation of the drone could potentially save mission makespan by taking advantage of geometry characteristics of hub-like locations.

The objective is to minimize the total mission time needed to complete all observation tasks including time spent traveling and conducting swapping services. Note that the total time the drone spends making observations is part of the mission but cannot be minimized because it is the sum of constant values.

To aid in our exposition of the problem and in the subsequent derivation of the mathematical formulation, we introduce the concept of a *nested unit* as shown in Figure 5.1. In a nested unit, the truck travels from location i to location j. Meanwhile, the drone departs from i, travels to and observes locations $\{k_1, k_2\}$, and finally meets up with the truck at location j. In summary, a nested unit consists of four components that are further explained as follows.

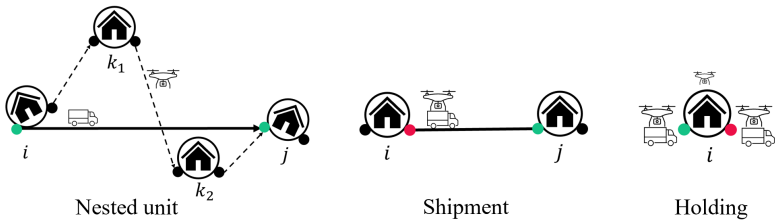


Figure 2.2: Illustration of a nested unit and its special formations.

- **Truck bridge** refers to the arc from i to j. It connects two consecutive locations where a battery swap occurs.

- **Drone path** refers to the collection of arcs that deviate from the truck bridge, and guides the drone to observe a subset of locations assigned to the nested unit. The fact that the drone speed is usually greater than the truck speed makes it possible for the drone to observe multiple locations while the truck travels from i to j.

- Node i is a **split location** from which the drone and the truck initiate their respective next tasks (i.e., taking observations, and delivery of a new battery to the next stop). Generally, the drone gets a fully charged battery and will take off right after the swapping process. Since the truck provides the swap service, it departs no earlier than the drone.

- Node j is a **rendezvous location** at which the drone path and truck bridge merge together. A battery swap happens right after the two vehicles meet.

- In special cases, the nested unit will be transformed and reduced to a **shipment** or **holding** pattern.

    1. A **shipment** pattern occurs when the truck and the drone both decide to traverse a relatively long arc without any observation task involved during the move. When a shipment happens, the truck ships the drone and executes a battery swap in transit. Depends on the required amount of time for completing a battery swap service, the shipment unit delays the mission by the maximum of the truck travel time and the battery swap service time.

    2. A **holding** pattern can occur at a location due to a relatively long observation period. In this case, the truck is held at the location and provides batteries to the drone both before and after the drone observes that location. In this case, the truck bridge degrades to a trivial point. Since the holding pattern only considers one location, it differs fundamentally from a cyclic operation, which takes into account at least two locations, as discussed in Assumption **A.8**.

The Nested-VRP solution can be viewed as a collection of nested units without overlapping tasks. If the drone has unlimited battery capacity (i.e., infinite endurance time), the optimal solution is that the drone gets a battery at a depot once, follows a TSP route to visit and observe all locations, and returns to the depot without additional swaps along the tour. Referring to Figure 5.1, the optimal solution is a single large nested unit with a split location (the originating depot), a rendezvous location (the depot), and a single drone path (the TSP route). In this case, no truck bridge would be involved. However, if the drone has limited battery capacity, the drone can only operate over relatively short time intervals without a battery swap. It follows then that the single large nested unit must be decomposed into a sequence of smaller nested units in which excessively long arcs and prolonged observation tasks will be accommodated into shipment and holding patterns, respectively. Most importantly, in each nested unit, the drone should be able to complete the specified travel and observation tasks using only its battery capacity. Therefore, the total mission is reduced to having the drone complete the relatively few tasks associated with each nested unit with battery swaps undertaken at the split and rendezvous locations.

We further require that the truck should always arrive before the drone's battery is depleted. This restriction is referred to as a **synchronization constraint**. However, there is no preference regarding the order in which the two vehicles arrive at a rendezvous location as long as the truck arrives before the drone battery charge has expired. Note that the drone may occasionally have to hover at the rendezvous location if it arrives before the truck.

To keep track of mission makespan, we define the interval between rendezvous (IBR) for each nested unit as the greater of the two vehicles' travel durations from the split location at the beginning of the nested unit to the rendezvous location at the end of the nested unit. Therefore, minimizing the mission makespan is equivalent to minimizing the summation of all the IBRs associated with nested units together with the total service times needed for battery swaps. Note that we regard a shipment as a special form of a nested unit whose IBR is the maximum of the truck travel time and the battery swap service time.

In the case where a nested unit reduces to a holding pattern, the truck remains stationary while the drone is observing a location. Therefore, the mission makespan increases by the amount of the drone's observation time at that location. The time increment is the IBR of the holding pattern.

2.3 Model Formulation

Given the comprehensive list of notations in Table 2.1, consider an undirected graph $\mathcal{G} = (\mathcal{H}, \mathcal{A})$, where $\mathcal{H} = \{0, 1, 2, \ldots, n+1\}$ is the set of locations, and $\mathcal{A} = \{(i, j) \mid i \in \mathcal{H} \setminus \{n+1\},\ j \in \mathcal{H} \setminus \{0\},\ i \neq j\}$ is the set of arcs. Location 0 is the origin, and location $n+1$ is the eventual destination, which we force to be the origin (so that the trip starts and stops at the same place). For each location $i \in \mathcal{H} \setminus \{0, n+1\}$, let o_i be its non-negative observation time, which is assumed to be smaller than T_{bl}. If o_i were to be greater than T_{bl}, then location i must be set as a swap stop, where at least one battery swap is required (depending on the observation time); and the "final" observation time (i.e., after the last swap) is the remainder of the quotient o_i/T_{bl}, that is, $o_i/T_{\text{bl}} - \lfloor o_i/T_{\text{bl}} \rfloor$, where $\lfloor \cdot \rfloor$ denotes the "floor" function.

Table 2.1: Notations used in the Nested-VRP MIP formulation.

Parameters	
T_{bl}	Battery capacity of drone.
T_{s}	Service time needed to swap a battery.
$\mathcal{H}$	Set of locations in graph $\mathcal{G} = (\mathcal{H}, \mathcal{A})$.
$\mathcal{A}$	Set of directed arcs in graph $\mathcal{G} = (\mathcal{H}, \mathcal{A})$.
$\mathcal{S}$	Set of arcs on the TSP route of a given graph $\mathcal{G}$.
o_i,$i \in \mathcal{H} \setminus \{0, n+1\}$	Observation time associated with location i.
τ_{ij}^D, $(i,j) \in \mathcal{A}$	Drone's flight time from i to j.
τ_{ij}^T, $(i,j) \in \mathcal{A}$	Truck's travel time from i to j.
Decision Variables	
$x_{ij} \in \{0,1\}$, $(i,j) \in \mathcal{A}$	If $x_{ij} = 1$, the drone flies from i to j, 0 otherwise.
$y_{ij} \in \{0,1\}$, $(i,j) \in \mathcal{A}$	If $y_{ij} = 1$, the truck travels from i to j, 0 otherwise.
$z_i^- \in \{0,1\}$, $i \in \mathcal{H} \setminus \{0\}$	If $z_i^- = 1$, the drone swaps batteries at location i immediately *before* it observes location i, 0 otherwise.
$z_i^+ \in \{0,1\}$, $i \in \mathcal{H} \setminus \{n+1\}$	If $z_i^+ = 1$, the drone swaps batteries at location i immediately *after* it observes location i, 0 otherwise.
$z_i \in \{0,1\}$, $i \in \mathcal{H}$	If $z_i = 1$, location i is selected as a battery swap stop, 0 otherwise.
Auxiliary Variables	
$t_i^- \in [0, T_{\text{bl}}]$, $i \in \mathcal{H} \setminus \{0\}$	Total travel and observation time from when drone departs the previous rendezvous location until it *arrives* at location i.
$t_i^+ \in [0, T_{\text{bl}}]$, $i \in \mathcal{H} \setminus \{n+1\}$	Total travel and observation time from when drone departs the previous rendezvous location until it *leaves* location i.
$u_i \in [0, n+1]$, $i \in \mathcal{H}$	Order index of location i in the solution of the drone route.
$w_{ij} \in \{0,1\}$, $(i,j) \in \mathcal{A}$	If $w_{ij} = 1$, the truck ships the drone from location i to j, 0 otherwise.
$l_i^- \in \mathcal{R}_+$, $i \in \mathcal{H} \setminus \{0\}$	IBR of a nested unit that terminates *before* the drone observes location i.
$l_i^+ \in \mathcal{R}_+$, $i \in \mathcal{H} \setminus \{n+1\}$	IBR of a nested unit that terminates *after* the drone observes location i.

For each arc $(i,j) \in \mathcal{A}$, the time metrics τ_{ij}^T (τ_{ij}^D) represent the truck (drone) travel times between pairs of locations. We assume that the travel times satisfy the triangle inequality. We further assume that the road segments between pairs of locations are straight lines. In

addition, the ground speed of the truck and the cruising speed of the drone are assumed to be constants.

A mission consists of planning the drone route x_{ij}, $(i,j) \in \mathcal{A}$, to visit all locations and designing the truck route y_{ij}, $(i,j) \in \mathcal{A}$, to delivery fully charged batteries. Specifically, the drone route and the truck route should intersect at a subset of locations z_i, $i \in \mathcal{H}$ where the truck performs battery swaps for the drone. The primary goal is to minimize the total mission time while operational constraints are satisfied.

Next, we explain different sets of operational constraints. Constraints from the same set serve a particular function. At the end of this section, we put together all of the constraints and present the full Nested-VRP formulation.

Drone Route Construction

The drone departs from location 0 and eventually returns to location $n+1$. To ensure each location is observed exactly once, we require that every location in $\mathcal{H} \setminus \{0, n+1\}$ has one incoming arc and one outgoing arc. Therefore, we have the following set of constraints:

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij} = 1, \quad \forall i \in \mathcal{H} \setminus \{n+1\} \tag{2.1}$$

$$\sum_{i:(i,j)\in\mathcal{A}} x_{ij} = 1, \quad \forall j \in \mathcal{H} \setminus \{0\} \tag{2.2}$$

The above constraints are not sufficient to construct the tour because they are also satisfied by subtours in the graph. We further introduce auxiliary variables u_i, $i \in \mathcal{H}$, which indicate the order of locations along the drone route. Any potential subtours in the graph will be eliminated by enforcing the following subtour elimination constraints (SEC) [28], denoted by MTZ.

$$u_0 = 0 \tag{2.3}$$

$$1 \le u_i \le n+1, \quad \forall\, i \in \mathcal{H} \setminus \{0\} \tag{2.4}$$

$$u_i - u_j + 1 \le (n+1)(1 - x_{ij}), \quad \forall (i,j) \in \mathcal{A},\, i \neq 0 \tag{2.5}$$

In theory, the linear programming (LP) relaxation of the Nested-VRP model can be further tighten up by considering the classical formulation of the SEC constraints: $\sum_{(i,j)\in\mathcal{A},i\in\mathcal{S},j\notin\mathcal{S}} x_{ij} \geq 2$, where $\mathcal{S} \subseteq \mathcal{H}$ [4], denoted by DFJ. In practice, the implementation of DFJ constraints involves generating exponential number of constraints which is time-consuming. As the size of problem increases, employing the DFJ constraint may prohibit Optimizer (e.g., Gurobi) return solution within cutoff time as demonstrated by [29], [30]).

Truck Route Construction

Likewise, the truck departs from location 0 and returns to location $n+1$ at the end of the trip. In the special case where the drone can finish the entire mission without any battery swaps, the truck parks at the origin 0. Note that the truck route is constructed in such a way that the truck only serves locations that are selected as battery swap stops (i.e., $z_i = z_i^- \vee z_i^+ = 1$). The above requirements are captured by constraints (2.6)-(2.10). Most importantly, the truck visits the battery swap locations in the same order as that of the drone. The precedence relationship between battery swap locations is enforced by constraint (2.11) which also eliminates truck subtours.

$$z_i \leq z_i^- + z_i^+, \quad \forall i \in \mathcal{H} \setminus \{0, n+1\} \tag{2.6}$$

$$z_i \geq z_i^-, \quad \forall i \in \mathcal{H} \setminus \{0\} \tag{2.7}$$

$$z_i \geq z_i^+, \quad \forall i \in \mathcal{H} \setminus \{n+1\} \tag{2.8}$$

$$\sum_{j:(0,j)\in\mathcal{A}} y_{0,j} \leq 1 \tag{2.9}$$

$$\sum_{i:(i,j)\in\mathcal{A}} y_{ij} = z_j; \qquad \sum_{k:(j,k)\in\mathcal{A}} y_{jk} = z_j, \quad \forall j \in \mathcal{H} \setminus \{0, n+1\} \tag{2.10}$$

$$u_i - u_j + 1 \leq (n+1)(1 - y_{ij}), \quad \forall (i,j) \in \mathcal{A},\, i \neq 0 \tag{2.11}$$

In theory, the linear programming (LP) relaxation of the Nested-VRP model can be further tightened up by replacing the MTZ subtour elimination constraints by the classical

formulation of the SEC constraints: $\sum_{(i,j)\in\mathcal{A},i\in\mathcal{S},j\notin\mathcal{S}} x_{ij} \geq 2$, where $\mathcal{S} \subseteq \mathcal{H}$ [4], denoted by DFJ. In the Nested-VRP model, however, substituting the MTZ subtour elimination constraints (2.4)–(2.5) with DFJ constraints for the drone and constraints (2.11) with DFJ constraints for the truck is not an equivalent transformation. First, the MTZ subtour elimination constraints (2.4)–(2.5) together with constraints (2.11) serve not only to eliminate subtours in the drone route and the truck route respectively, but also to ensure that the truck travels in the same direction as the drone by visiting lower rank to higher rank locations encoded in variable u_i. Second, rather than visiting all locations, the truck only travels to and visits locations that are selected as battery swap locations. Only when the set of battery swap locations is determined can we explicitly formulate the DFJ constraints for the truck route. Given the above analysis, we pursue the goal of strengthening the MTZ subtour elimination constraints of the drone by examining lifting technique and Reformulation-Linearization technique (RLT) in section 3.4.1.

To ensure the synchronization constraint, the truck must spend no more than T_{bl} time in transit between two consecutive rendezvous locations. In the special case where the truck ships the drone (i.e., $w_{ij} = 1$), the synchronization constraint becomes redundant. This can be captured by the following constraint in which $M_1 = \max_{(i,j)\in\mathcal{A}} \tau^T_{ij}$.

$$\tau^T_{ij} y_{ij} \leq T_{\text{bl}} + M_1 w_{ij}, \quad \forall (i,j) \in \mathcal{A} \tag{2.12}$$

Time Flow Balance

To keep track of the drone's battery consumption, we create an artificial timer. The timer records the current battery consumption of the drone by accumulating the total travel and observation time since leaving the previous rendezvous location. At each location, we introduce auxiliary variables $t^-_j, t^+_j \in [0, T_{\text{bl}}]$, which we refer to as the state of timer at location j. In particular, t^-_j denotes the state of the timer when the drone arrives at location j and t^+_j denotes the state of the timer when the drone is about to leave location j. Once a timer is about to exceed the battery capacity T_{bl}, the timer is reset to 0, corresponding to a

battery swap. Constraints (2.13) - (2.14) state the maximum value of the timer.

Constraint (2.15) defines variable t_j^+ which is the total travel and observation time from when the drone departs the previous rendezvous location until it leaves location j. It depends on the state of the timer t_j^- as well as the battery swap decision z_j^- when the drone arrives at location j. (a) If $z_j^- = 0$, the drone continues the flight and observes location j. Thus, the timer keeps accumulating the drone flight time and increases by the amount of o_j, that is $t_j^+ = t_j^- + o_j$. (b) If $z_j^- = 1$, the drone requires a battery swap service before observing location j. This is equivalent to end the previous nested unit and restart a new nested unit for the latter mission. In this case, the timer starts from 0 and becomes o_j once the drone completes the observation task at location j, that is $t_j^+ = o_j$.

Constraints (2.16)–(2.22) are used to the relationship between variable t_i^+ and t_j^- due to the drone travel from location i to j. The battery consumption on any arc $(i, j) \in \mathcal{A}$ depends on both the drone route decision x_{ij} and the truck shipment decision w_{ij} associated with the arc (i, j). Constraints (2.16)–(2.20) ensure that the truck ships the drone (i.e., $w_{ij} = 1$) if and only if the two vehicles have decided to traverse the same arc (i, j) (i.e., $x_{ij} = y_{ij} = z_i^+ = z_j^- = 1$).

Typically, the drone travels from location i to location j alone, i.e., $x_{ij} = 1$ and $w_{ij} = 0$. Recall that the timer state is t_i^+ when the drone is about to leave location i and t_j^- when the drone arrives at location j. According to constraints (2.21)–(2.22), if arc (i, j) is activated as part of the drone route, the states of the timer at both sides of the arc (i, j) are regulated by the equation $t_j^- = t_i^+(1 - z_i^+) + \tau_{ij}^D$. Specifically, if the drone has sufficient battery to cover the arc (i, j), the drone requires no additional battery swaps (i.e., $z_i^+ = 0$) before leaving i. Therefore, the timer increases by the amount of traveling time τ_{ij}^D, and t_j^- becomes $t_i^+ + \tau_{ij}^D$ when the drone arrives at location j. However, if the drone requires a battery swap to be able to cover arc (i, j), a rendezvous location is added when the drone departs location i (i.e., $z_i^+ = 1$). In this case, since the timer is set to 0 at the beginning of the arc traveling, the timer accumulates the amount of the drone traveling time and becomes

τ_{ij}^D when the drone reaches the endpoint of arc (i, j).

Constraints (2.21)–(2.22) regulate the states of the timer at both sides of arc (i, j). When the shipment happens, the drone's timer is set to 0 at the time the drone arrives at location j which corresponds to recieving a new battery. Note that the truck does not perform battery swaps at the rendezvous locations placed at both endpoints of the arc (i, j). Therefore, these two rendezvous locations do not delay the mission (see objective function (6.1)).

Arc (i, j) has no impact on the state of the timer if it is not part of the drone route. This can be captured by constraints (2.21)–(2.22) in which $M_2 = T_{\text{bl}}$. In Figure 2.3, we illustrate how the state of the timer is updated as the mission continues.

$$t_j^- \leq T_{\text{bl}}, \quad \forall j \in \mathcal{H} \setminus \{0\} \tag{2.13}$$

$$t_j^+ \leq T_{\text{bl}}, \quad \forall j \in \mathcal{H} \setminus \{n+1\} \tag{2.14}$$

$$t_j^+ = t_j^-(1 - z_j^-) + o_j, \quad \forall j \in \mathcal{H} \setminus \{0, n+1\} \tag{2.15}$$

$$w_{ij} \leq x_{ij}, \quad \forall (i, j) \in \mathcal{A} \tag{2.16}$$

$$w_{ij} \leq y_{ij}, \quad \forall (i, j) \in \mathcal{A} \tag{2.17}$$

$$w_{ij} \leq z_i^+, \quad \forall (i, j) \in \mathcal{A} \tag{2.18}$$

$$w_{ij} \leq z_j^-, \quad \forall (i, j) \in \mathcal{A} \tag{2.19}$$

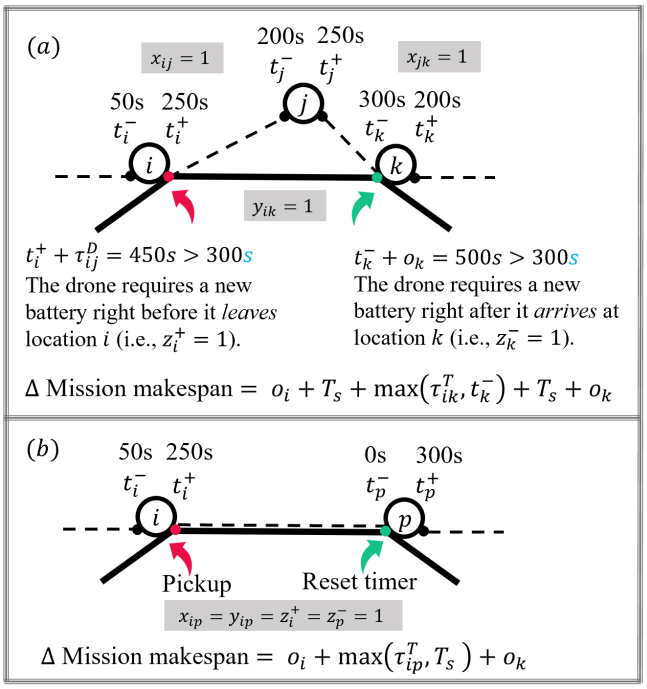
$$x_{ij} + y_{ij} + z_i^+ + z_j^- \leq 3 + w_{ij}, \quad \forall (i, j) \in \mathcal{A} \tag{2.20}$$

$$t_j^- \leq t_i^+(1 - z_i^+) + \tau_{ij}^D(1 - w_{ij}) + M_2(1 - x_{ij}), \quad \forall (i, j) \in \mathcal{A} \tag{2.21}$$

$$t_j^- \geq t_i^+(1 - z_i^+) + \tau_{ij}^D(1 - w_{ij}) - M_2(1 - x_{ij}), \quad \forall (i, j) \in \mathcal{A} \tag{2.22}$$

Interval Between Rendezvous

With the help of the timer from the previous section, we now derive the IBR for each nested unit. A nested unit can only terminate at rendezvous location j either before or after the drone observes the location j. Let l_j^- denote the IBR for the nested unit that terminates when the drone arrives at location j. Likewise, let l_j^+ denote the IBR for the nested unit



Parameters	Times
T_s	50s
T_{bl}	300s
o_i	200s
o_j	50s
o_k	200s
o_p	300s
τ_{ij}^D	200s
τ_{jk}^D	50s
τ_{ik}^T	100s
τ_{ip}^T	100s

Figure 2.3: Time flow balance example. (a) Typically, a battery swap can occur when the drone is about to leave a location (red dot) or when it just arrives at a location (green dot). (b) In the special case where the truck ships the drone, a rendezvous location, corresponding to a pickup, is placed before they start traveling (red dot). A rendezvous location is also placed at the end of travel (green dot). Notice that the timer is in state 0 when the two vehicles arrive at the designated location p. In addition, the stops at the two ends of arc ip serve special purposes and do not delay the mission.

that ends after the drone observes location j.

Typically, the IBR of a nested unit should not exceed T_{bl}. In the special case when the nested unit reduces to a "Shipment" pattern, it must terminate at the arrival of a location j (i.e., $z_j^- = 1, \sum_{i:(i,j)\in\mathcal{A}} w_{ij} = 1$). Thus, the IBR of such unit l_j^- is unrestricted. Notice that for a location that is not a battery swap stop, the IBR of such a location does not exist and, therefore, is trivial. Constraints (2.23)–(2.24) capture the above restrictions.

$$l_j^- \leq T_{\text{bl}} z_j^- + M_2 \sum_{i:(i,j)\in\mathcal{A}} w_{ij}, \quad \forall j \in \mathcal{H} \tag{2.23}$$

$$l_j^+ \leq T_{\text{bl}} z_j^+, \quad \forall j \in \mathcal{H} \tag{2.24}$$

As indicated by constraints (2.25)-(2.26), IBR l_j^- is determined by comparing the the

drone's surveillance time t_j^- and truck's travel time $\sum_{i:(i,j)\in\mathcal{A}} \Big(\tau_{ij}^T y_{ij} + (\max(\tau_{ij}^T, T_\mathrm{s}) - \tau_{ij}^T) w_{ij}\Big)$ since they both leave the previous rendezvous location. In the special case when the nested unit is a shipment unit connecting, for example, location k and j, the drone timer t_j^- at location j becomes 0 due to constraints (2.21)–(2.22). In this case, the drone's surveillance time becomes trivial. As a result, the IBR l_j^- should be no smaller than the amount of time the truck spends in shipping the drone from k to j while completing a battery swap service, that is $\max(\tau_{kj}^T, T_\mathrm{s})$.

As indicated by constraints (2.27)-(2.28), IBR l_j^+ is the maximum of the drone's surveillance time t_j^+ and the truck's travel time $\sum_{i:(i,j)\in\mathcal{A}} \tau_{ij}^T y_{ij}(1 - z_j^-)$ for the nested unit that ends after the drone observes location j. Notably, if the nested unit reduces to a holding pattern (i.e., $z_j^- = z_j^+ = 1$), then the truck's travel time becomes 0, and thus IBR l_j^+ should be no smaller than the drone's surveillance time t_j^+.

As a summary, the following constraints (2.25)–(2.28) are necessary to define IBRs for all possible swap locations.

$$l_j^- \geq t_j^- - M_2(1 - z_j^-), \quad \forall j \in \mathcal{H} \tag{2.25}$$

$$l_j^- \geq \sum_{i:(i,j)\in\mathcal{A}} \Big(\tau_{ij}^T y_{ij} + \big(\max(\tau_{ij}^T, T_\mathrm{s}) - \tau_{ij}^T\big) w_{ij}\Big) - M_1(1 - z_j^-), \quad \forall j \in \mathcal{H} \tag{2.26}$$

$$l_j^+ \geq t_j^+ - M_2(1 - z_j^+), \quad \forall j \in \mathcal{H} \tag{2.27}$$

$$l_j^+ \geq \sum_{i:(i,j)\in\mathcal{A}} \tau_{ij}^T y_{ij}(1 - z_j^-) - M_1(1 - z_j^+), \quad \forall j \in \mathcal{H} \tag{2.28}$$

Overall Formulation

Our objective is to minimize the mission makespan, which consists of the IBRs of all nested units in the solution and the total service time for battery swaps. In the case where the truck ships the drone from location i to j (i.e., $w_{ij} = 1$), the model enforces $z_i^+ = z_j^- = 1$ for adding meetup stops at the two ends of arc (i, j). Since no battery swaps happen at these two stops, the objective function adjusts for over counting the battery swaps service time

by adding $-\sum_{(i,j)\in\mathcal{A}} 2w_{ij}$. To initialize the mission, we assume that the drone is equipped with a new battery before leaving origin 0, so that $z_0^+ = 1$ and $t_0^+ = 0$. The complete Nested-VRP model is given as follows.

$$\text{(Nested-VRP)} \quad \min \quad \sum_{i\in\mathcal{H}}(l_i^- + l_i^+) + T_{\text{s}}\big(\sum_{i\in\mathcal{H}}(z_i^- + z_i^+) - \sum_{(i,j)\in\mathcal{A}} 2w_{ij}\big) \tag{2.29}$$

$$\text{s.t.} \quad \text{constraints (2.1)–(2.28)}$$

$$z_0^+ = 1,\ t_0^+ = 0 \tag{2.30}$$

$$x_{ij} \in \{0,1\},\ y_{ij} \in \{0,1\},\ w_{ij} \in \{0,1\}, \forall (i,j) \in \mathcal{A} \tag{2.31}$$

$$z_i^-, \in \{0,1\},\ t_i^-, \in \mathcal{R}_+,\ l_i^-, \in \mathcal{R}_+, \forall i \in \mathcal{H} \setminus \{n+1\} \tag{2.32}$$

$$z_i^+, \in \{0,1\},\ t_i^+, \in \mathcal{R}_+,\ l_i^+, \in \mathcal{R}_+, \forall i \in \mathcal{H} \setminus \{0\} \tag{2.33}$$

$$z_i, u_i \in [0, n+1], \forall i \in \mathcal{H} \tag{2.34}$$

2.4 Model Comparison

In this section, we compare the Nested-VRP model to what we regard as the state-of-the-art model in terms of their relaxed polyhedra. From a modeling perspective, it is of particular interest to compare the compactness of the proposed MIP model to other state-of-the-art models. To the best of our knowledge, the most-similar model concerning truck-drone coordinated routing is that of [22] — the Truck Drone Team Logistics (TDTL) model. Since the TDTL model does not consider the observation times associated with each location nor the service time in swapping the battery, we will first derive a special version of the Nested-VRP model where we neglect the battery swap service time and set the observation time to zero for every location. Our special version of the Nested-VRP is called Zero Observation Nested-VRP (ZONVRP). Since the two models to be compared apply different notations with different physical meanings, a linear transformation Φ that maps the ZONVRP variables to that of TDTL is needed. We will show that the LP relaxation of the Nested-VRP model is tighter than the LP relaxation of the TDTL with respect to a linear transformation.

This is noteworthy because the majority of commercial MIP Optimizers have a branch-and-bound component that leverages the associated LP to iteratively search for the optimal solution. Thus, a tighter formulation usually requires the evaluation of fewer branching nodes thereby reducing computation time. Further, even if the optimal solution can not be obtained within the time limit, the MIP Optimizer can provide a better bound on the optimal value of the problem at termination when using a tighter formulation.

Theorem 2.4.1. *Denote the feasible set of ZONVRP under a linear transformation* Φ *as* P_1*, and denote the feasible set of TDTL as* P_2*. Then* P_1 *is a proper subset of* P_2*.*

2.4.1 Proof of Theorem 2.4.1

Proof. First, we set forth the ZONVRP formulation. Since there are no observations, we do not distinguish battery swaps that happen before or after observing a location. Therefore, for each location, we define $z_j \in \{0,1\}, \forall j \in \mathcal{H}$. If $z_j = 1$, location j is a battery swap location, 0 otherwise. Likewise, each location only requires a variable $l_j \in \mathcal{R}_+, \forall j \in \mathcal{H}$ to record the IBR time. The ZONVRP model is presented as follows:

$$
\begin{aligned}
(\text{ZONVRP}) \quad \min \quad & \sum_{i\in\mathcal{H}} l_i + T_s \sum_{i\in H} z_i - T_s \sum_{(i,j)\in A} 2w_{ij} \\
\text{s.t.} \quad & \sum_{j:(i,j)\in\mathcal{A}} x_{ij} = 1, \quad \forall i \in \mathcal{H}\setminus\{n+1\} && (2.35)\\
& \sum_{i:(i,j)\in\mathcal{A}} x_{ij} = 1, \quad \forall j \in \mathcal{H}\setminus\{0\} && (2.36)\\
& u_0 = 0 && (2.37)\\
& 1 \le u_i \le n+1, \quad \forall\, i \in \mathcal{H}\setminus\{0\} && (2.38)\\
& u_i - u_j + 1 \le (n+1)(1-x_{ij}), \quad \forall (i,j)\in\mathcal{A},\, i\neq 0 && (2.39)\\
& \sum_{j:(0,j)\in\mathcal{A}} y_{0,j} \le 1 && (2.40)\\
& \sum_{i:(i,j)\in\mathcal{A}} y_{ij} = z_j; \qquad \sum_{k:(j,k)\in\mathcal{A}} y_{jk} = z_j, \quad \forall j \in \mathcal{H}\setminus\{0,n+1\} && (2.41)
\end{aligned}
$$

$$u_i - u_j + 1 \leq (n+1)(1 - y_{ij}), \quad \forall (i,j) \in \mathcal{A},\ i \neq 0 \tag{2.42}$$

$$\tau_{ij}^T y_{ij} \leq T_{\text{bl}} + M_1 w_{ij}, \quad \forall (i,j) \in \mathcal{A} \tag{2.43}$$

$$t_j^- \leq T_{\text{bl}}, \quad \forall j \in \mathcal{H} \setminus \{0\} \tag{2.44}$$

$$t_j^+ \leq T_{\text{bl}}, \quad \forall j \in \mathcal{H} \setminus \{n+1\} \tag{2.45}$$

$$t_j^+ = t_j^-(1 - z_j), \quad \forall j \in \mathcal{H} \setminus \{0, n+1\} \tag{2.46}$$

$$w_{ij} \leq x_{ij}, \quad \forall (i,j) \in \mathcal{A} \tag{2.47}$$

$$w_{ij} \leq y_{ij}, \quad \forall (i,j) \in \mathcal{A} \tag{2.48}$$

$$w_{ij} \leq z_i, \quad \forall (i,j) \in \mathcal{A} \tag{2.49}$$

$$w_{ij} \leq z_j, \quad \forall (i,j) \in \mathcal{A} \tag{2.50}$$

$$x_{ij} + y_{ij} + z_i + z_j \leq 3 + w_{ij}, \quad \forall (i,j) \in \mathcal{A} \tag{2.51}$$

$$t_j^- \leq t_i^+(1 - z_i) + \tau_{ij}^D(1 - w_{ij}) + M_2(1 - x_{ij}), \quad \forall (i,j) \in \mathcal{A} \tag{2.52}$$

$$t_j^- \geq t_i^+(1 - z_i) + \tau_{ij}^D(1 - w_{ij}) - M_2(1 - x_{ij}), \quad \forall (i,j) \in \mathcal{A} \tag{2.53}$$

$$l_j \leq T_{\text{bl}} z_j + M_2 \sum_{i:(i,j)\in\mathcal{A}} w_{ij}, \quad \forall j \in \mathcal{H} \tag{2.54}$$

$$l_j \geq t_j^- - M_2(1 - z_j), \quad \forall j \in \mathcal{H} \tag{2.55}$$

$$l_j \geq \sum_{i:(i,j)\in\mathcal{A}} \Big(\tau_{ij}^T y_{ij} + \big(\max(\tau_{ij}^T, T_{\text{s}}) - \tau_{ij}^T \big) w_{ij} \Big) - M_1(1 - z_j), \quad \forall j \in \mathcal{H} \tag{2.56}$$

$$z_0 = 1, t_0^+ = 0 \tag{2.57}$$

$$x_{ij} \in \{0,1\}, y_{ij} \in \{0,1\}, w_{ij} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{A} \tag{2.58}$$

$$z_i \in \{0,1\}, u_i \in [0, n+1], t_i^-, t_i^+ \in [0, T_{\text{bl}}], l_i \in \mathcal{R}_+, \quad \forall i \in \mathcal{H} \tag{2.59}$$

To make the proof self-contained, we introduce the notations that are applied in TDTL

Table 2.2: Notations used in the TDTL MIP formulation.

Sets	
$\mathcal{N}$	Set of nodes of graph $G = (\mathcal{N}, \mathcal{A})$.
$\mathcal{A}$	Set of directed links in $G = (\mathcal{N}, \mathcal{A})$.
o/e	The origin/ending node of the mission, $o, e \in \mathcal{N}$.
$\delta^+(i)$	Nodes that can be reached from i, $i \in \mathcal{N}$.
$\delta^-(i)$	Nodes that can reach to node i, $i \in \mathcal{N}$.
Parameters	
Q	Battery capacity expressed in time units.
$t_{ij}^T/$	Truck travel time at link $(i, j) \in \mathcal{A}$.
t_{ij}^D	Drone travel time at link $(i, j) \in \mathcal{A}$.
M	A big enough constant.
Variables	
u_{ij}	If $u_{ij} = 1$, link (i, j) is traversed by the truck.
v_{ij}	If $v_{ij} = 1$, link (i, j) is traversed by the drone.
s_i	The earliest departure time from node $i \in \mathcal{N}$.
b_i^-	Drone battery level when if drone is just coming to the node $i \in \mathcal{N}$.
b_i^+	Drone battery level when the drone is just departing from node $i \in \mathcal{N}$.

model and reproduce the TDTL model from [22].

$$\text{(TDTL)} \quad \min \quad s_e$$

$$\text{s.t.} \quad \sum_{j \in \delta^-(i)} u_{ji} \leq 1, \quad \forall i \in \mathcal{N} \setminus \{o, e\} \tag{2.60}$$

$$\sum_{j \in \delta^+(i)} u_{ij} - \sum_{j \in \delta^-(i)} u_{ji} = 0, \quad \forall i \in \mathcal{N} \setminus \{o, e\} \tag{2.61}$$

$$\sum_{j \in \delta^+(o)} u_{oj} = 1 \tag{2.62}$$

$$\sum_{i \in \delta^-(e)} u_{ie} = 1 \tag{2.63}$$

$$\sum_{j \in \delta^-(i)} v_{ji} \leq 1, \quad \forall i \in \mathcal{N} \setminus \{o, e\} \tag{2.64}$$

$$\sum_{j \in \delta^+(i)} v_{ij} - \sum_{j \in \delta^-(i)} v_{ji} = 0, \quad \forall i \in \mathcal{N} \setminus \{o, e\} \tag{2.65}$$

$$\sum_{j \in \delta^+(o)} v_{oj} = 1 \tag{2.66}$$

$$\sum_{i \in \delta^-(e)} v_{ie} = 1 \tag{2.67}$$

$$\sum_{i \in \delta^-(j)} u_{ij} + \sum_{i \in \delta^-(j)} v_{ij} \geq 1, \quad \forall j \in \mathcal{N} \setminus \{o\} \tag{2.68}$$

$$s_j \geq s_i + t_{ij}^T u_{ij} - M\Big(1 - u_{ij}\Big), \quad (i,j) \in \mathcal{A} \tag{2.69}$$

$$s_j \geq s_i + t_{ij}^D v_{ij} - M\Big(1 - v_{ij} + u_{ij}\Big), \quad (i,j) \in \mathcal{A} \tag{2.70}$$

$$s_o = 0 \tag{2.71}$$

$$b_j^- \leq Q + M\Big(2 - v_{ij} - u_{ij}\Big), \quad (i,j) \in \mathcal{A} \tag{2.72}$$

$$b_j^- \geq Q - M\Big(2 - v_{ij} - u_{ij}\Big), \quad (i,j) \in \mathcal{A} \tag{2.73}$$

$$b_j^+ \leq Q + M\Big(2 - v_{ij} - u_{ij}\Big), \quad (i,j) \in \mathcal{A} \tag{2.74}$$

$$b_j^+ \geq Q - M\Big(2 - v_{ij} - u_{ij}\Big), \quad (i,j) \in \mathcal{A} \tag{2.75}$$

$$b_j^- \leq b_i^+ - t_{ij}^D + M\Big(1 - v_{ij} + u_{ij} + \sum_{k \neq i} u_{kj}\Big), \quad (i,j) \in \mathcal{A} \tag{2.76}$$

$$b_j^- \geq b_i^+ - t_{ij}^D - M\Big(1 - v_{ij} + u_{ij} + \sum_{k \neq i} u_{kj}\Big), \quad (i,j) \in \mathcal{A} \tag{2.77}$$

$$b_j^+ \leq b_j^- + M\Big(1 - v_{ij} + u_{ij} + \sum_{k \neq i} u_{kj}\Big), \quad (i,j) \in \mathcal{A} \tag{2.78}$$

$$b_j^+ \geq b_j^- - M\Big(1 - v_{ij} + u_{ij} + \sum_{k \neq i} u_{kj}\Big), \quad (i,j) \in \mathcal{A} \tag{2.79}$$

$$b_j^- \leq b_i^+ - t_{ij}^D + M\Big(1 - v_{ij} + u_{ij} + 1 - \sum_{k \neq i} u_{kj}\Big), \quad (i,j) \in \mathcal{A} \tag{2.80}$$

$$b_j^- \geq b_i^+ - t_{ij}^D - M\Big(1 - v_{ij} + u_{ij} + 1 - \sum_{k \neq i} u_{kj}\Big), \quad (i,j) \in \mathcal{A} \tag{2.81}$$

$$b_j^+ \leq Q + M\Big(1 - v_{ij} + u_{ij} + 1 - \sum_{k \neq i} u_{kj}\Big), \quad (i,j) \in \mathcal{A} \tag{2.82}$$

$$b_j^+ \geq Q - M\Big(1 - v_{ij} + u_{ij} + 1 - \sum_{k \neq i} u_{kj}\Big), \quad (i,j) \in \mathcal{A} \tag{2.83}$$

$$b_o^+ = Q \tag{2.84}$$

Let the polyhedron of the linear relaxation of models ZONVRP and TDTL be defined by

$$P(\text{ZONVRP}) = \left\{(l, t^-, t^+, z, u, w, x, y) \in \mathcal{R}_{\geq 0}^{5(n+1)} \times [0,1]^{3n^2} \middle| \text{constraints: (2.35)–(2.59)}\right\}$$

$$P(\text{TDTL}) = \left\{(s, b^-, b^+, u, v) \in \mathcal{R}_{\geq 0}^{3(n+1)} \times [0,1]^{2n^2} \middle| \text{ constraints: (2.60)–(2.84)}\right\}$$

Given the linear transformation Φ in (2.85)–(2.89) below, we will show that each constraint in $P(\text{TDTL})$ is implied by that in $P(\text{ZONVRP})$. We use the notation TDTL.(k) to refer to the constraint (k) in the TDTL model.

$$x_{ij} = v_{ij}, \quad \forall (i,j) \in \mathcal{A} \tag{2.85}$$

$$y_{ij} = u_{ij}, \quad \forall (i,j) \in \mathcal{A} \tag{2.86}$$

$$z_j = \sum_{i:(i,j)\in\mathcal{A}} u_{ij}, \quad \forall j \in \mathcal{N} \tag{2.87}$$

$$t_i^- = Q - b_i^-, \quad \forall i \in \mathcal{N} \tag{2.88}$$

$$t_i^+ = Q - b_i^+, \quad \forall i \in \mathcal{N} \tag{2.89}$$

TDTL.(2.60)

$$\sum_{j\in\delta^-(i)} u_{ji} = \sum_{j:(j,i)\in\mathcal{A}} y_{ji} = z_i \leq 1$$

TDTL.(2.61)–(2.63) are implied by constraint (2.40)–(2.42).

TDTL.(2.64)–(2.67) are implied by constraints (2.35)–(2.39) directly.

TDTL.(2.68)

$$\sum_{i\in\delta^-(j)} u_{ij} + \sum_{i\in\delta^-(j)} v_{ij} = z_j + \sum_{i:(i,j)\in\mathcal{A}} x_{ij} = z_j + 1 \geq 1$$

TDTL.(2.69) is in charge of setting the departure time at node j given that arc (i,j) is

traversed by the truck.

$$
\begin{aligned}
& s_j - s_i - t_{ij}^T u_{ij} + M(1 - u_{ij}) \\
& = s_j - s_i - \tau_{ij}^T y_{ij} + M(1 - y_{ij}) \\
& \geq s_j - s_i - \tau_{ij}^T, \quad \text{implied by } y_{ij} = 1 \\
& = \max\left\{\tau_{ij}^T, \max\{\tau_{ij}^T, T_s\}\right\} - \tau_{ij}^T, \text{where } s_j - s_i \text{ is no less than the truck travel time from location i to j} \\
& \geq 0
\end{aligned}
$$

TDTL.(2.70) regulates the departure time at node j given that arc (i, j) is traversed by the drone.

$$
\begin{aligned}
& s_j - s_i - t_{ij}^D v_{ij} + M(1 - v_{ij} + u_{ij}) \\
& = s_j - s_i - \tau_{ij}^D x_{ij} + M(1 - x_{ij} + y_{ij}) \\
& \geq s_j - s_i - \tau_{ij}^D, \quad \text{implied by } x_{ij} = 1, y_{ij} = 0 \\
& = \tau_{ij}^D - \tau_{ij}^D \\
& \geq 0
\end{aligned}
$$

TDTL.(2.71) is implied by constraint (2.57).

TDTL.(2.72)–(2.75) describe the drone's battery level at the time it just arrives at or departs from a node j.

$$
\begin{aligned}
& b_j^- - Q - M(2 - v_{ij} - u_{ij}) = -t_j^- - M(2 - x_{ij} - y_{ij}) \leq 0 \\
& b_j^- - Q + M(2 - v_{ij} - u_{ij}) = -t_j^- + M(2 - x_{ij} - y_{ij}) \geq 0, \quad \text{implied by constraints (2.47)– (2.53)} \\
& b_j^+ - Q - M(2 - v_{ij} - u_{ij}) = -t_j^+ - M(2 - x_{ij} - y_{ij}) \leq 0 \\
& b_j^+ - Q + M(2 - v_{ij} - u_{ij}) = -t_j^+ + M(2 - x_{ij} - y_{ij}) \geq 0, \quad \text{implied by constraints (2.46)– (2.53)}
\end{aligned}
$$

TDTL.(2.76), $\forall (i,j) \in \mathcal{A}$

$$
\begin{aligned}
& b_j^- - b_i^+ + \tau_{ij}^D - M\Big(1 - v_{ij} + u_{ij} + \sum_{k \neq i} u_{kj}\Big) \\
& = t_i^+ - t_j^- + \tau_{ij}^D - M(1 - x_{ij} + z_j) \\
& \leq t_i^+ - t_j^- + \tau_{ij}^D, \quad \text{implied by } x_{ij} = 1, z_j = 0 \\
& = -\tau_{ij}^D + \tau_{ij}^D \\
& \leq 0
\end{aligned}
$$

TDTL.(2.77), $\forall (i,j) \in \mathcal{A}$

$$
\begin{aligned}
& b_j^- - b_i^+ + \tau_{ij}^D + M\Big(1 - v_{ij} + u_{ij} + \sum_{k \neq i} u_{kj}\Big) \\
& = t_i^+ - t_j^- + \tau_{ij}^D + M(1 - x_{ij} + z_j) \\
& \geq t_i^+ - t_j^- + \tau_{ij}^D, \quad \text{implied by } x_{ij} = 1, z_j = 0 \\
& = -\tau_{ij}^D + \tau_{ij}^D \\
& \geq 0
\end{aligned}
$$

TDTL.(2.78), $\forall (i,j) \in \mathcal{A}$

$$
\begin{aligned}
& b_j^+ - b_j^- - M\Big(1 - v_{ij} + u_{ij} + \sum_{k \neq i} u_{kj}\Big) \\
& = t_j^- - t_j^+ - M(1 - x_{ij} + z_j) \\
& \leq t_j^- - t_j^-(1 - z_j), \quad \text{implied by } x_{ij} = 1, z_j = 0, \text{constraint (2.46)} \\
& \leq 0
\end{aligned}
$$

TDTL.(2.79), $\forall (i,j) \in \mathcal{A}$

$$
\begin{aligned}
& b_j^+ - b_j^- + M\Big(1 - v_{ij} + u_{ij} + \sum_{k \neq i} u_{kj}\Big) \\
& = t_j^- - t_j^+ + M(1 - x_{ij} + z_j) \\
& \geq t_j^- - t_j^-(1 - z_j), \quad \text{implied by } x_{ij} = 1, z_j = 0, \text{constraint (2.46)} \\
& \geq 0
\end{aligned}
$$

TDTL.(2.80)

$$
\begin{aligned}
& b_j^- - b_i^+ + \tau_{ij}^D - M\Big(1 - v_{ij} + u_{ij} + 1 - \sum_{k \neq i} u_{kj}\Big) \\
& = -t_j^- + t_i^+ + t_{ij}^D - M\Big(1 - x_{ij} + y_{ij} + 1 - \sum_{k \neq i} y_{kj}\Big) \\
& \leq -t_j^- + t_i^+ + t_{ij}^D, \quad \text{implied by } x_{ij} = 1, y_{ij} = 0, z_j = 1 \\
& = -t_{ij}^D + t_{ij}^D \\
& \leq 0
\end{aligned}
$$

TDTL.(2.81), $\forall (i,j) \in \mathcal{A}$

$$
\begin{aligned}
& b_j^- - b_i^+ + \tau_{ij}^D + M\Big(1 - v_{ij} + u_{ij} + 1 - \sum_{k \neq i} u_{kj}\Big) \\
& = -t_j^- + t_i^+ + t_{ij}^D + M\Big(1 - x_{ij} + y_{ij} + 1 - \sum_{k \neq i} y_{kj}\Big) \\
& \geq -t_j^- + t_i^+ + t_{ij}^D, \quad \text{implied by } x_{ij} = 1, y_{ij} = 0, z_j = 1 \\
& = -t_{ij}^D + t_{ij}^D \\
& \geq 0
\end{aligned}
$$

TDTL.(2.82), $\forall (i,j) \in \mathcal{A}$

$$
\begin{aligned}
& b_j^+ - Q - M\Big(1 - v_{ij} + u_{ij} + 1 - \sum_{k \neq i} u_{kj}\Big) \\
& = -t_j^+ - M\Big(1 - x_{ij} + y_{ij} + 1 - \sum_{k \neq i} y_{kj}\Big) \\
& \leq -t_j^-(1 - z_j), \quad \text{implied by } x_{ij} = 1, y_{ij} = 0, z_j = 1 \\
& \leq 0
\end{aligned}
$$

TDTL.(2.83), $\forall (i,j) \in \mathcal{A}$

$$
\begin{aligned}
& b_j^+ - Q + M\Big(1 - v_{ij} + u_{ij} + 1 - \sum_{k \neq i} u_{kj}\Big) \\
& = -t_j^+ - M\Big(1 - x_{ij} + y_{ij} + 1 - \sum_{k \neq i} y_{kj}\Big) \\
& \geq -t_j^-(1 - z_j), \quad \text{implied by } x_{ij} = 1, y_{ij} = 0, z_j = 1 \\
& \geq 0
\end{aligned}
$$

TDTL.(2.84) is implied by constraint (2.57).

We have finally shown that $\Phi(P(\text{ZONVRP})) \subseteq P(\text{TDTL})$. To further establish that the feasible region of $\Phi(P(\text{ZONVRP}))$ is strictly contained in $P(\text{TDTL})$, it is sufficient to give a solution that is valid in TDTL model but infeasible in ZONVRP model.

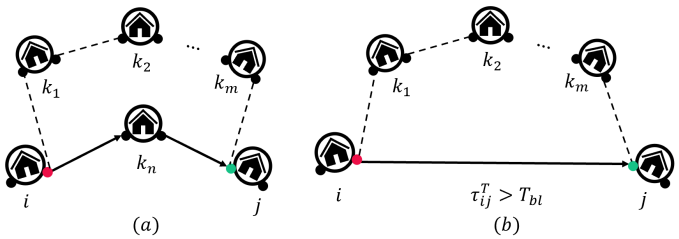


Figure 2.4: Partial solutions that are avoided in ZONVRP.

In Figure 2.4 (a), we depict a nested unit in which the truck visits location k_n before

drives to rendezvous location j. This nested unit could be part of any TDTL solution. However, such a solution is infeasible to the ZONVRP model since constraints (2.1)–(2.2) require that each location has to be visited by the drone exactly once. In this case shown in Figure 2.4 (b), the truck travels for more than T_{bl} time units to the designated destination j; and here the drone has depleted its battery and has been forced to land on the ground. But the ZONVRP excludes this situation via the constraint (2.12). □ □

CHAPTER 3
SOLVING LARGE SCALE MIXED INTEGER QUADRATICALLY-CONSTRAINED PROGRAM

3.1 Overview

With the Nested-VRP model defined in chapter 2, we dive deep into the solution methodology in solving the Nested-VRP model in this chapter. Essentially, we are interested in solving a Mixed Integer Quadratically Constrained Programming (MIQCP) problem of the following form:

$$\begin{aligned} (\text{MIQCP})\ \min \quad & c^T x \\ s.t. \quad & Ax = b \\ & x^T Q x + q^T x \leq b_i \\ & l \leq x \leq u \\ & x \in \mathcal{R}^{n-p} \times \mathcal{I}^p \end{aligned}$$

Where x is a vector of variables that are lower bounded by vector l and upper bouned by vector u. The MIQCP distinguishes itself by incorporating a mixture of continuous and integer variables into the model. In addition, by definition, the MIQCP model, considers linear objective function and only nonlinearity in a form of quadratic terms in the constraints.

There are two ways that a MIQCP can be reduced to a Mixed Integer Linear Program (MILP) that only consider linear objective function and linear constraints. If the right-hand side of the quadratic constraint b_i is large enough, the corresponding quadratic constraint becomes redundant. As a result, the MIQCP with redundant quadratic constraints becomes equivalent to the MILP. Another strategy is to perform linear transformation on

the quadratic terms to a set of linear constraints by introducing additional variables into MIQCP while maintain the behavior of the model.

We can see that the MIQCP is closely related to the MILP, which is well-known to be an NP-hard problem. Over the last decade, there has been fruitful research into the techniques for solving the MILP. One line of research considers exact algorithms that always solve an optimization problem to the optimality. For example, the branch-and-bound algorithm and the cutting plane algorithm. Because of the hardness of MILP, solving problems of realistic size could be intractable. As a result, research into the heuristic method as an alternative to the exact approach have been receiving a lot of attention. Heuristic methods have the ideal properties of reduced computation time and easy-to-understand logic, which are achieved at a price of sacrificing the optimality of the solution.

To solve the Nested-VRP, in this chapter, we first analyze the complexity of the Nested-VRP model from multiple perspectives in section 3.2. Then, we review and propose a set of criteria to evaluate the effectiveness and efficiency of solution approaches in section 3.3. Next, in section 3.4, an enhancement of the MIQCP model for the Nested-VRP is achieved by deriving the equivalent Mixed Integer Linear Programming (MILP) formulation as well as leveraging lifting and Reformulation-Linearization techniques to strengthen the subtour elimination constraints of the drone. Given the NP-hard nature of the Nested-VRP, we further propose an efficient neighborhood search (NS) heuristic in which we generate and improve on a good initial solution by iteratively solving the Nested-VRP on a local scale in section 3.5. Finally, we compare exact approaches based on MIQCP or its enhanced formulations and NS heuristic methods in small and large problem sizes, and present the results of a computational study to demonstrate the effectiveness of the MIQCP model and its variants, as well as the efficiency of the NS heuristic, including for a real-life instance with 631 locations. We envision that this framework will facilitate the planning and operations of combined drone-truck missions. This chapter contribute to the literature contributions in following ways:

- We propose an effective Neighborhood Search (NS) heuristic to solve the Nested-VRP. Although the NS heuristic is widely studied in solving combinatorial optimization problems, the proposed heuristic includes innovations in evaluating the goodness of local geometry by measuring how efficiently the drone battery can be used in nested units.

- An absolute lower bound on the mission makespan of the Nested-VRP is provided and serves as a benchmark for assessing the quality of heuristic solutions.

- We conduct extensive computational experiments from which we empirically examine the improvement of the Nested-VRP model by applying linearization and constraint strengthening techniques, demonstrate the effectiveness and efficiency of the proposed NS heuristic, and extract valuable insights for practitioners.

3.2 The Complexity of the Nested-VRP Model

3.2.1 Model Complexity without Prior Information

To understand the complexity of the Nested-VRP model, we will evaluate the computational effort required throughout the decision-making process. (I) First, the drone route is constructed by sequencing n locations. Together with the origin 0 and eventual destination $n+1$, there exists $n!$ different drone routes. Each possible drone route has a unique battery consumption pattern along the tour. (II) Second, given a specific possible drone route, the swap stops assignment is a problem of finding a subset of locations that naturally split the drone route into path segments. In particular, the drone's flight duration and the truck's ground travel time for each of these segments should both be within the battery limit. We can see that the choices of the set of charging locations is highly sensitive to the battery usage corresponding to the drone route. Thus, if implementing a drone route requires excessive battery swaps to ensure flight continuity, an adjustment to the drone route may reduce the coordination effort for the truck to serve batteries. As the name of the model suggests,

the decisions regarding the drone route and truck route are intertwined dynamically and tied by the decisions on swap locations. To solve the Nested-VRP problem, we should navigate through the tasks of planning a good drone route and scheduling battery swaps in a versatile manner. Neither component seems to dominate the other, and that question merits further investigation—if partial information about the Nested-VRP solution is given, how much effort is needed to obtain the complete Nested-VRP solution?

3.2.2 Model Complexity with Prior Information

In the following, we will demonstrate that, given a fixed order for the drone to visit all locations, we can solve within polynomial time the remaining Nested-VRP solution—including the truck route and the placement of swap stops—that minimizes the mission makespan.

Theorem 3.2.1. *Given a fixed order of a set of locations representing a known drone route, the partial Nested-VRP solution —including a subset of locations as swap stops and the truck route—can be solved in polynomial time.*

Proof. With the drone route specified, the optimal Nested-VRP solution can obtained by finding the cheapest collection of non-overlapping nested units such that the union of the drone paths from each unit aligns with the predetermined drone route. In the following, we will first construct the set of all feasible nested units. Each of these nested units is associated with a cost (i.e., IBR plus battery service time). Then, the collection of nested units (CNU) with the smallest mission time is obtained by solving an integer program efficiently. We name it the CNU problem.

Let $(s_0, s_1, \ldots, s_{n+1})$ be a permutation of the node set $\mathcal{H}$, where $s_0 = 0$ and $s_{n+1} = n + 1$. In following this order, the drone departs from a location s_i, travels to the next location s_{i+1} which takes time $\tau^D_{s_i s_{i+1}}$, and spends $o_{s_{i+1}}$ time for surveying location s_{i+1}. Let $W = (w_0, w_1, w_2, \ldots, w_{2n+1})$ denote the consecutive tasks to be completed by the drone in the mission, where $w_0 = o_{s_0} = 0, w_1 = \tau^D_{s_0 s_1}, w_{2k} = o_{s_k}, w_{2k+1} = \tau^D_{s_k s_{k+1}}, \forall k = 1, \ldots, n$. Without loss of generality, we assume that a rendezvous location is placed at the beginning

of the mission which corresponds to equipping the drone with a full battery. In addition, for simplicity, we assume that any other rendezvous location $i \in \{0, 1, \ldots, 2n+1\}$ should be limited to where the task w_i is completed (i.e., the drone just arrives at a location or just completes an observation task).

For a nested unit with the first rendezvous location at the end of completing task w_i and the second rendezvous location at the end of completing task w_j, we denote the nested unit as $(i, j, l_{ij}), i, j \in \{0, 1, \ldots, 2n+1\}$, $i < j$, where l_{ij} is the cost of the nested unit measured in time. Specifically, variables l_{ij} is the sum of the battery swap service time for obtaining a new battery at the start of the nested unit and the interval between rendezvous of the unit. Recall that when the truck ships the drone, the nested unit is in a special form of "Shipment" which delays the mission by the maximum of the truck travel time and the battery swap service time. To determine l_{ij}, we further define $D_{ij} = \sum_{k=i+1}^{j} w_k$ as the total drone travel and surveillance time for completing the $(i+1)$th task and all others up to and including the jth task. Likewise, define T_{ij} as the truck travel time moving from where the i task is completed to where the jth task will be completed by the drone. A nested unit is feasible if both D_{ij} and T_{ij} are within the battery limit T_{bl} unless the nested unit is in a form of "Shipment". The nested unit, if picked, will delay the mission by $l_{ij} \in \mathcal{R}_+$, which is defined as follows.

$$l_{ij} = \begin{cases} \max(T_{ij}, T_s) & \text{if } i \text{ is even, } j = i+1, \text{"Shipment"} \\ \max(D_{ij}, T_{ij}) + T_s & \text{if } j > i+1, D_{ij} \le T_{\text{bl}} \text{ or } T_{ij} \le T_{\text{bl}}, \text{"Nested unit"} \\ \infty & \text{otherwise} \end{cases}$$

Let $U = \{(i, j, l_{ij}) \ : \forall i, j \in \{0, 1, \ldots, 2n+1\}, \ i < j\}$ denote the set of all possible formations of nested units, the CNU problem is to find the least-delayed subset of nested units such that all tasks are completed/covered exactly once. We now formulate the CNU problem. Define the binary matrix $A \in \{0, 1\}^{|U||W|}$. For each nested unit $u \in U$, we have $A_{uk} = 1$ if task w_k is covered by nested unit u, and 0 otherwise. For simplicity, let

l_u represents the time cost of nested unit u. Define decision variables $x_u = \{0, 1\}, \forall u \in \{1, \ldots, |U|\}$. If $x_u = 1$, then nested unit u is selected, 0 otherwise.

$$\begin{aligned} (\text{CNU}) \quad \min \quad & \sum_{u=1}^{|U|} l_u x_u \\ \text{s.t.} \quad & \sum_{u=1}^{|U|} A_{uk} x_u = 1, \quad \forall k \in \{1, 2, \ldots, 2n+1\} \\ & x_u \in \{0, 1\}, \quad \forall i \in \{1, 2, \ldots, |U|\} \end{aligned} \tag{3.1}$$

The matrix A is total unimodular (TU) since each row of matrix A consists of consecutive ones [31]. Then since A is TU, the non-empty polyhedron $P(b) = \{Ax = b, x \geq 0\}$ has integral vertices for the all-integral vector b [32]; in our case, b is a vector of all 1s. Therefore, we can solve the CNU model by solving its linear relaxation and still achieve integer solutions. The time effort in solving a linear program is polynomially bounded by the total number of variables $|U|$ [33]. □

3.3 Criteria for Comparing Optimization Methodologies

3.3.1 Efficiency and Accuracy Trade-off

Given an optimization problem, making a fair and an unbiased assessment of various optimization methodologies and identifying the most appropriate one to solve the problem is complicated. Typically, this evaluation procedure entails implementing methodologies and empirically applying each methodology to a given set of problem instances. Performance metrics are obtained by statistically summarizing how quickly a problem can be solved and how good the solution is. When performed correctly, such an analysis can help the decision-maker in selecting the most appropriate methodology, taking into account the time budget and quality requirements for the solution.

In general, performance metrics center on two criteria: (1) solution accuracy and (2) runtimes (i.e., either the time required to solve a problem instance and, if not, return infea-

sibility notice, or the total number of unit operations to complete the computation process). This section focuses on reviewing performance metrics for evaluation methodologies in solving single-objective optimization problem sequentially (i.e., not considering parallel processing scheme). We briefly summarize the performance criteria in Figure 3.1.

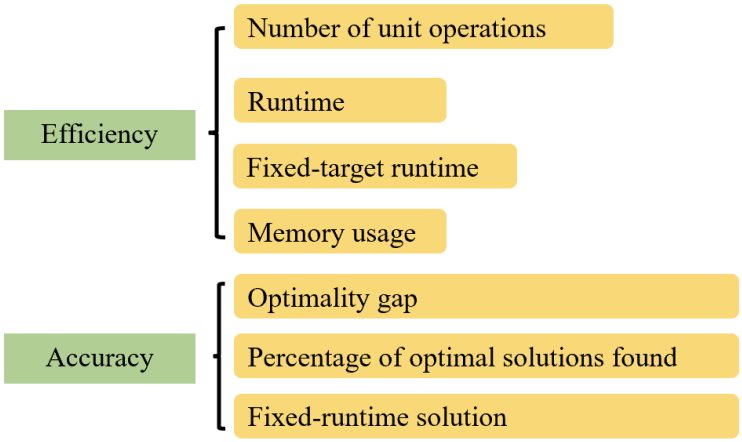


Figure 3.1: Performance metrics.

For exact methods that guarantee optimal results, there is no room for improvement in the accuracy of the solution. Consequently, the efficiency of the methodologies becomes the main priority. When solving a specific instance of a problem, we assess the number of unit operations and amount of time required to produce the optimal solution. These two criteria are not necessary independent. In the case of the branch-and-bound algorithm, for instance, a decrease in branching operations indicates a decrease in unit operations. However, depending on the structure of the sub problems, the computation time for solving a sub problem associated with each sub node may require a significant amount of time, resulting in an increase in the overall runtime. In situations where we are only concerned with the amount of time required to reach a solution that meets a predetermined quality threshold, we evaluate the methodology again based on its efficiency rather than its accuracy. Memory space is crucial when solving large-scale problems that require intensive read and write operations. For instance, if memory becomes constrained, branch-and-bound may be

unable to evaluate all nodes due to the failure to preserve the tree structure in the machine.

For heuristics, there is no guarantee that they will produce an optimal solution; therefore, a fair evaluation relies on a combination of efficiency and accuracy measurements. An essential concept in the optionality gap is the percentage deviation between the generated solution and the best known solution. In the case of a predetermined set of test instances, the greater a heuristic's ability to produce solutions that align with the global optimal solution, the more accurate the heuristic method. Typically, heuristics involve incrementally enhancing a feasible solution and can run forever to polish it. Instead of running the heuristic endlessly, it is common practice to terminate the polishing process based on a set of termination criteria or specify a maximum runtime and then focus on the solution quality at termination. In this situation, we evaluate the fixed-runtime solution.

If multiple methodologies are evaluated simultaneously and we are only interested in evaluating their relative performance, we can compare the efficiency metrics based on the fastest methodology and the accuracy based on the most accurate solution.

However, the best-known or optimal solutions are not always available, particularly when solving difficult problems with practical sizes. In this situation, it would be advantageous to derive an optimistic estimate of the optimal solution and compare it to the solutions produced by candidate methodologies. In the following section, we derive an absolute global lower bound for the Nested-VRP objective function.

3.3.2 Lower Bounding Method

For an optimization problem, a lower bound is a value that is known to be less than or equal to the optimum. Generally, a lower bound is used for evaluating the quality of a solution solved by a heuristic when the optimal solution is unattainable. Ideally, a tighter lower bound gives a more-qualified guarantee of a near-optimal solution. In this section, we focus on deriving a lower bound on the mission makespan of the Nested-VRP problem. We compare the solutions obtained by the NS heuristic to the lower bound value. The

tightness of the proposed lower bound will be evaluated in future work.

We start by investigating the battery usage in each of the nested units in the solution. Given a nested unit u, the drone surveillance time includes time spent on traveling between locations and completing observing tasks. As depicted in Figure 3.2(a), if the truck arrives at a rendezvous location first, then once the drone arrives at the rendezvous location, the drone relinquishes all remaining battery life before it obtains a new battery. The battery slackness is denoted as δ_u. However, in Figure 3.2(b), if the drone arrives at the rendezvous location first, the drone will idle for time Δ_u while waiting for the truck. Once the truck arrives, the drone lands on the truck and releases all remaining battery life δ_u.

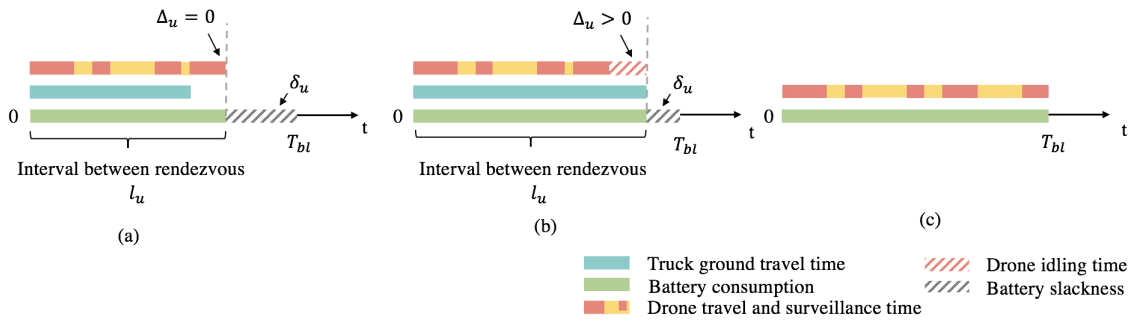


Figure 3.2: Battery usage in a nested unit. (a) The drone arrives later than the truck and thus the IBR l_u of the nested unit is determined by the drone's surveillance time (i.e., traveling and observing). The drone's idling time Δ_u is 0. The wasted battery energy δ_u is the difference between T_{bl} and l_u. (b) The truck arrives later than the drone and thus the IBR l_u of the nested unit is determined by the truck's traveling time. The drone arrives at the rendezvous location and idles for time Δ_u. The wasted battery energy δ_u is the difference between T_{bl} and l_u. (c) The drone battery can be swapped at anytime, anywhere. Thus, the drone's idling time δ_u and the wasted battery time $T_{\text{bl}} - l_u$ are trivial.

A lower bounding technique is based on relaxing some of the constraints in the original Nested-VRP model. First, instead of restricting the swap stops to take place at locations, the drone is allowed to replace its battery either en route from one location to the other or while observing at a location. Second, we simplify the synchronization constraint that ordinarily requires the truck to meet up the drone before the drone battery charge has expired; now we allow the drone to replace the battery itself without the truck being involved. This relaxed

Nested-VRP is illustrated in Figure 3.2(c). In this case, the drone idling time Δ_u and battery slackness δ_u are trivial. It is clear that the total number of swap stops is purely proportional to the battery consumption that occurs only during drone travel and surveillance activities.

Given the above relaxations, an optimistic estimate of the mission makespan consists of the following three components: (i) the minimum time spent in drone routing (i.e., the TSP route); (ii) the constant time spent in observing locations; and (iii) the smallest number of swap stops multiplied by the battery swap service time. We formally establish the lower bound on the objective value of the Nested-VRP model in Theorem 3.3.1.

Theorem 3.3.1. *Given a Nested-VRP instance described in graph $\mathcal{G} = (\mathcal{H}, \mathcal{A})$, let $\mathcal{S}$ denote the set of arcs on the TSP route where $\mathcal{S} = \{(i,j) : (i,j) \in \mathcal{A}, (i,j)$ is on the TSP route.$\}$. The lower bound on the value of an optimal solution of the Nested-VRP problem, LB, can be computed by Equation (3.2).*

$$\text{LB} = \sum_{(i,j)\in\mathcal{S}} \tau_{ij}^{D} + \sum_{k\in\mathcal{H}} o_k + \left\lfloor \frac{1}{T_{\text{bl}}} \Big(\sum_{(i,j)\in\mathcal{S}} \tau_{ij}^{D} + \sum_{k\in\mathcal{H}} o_k \Big) \right\rfloor T_{\text{s}} \tag{3.2}$$

Proof of Theorem 3.3.1

Proof. Given a Nested-VRP described in graph $\mathcal{G} = (\mathcal{H}, \mathcal{A})$, the optimal solution to the problem consists of the drone route $\mathcal{X} = \{(i,j) \mid (i,j) \in \mathcal{A}, \text{ and } x_{ij} = 1\}$ and the truck route $\mathcal{Y} = \{(i,j) \mid (i,j) \in \mathcal{A}, \text{ and } y_{ij} = 1\}$. The optimal solution can also be described as a set of non-overlapping nested units U. Mathematically, a nested unit $u \in U$ can be viewed as a sub-graph of $\mathcal{G}$. The sub graph contains a set of locations $V(u)$ to be observed by the drone and the corresponding drone path $E(u)$. Most importantly, one can see that $\mathcal{X} \equiv \{(i,j) \mid (i,j) \in E(u), u \in U\}$ and $\mathcal{H} \equiv \{k \mid k \in V(u), u \in U\}$.

In a nested unit u, when the drone is about to meet with the truck, the drone arrives at the rendezvous location either earlier than the truck and idles for Δ_u time units or later than the truck without idling. We define an indicator function $\mathbb{1}_u$ for each nested unit u that forms the optimal Nested-VRP solution. In a nested unit u, if the drone arrives earlier than the

truck at rendezvous, $\mathbb{1}_u = 1$, otherwise $\mathbb{1}_u = 0$. After two vehicles meet up successfully, the drone relinquishes all remaining battery life before it obtains a new battery. This portion of unused battery life is denoted as δ_u.

A battery is either used for drone surveillance (e.g., routing between locations, surveying locations, and possibly waiting for the truck) or wasted after the two vehicles meet up. To derive the lower bound of the mission makespan of a Nested-VRP solution, we first investigate the battery consumption associated with a Nested-VRP solution. Denote the IBR l_u of a nested unit as in Equation (3.3). We summarize the battery consumption breakdowns in Equation (3.4).

$$l_u = \sum_{(i,j)\in E(u)} \tau_{ij}^D x_{ij} + \sum_{k\in V(u)} o_k + \mathbb{1}_u \Delta_u \tag{3.3}$$

$$T_{\text{bl}} = l_u + \delta_u \tag{3.4}$$

Due to the conservation of energy, the amount of energy extracted from all battery replacements scheduled en route should be able to balance off the total battery consumption needed for the mission. Therefore, we can derive the necessary number of battery swaps N_{s} as follows:

$$N_{\text{s}} T_{\text{bl}} = \sum_{u\in U} (l_u + \delta_u) = \sum_{u\in U} \Big(\sum_{(i,j)\in E(u)} \tau_{ij}^D x_{ij} + \sum_{k\in V(u)} o_k + \mathbb{1}_u \Delta_u + \delta_u \Big)$$

$$N_{\text{s}} = \frac{1}{T_{\text{bl}}} \sum_{u\in U} \Big(\sum_{(i,j)\in E(u)} \tau_{ij}^D x_{ij} + \sum_{k\in V(u)} o_k + \mathbb{1}_u \Delta_u + \delta_u \Big)$$

Since the mission makespan is the sum of the IBRs $l_u, \forall u \in U$ and battery swap service times:

$$\text{makespan} = \sum_{u\in U} l_u + N_{\text{s}} T_{\text{s}}$$

$$= \sum_{u \in U} \Big(\sum_{(i,j) \in E(u)} \tau_{ij}^{D} x_{ij} + \sum_{k \in V(u)} o_k + \mathbb{1}_u \Delta_u \Big) + \frac{T_{\mathrm{s}}}{T_{\mathrm{bl}}} \sum_{u \in U} \Big(\sum_{(i,j) \in E(u)} \tau_{ij}^{D} x_{ij} + \sum_{k \in V(u)} o_k + \mathbb{1}_u \Delta_u + \delta_u \Big)$$

We relax constraints imposed on how battery swap could happen and allow the drone to charge itself at any time when its battery has depleted. Therefore, the mission makespan of a relaxed version of Nested-VRP instance is given by:

$$\begin{aligned}
\text{makespan} &= \sum_{u \in U} \Big(\sum_{(i,j) \in E(u)} \tau_{ij}^{D} x_{ij} + \sum_{k \in V(u)} o_k \Big) + \frac{T_{\mathrm{s}}}{T_{\mathrm{bl}}} \sum_{u \in U} \Big(\sum_{(i,j) \in E(u)} \tau_{ij}^{D} x_{ij} + \sum_{k \in V(u)} o_k \Big) \\
&= \sum_{(i,j) \in \mathcal{X}} \tau_{ij}^{D} x_{ij} + \sum_{k \in \mathcal{H}} o_k + \frac{T_{\mathrm{s}}}{T_{\mathrm{bl}}} \Big(\sum_{(i,j) \in \mathcal{X}} \tau_{ij}^{D} x_{ij} + \sum_{k \in \mathcal{H}} o_k \Big) \qquad (3.5) \\
&\geq \sum_{(i,j) \in \mathcal{S}} \tau_{ij}^{D} + \sum_{k \in \mathcal{H}} o_k + \left\lfloor \frac{1}{T_{\mathrm{bl}}} \Big(\sum_{(i,j) \in \mathcal{S}} \tau_{ij}^{D} + \sum_{k \in \mathcal{H}} o_k \Big) \right\rfloor T_{\mathrm{s}}, \qquad (3.6)
\end{aligned}$$

Recall that $\mathcal{S}$ is the collection of arcs that are in the TSP route. Since the union of the drone paths in all nested units will produce a Hamiltonian cycle containing all of the locations and whose total length is no shorter than the TSP route, Equation (3.5) will be lower bounded by specifying the drone route as the TSP route as well as rounding down the total number of battery swaps to an integer. The result is that, the objective function value of the original Nested-VRP model is lower bounded by term (3.6). □

3.4 Exact Methodology with Constraint Strengthening

3.4.1 Model Linearization

One possible way to reduce model complexity is to transform the structure of the model into a more-amiable one. Specifically, the mathematical formulation of the Nested-VRP model consists of a handful of quadratic terms in the constraints, which leads to a Mixed Integer Quadratically Constrained Program (MIQCP). An efficient methodology for directly

tackling the MIQCP model requires careful design and relies on a combination of special techniques for decreasing the "upper bound" (e.g., heuristics to obtain integral solutions) or increasing the lower bound (e.g., valid inequality) which is challenging for commercial optimizers. Tracing back to the seventies, [34] showed that solving an MIQCP is deemed as a formidable task due to its notorious computational intractability. This motivates us to linearize the MIQCP model to its MILP equivalent. Solving an MILP model is one of the most-successful achievements in computational optimization. Theoretical and computational research progress over the last six decades has shown that MILPs can be solved in many if not all practical settings. We further propose an MILP equivalent of the Nested-VRP model in Proposition 3.4.1.

Proposition 3.4.1. *The following pairs of constraints are equivalent: (i) linear constraints* $(15^\star)$ *and quadratic constraints (2.15); (ii) linear constraints* $(21^\star) - (22^\star)$ *and quadratic constraints (2.21) – (2.22); (iii) linear constraints* $(28^\star)$ *and quadratic constraints (2.28)*

$$\begin{cases} P_j \in [0, T_{bl}] \\ P_j \leq T_{bl} z_j^- \\ P_j \leq t_j^- \\ P_j \geq t_j^- - T_{bl}(1 - z_j^-) \\ t_j^+ = t_j^- - P_j + o_j, \quad \forall j \in H \setminus \{0, n+1\} \end{cases} \tag{$15^\star$}$$

$$
\begin{cases}
F_i \in [0, T_{bl}] \\
F_i \leq T_{bl} z_i^+ \\
F_i \leq t_i^+ \\
F_i \geq t_i^+ - T_{bl}(1 - z_i^+) \\
t_j^- \leq t_i^+ - F_i + \tau_{ij}^D (1 - w_{ij}) + M_2(1 - x_{ij}), \forall (i,j) \in A \\
t_j^- \geq t_i^+ - F_i + \tau_{ij}^D (1 - w_{ij}) - M_2(1 - x_{ij}), \forall (i,j) \in A
\end{cases}
\tag{21*) – (22*}
$$

$$
\begin{cases}
Q_{ij} \leq y_{ij}, \quad \forall (i,j) \in A \\
Q_{ij} \leq z_j, \quad \forall (i,j) \in A \\
Y_{ij} + z_j \leq Q_{ij} + 1, \quad \forall (i,j) \in A \\
l_j^+ \geq \sum_{i:(i,j)\in\mathcal{A}} \tau_{ij}^T \big(y_{ij} - Q_{ij}\big) - M_1(1 - z_j^+), \quad \forall j \in \mathcal{H}
\end{cases}
\tag{28*}
$$

where auxiliary variables $P_j = t_j^- z_j^-, \forall j \in H \setminus \{0, n+1\}$, $F_i = t_i^+ z_i^+, \forall i \in H \setminus \{n+1\}$, *and* $Q_{ij} = y_{ij} z_j^-, \forall (i,j) \in A$.

3.4.2 Constraint Enhancement

Another way to possibly speed up the problem-solving process is to strengthen the Nested-VRP formulation. When solving the Nested-VRP model, as we stated earlier, the commercial optimizer typically adopts the spirit of branch-and-bound algorithm at its heart which involves repeatedly solving the LP relaxation and leveraging the generated bounds to guide branching decisions. Such an exact approach favors two strategies to resolve the need for integrality for variables. The first strategy is to identify effective cutting planes with the goal of iteratively reducing the search space by introducing linear inequalities. The second strategy aims at tightening up the polyhedral representation of the mathematical model at the root node by reformulating the model. In this chapter, we adopt the second strategy

and employ two different techniques to tighten up the MTZ subtour elimination constraints (2.4)–(2.5) that describe the routing decisions of the drone $x_{ij}, \forall (i,j) \in A$.

Inspired by research work [35] that utilizes a lifting technique to strengthen the MTZ subtour elimination constraints for the Traveling Salesman Problem, we apply their ideas with minor variations for the Nested-VRP.

Proposition 3.4.2. *The constraints*

$$1 + (1 - x_{0i}) + (n-2)x_{i,n+1} \leq u_i \leq (n+1) - (n-1)x_{0i} - (1 - x_{i,n+1}), \quad \forall i \in H \setminus \{0, n+1\} \tag{DL-1}$$

$$u_i - u_j + (n+1)x_{ij} + (n-1)x_{ji} \leq n, \quad \forall (i,j) \in A, i \neq 0 \tag{DL-2}$$

are valid inequalities for the Nested-VRP.

Proof of Proposition 3.4.2

Proof. The following analysis applies to general cases where the total number of locations n is more than two.

Consider the left-hand side of constraints (DL-1) with the coefficient of term $x_{j,n+1}$ being replaced by $a_{j,n+1}$, we compute the largest possible value for $a_{j,n+1}$ such that the inequality remains valid.

$$1 + (1 - x_{0j}) + a_{j,n+1}x_{j,n+1} \leq u_j \tag{3.7}$$

case 1. $x_{0j} = 1$. Since location j is right after location 0, we have $u_j = 1$. Meanwhile, location j should not be connected to the designated location $n+1$ when $n > 2$ (i.e., $x_{j,n+1} = 0$), otherwise, the drone misses out locations before returning back to the depot. In this case, constraint (3.7) becomes $0 \leq 1$ which is valid.

Case 2. $x_{0j} = 0$. Since the location j is not the one right after 0, the order u_j of location j must satisfy $2 \leq u_j \leq n+1$. Furthermore, if $x_{j,n+1} = 0$, we have $2 \leq u_j$ which is valid. However, if $x_{j,n+1} = 1$, we have $u_j = n$ and the constraint (3.7) becomes $a_{j,n+1} \leq u_j - 2 = n - 2$. Therefore, we set $a_{j,n+1} = n - 2$.

Next, we consider the right-hand side of constraint (DL-1) with the coefficient of term x_{0j} being replaced by b_{0j}. Similarly, we compute the largest possible value of b_{0j}.

$$u_j \leq (n+1) - b_{0j}x_{0j} - (1 - x_{j,n+1}) \tag{3.8}$$

Case 1. $x_{j,n+1} = 1$. Since location j is the last location visited before the drone coming back to depot $n+1$, we know the rank of location j is n (i.e., $u_j = n$). Moreover, location j must not be right after depot 0 Thus, the inequalities (3.8) become $n \leq n+1$ which are satisfied at all times.

Case 2. $x_{j,n+1} = 0$. If $x_{0j} = 0$, since location j could be anywhere not right after 0 and right before $n+1$, the rank of location j must satisfies $2 \leq u_j \leq n$ inequalities (3.8) become $u_j \leq n$ which is satisfied at all times. However, if $x_{0j} = 1$, we have $u_j = 1$ and $b_{0j} \leq n - 1$. Therefore, we choose $b_{0j} = n - 1$.

Consider the constraint (DL-2) with the coefficient of term x_{ji} being replaced by c_{ji}, if $c_{ji} = 0$, the constraint is the same with constraint (2.5)

$$u_i - u_j + (n+1)x_{ij} + c_{ji}x_{ji} \leq n \tag{3.9}$$

Case 1. $x_{ji} = 0$. The constraint is the same with constraint (2.5).

Case 2. $x_{ji} = 1$. This means that $x_{ij} = 0$ and $u_i \geq u_j + 1$ in the drone order. Thus, constraint (DL-2) becomes $u_i - u_j + c_{ji} \leq n$. We have $c_{ji} \leq n - 1$. □

Research work [36] proposed a novel RLT to derive even tighter relaxations for the Asymmetric Traveling Salesman Problem (ATSP) that is based on the MTZ formulation. The resulting new formulation of the ATSP has been theoretically and computationally

shown to outperform the lifted-MTZ formulation proposed by [35]. We apply the proposed RLT on strengthening the MTZ subtour elimination constraints in the Nested-VRP model.

Proposition 3.4.3. *The constraints*

$$\sum_{j:(i,j)\in A, j\neq n+1} y_{ij} + nx_{i,n+1} - u_i = 0, \quad \forall i \in H \setminus \{n+1\} \tag{SD-1}$$

$$\sum_{i:(i,j)\in A} y_{ij} + 1 = u_j, \quad \forall j \in H \setminus \{0\} \tag{SD-2}$$

$$x_{ij} \leq y_{ij} \leq nx_{ij}, \quad \forall (i,j) \in A, i \neq 0 \tag{SD-3}$$

$$u_i + (n+1)(x_{ij} - 1) + nx_{ji} \leq y_{ij} + y_{ji} \leq u_i - (1 - x_{ij}), \quad \forall (i,j) \in A, i \neq 0 \tag{SD-4}$$

$$1 + (1 - x_{0i}) + (n-2)x_{i,n+1} \leq u_i \leq n + 1 - (n-1)x_{0i} - (1 - x_{i,n+1}), \quad \forall i \in H \setminus \{0, n+1\} \tag{SD-5}$$

are valid inequalities for the Nested-VRP.

Proof of Proposition 3.4.3

Proof. We would like to show that constraints (SD-1) – (SD-5) are valid restatement of constraints (2.4) and (2.5). To achieve the goal, we follow the RTL technique proposed in [36] which consists of a reformulation and linearization steps.

We restate the MTZ subtour elimination constraints as follows:

$$u_j x_{ij} = (u_i + 1)x_{ij}, \quad \forall (i,j) \in A, i \neq 0 \tag{3.10}$$

$$u_j x_{0j} = x_{0j}, \quad \forall j \in H \setminus \{0\} \tag{3.11}$$

$$u_j x_{j,n+1} = nx_{j,n+1}, \quad \forall j \in H \setminus \{n+1\} \tag{3.12}$$

$$1 \leq u_j \leq n + 1, \quad \forall j \in H \setminus \{0\} \tag{3.13}$$

Constraint (3.10) – (3.13) excludes any subtour in the drone route. To see that, if $x_{ij} = 1$, location j increases rank by 1 as compared to that of the location i. Specifically,

if $x_{0j} = 1$, location j must have rank 1. If $x_{j,n+1} = 1$, location j has rank n. Note that Constraint (3.10) – (3.12) contain quadratic terms, we therefore perform linearization process by introducing new variables.

Reformulation: We reformulate the constraints (3.10) – (3.13) by generating additional implied constraints:

$$(\text{r1}) : u_i\Big(\sum_{j:(i,j)\in A} x_{ij} - 1\Big) = 0, \quad \forall i \in H \setminus \{n+1\}$$

$$(\text{r2}) : u_j\Big(\sum_{i:(i,j)\in A} x_{ij} - 1\Big) = 0, \quad \forall j \in H \setminus \{0\}$$

$$(\text{r3}) : (u_i - 1)x_{ij} \geq 0, \quad \forall (i,j) \in A, i \neq 0$$

$$(\text{r4}) : (n + 1 - u_i)x_{ij} \geq 0, \quad \forall (i,j) \in A, i \neq 0$$

$$(\text{r5}) : (u_i - 1)(1 - x_{ij} - x_{ji}) \geq 0, \quad \forall (i,j) \in A, i \neq 0$$

$$(\text{r6}) : (n + 1 - u_i)(1 - x_{ij} - x_{ji}) \geq 0, \quad \forall (i,j) \in A, i \neq 0$$

$$(\text{r7}) : (u_i - 2)(1 - x_{0i} - x_{i,n+1}) \geq 0, \quad \forall i \in H \setminus \{0, n+1\}$$

$$(\text{r8}) : (n - 1 - u_i)(1 - x_{0i} - x_{i,n+1}) \geq 0, \quad \forall i \in H \setminus \{0, n+1\}$$

Linearization:

Let $y_{ij} = u_i x_{ij}, \forall (i,j) \in A, i \neq 0$ and $z_{ij} = u_j x_{ij}, \forall (i,j) \in A, i \neq 0$. Constraints (3.10)–(3.12) become:

$$\begin{cases} z_{ij} = y_{ij} + x_{ij}, & \forall (i,j) \in A, i \neq 0 \\ u_j x_{0j} = x_{0j}, & \forall j \in H \setminus \{0\} \\ u_j x_{j,n+1} = n x_{j,n+1}, & \forall j \in H \setminus \{n+1\} \end{cases} \tag{3.14}$$

We then linearize constraints (r1)–(r8) one by one by leveraging the equations in (3.14).

r1:

$\forall i \in H \setminus \{n+1\}$:

$$
\begin{aligned}
& u_i\big(\sum_{j:(i,j)\in A} x_{ij} - 1\big) = 0 \\
\iff & u_i x_{i1} + u_i x_{i2} + \cdots + u_i x_{i,n+1} - u_i = 0 \\
\iff & \sum_{j:(i,j)\in A, j\neq n+1} y_{ij} + n x_{i,n+1} - u_i = 0 \\
\implies & \text{constraint (SD-1)}
\end{aligned}
$$

r2:

$\forall j \in H \setminus \{0\}$:

$$
\begin{aligned}
& u_j\big(\sum_{i:(i,j)\in A} x_{ij} - 1\big) = 0 \\
\iff & \sum_{i:(i,j)\in A} (x_{ij} + y_{ij}) - u_j = 0 \\
\iff & \sum_{i:(i,j)\in A} y_{ij} + 1 - u_j = 0 \\
\implies & \text{constraint (SD-2)}
\end{aligned}
$$

r3:

$\forall (i,j) \in A, i \neq 0$:

$$
\begin{aligned}
& (u_i - 1)x_{ij} \geq 0 \\
\iff & x_{ij} \leq y_{ij} \\
\implies & \text{constraint (SD-3) lower bound}
\end{aligned}
$$

r4:

$\forall (i,j) \in A, i \neq 0$:

$$(n + 1 - u_i)x_{ij} \geq 0$$

$$\iff (n+1)x_{ij} - y_{ij} \geq 0$$

$$\iff y_{ij} \leq (n+1)x_{ij}$$

$$\implies \text{constraint (SD-3) upper bound}$$

r5:

$\forall (i,j) \in A, i \neq 0$:

$$(u_i - 1)(1 - x_{ij} - x_{ji}) \geq 0$$

$$\iff u_i - y_{ij} - z_{ij} - 1 + x_{ij} + x_{ji} \geq 0$$

$$\iff y_{ij} + y_{ji} \leq u_i - (1 - x_{ij})$$

$$\implies \text{constraint (SD-4) upper bound}$$

r6:

$\forall (i,j) \in A, i \neq 0$:

$$(n + 1 - u_i)(1 - x_{ij} - x_{ji}) \geq 0$$

$$\iff (n+1) - (n+1)x_{ij} - (n+1)x_{ji} - u_i + y_{ij} + x_{ji} + y_{ji} \geq 0$$

$$y_{ij} + y_{ji} \geq u_i + (n+1)(x_{ij} - 1) + nx_{ji}$$

$$\implies \text{constraint (SD-4) lower bound}$$

r7:

$\forall i \in H \setminus \{0, n+1\}$:

$$(u_i - 2)(1 - x_{0i} - x_{i,n+1}) \geq 0$$

$$\iff u_i \geq 2 + x_{0i} + nx_{i,n+1} - 2x_{0i} - 2x_{i,n+1}$$

$$\iff u_i \geq 1 + (1 - x_{0i}) + (n-2)x_{i,n+1}$$

$$\implies \text{constraint (SD-5) lower bound}$$

r8:

$\forall i \in H \setminus \{0, n+1\}$:

$$
\begin{aligned}
& (n-1-u_i)(1-x_{0i}-x_{i,n+1}) \geq 0 \\
\iff & n-1-(n-1)x_{0i}-(n-1)x_{i,n+1}-u_i+x_{0i}+nx_{i,n+1} \\
\iff & u_i \leq n-(n-2)x_{0i}-(1-x_{i,n+1}) \\
\implies & \text{constraint (SD-5) upper bound}
\end{aligned}
$$

□

Table 3.1 presents the original Nested-VRP model and three new models with different choices of linearization and strengthening techniques. For simplicity, the original Nested-VRP model is referred to as MIQCP and the linearized Nested-VRP model is referred to as MILP. Similarly, we refer to the MILP as MILP+DL or MILP+SD when the drone's MTZ subtour elimination constraints are replaced by constraints (DL-1)–(DL-2) or (SD-1)–(SD-5). In addition, Table 3.1 compares the size of the models in terms of the numbers of variables and constraints. In spite of the fact that the linearization process inevitably introduces $O(n^2 + 2n)$ additional variables and $O(3n^2 + 12n)$ additional constraints, it does convert the MIQCP model into one for which a large number of solution algorithms have been developed. The formulation strengthening techniques, as stated in Proposition 3.4.2 and Proposition 3.4.3, do not bring more variables into either the MILP+DL or the MILP+SD model. However, with the use of the RLT, we introduce $O(n^2 + 3n)$ additional constraints into the MILP+SD model as compared to the MILP model.

3.5 Heuristic Methodology via Neighborhood Search

The Nested-VRP problem, being an extension of the Traveling Salesman Problem, is difficult from a theoretical perspective. In particular, as the size of the problem increases, the complexity of the MIP model and limited computation time do not allow for an exact

Table 3.1: Compare MIQCP, MILP, MILP+DL, and MILP+SD formulations and corresponding size as a function of number of locations n.

	MIQCP	MILP	MILP+DL	MILP+SD
Formulation	(2.1)–(2.34)	(2.1)–(2.34) with (2.15) substituted by $(15^\star)$ with (2.21) – (2.22) substituted by $(21^\star) - (22^\star)$ with (2.28) substituted by $(28^\star)$	(2.1)–(2.34) with (2.15) substituted by $(15^\star)$ with (2.21) – (2.22) substituted by $(21^\star) - (22^\star)$ with (2.28) substituted by $(28^\star)$ with (2.4) – (2.5) substituted by (DL-1) – (DL-2)	(2.1)–(2.34) with (2.15) substituted by $(15^\star)$ with (2.21) – (2.22) substituted by $(21^\star) - (22^\star)$ with (2.28) substituted by $(28^\star)$ with (2.4) – (2.5) substituted by (SD-1) – (SD-5)
Binary variables	$O(3n^2 + 9n)$	$O(4n^2 + 11n)$	$O(4n^2 + 11n)$	$O(4n^2 + 11n)$
Continuous variables	$O(5n)$	$O(7n)$	$O(7n)$	$O(7n)$
Constraints	$O(10n^2 + 35n)$	$O(13n^2 + 47n)$	$O(13n^2 + 47n)$	$O(14n^2 + 50n)$

solution. This motivates us to develop a heuristic methodology that is expected to produce good solutions given a computational time budget.

The neighborhood search (NS) framework, first introduced in [37], has been demonstrated as a powerful tool for solving difficult combinatorial optimization problems. A generic NS starts with an initial feasible solution to the problem of interest. Each NS iteration involves destructing the current best-known solution and reconstructing a better candidate by modifying local decisions. The best solution obtained by the time of termination is recorded as the final result.

As we have discussed, the optimal Nested-VRP solution can be characterized as the time-minimizing collection of non-overlapping nested units each of which obeys the battery capacity limit and the synchronization constraint. Given a Nested-VRP instance, we denote its optimal solution as I_{opt} which consists of the optimal configuration of nested units. In our NS framework, we start with a feasible solution I of the same instance. Initially, the feasible solution I forms the nested units in a suboptimal way (i.e., at least one nested unit does not match that in the optimal solution sol). Next, I can be further improved by iteratively destructing mismatched nested units and regrouping them with their neighboring units. When the search process terminates, the heuristic returns the best known solution I^*. Therefore, the key to success of an NS heuristic approach to solving the nested-VRP problem boils down to: (i) Finding a good initial feasible solution that contains the least possible mismatched nested units in comparison to the true (unknown) optimal solution as a starting point. (ii) Identifying the set of undesirable nested units that are more likely mismatched compared to the (unknown) optimal solution. (iii) An effective destruction and reconstruction process to fix the local mismatches.

We summarize the overall NS heuristic in Figure 3.3. The initialization, destruction, reconstruction, and termination components are further discussed in section 3.5, 3.5, 8, and 2, respectively. In addition, a formal description of the NS heuristic is stated at the end of this section.

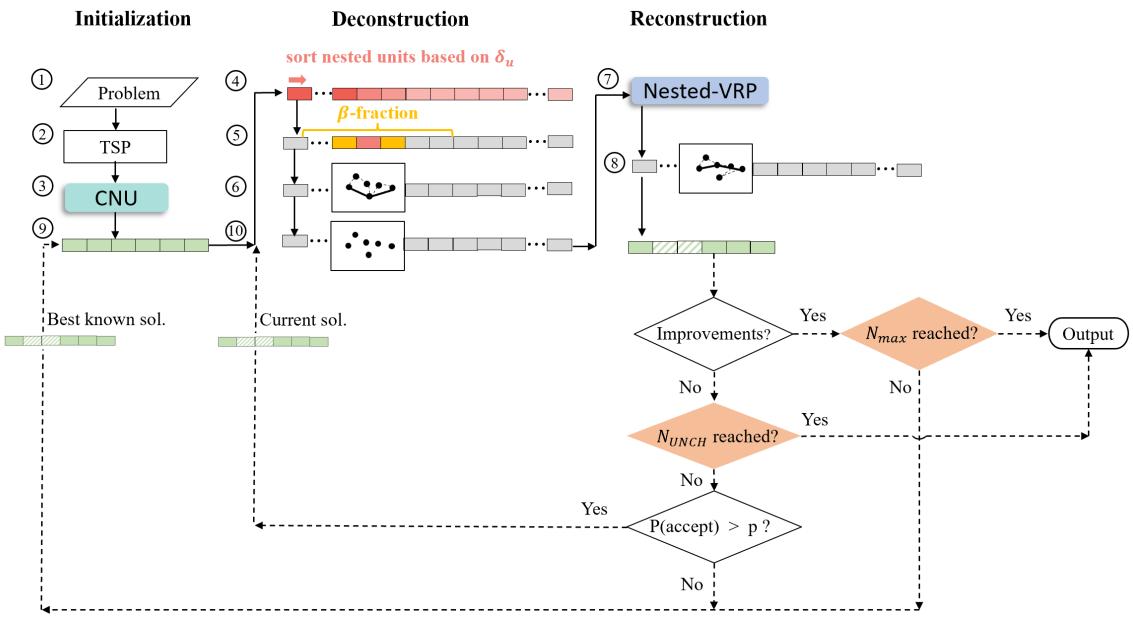


Figure 3.3: Flow chart for the NS heuristic, including Initialization, Destruction, and Reconstruction phases.

Initialization Suppose that the order to visit all locations (i.e., the drone route) is given. By Theorem 3.2.1, the full set of Nested-VRP solutions can be determined by further solving the CNU problem (3.1). Therefore, finding a good initial drone route is critical in obtaining a feasible Nested-VRP solution.

We expect that the initial feasible solution has as few mismatched nested units compared to the unknown optimal solution. Empirically, we have observed from the solutions of small-size problems (whose exact optimal solutions are actually obtainable) that even if the TSP order is not always guaranteed to be the best drone route, it at least aligns with the optimal drone routing most of the time. In fact, in the situation where the true optimal drone route is not the TSP route, we have still empirically observed that a large portion of nested units, both in the heuristic solution and the optimal solution, share the same configurations.

Thus, in the initial Nested-VRP solution, we enforce that the drone route is the same as the TSP route, i.e., the shortest-time tour to visit every location exactly once. With the known drone route, the decisions on the truck route and the assignment of battery stops along the tour can be obtained by further solving the CNU problem (see Figure 3.3, steps

1–3).

Destruction Given a feasible solution to be improved, the destruction process is to remove the nested units that are likely misaligned with those in the true optimal solution. Since the Nested-VRP problem favors a solution with a minimum makespan, each nested unit is expected to pack in as many observation tasks and be as efficient with battery energy as possible. Therefore, if one nested unit only consumes a small fraction of battery capacity and leaves a great amount of energy wasted when a new battery swap occurs, the nested unit is identified as undesirable. A nested unit u is deemed to be less desirable if the amount of wasted battery capacity, named battery slackness δ_u is relative large. Motivated by this, the destruction process involves sorting all nested units in *non-increasing* order of δ_u and randomly choosing a nested unit from the top β-fraction of the list as the bad unit at the current iteration. Intuitively, an NS heuristic parameterized by a larger β is less tolerant of inefficient battery energy usage.

With the bad nested unit identified, the NS heuristic proceeds to destroy the bad nested unit itself as well as its neighbors. Regarding the severity of the destruction process, if the impacted neighborhood is limited, there is not much freedom for the reconstruction process to identify a better choice. On the contrary, once a large portion of the initial solution is destructed, the reconstruction process is equivalent to resolving a Nested-VRP of a relatively larger size. To alleviate the brunt of computational complexity per iteration, we currently restrict the destruction process to the bad unit plus a single neighbor located either immediately before or after the bad unit. At the end of the destruction process (see Figure 3.3, steps 4–6), we obtain a set of locations that were previously covered by the bad unit and its neighbor. To complete the observation tasks at these locations, we still need to determine the drone route, truck route, and assignment of the battery swap stops for coordinating the truck and the drone. We present the destruction function in Algorithm 1.

Algorithm 1: Destruction Function

Data: I: The current feasible solution;

Result: U: bad nested units to be destroyed;

L: the set of locations that are included in set U;

$pool \leftarrow$ decompose the current Nested-VRP solution to a set of nested units ;

for *nested unit* u *in* $pool$ **do**

  compute battery slackness δ_u ;

end

$pool \leftarrow$ sort all nested units in an non-increasing order of δ_u;

$l \leftarrow$ size of the pool;

$U \leftarrow$ randomly choose a unit from the first to βl-th units from $pool$ and pick one of its neighboring units ;

$L \leftarrow$ locations that are covered by nested units in U ;

Reconstruction Nested units that have been destroyed at the end of the destruction process are in the form of free locations in the graph. In the reconstruction process, our goal is to sequence the free locations and pick a subset of the free locations as swap stops to minimize the total time for the drone to receive battery replacements, travel, and complete observation tasks associated with these free locations. This can be carried out by solving the Nested-VRP model on the free locations locally and exactly.

Even though the Nested-VRP model suffers from computational complexity as the size of the problem increases, in the reconstruction process, the number of locations that need to be solved in each iteration is relatively small. This enables us to take advantage of the Nested-VRP MIP model that produces a local operation plan with the smallest makespan (see Figure 3.3, steps 7–8). Interestingly, during the local reconstruction, the order of the free locations will be altered to explore the portion of time-saving benefits lost due to prefixing the drone route.

At the end of the reconstruction process, we obtain a new set of nested units. Compared to the grouping before being destructed, if the new sets have a smaller makespan, we accept the solution and replace the old grouping with the new one (see Figure 3.3, step 9). Otherwise, we will accept it with a probability $1/2$. Accepting a worse solution allows the heuristic to step out of a local minimum and explore for the global minimum. If the solution is rejected, the heuristic proceeds to the next iteration with the same best-known solution (Figure 3.3, step 10). After that, the destruction process picks a different bad unit and continues the search process. The reconstruction function is detailed in Algorithm 2.

Algorithm 2: Reconstruction Function

Data: I: The current feasible solution;

U: The set of bad units;

L: The set of locations that are covered by bad units;

Result: I': a candidate solution;

1 L' = Nested-VRP(L) ; `// L' is a collection of nested units obtained by solving the Nested-VRP model on locations in L`

2 $I' \leftarrow (I \backslash U) \cup L'$; `// merge good nested units with the newly formed nested units`

Termination Criteria The destruction and reconstruction process is repeated until the makespan savings between iterations become marginal. Specifically, if there is no improvement achieved for N_{UNCH} consecutive iterations, then the loop is terminated. Also, to safeguard the run-times, we restrict the total number of iterations to N_{max}, which is instance dependent.

As a reflection, the art of performing this heuristic involves starting with a good feasible solution, exploring possible nested units for potential improvements, and optimizing Nested-VRP exactly on a local scale. At a higher level, the iterations between destruction and reconstruction can be viewed as a negotiation between the drone and truck routing decisions. Given a drone route, the truck can accept part of the workload for good nested

units and reject the leftover workload required by bad nested units. In return, the drone will change its route with the hope that both parties are satisfied. The overall heuristic methodology is formally stated in Algorithm 3.

Algorithm 3: NS Heuristic Overview

Data: Nested-VRP Problem;

Result: I^*: the best known collection of nested units;

```
S = TSP(Problem);
I = CNU(S);
while Stopping condition is not satisfied do
    U, L = Destruction(I);
    L' = Nested-VRP(L);
    I' = Reconstruction(I\U, L');
    if I' shows improvement then
        I ← I';
        Update stopping criteria;
    else
        I ← I' with probability 1/2;
        Update stopping criteria;
    end
end
return I* = I
```

3.6 Computational Experiments

3.6.1 Experimental Setup

In this section, we conduct a series of experiments to investigate the performance of two different approaches: the exact approaches, such as MIQCP and its variants, and the NS

heuristic. Our goals are three-fold: (i) From a modeling perspective, we will examine and compare the strength of Nested-VRP model and its variants that apply linearization and constraint strengthening techniques; (ii) From an algorithm design standpoint, we will demonstrate that the proposed heuristic is adequate to support the drone-truck surveillance mission; (iii) From an operational standpoint, we will further examine how to achieve the most-economic solution by carefully analyzing the model parameters.

All experiments are performed on a set of benchmark instances from [10] where the authors randomly generate locations on a 2D plane following different patterns, which are labeled: uniform, single-center, and double-center. The uniform pattern consists of locations whose x and y coordinates are uniformly sampled from $\{0, \ldots, 100\}$ independently. The single-center represents a circular city in which locations are closer to the center with higher probability. The double-center pattern mimics a city with two centers that are 200 distance units away from each other. Around each center, locations are distributed in the same way as the single-center pattern. The benchmark dataset is naturally split into small cases and large cases according to the number of locations, N. The "small" set considers possible location numbers $\{5, 6, 7, 8, 9, 10\}$ while the "large" set considers numbers in the pool of $\{20, 50, 75, 100, 175, 250\}$. By default, the drone's cruising speed is 1 unit distance per unit time. By varying the truck speed among $\{1, 0.5, 0.3333\}$, while keeping the drone speed as 1, we achieve the speed ratios $\alpha = \{1, 2, 3\}$ between the two vehicles. To give practical sense to the data, we treat 1 unit distance as 100 meters and constant drone cruising speed 1 unt distance per unit time as 30 meters per second. For clarity, a scenario is referred to as a subset of data sharing the same $(\text{pattern}, N, \alpha)$ characteristics. One scenario includes 10 instances. For example, the scenario $(\text{pattern=uniform}, N = 5, \alpha = 1)$ consists of 10 instances, where each instance corresponds to a Nested-VRP in which the drone, with the same speed as the truck, observes 5 locations that are randomly generated by following a uniform pattern.

Additional information is needed for solving the Nested-VRP. First, the battery capac-

ity is set to 900 seconds by default, which is enough for the drone to complete a round trip along the diagonal lines of the largest 2D plane across all instances from the small data set. Second, a single battery swap service takes 100 seconds. Since the benchmark data does not provide any information about the observation times associated with locations, we randomly generate the observation times at each location by drawing uniformly from $[0, 250]$ seconds. In short, given a scenario characterized as $(\text{pattern}, N, \alpha)$, one of its instances is further characterized by observation times O for N locations, battery capacity T_{bl}, battery swap service time T_{s}; and this is described as $(\text{pattern}, N, \alpha, O, T_{\text{bl}}, T_{\text{s}})$. All the input data features are listed in Table 3.2.

Table 3.2: Summary of data features.

Notations	**Features**	**Values**
P	patterns	{uniform, single-center, double-center}
N	number of locations	{ 5, 6, 7, 8, 9, 10 } { 20, 50, 75, 100, 175, 250 }
T_{bl}	battery capacity	900 seconds
T_{s}	battery swap service time	100 seconds
O	observation time	*Uniform*[0, 250] seconds
α	speed ratio of truck and drone	{ 1, 2, 3 }

By varying the shapes of the location pattern, the number of locations, and the speed ratios of the two vehicles, a total of $6 \times 3 \times 3 \times 10$ computational experiments were conducted on a computer with an Intel® Xeon® CPU E5-2687W v4 3.00 GHz processor and 64.0 GB installed RAM. All the algorithms and models are coded in Python 3.7.8. The MIQCP, MILP, MILP+DL, MILP+SD models are solved via Gurobi 9.1.2. We limit Gurobi runtimes to 15 minutes.

3.6.2 Formulation Comparison Results

The first set of experiments aims at empirically assessing the impact of pure linearization process on the model performance and examine whether the linearized Nested-VRP model with the original MTZ subtour elimination constraints being strengthened per Proposition

3.4.2 or Proposition 3.4.3 would exhibit improvements in terms of runtime, number of nodes explored to achieve optimality, and tightness of the lower bound at the root node. Therefore, we solve data instances parameterized in the pattern = {uniform, singlecenter, doublecenter}, $N = 8$, α = {1, 2, 3} using the MIQCP, MILP, MILP+DL, and MILP+SD formulations presented in Table 3.1, respectively. When solving each one of the mentioned formulations, Table 3.3 presents the average runtime T_{sol} in seconds, the average number of nodes explored to achieve optimality N_{node}, and the average gap at the root node γ_0 in percentage. Moreover, we compare the relative performance of the MILP, MILP+DL, MILP+SD models to that of the MIQCP model in the second row of the table (i.e., changes with respect to the MIQCP model).

This study reveals that the pure linearization process is able to reduce runtime by 35.67 seconds on average — which is equivalent to approximately a 25.16% runtime reduction with respect to the MIQCP model. This observation can be further explained by the fact that when the Gurobi Optimizer solves the MILP model, the Optimizer explores on average 451195 fewer nodes in the branch-and-bound search process as compared to solving the MIQCP model. Compared to the MILP model, MILP+DL makes slightly further progress in both reducing the runtime and the number of nodes explored. Most fortuitously, the MILP+SD outperforms all previous models by offering 92.29 seconds runtime reduction, which is 64.88% less than that needed by the MIQCP model; and by requiring 639639 fewer node explorations, which is 84.15% less than that required by the MIQCP model. However, it is somewhat surprising that all MILP, MILP+DL, and MILP+SD models exhibit slight but not obvious improvements in tightening up the lower bound produced by the LP relaxation at the root node. These computational results clearly demonstrate that the enhancement of the Nested-VRP model by linearization and constraints tightening techniques improves the model performance and reduces runtime to produce provably optimal solutions. In addition, the observed speedup is not caused by tightening the polyhedral representation of the original Nested-VRP model at the root node. To further enhance the Nested-VRP

model, future research could shift the focus to identifying effective cutting planes according to the data structure of the model.

Given the superior performance of the MILP+SD model, we are interested in extending the computational experiments and further assessing and comparing the MILP+SD's capability in terms of closing gap during optimizing process with respect to the MIQCP model in the following sections.

Table 3.3: Performance attained by employing MIQCP, MILP, MILP+DL, and MILP+SD in solving Nested-VRP.

	MIQCP			MILP			MILP+DL			MILP+SD		
	T_{sol} (seconds)	N_{node}	γ_0 (%)	T_{sol} (seconds)	N_{node}	γ_0 (%)	T_{sol} (seconds)	N_{node}	γ_0 (%)	T_{sol} (seconds)	N_{node}	γ_0 (%)
Mean	141.79	760138.10	119.82	106.13	308942.80	118.89	99.75	296040	118.70	49.50	120498.60	118.71
Change wrt MIQCP	-	-	-	-35.67	-451195	-0.93	-42.04	-464098	-1.12	-92.29	-639639	-1.12

3.6.3 Small Dataset Results

In this section, we solve in three ways the Nested-VRP for each instance contained in the small data set via the MIQCP model, the MILP+SD model, and the NS heuristic. In Figure 3.4, we present the computational results obtained from solving instances belonging to uniform, single-center, and double-center scenarios. Detailed statistics are documented in Tables 3.4, 3.5, 3.6, respectively. Within each table, we report the results in three subgroups by differentiating $\alpha = \{1, 2, 3\}$. In particular, when solving the MIQCP and MILP+SD models, we record the average optimal mission makespan $C_{\text{MIQCP}}/C_{\text{MILP+SD}}$ for instances belonging to the same scenario. Additionally, we track the optimality gap $\gamma_{\text{MIQCP}}/\gamma_{\text{MILP+SD}}$ and runtime $T_{\text{MIQCP}}/T_{\text{MILP+SD}}$ reported from the Gurobi Optimizer. In the following experiments, the NS heuristic is parameterized by setting $\beta = 0.25$, $N_{\text{UNCH}} = 5$, and $N_{\text{max}} = 20$ as the termination criteria. Plus, the reconstruction process of the NS heuristic employs the MILP+SD model whose superior performance has been demonstrated in section 3.6.2. In Figure 3.4, we compare the performance of the NS heuristic to the exact approaches by looking at the total number of instances, $N^{\star}$, where the NS heuristic achieves a solution with lower mission makespan as compared to the best solution provided by the Gurobi

Optimizer from solving either the MIQCP or MILP+SD models, for each scenario. The average run-times T_{NS} are recorded as well.

Table 3.4: Results from solving instances from the uniform pattern in the small data set.

uniform	**MIQCP**			**MILP+SD**			**LB**	**NS Heuristic**	
N	$C_{\text{MIQCP}}(s)$	$\gamma_{\text{MIQCP}}(\%)$	$T_{\text{MIQCP}}(s)$	$C_{\text{MILP+SD}}(s)$	$\gamma_{\text{MILP+SD}}(\%)$	$T_{\text{MILP+SD}}(s)$	$\gamma_{\text{lb}}(\%)$	$N^{\star}$	$T_{\text{NS}}(s)$
$\alpha = 1$									
5	1615.5	0.0	0.4	1615.5	0.0	0.4	-0.8	10/10	0.7
6	1795.1	0.0	2.2	1795.1	0.0	2.7	-0.8	10/10	1.6
7	2313.9	0.0	26.4	2313.9	0.0	16.3	-1.2	10/10	2.0
8	2484.2	0.0	246.4	2484.2	0.0	59.8	-1.5	10/10	3.5
9	2901.8	36.6	901.7	2900.7	1.6	446	-1.4	10/10	4.1
10	3004.9	54.8	901.8	3002.5	54.4	900.4	-1.6	10/10	5.9
$\alpha = 2$									
5	1431.9	0.0	1.0	1431.9	0.0	0.5	-3.7	10/10	0.7
6	1577.6	0.0	4.5	1577.6	0.0	2.1	-2.2	10/10	1.4
7	1915.1	0.0	43.8	1915.1	0.0	11.1	-2.9	10/10	2.5
8	2305.1	0.0	374.8	2305.1	0.0	72.2	-2.8	10/10	3.5
9	2596.9	36.8	902.2	2591.8	5.1	475.4	-3.0	10/10	4.9
10	2364.1	60.2	901.9	2361.1	52.8	900.4	-2.9	10/10	7.6
$\alpha = 3$									
5	1474.8	0.0	1.1	1474.8	0.0	0.4	-4.2	10/10	0.9
6	1414.8	0.0	4.6	1414.8	0.0	0.8	-2.4	10/10	1.4
7	1880	0.0	30.5	1880	0.0	5.5	-4.1	10/10	2.3
8	2227.1	0.0	231.1	2227.1	0.0	62.8	-1.9	10/10	3.1
9	2562.5	40.4	902.0	2562.4	3.5	500.9	-3.2	9/10	4.7
10	2449	62.6	901.8	2429.1	47.0	900.3	-3.0	9/10	6.5
Summary	120/180 of instances reach optimality			145/180 instances reach optimality					

Table 3.5: Results from solving instances from the single-center pattern in the small data set.

single-center	**MIQCP**			**MILP+SD**			**LB**	**NS Heuristic**	
N	$C_{\text{MIQCP}}(s)$	$\gamma_{\text{MIQCP}}(\%)$	$T_{\text{MIQCP}}(s)$	$C_{\text{MILP+SD}}(s)$	$\gamma_{\text{MILP+SD}}(\%)$	$T_{\text{MILP+SD}}(s)$	$\gamma_{\text{lb}}(\%)$	$N^{\star}$	$T_{\text{NS}}(s)$
$\alpha = 1$									
5	1546	0.0	0.4	1546	0.0	0.3	-1.2	10/10	0.7
6	2156.7	0.0	2.0	2156.7	0.0	2.3	-1.6	10/10	1.2
7	2233.6	0.0	19.4	2233.6	0.0	19.0	-1.3	10/10	1.8
8	2562.4	0.0	194.4	2562.4	0.0	178.5	-1.3	10/10	2.9
9	2964.2	34.9	901.7	2960.3	15.0	778.5	-1.6	10/10	4.0
10	3471.4	51.0	901.8	3461.2	32.3	900.5	-2.3	9/10	5.8
$\alpha = 2$									
5	1340.6	0.0	0.9	1340.6	0.0	0.5	-3.1	10/10	1.2
6	1741.3	0.0	5.0	1741.3	0.0	1.2	-4.9	10/10	1.6
7	1917.1	0.0	32.5	1917.1	0.0	9.1	-2.3	10/10	2.3
8	2241.6	0.0	363.3	2241.6	0.0	67.3	-4.0	9/10	3.2
9	2348.9	37.7	902.4	2338.8	6.4	301.9	-3.3	8/10	3.8
10	3213.4	57.5	901.6	3191.1	1.8	406.2	-3.3	7/10	6.0
$\alpha = 3$									
5	1299.6	0.0	1.1	1299.6	0.0	0.4	-2.4	10/10	1.3
6	1484.9	0.0	4.5	1484.9	0.0	0.7	-3.8	10/10	1.3
7	1874.1	0.0	25.8	1874.1	0.0	3.0	-3.0	10/10	2.2
8	2418	0.0	268.8	2418	0.0	27.0	-2.8	9/10	2.2
9	2475.5	37.5	902.0	2468.8	1.8	104.7	-2.8	10/10	5.2
10	2713.9	61.8	901.8	2673.5	0.9	245.8	-1.6	4/10	6.6
Summary	120/180 of instances reach optimality			156/180 instances reach optimality					

First, we evaluate the performance of the MIQCP and MILP+SD exact approaches. As

Table 3.6: Results from solving instances from the double-center pattern in the small data set.

double-center	**MIQCP**			**MILP+SD**			**LB**	**NS Heuristic**	
N	$C_{\text{MIQCP}}(s)$	$\gamma_{\text{MIQCP}}(\%)$	$T_{\text{MIQCP}}(s)$	$C_{\text{MILP+SD}}(s)$	$\gamma_{\text{MILP+SD}}(\%)$	$T_{\text{MILP+SD}}(s)$	$\gamma_{\text{lb}}(\%)$	$N^{\star}$	$T_{\text{NS}}(s)$
$\alpha = 1$									
5	2645.8	0.0	0.4	2645.8	0.0	0.2	-1.0	10/10	0.8
6	2889.1	0.0	1.2	2889.1	0.0	0.9	-0.6	10/10	1.1
7	3254	0.0	9.8	3254	0.0	6.2	-0.8	10/10	1.9
8	3687.1	0.0	78.0	3687.1	0.0	41.3	-1.2	9/10	2.6
9	3568.4	15.5	746.6	3567.3	5.0	394.9	-0.9	10/10	4.7
10	4179.2	31.4	901.6	4176.4	7.6	900.5	-1.4	10/10	5.0
$\alpha = 2$									
5	1855.7	0.0	0.5	1855.7	0.0	0.2	-3.9	10/10	1.1
6	2429.1	0.0	3.5	2429.1	0.0	0.3	-4.8	10/10	1.4
7	2354.9	0.0	23.4	2354.9	0.0	0.9	-3.2	10/10	2.0
8	3027.8	0.0	372.5	3027.8	0.0	6.7	-4.5	10/10	3.0
9	3296.2	31.2	901.8	3289.1	0.0	73.6	-4.5	10/10	4.5
10	3362.6	50.5	901.6	3353.2	0.0	417.9	-4.6	8/10	6.4
$\alpha = 3$									
5	1612.6	0.0	0.4	1612.6	0.0	0.1	-2.6	9/10	1.0
6	1711.9	0.0	1.8	1711.9	0.0	0.3	-2.7	10/10	1.4
7	2350.4	0.0	20.8	2350.4	0.0	0.6	-2.2	9/10	2.2
8	2429.4	0.0	142.7	2429.4	0.0	2.6	-2.3	10/10	3.3
9	2806.4	35.7	901.8	2794.9	0.0	23.3	-3.0	9/10	4.8
10	3273.3	53.8	902.3	3251.7	0.0	93.9	-2.6	6/10	6.3
Summary	123/180 of instances reach optimality			173/180 instances reach optimality					

a highlight, the MILP+SD model outperforms the MIQCP model in that it can provably solve 474/540 instances to optimality, whereas the MIQCP can only certify the optimality for 363/540 instances given the 15-minute cutoff time. In particular, both exact approaches achieve a 0 gap for all data instances with no more than 8 locations. When the number of locations n is increased to 9, the MIQCP model fails to produce optimal solutions for any instance and achieves an average optimality gap at 34.03%. As a contrast, the MILP+SD model is able to aggressively close the gap to 4.30% on average. Moreover, when $N = 10$, the MIQCP model produces solutions with an average optimality gap of 53.73% which is more than double the gap obtained by the MILP+SD model, that is 21.87%. The loss of optimality due to the increased complexity of the problem has also been observed by [22], who solved an MIP model sharing similar features. These observations align with those found in 3.6.2 and further confirm that the MILP+SD model tends to solve small-sized instances to optimality significantly faster than the original MIQCP model by leveraging the linearization scheme and RLT. More precisely, the MILP+SD model achieves aggressive speedups by drastically reducing the number of nodes explored during the branch-and-

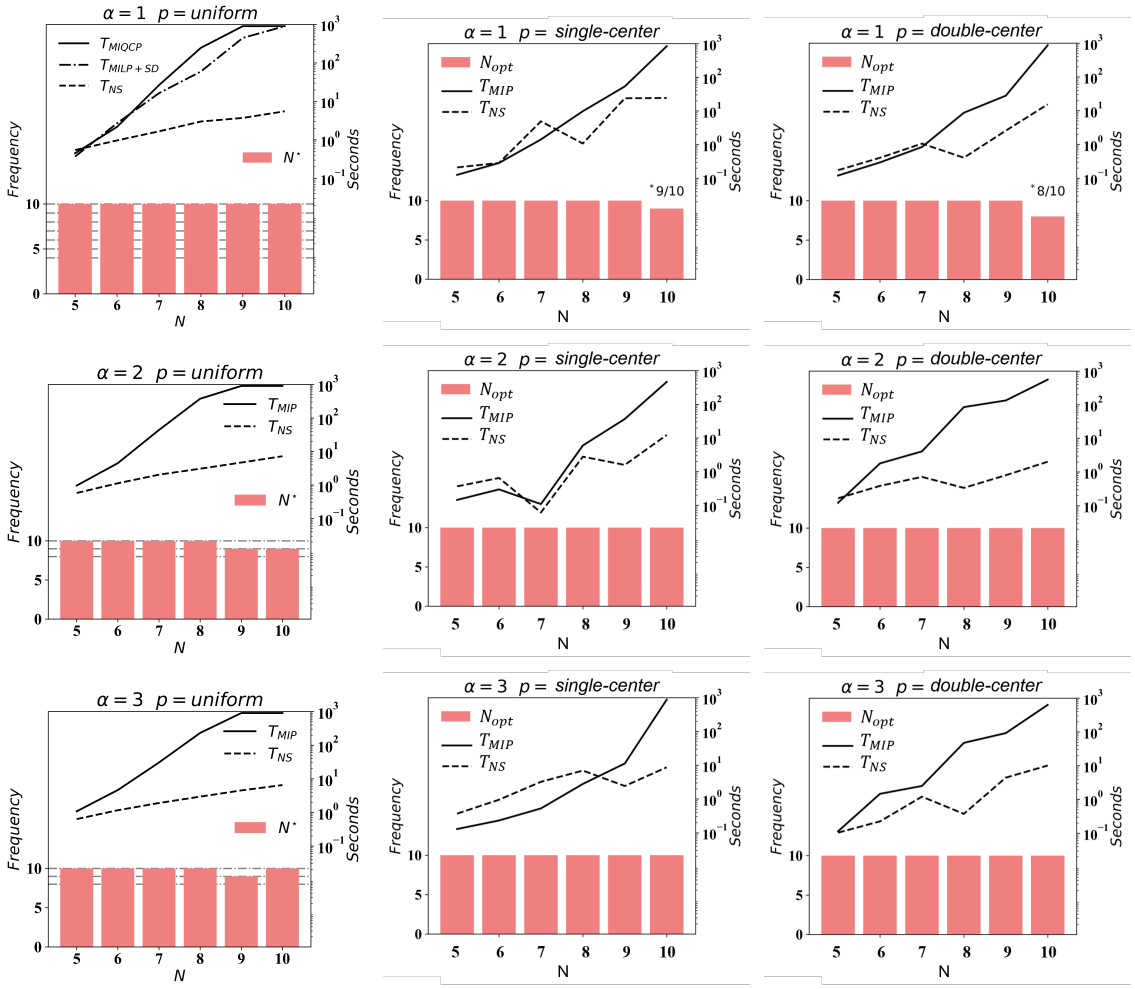


Figure 3.4: Frequency of optimal solutions found and comparison of run-times of the MIQCP and MILP+SD exact approaches as well as the NS heuristic. In a particular row, we consider the uniform, single-center, and double-center pattern shapes. In the vertical direction, we arrange the figures by increasing the speed ratio of the two vehicles from $\alpha = 1$ to $\alpha = 3$. In each subfigure, a red column corresponds to a scenario characterized by $(\text{pattern}, N, \alpha)$. Each scenario is based on the 10 instances provided by the benchmark dataset. Recall that for each scenario, $N^\star$ counts the total number of instances where the NS heuristic finds a better solution than the exact approach. For example, the NS heuristic does better in 9 out of 10 instances in scenario $(\text{single-center}, N = 10, \alpha = 1)$. In addition, the solid (dashed) line depicts the trend of computational time growth of the MIP approach (NS heuristic) as N increases.

bound process.

Regarding the computation efficiency of the proposed exact approaches, we further compare our original MIQCP model without performing the linearization and constraints strengthening scheme to the TDTL model introduced by [22]. Recall that the Nested-VRP

considers a more-complex operational scenario where each location associates with a non-zero observation time. In addition, the truck and drone are allowed to travel simultaneously from one location to another and complete battery swaps in transit. We observe that the MIQCP model is able to solve data instances (single-center, $N = 10$, $\alpha = 3$) and obtains a solution within a 62.6% average optimality gap in around 900 seconds. However, the TDTL model provides a sub-optimal solution with a 345.2% optimality gap after spending 1322.2 seconds in solving the same data instances. The same reasoning can be applied to data instances (pattern = {single-center, double-center}, $N = 10$, α = {1, 2, 3}). The observed superior performance of the MIQCP model is a byproduct of Theorem 1. From the above observations, we can conclude that the proposed MIQCP model as well as its improved variants have good performance when the problem size is relatively small.

Second, we investigate the relationship between vehicle speed ratios α and the mission makespan C_{MIP}. The purpose is to provide practical guidelines to pair a drone and a truck, with different speed settings, to achieve more operational efficiency. We do so by comparing the average percentage time savings (APTS) due to the change of vehicle speed ratio from α_1 to α_2 for a specific geometrical pattern p of interest. Given the pattern of interest $p \in \{\text{uniform, single-center, double-center}\}$, vehicle speed ratios of interest $\alpha_1, \alpha_2 \in \{1, 2, 3\}$, and possible number of locations $n \in \{5, 6, 7, 8, 9, 10\}$, let $C_{\text{MILP+SD}}(p, n, \alpha_k), k \in \{1, 2\}$ denote the average mission makespan of scenario (pattern $= p, N = n, \alpha = \alpha_k$). Then, we can define $\text{APTS}(p, \alpha_1, \alpha_2)$ as follows.

$$\text{APTS}(p, \alpha_1, \alpha_2) \equiv \sum_{n \in \{5,6,7,8,9,10\}} \frac{C_{\text{MILP+SD}}(p, n, \alpha_2) - C_{\text{MILP+SD}}(p, n, \alpha_1)}{6\, C_{\text{MILP+SD}}(p, n, \alpha_1)}$$

A more-negative $\text{APTS}(p, \alpha_1, \alpha_2)$ indicates that more time savings can be achieved by changing the speed ratio from α_1 to α_2 while the pattern p remains the same. Take the set of data from the uniform pattern as an example, in which case $\text{APTS}(\text{uniform}, 1, 2) = -13.32\%$ and $\text{APTS}(\text{uniform}, 1, 3) = -14.96\%$. In other words, increasing the vehicle speed ratio from 1 to 2 only provides additional time savings by 1.57%. However, in the

double-center scenario, increasing the vehicle speed ratio from 1 to 2 further reduces the Nested-VRP mission makespan by $|(-30.91\%)-(-19.80\%)| = 11.17\%$. This observation has significant implications for how to invest in a truck-drone team to survey the local region. A drone model with a higher maximum cruising speed is typically more expensive than a standard one. As a result, understanding how the observation tasks are distributed geometrically in the local region and what is the appropriate, but not necessarily maximum, speed ratio between the two vehicles can achieve satisfactory operation efficiency within budget.

Third, with regard to the performance of the NS heuristic, we observe that the NS heuristic is able to produce better solutions compared to the exact approach for **514** out of **540** total small data instances. This is supported by the evidence in Figure 3.4. A closer look at instances where the NS heuristic fails to compete over the exact technique reveals that the NS heuristic has difficulty in searching for the optimal Nested-VRP solution when dealing with complex geometry, double-center pattern. However, there is no conclusive results on how the total number of locations and the vehicle speed ratio affect the performance of the proposed NS heuristic.

In terms of computation efficiency, the NS heuristic can obtain Nested-VRP solutions of high quality with significantly less computation time than the exact technique with a 15-minute cutoff time for 514 out of 540 small data instances. In particular, the run times for the NS heuristic to solve a small data instance is on average 3.04 seconds with a standard deviation of 1.90 seconds. In the next section, we further examine how the NS heuristic performs when dealing with problems of larger size.

3.6.4 Large Dataset Results

For solving larger-scale problems, we first perform the MIQCP and the MILP+SD exact approaches via Gurobi Optimizer with a 15-minute cutoff time. In particular, we warm start the exact approach using the initial feasible Nested-VRP solution provided at the end

of initialization phase of the NS heuristic as discussed in section 3.5. However, these two exact approaches fail to provide satisfactory solutions. Therefore, we apply the NS heuristic to solve Nested-VRP involving large data sets. In the following experiments, the heuristic is parameterized by setting $\beta = 0.25$, $N_{\text{UNCH}} = 5$ and $N_{\text{max}} = 50$ as the termination criteria. By default, the reconstruction process of the NS heuristic employs the MILP+SD formulation. To evaluate the quality of the solutions, we leverage the lower bound value introduced in section 3.3.2.

Recall that the large data set considers locations of sizes $\{20, 50, 75, 100, 175, 250\}$ from uniform, single-center, and double-center geometrical patterns. Additionally, the speed ratio between the drone and the truck varies from $\{1, 2, 3\}$. Each scenario, characterized by $(\text{pattern}, N, \alpha)$, includes 10 instances.

Tables 3.7, 3.8, and 3.9 provide a comparison of the computational performance resulting from the application of various approaches on data from uniform, single-center, double-center patterns, respectively. Within a table, each row summarizes the statistics of interest per scenario. For each scenario, the statistics considered are:

- The average mission makespan of the solutions obtained via Gurobi with warm-start (15-minute cutoff time) using the MIQCP model $C^{\text{MIQCP}}_{\text{15mins}}$ or the MILP+SD model $C^{\text{MILP+SD}}_{\text{15mins}}$; via solving the CNU problem C_{CNU} (initialization of the NS Heuristic); and the NS heuristic C_{NS}.
- The estimate of the lower bound on mission makespan C_{lb}.
- The relative optimality gaps $\gamma^{\text{MIQCP}}_{\text{15mins}}$, $\gamma^{\text{MILP+SD}}_{\text{15mins}}$, γ_{CNU}, γ_{NS} with respect to the estimate of the lower bound.
- For the NS heuristic, we also report the average run-times T_{NS}, number of iterations $\#_{\text{iter}}$, and number of recharge stops N_{s}.

First, we examine the performance of the two exact approaches in solving large data instances. Recall that for solving each large instance, we warm-start the exact approaches

Table 3.7: Results from solving instances from a uniform pattern in the large data set.

uniform	**MIQCP**	**MILP+SD**	**NS Heuristic**					**LB**	**Gap Comparison (%)**			
N	$C^{\text{MIQCP}}_{\text{15mins}}$	$C^{\text{MILP+SD}}_{\text{15mins}}$	$T_{\text{NS}}\ (s)$	$\#_{\text{iter}}$	N_{s}	C_{CNU}	C_{NS}	C_{lb}	γ_{CNU}	$\gamma^{\text{MIQCP}}_{\text{15mins}}$	$\gamma^{\text{MILP+SD}}_{\text{15mins}}$	γ_{NS}
$\alpha = 1$												
20	5630.7	5630.7	4.9	3.8	17.1	5630.7	5242.7	5029.5	12.05	12.05	12.05	4.18
50	13505.8	13505.8	13.3	6.2	30.3	13505.8	12976.1	12637.2	6.88	6.88	6.88	2.66
75	20597.7	20597.7	25.2	10.6	50.2	20597.7	20040.7	18995.5	8.43	8.43	8.43	5.50
100	26979.9	26979.9	40.1	14.8	64.7	26979.9	26430.4	24841.1	8.63	8.63	8.63	6.42
175	46891.6	46891.6	99.8	24.0	104.6	46891.6	46249.2	43882.2	6.87	6.87	6.87	5.40
250	66213	66213	186.9	34.1	143.7	66213.0	65514.7	62437.5	6.06	6.06	6.06	4.94
$\alpha = 2$												
20	5265.7	5265.7	5.3	2.9	11.9	5265.7	5094.0	4953.8	6.42	6.42	6.42	2.91
50	13123.6	13123.6	17.4	6.7	27.4	13123.6	12557.4	12399.0	5.94	5.94	5.94	1.33
75	19915	19915	27.8	9.7	44.1	19915.0	19254.2	18885.7	5.45	5.45	5.45	1.95
100	26300.2	26300.2	47.8	11.4	57.2	26300.2	25618.8	24957.4	5.38	5.38	5.38	2.65
175	45664.3	45664.3	107.1	22.7	92.4	45664.3	44951.3	43731.9	4.42	4.42	4.42	2.79
250	64342.6	64342.6	204.1	34.3	134.3	64342.6	63605.2	61824.4	4.08	4.08	4.08	2.88
$\alpha = 3$												
20	5194.5	5194.5	12.5	3.3	9.5	5194.5	5036.0	4885.1	6.29	6.29	6.29	3.13
50	13427.2	13427.2	22.1	6.8	29.7	13427.2	12814.7	12680.9	5.93	5.93	5.93	1.05
75	18847.2	18847.2	42.5	10.9	41.8	18847.2	18213.5	18001.5	4.71	4.71	4.71	1.17
100	26282.7	26282.7	51.6	14.3	57.6	26282.7	25589.0	25186.9	4.36	4.36	4.36	1.60
175	45199.6	45199.6	140.3	27.6	94.8	45199.6	44466.9	43584.7	3.71	3.71	3.71	2.02
250	64445.2	64445.2	229.9	35.7	136.7	64445.2	63678.9	62349.8	3.36	3.36	3.36	2.13
Summary	The NS heuristic solves 146/180 instances to within 5% of the C_{lb}											
	The NS heuristic solves 180/180 instances to within 10% of the C_{lb}											

Table 3.8: Results from solving instances from a single-center pattern in the large data set.

single-center	**MIQCP**	**MILP+SD**	**NS Heuristic**					**LB**	**Gap Comparison (%)**			
N	$C^{\text{MIQCP}}_{\text{15mins}}$	$C^{\text{MILP+SD}}_{\text{15mins}}$	$T_{\text{NS}}\ (s)$	$\#_{\text{iter}}$	N_{s}	C_{CNU}	C_{NS}	C_{lb}	γ_{CNU}	$\gamma^{\text{MIQCP}}_{\text{15mins}}$	$\gamma^{\text{MILP+SD}}_{\text{15mins}}$	γ_{NS}
$\alpha = 1$												
20	6107.7	6053.8	2.3	4.5	21.2	6107.7	5730.6	5594.5	9.18	9.18	8.26*	2.43
50	14435.1	14435.1	15.7	11.8	43.1	14435.1	13938.7	13633.9	5.87	5.87	5.87	2.23
75	22012.6	22012.6	24.0	14.1	65.4	22012.6	21431.1	20665.6	6.51	6.51	6.51	3.69
100	29020.6	29020.6	30.9	16.8	75.7	29020.6	28402.6	27087.9	7.13	7.13	7.13	4.85
175	47608.6	47608.6	67.0	21.2	116.5	47608.6	46952.1	43953.1	8.31	8.31	8.31	6.81
250	67988.1	67988.1	311.7	35.7	160.8	67988.1	67288.9	62497.5	8.79	8.79	8.79	7.67
$\alpha = 2$												
20	5923.1	5923.1	7.3	3.6	11.1	5923.1	5814.8	5670.4	4.52	4.52	4.52	2.63
50	13247.3	13247.3	19.6	7.8	30.1	13247.3	12746.3	12586.6	5.26	5.26	5.26	1.29
75	19940.3	19940.3	30.5	9.8	44.1	19940.3	19328.5	19035.8	4.75	4.75	4.75	1.54
100	26604.5	26604.5	47.1	15.8	58.8	26604.5	26037.7	25333.0	5.01	5.01	5.01	2.77
175	45876	45876	115.3	29.9	103.3	45876.0	45166.6	43461.6	5.55	5.55	5.55	3.92
250	65259.4	65259.4	167.5	31.8	133.6	65259.4	64567.7	61760.9	5.66	5.66	5.66	4.54
$\alpha = 3$												
20	5642.4	5642.4	7.3	1.0	7.9	5642.4	5498.8	5352.5	5.44	5.44	5.44	2.69
50	13334.9	13334.9	19.0	7.8	30.0	13334.9	13092.7	12803.1	4.17	4.17	4.17	2.21
75	20214	20214	28.6	8.8	46.4	20214.0	19552.3	19316.6	4.67	4.67	4.67	1.21
100	26984.7	26984.7	45.2	14.5	55.3	26984.7	26310.0	25924.0	4.09	4.09	4.09	1.49
175	46005.2	46005.2	88.3	20.6	99.1	46005.2	45303.4	44111.9	4.29	4.29	4.29	2.69
250	65213	65213	273.2	34.2	135.9	65213.0	64472.3	62415.8	4.48	4.48	4.48	3.30
Summary	The NS heuristic solves 157/180 instances to within 5% of the C_{lb}											
	The NS heuristic solves 180/180 instances to within 10% of the C_{lb}											

by using the initial feasible solution obtained by the CNU model. As can be observed in columns $\gamma^{\text{MIQCP}}_{\text{15mins}}$, $\gamma^{\text{MILP+SD}}_{\text{15mins}}$, and γ_{CNU} in each of Tables 3.7–3.9, both exact approaches expe-

Table 3.9: Results from solving instances from a double-center pattern in the large data set.

double-center	**MIQCP**	**MILP+SD**	**NS Heuristic**					**LB**	**Gap Comparison (%)**			
N	$C_{15\text{mins}}^{\text{MIQCP}}$	$C_{15\text{mins}}^{\text{MILP+SD}}$	$T_{\text{NS}}\ (s)$	$\#_{\text{iter}}$	N_{s}	C_{CNU}	C_{NS}	C_{lb}	γ_{CNU}	$\gamma_{15\text{mins}}^{\text{MIQCP}}$	$\gamma_{15\text{mins}}^{\text{MILP+SD}}$	γ_{NS}
$\alpha = 1$												
20	7424.6	7424.6	2.4	3.5	21.9	7424.6	7089.6	6933.5	7.13	7.13	7.13	2.25
50	15512.9	15512.9	10.4	8.6	47.0	15512.9	15060.0	14913.1	4.06	4.06	4.06	1.00
75	22348	22348	23.1	14.0	68.0	22348.0	21810.7	21347.0	4.70	4.70	4.70	2.17
100	29956.4	29956.4	39.3	22.7	92.7	29956.4	29421.4	28390.7	5.51	5.51	5.51	3.63
175	50159.1	50159.1	100.3	29.1	121.5	50159.1	49555.0	46999.4	6.72	6.72	6.72	5.44
250	69377.8	69377.8	260.0	40.5	178.8	69377.8	68733.0	64420.0	7.70	7.70	7.70	6.69
$\alpha = 2$												
20	6405.8	6405.8	5.5	1.7	11.6	6405.8	6015.4	5891.2	9.09	9.09	9.09	2.21
50	13840.9	13840.9	17.7	5.5	28.7	13840.9	13308.6	13145.2	5.31	5.31	5.31	1.25
75	20762.9	20762.9	30.7	9.8	47.3	20762.9	20154.9	19899.4	4.35	4.35	4.35	1.28
100	26963.8	26963.8	47.4	12.9	57.0	26963.8	26323.6	25805.4	4.49	4.49	4.49	2.00
175	46667.6	46667.6	102.5	19.6	105.6	46667.6	46011.9	44461.9	4.96	4.96	4.96	3.49
250	66497.8	66497.8	297.9	36.6	146.5	66497.8	65816.6	63210.3	5.20	5.20	5.20	4.12
$\alpha = 3$												
20	5701	5587.7	6.8	3.2	9.8	5701.0	5282.0	5111.8	11.65	11.65	$9.44^{\star}$	3.38
50	13400.3	13400.3	25.9	6.9	26.8	13400.3	12863.3	12736.8	5.32	5.32	5.32	1.11
75	20255.2	20255.2	40.1	10.4	39.3	20255.2	19670.6	19512.0	3.82	3.82	3.82	0.80
100	26478.9	26478.9	52.8	14.3	54.1	26478.9	25825.0	25472.4	3.95	3.95	3.95	1.38
175	45429.1	45429.1	108.2	17.4	97.7	45429.1	44716.7	43657.1	4.06	4.06	4.06	2.43
250	65414	65414	298.4	31.6	139.3	65414.0	64697.7	62760.9	4.23	4.23	4.23	3.08
Summary	The NS heuristic solves 155/180 instances to within 5% of the C_{lb}											
	The NS heuristic solves 180/180 instances to within 10% of the C_{lb}											

rience difficulty in further reducing the mission makespan upon the initial feasible solution when the searching process is curtailed due to the limited run-time constraint. However, the MILP+SD model shows slight improvement in solving data scenarios characterized as (single-center, N = 20, α = 1) and (double-center, N = 20, α = 3) as highlighted by $\star$. These results indicate the failure in leveraging exact approaches for solving large data instances, which aligns with the conclusions in [22]. On the one hand, the difficulties in closing the gap come from the effects of the number of locations, the geometrical distribution of the locations, and the vehicles' speed ratio. We hypothesize that as the distribution of the locations becomes more skewed, the speed ratio between the two vehicles becomes more incompatible, and the size of the problem increases, so that the drone experiences more difficulties in collaborating with the truck to replace batteries. From a computational complexity perspective, when solving the Nested-VRP, finding the optimal solution requires an intensive search for the best combination of drone route, truck route, and battery swap locations. Therefore, the exact approach can only achieve a less-than-optimal solution given

the run-time limit of 15 minutes. On the other hand, when using the Gurobi optimizer and given a near-optimal feasible solution, the failure to achieve optimality could be partially explained by the slow converging behavior of the lower bound in the branch-and-bound process.

Second, we explore the advantages of applying the proposed NS heuristic. Compared to the exact approaches, the NS heuristic improves the solution quality by combining effective initialization strategy and local improvement scheme. To assess the effectiveness of the initialization strategy, when solving each Nested-VRP instance, we evaluate the relative optimality gap γ_{CNU} of the solution obtained by solving the CNU problem with respect to the lower bound value. The column γ_{CNU} suggests that solving the CNU problem produces robust feasible Nested-VRP solutions with a 5.83% average gap and a 1.91% standard deviation relative to the lower bound value C_{lb} across all large instances. We further examine the effectiveness of the pure local search process of the NS heuristic by assessing its contribution to optimality. By referring to columns γ_{CNU} and γ_{NS}, across all scenarios, the local search process contributes to a further 2.86% gap reduction on average given the initial feasible solution provided by solving the CNU problem.

It is also important to highlight that further reduction, obtained by the local search process, comes from exploring a relatively small number of nested units specified by the initial feasible solution. To see this, the number of battery swaps N_{s} is an indicator of the total number of nested units that are included in the final solution, while the number of iterations $\#_{\text{iter}}$ tells how many bad nested units are reorganized before reaching the final decision. As an example, in Table 3.8, in the scenario (single-center, $N = 250$, $\alpha = 1$), the NS heuristic takes on average 35.7 iterations to finalize the Nested-VRP solution, which consists of 160.8 nested units. Approximately 22.20% of the units are reorganized before the NS heuristic terminates. A similar analysis can be applied to other scenarios.

Third, we assess the quality of the solutions provided by the NS heuristic. The NS heuristic is able to produce solutions with objective values that are reasonably close to

the lower bound. Specifically,according to the summary of the Tables 3.7–3.9, $(146 + 151 + 155)/(180 + 180 + 180) = 84.81\%$ of instances achieve results that deviate from the lower bound solution by less than 5%. Moreover, all instances can be solved within a 10% gap with respect to the proposed lower bound value C_{lb}. It is worth noticing that the LB produces an over-optimistic estimation of the optimal value. Further tightening the LB will potentially gain more-accurate insights into the performance of the NS heuristic method. In terms of the NS's computation efficiency, the column T_{NS} in Tables 3.7–3.9 show that the time needed for NS to solve a Nested-VRP instance scales up linearly as a function of total number of locations. To summarize the above discussion, the proposed NS heuristic is effective and efficient in producing high-quality solutions to the Nested-VRP.

3.6.5 Real-world Case Study

In this section, we focus on a realistic application of the Nested-VRP to surveillance in the aftermath of the 2017 Santa Rosa Wildfire. Our goal is to assess the effectiveness and efficiency of the NS heuristic in solving practical applications. Most importantly, we will further demonstrate the robustness of the proposed heuristic by empirically investigating its convergence behavior.

After the devastating fire disaster, an insurance company decided to send out a truck and a drone to inspect and collect home damage evidence from 631 clients' properties. The information regarding all house locations is given. We set the truck speed as 5 m/s (considering the difficult ground conditions) and the drone speed as 10 m/s; thus, $\alpha = 2$. The battery capacity is set to 10 minutes. The data cleaning process includes bundling observation tasks for apartments in the same building as a single observation task at the building's location. Correspondingly, the observation time is the sum of that for each apartment. We perform the NS heuristic on this specific instance for 100 independent runs to account for the randomness in the searching process. In each run, the NS heuristic search bad nested units on the top $\beta = 0.3$ fraction list per iteration and will terminate if the program observes

no improvements for $N_{\text{UNCH}} = 5$ consecutive iterations or the total number of iterations exceeds $N_{\max} = 50$. We depict the best-known solution in Figure 3.5.

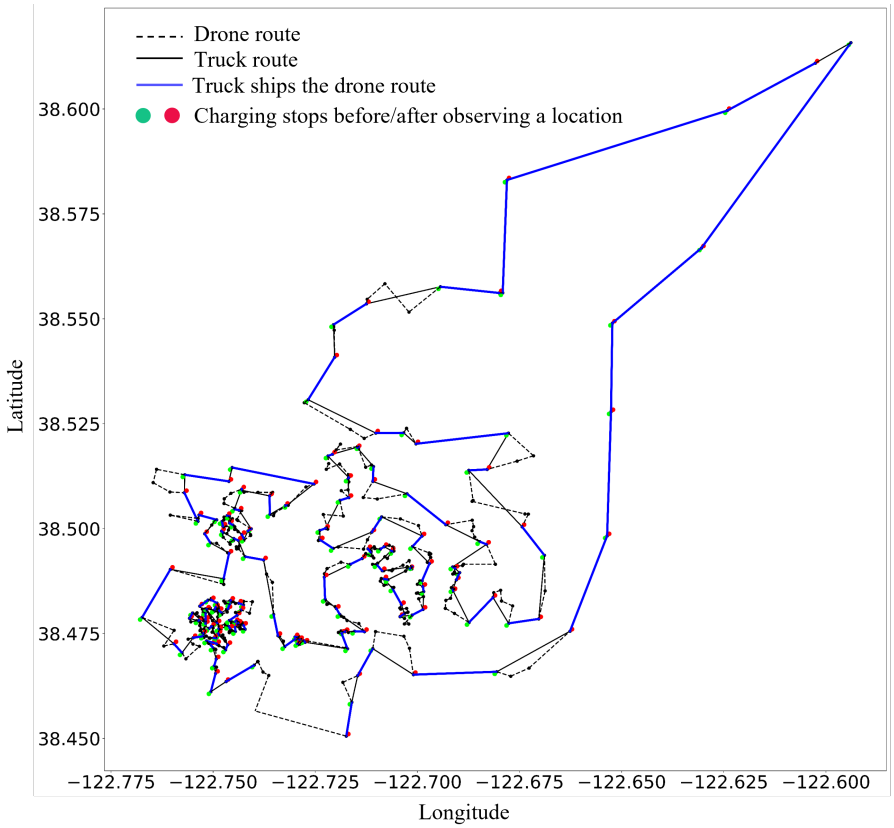


Figure 3.5: Nested-VRP solution for the practical instance. The black solid line is the truck route, while the black dashed line represents the drone route. The blue line corresponds to the truck ships the drone. Circles depict the subset of locations serving as swap stops. Green and red colors differentiate the battery swaps that happen before or after the drone observes a location.

The best-known solution suggests that at least 26 hours 51 minutes are required to complete the entire mission, which consists of vising 218 meet-up stops along the tour. We summarize the performance of the NS heuristic in Figure 3.6. In this figure, the ith blue box describes the distribution of γ_{NS} at the ith iteration across 100 runs. The average trend of γ_{NS} as the number of iterations increases is presented in the red curve. In addition, the green curve reports the average amount of time savings obtained per iteration.

Run-times: As the NS heuristic suggests, the average run-times for solving this Nested-

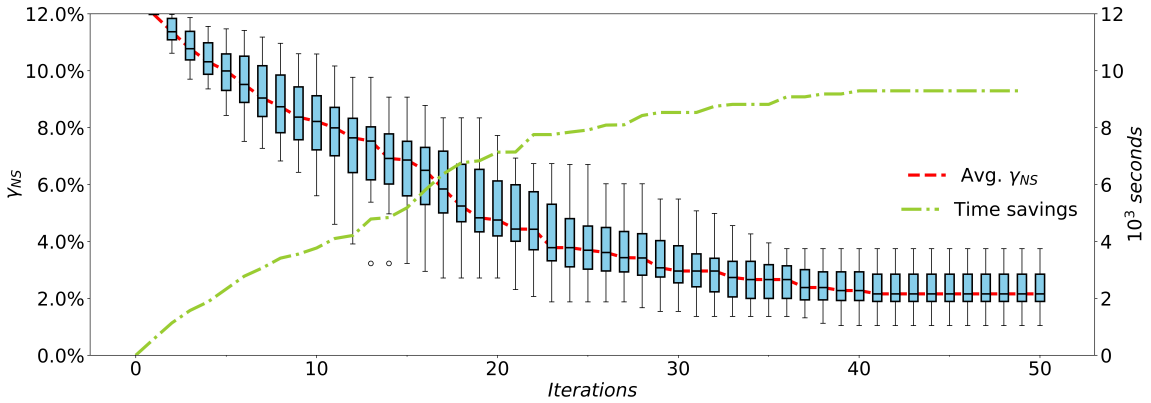


Figure 3.6: NS heuristic performance when solving the practical instance 100 times. The red line shows that the average optimality gap γ_{NS} decreases as the iteration number increases across the 100 tries. The green line shows that, compared to the mission makespan of the initial Nested-VRP solution obtained by solving CNU program, the NS heuristic finds a better solution via an effective local search scheme iteration by iteration. A better solution corresponds to a solution with a shorter mission makespan and thus corresponds to a larger time savings.

VRP instance of size 631 is 1032.6 seconds (17 minutes 13 seconds), which can be considered as efficient for planning a full-day mission.

Convergence: Referring to Figure 3.6, the red line summarizes its average trend across 100 runs. The red line suggests that the NS heuristic produces a Nested-VRP solution of smaller mission makespan as the local search process proceeds. Specifically, after solving the CNU model in the initialization stage, the NS heuristic produces a feasible Nested-VRP solution of 11.98% average optimality gap away from the estimate of lower bound value. Even though the red line has some "almost flat" segments (i.e., the 23th to 26th iterations), which indicate a marginal improvement between iterations in the destruction and reconstruction process, these stationary stages were transient and eventually progressed to a better solution. This is evidence to show NS's ability to escape local minima. Promisingly, the NS heuristic saves on average 9292.5 seconds (2 hour 35 minutes) from an initial CNU feasible solution and produces final results that are 2.16% closer to the lower bound on average.

It is worth noting that for a realistic Nested-VRP instance, implementing a drone-truck

surveillance team to complete observation tasks may occasionally provide no time-saving benefits. Referring to Figure 3.5's upper-right corner, we observe that the truck travels to multiple sites with the drone on-board. This is due to the geometrical configuration constituting the locations of the sites: relatively far away from each other and out of the drone's reach from one location to the other. The solution is almost reduced to having a truck complete the surveillance task alone by following a partial TSP tour. In this case, the drone, even though it has the advantage of high cruising speed, does not contribute to speeding up the mission. Therefore, we lose the potential time-saving benefit of hiring a drone-truck surveillance team. This observation suggests that while matching the speed ratio between the truck and drone is crucial, picking the appropriate drone model with sufficient flight duration to support the mission is also important to the overall mission performance.

The results from solving small, large, and real-world Nested-VRP instances shed light on how to use the proposed methodology to solve a broader problem involving the coordination of multiple trucks and drones to complete a surveillance mission. Consider a large area with thousands of locations to observe. One strategy is to divide the area into dense, sub-dense, sparse, and truck-only sub-areas. Locations in the truck-only sub-area are typically located far apart from one another in the same sub-area and from the rest. In this case, using a drone and truck team to perform surveillance tasks is not beneficial. Then, for each sub-area except for the truck-only sub-area, we employ a single truck or a team of a single truck and a single drone. The proposed strategy has three benefits: (i) Excluding locations in the truck-only sub-area reduces the size of the Nested-VRP; (ii) Pairing a single-truck-single-drone team with a sub-area allows the practitioner to take a finer look at the geometrical distribution of the locations within the sub-area and adjust the speed ratio of the truck and the drone so as to obtain the most time savings; (iii) Solving the Nested-VRP for each sub-area allows the Nested-VRP model to be solved in parallel, potentially resulting in significant run-time savings.

3.7 Conclusion

In part I of the thesis, the Nested-VRP problem is formally defined and formulated as a MIP program. Given the operational assumptions described in section 2.2, our model is capable of finding the best trade-off between the drone's routing plan and swap assignments along the route such that the total mission duration is minimized. In situations where the Gurobi Optimizer fails to solve the Nested-VRP exactly, we further propose an NS heuristic approach that is based on destruction and local reconstruction principles. Our extensive experiments on small, large, and realistic instances have demonstrated the proposed heuristic's ability to obtain reasonably good results while requiring substantially lower run-times compared to the MIP exact approach when the size of the problem is large.

Our empirical study has the following implications for future practitioners. (i) The geometrical distribution of the set of locations should be evaluated first to see the potential time-savings benefits that are achievable by implementing the drone-truck surveillance team. (ii) The speed of the two vehicles should be matched at the right ratio to maximize the overall time savings.

The model developed in this paper could enable a spectrum of applications in the field of aerial surveillance. Future work could include the ground traffic as a stochastic element into the model such that the ground delays will factor in the planning. Regarding the drone's flight performance, further research into how the drone's energy consumption in observation and flying mode affects routing decisions is critical for real-world application. In terms of algorithmic design, one might propose a sophisticated exact approach inspired by branch-and-bound or a more-efficient heuristic that produces near-optimal solutions with reduced run-times. From a modeling perspective, while we only consider the case with a single truck and a single drone, one possible direction could be coordinating a team of drones together with multiple trucks to accomplish the goals and assessing the benefits of introducing more agents.

Part II

Trajectory Planning under Extreme Uncertainty

CHAPTER 4

FLEXIBILITY, ROBUSTNESS, AND UNCERTAINTY

4.1 Overview

The issue of uncertainty is pervasive in decision-making problems. Typically, the desirability of an action is determined by the probability distribution of its rewards given the outcome, which may vary as new information becomes available. A course of action whose desirability is very resistant to change is deemed robust in the presence of uncertainty. In other words, a robust solution is less likely to incur unacceptable profits or losses.

Robustness is usually measured in a probabilistic sense. For example, the robustness of an aircraft landing procedure is 99% which means that the aircraft can land successfully 99% percent of the time by following that procedure. Measuring the robustness of a solution is difficult when there are no probability models to describe the sources of uncertainty completely. Furthermore, under extreme uncertainty, any probabilistic measurement may be subject to unknown disturbance, providing no meaningful insights to guide decision-making.

While the robustness deals with the uncertainty by improving the quality of solutions, another term highlights the metrics of preserving a large quantity of choices - *Flexibility*. *Flexibility*, as defined by the Oxford English Dictionary, is the ability to change or be changed easily according to the situation[38]. When making a decision, if the decision maker is certain about the situation right now as well as how it evolves in the future, he only needs to take one action in response to a series of deterministic events. However, if the decision maker lacks confidence in future changes, he may value action flexibility because by appropriately adjusting actions based on perceived information, he may be able to achieve as close to the best interest as possible. Therefore, flexibility is desirable

whenever there is an opportunity to react after new information is received or the current action will affect the quality and magnitude of the set of future actions.

In this chapter, we first review works in section 4.1.1 that use the concept of robustness and flexibility to mitigate uncertainty across multiple disciplines. This provides a justification for utilizing these two concepts in the trajectory planning problem of our particular interest. In section 4.2, we introduce the sequential decision-making framework, which provides a solid basis for further developing design and performance metrics. Next, a special case of a sequential decision-making problem—trajectory planning for autonomous vehicles—is considered in section 4.2.1. We propose a set of metrics to quantify the uncertainty level of a planned trajectory as well as a flight mission in section 4.3. Methodologies to compute these metrics are further proposed in section 4.4 and section 4.5.

4.1.1 Related Work

The concept of robustness has increasingly been recognized by system designers and decision makers in various field to cope with uncertainty, such as engineering, marketing, biology, and psychology(see e.g., [39, 40, 41, 42]). Nevertheless, there is a relatively small body of literature on the subject of flexibility. According to [43], the preservation of flexibility is a neglected factor when one is confronted with uncertainty and required to act under risk. Inadequate research is due to the inherent difficulty in defining flexibility in a way that applies to universal applications. In this review, we provide a historical perspective on the role of flexibility in various disciplines, as well as justification for leveraging the proposed metrics in solving the trajectory planning problem under uncertainty.

The concept of flexibility is crucial for a wide range of economic decisions. An early debate focused on the investment decisions that allow a company to profit from satisfying consumers' preferences, which are inherently uncertain. Hart argued in 1940 that if a production plant is flexible, it is possible to deviate from planned values some time after the production decision is made [44]. His argument, in particular, emphasized that investing

in low-cost plant that can afford a wide range of output levels may increase the expected profit, as opposed to a plant that costs the least to produce at a specific level but is extremely expensive at other levels due to the uncertainty of consumers' preferences. Hart's work was one of the first attempts to incorporate uncertainty about future preferences into the present-day decisions, according to Koompmans [45]. Koompmans studied the same topic and proposed that an economic planner may prefer to keep flexibility and defer a production choice so that he can adapt to newly observed tastes and desire trends. For Stigler [46] and Baumol [47], flexibility was defined as the rate of change of marginal cost in a scenario involving static decision-making. When the second derivative of a plant's total cost becomes smaller, the plant's flexibility increases.

In the context of environmental issues, Gersonius et al. [48] has demonstrated the economic benefits of including flexibility into the design of urban drainage infrastructure. Specifically, options that represent physical configurations of the infrastructure can offer the flexibility necessary to deal with climate change uncertainty. In management science, Graves et al. [49] investigated the design of a supply chain that incorporates process flexibility in anticipation of uncertain future product demands. Instead of building dedicated plants with sufficient capacity to serve maximum possible demand for one product, they proposed a strategy that enables plants to process multiple products. In short, manufacturing flexibility enables a company to respond to consumer demands in a timely manner without spending excessive expenditures and increasing an excessive number of resources (e.g., [50, 51, 52, 53]).

In the realm of robotics, there is a growing emphasis on flexibility. Traditional robots have a restricted functional workspace and, as a result, struggle to carry big loads and execute maneuvers with high accuracy. In research work [54], Ider and Korkmaz developed motion control algorithm that allows parallel robots to handle high-precision manipulations by exploiting the joint flexibility. Good et al. [55] also demonstrated that disregarding joint flexibility in the controller design of industrial robots results in performance degradation.

Study on the flexibility of robot arms can be further consulted in [56].

Flexibility-focused research is dispersed throughout numerous areas. The importance of incorporating the concept of flexibility into engineering design and management science has been empirically established and is evident. As mentioned above, the challenge of defining flexibility in a general sense hinders its prevalence in applications. We should recognize this fact that even within the same discipline, the meaning of the term "flexibility" varies considerably. In this thesis, we examine the topic of planning safe paths under conditions of uncertainty. We explicitly describe the trajectory flexibility of an autonomous vehicle and make use of the trajectory flexibility to enhance flight safety.

4.2 Sequential Decision-Making Structure and Notation

In this section, we introduce the sequential decision-making structure that provide basic notations and conceptual framework for organizing our thoughts about robustness and flexibility.

A sequential decision-making problem usually involve a series of activities over a time period, T which might be continuous or discrete. Of our particular interest, we consider discrete time space $\mathcal{T} = \{0, 1, 2, \cdots, T\}$. Activities can happen at time points $t \in T$.

State is a summary of the conditions of a system/entity/program. The transition from one state to the other is accomplished by taking an action intentionally. Such transition is also subject to internal uncertainty described as ζ and external uncertainty $\mathcal{W}$. Let s_t be the state of the system at the beginning of the time point t and a_t be the action taken right after observing the state s_t, we describe the transition between states and the state-action path of the system as:

$$s_t = f(s_{t-1}, a_{t-1}, \zeta_{t-1}) + \mathcal{W}_t \tag{4.1}$$

$$\{s_0, a_0, s_1, a_1, \cdots, a_t, s_{t+1}, \cdots, s_T\} \tag{4.2}$$

Where $f(\cdot)$ encodes the dynamics of the system. Given a fixed initial state s_0, the state path

of the system is determined by the sequence of actions planned. Let A_t be the set of the actions that are available to the system at time t, we can obtain in total $\mathcal{F} := A_1 \times A_2 \times \cdots \times A_{T-1}$ possible action plans which correspond to $|\mathcal{F}|$ possible state path of the system.

Typically, a system is designed to perform a certain task by exactly following the state path $\{s_0, s_1, \cdots, s_T\}$ to reach the goal state s_T = Goal. The robustness of a state path is determined by the probability of carrying out the exact path as planned in presence of uncertainty. The system's capability to alter the current planned state path to another is proportional to the number of actions available immediately at current time t and how many alternative choices (i.e., state paths) are available driving the system from current state to the goal in the long term.

4.2.1 Space and Time Quantization

We consider the trajectory planning problem for a fixed flight level. To count the number of trajectories, we first establish a discrete representation of space and time. Specifically, time is discretized into equal time steps that are Δ_t apart and space is discretized into rectangular cells of dimension $\Delta_x \times \Delta_y$ as shown in Figure 4.1. The state of a vehicle is denoted by $(x_i, y_i, h_i, v_i, t_i) \in \mathcal{R}^5$ where h_i is the heading and v_i is the speed. A motion primitive (i.e., action plans) is defined by the pairing of heading and speed changes, that is $(\delta_h, \delta_v) \in \mathcal{R}^2$. The set of all possible motion primitives is denoted by $\mathcal{A}$, where the cardinality of $\mathcal{A}$ indicates the maneuverability of a vehicle.

Next, we introduce the mapping $c : \mathcal{R}^3 \mapsto \mathcal{N}^3$ between a way point in continuous space and time and a cell defined by $c(x, y, t) = (\lfloor x \rceil, \lfloor y \rceil, \lfloor t \rceil)$ where $\lfloor \cdot \rceil$ rounds the element $\cdot$ to the nearest integer. We further define the partial trajectory, connecting cell $c(x_i, y_i, t_i)$ to $c(x_{i+1}, y_{i+1}, t_{i+1})$ in discrete space and time, as a segment ls_i. Equivalently, a trajectory l can also be described as a sequence of segments, that is $l = [ls_0, ls_1, \ldots, ls_{T-1}]$, where $[\cdot]$ is an ordered list. Depending on the resolution of the discretized time and space, a segment l_{s_i} may degenerate to a cell (i.e., two end cells are identical) or cross multiple cells

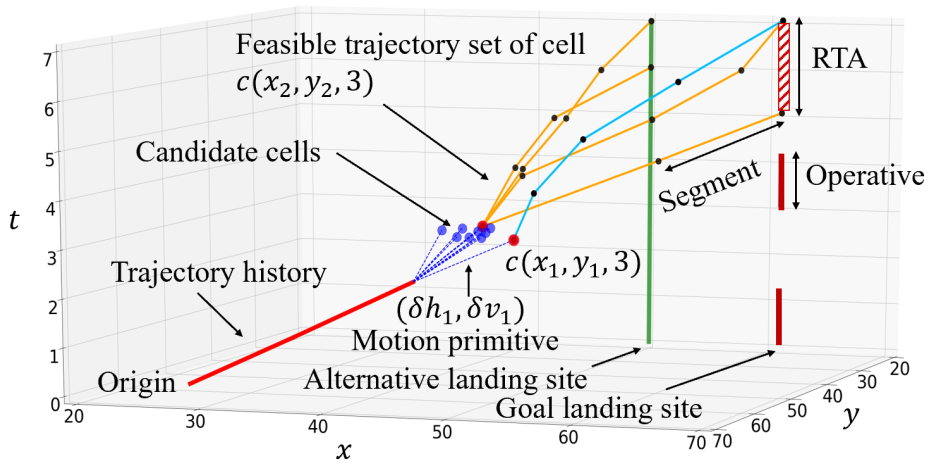


Figure 4.1: Discrete representation of space and time. Since the goal/alternative landing sites are built infrastructures, they block cells in space and time that correspond to their physical locations at all time. While the goal landing site is only open to the ownship during the RTA period, the alternative landing site is open at all times.

between two end cells. Cells that are blocked at time t are denoted by $\mathcal{B}(t) \subset \mathcal{N}^3$, where the information on the environment is refreshed at a certain rate. Further, we apply the following Assumptions 4.2.1 and 4.2.2.

Assumption 4.2.1. *We consider intruders as the only sources of uncertainty in the environment, and that a cell is completely blocked if any part of it is crossed by an intruder's trajectory.*

Assumption 4.2.2. *A straight line passing through multiple cells can be used to approximate a trajectory segment. Let $c(x_i, y_i, t_i)$ and $c(x_j, y_j, t_i + \epsilon_t)$ denote two end cells of a segment. A segment is feasible if all cells that are crossed by the segment are accessible. Specifically,*

$$c(x_i + \beta(x_j - x_i), y_i + \beta(y_j - y_i), t_i + \beta\epsilon_t) \notin \mathcal{B}(t_i),$$

where $\beta \in [0, 1]$.

Cell quantization has an impact on trajectory quality and safety. If the cell dimensions are small relative to the required safe separation, blocking the cells traversed by the trajec-

tory does not provide sufficient separation from other vehicles. In such cases, Assumption 4.2.2 is leveraged to block cells crossed by the trajectory segment as well as those within a defined radius of the trajectory segment. Typically, such a radius corresponds to the required safe separation between two vehicles. If the cell dimensions are large relative to the motion primitives, a trajectory may not be able to transit from one cell to another in a single time step. Further, large cell dimensions and time steps may result in trajectories that are too coarse.

4.2.2 Trajectory Planning Problem Definition

We consider a special case of sequential decision problem where an ownship departs from its origin (x_0, y_0) at time t_0 and arrives at its goal landing site (x_f, y_f) during the time interval $[t_{f_l}, t_{f_u}]$ that defines its Required Time of Arrival (RTA). At the time of arrival, the heading of the vehicle h_f depends on the landing pad's orientation and thus is limited to a finite set of discrete values $\{h_1, \ldots, h_m\}$, where $h_i \in [0, 2\pi]$. In addition, we require that the vehicle maintains a minimum speed $v_f = v_{min}$ at the moment of arrival. We denote by $\Omega(t)$ the set of all valid terminal states. In the following chapters, other participant in the airspace is referred to as intruder.

The ownship is capable of adapting its trajectory dynamically to account for uncertainty. A flight mission consists of T decision points set $\epsilon_t \in \mathcal{R}_+$ apart. The choice of ϵ_t depends on the frequency that the information on environment is updated. Typically, we consider $\epsilon_t > \Delta_t$. The mission completion time should not exceed the maximum flight endurance, which is determined by the ownship's fuel capacity. Regarding the vehicle dynamics, we assume that the ownship is allowed to change its heading h and speed v at a decision point by applying a pre-defined control primitive $(\delta_h, \delta_v) \in \mathcal{A}$, where $\delta_h \in [-\delta_{h_l}, \delta_{h_u}]$ and $\delta_v \in [-\delta_{v_l}, \delta_{v_u}]$. The vehicle maintains constant heading and speed until it reaches the next decision point. Therefore, at each decision point, the core decision-making problem involves optimizing the trajectory plan so as to maximize a certain objective function while

respecting all constraints imposed on trajectories.

We focus on the task of computing a feasible trajectory that directs the ownship to the goal landing site subject to the constraint that it must always be (or at least have a very high probability of being) able to reach an alternative landing site should the goal landing site become unreachable.

4.3 Metrics

The precise quantitative measurement of attributes, also known as metrics, of known objects is critical in improving a vehicle's understanding of a situation and environment. Whether metrics are used explicitly to influence the behaviors of an autonomous vehicle or simply to describe an intrinsic property of the environment's elements, they will affect the vehicle's actions and decisions, resulting in varying mission performance. In short, selecting the proper metrics is essential for an autonomous vehicle to successfully complete a mission.

One difficulty in the development of metrics is the variety of different types of metrics. In the trajectory planning problem, we classify metrics based on their relevance to the ultimate goal—completing or surviving a flight mission successfully. Accordingly, we identify basic, design, and performance metrics to be used in planning a safe trajectory for autonomous vehicles.

- Basic metrics: Metrics pertain to resource availability and the tightness of operational constraints. Basic metrics report the space-based and time-based inherent features of an event. Basic measurements can be observed more easily than other sorts of metrics. For instance, at any given point in location and time, the autonomous vehicle requires an exact description of its remaining lifetime and safe distance from other vehicles, etc.

- Design metrics: Metrics that are derived from basic metrics and user interests, and

whose quantification requires a rigorous computation and measurement method. For instance, the autonomous vehicle is interested in determining the reliability of the current planned trajectory. Such a probabilistic assessment necessitates an accurate model of the sources of uncertainty arising from dynamic obstacles and a simulation based on sampling to study the effects of uncertainty on the trajectory of interest.

- Performance metrics: At the highest level, performance metrics are responsible for integrating measurements across all functions and aligning the behavior of autonomous vehicles to fulfill mission-level objectives.

In the following, we list a set of metrics that serves the basis of the decision-making engine of the autonomous vehicle by category.

4.3.1 Basic Metrics

Definition 4.3.1 (Feasibility of a trajectory plan)**.** *A trajectory is feasible if the following constraints are satisfied:*

(1) Vehicle dynamics: The vehicle is able to follow the trajectory without violating its maneuverability constraints (i.e., turning rate and acceleration ranges, and maximum speed).

(2) Terminal constraint: If the location (x_f, y_f) corresponds to the goal landing site, the arrival time T should be in the range of RTA, that is $T \in [t_{f_l}, t_{f_u}]$. Else, if location (x_f, y_f) corresponds to an alternative landing site, t_f is unbounded. In both cases, the terminal states must adhere to the heading and velocity requirements at the corresponding landing site.

(3) Conflict avoidance constraint: For any segment that belongs to the trajectory l, the cells, crossed by the segment, should be outside of blocked zones $\mathcal{B}(t)$.

(4) Safe distance from alternative destination (used in chapter 5 only): The vehicle must remain in r_{safe} distance from at least one valid landing site.

4.3.2 Design Metrics

Definition 4.3.2 (Flexibility of a trajectory plan)**.** *The flexibility of a trajectory plan l, $\mathcal{F}_l$ is defined as the total number of feasible alternatives at the current space and time point (x, y, t).*

$$\mathcal{F}_l = |\{l^{'} \mid l_0^{'} \in c(x, y, t), l_{end}^{'} = c(x_f, y_f, [t_{f_l}, t_{f_u}])\}| \tag{4.3}$$

where $l^{'}$ represent a feasible trajectory that starts from cell $l_0^{'}$ and terminates at $l_{end}^{'}$.

Definition 4.3.3 (Robustness of a trajectory plan)**.** *The robustness of a trajectory l, $\mathcal{P}_l$ is defined as the likelihood that the trajectory l will remain feasible despite the occurrence of disturbances that pose a constraint violation risk. The robustness of a trajectory is expressed as:*

$$\mathcal{P}_l = \Big(\prod_{i=0}^{T-1} p_i \Big) \alpha_T, \tag{4.4}$$

where p_i is the probability that segment l_i remains feasible in the presence of disturbances, and α_T is the probability that the corresponding landing site is available at time T.

4.3.3 Performance Metrics

At a given point in space and time, a vehicle may have a set of feasible trajectories $\mathcal{L} = \mathcal{L}_g \cup \mathcal{L}_a$, where the trajectories in the set $\mathcal{L}_g = \{l_1, l_2, \ldots, l_n\}$ terminate at the goal landing site, while the trajectories in the set $\mathcal{L}_a = \{l_1, l_2, \ldots, l_m\}$ lead to an alternative landing site. A mission is deemed to be **successful** if there is at least one feasible trajectory that reaches the goal landing site (i.e., $\mathcal{L}_g \neq \emptyset$). Similarly, a mission is deemed to be **survivable** if there is at least one feasible trajectory that leads to either the goal landing site or an alternative landing site (i.e., $\mathcal{L} \neq \emptyset$). Therefore, the probability that a mission is successful or survivable is highly dependent on the robustness of each trajectory in the set $\mathcal{L}_g/\mathcal{L}$ and its cardinality.

Definition 4.3.4 (Robustness of a mission)**.** *Consider the set of trajectories $\mathcal{L}_g = \{l_1, l_2, \ldots, l_n\}$ that terminate at the goal landing site. With a slight abuse of notation, we denote the robustness of a trajectory l_i as $\mathcal{P}_{l_i} \in [0, 1]$, and the probability that the mission is successful as:*

$$\mathcal{P}^{succeed}(\mathcal{L}) = 1 - \prod_{i=1, l_i \in \mathcal{L}_g}^{n} \Big(1 - \mathcal{P}_{l_i}\Big) \tag{4.5}$$

Definition 4.3.5 (Survivability of a mission)**.** *Given the set of feasible trajectories $\mathcal{L} = \{l_1, \ldots, l_m\}$ that terminate at a valid landing site (i.e., at either the goal landing site or an alternative landing site), we denote the robustness of an individual trajectory l_i as $\mathcal{P}_{l_i} \in [0, 1]$, and the probability that the mission is survivable as:*

$$\mathcal{P}^{survive}(\mathcal{L}) = 1 - \prod_{i=1, l_i \in \mathcal{L}}^{m} \Big(1 - \mathcal{P}_{l_i}\Big) \tag{4.6}$$

Important Properties of Performance Metrics

Theorem 4.3.1. *Let $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$ be a set of independent, feasible trajectories, where the robustness of trajectory $l_i \in \mathcal{L}$ is $\mathcal{P}_{l_i} \in [0, 1]$. Then, given the trajectory set $\mathcal{L}$, the survivability of the mission is expressed in Equation (4.6).*

i. Let $\mathcal{P}'_{l_i} \in [0, 1]$ be a new robustness measurement of trajectory l_i such that $\mathcal{P}'_{l_i} > \mathcal{P}_{l_i}$. Then, the survivability of the mission increases by the amount $\Delta\mathcal{P}_1 \geq 0$.

ii. Let the trajectory set $\mathcal{L}$ increase in size through the addition of a feasible trajectory l_{n+1} with the robustness $\mathcal{P}_{l_{n+1}} \in [0, 1]$. Then, the survivability of the mission increases by the amount $\Delta\mathcal{P}_2 \geq 0$.

Proof. For simplicity, define $A = \prod_{i=1}^{n}(1 - \mathcal{P}_{l_i})$ and $A \in [0, 1]$. To show $i.$, we express the increment in the survivability of the mission due to the inclusion of an existing trajectory l_i when $\mathcal{P}_{l_i} \in [0, 1)$ as follows:

$$\Delta\mathcal{P}_1 = \mathcal{P}'^{survive} - \mathcal{P}^{survive} = 1 - A\frac{(1 - \mathcal{P}'_{l_i})}{(1 - \mathcal{P}_{l_i})} - (1 - A)$$

$$= A\frac{\mathcal{P}'_{l_i} - \mathcal{P}_{l_i}}{1 - \mathcal{P}_{l_i}} \geq 0$$

In a special case where $\mathcal{P}_{l_i}$, since $\mathcal{P}'_{l_i} \geq \mathcal{P}_{l_i} = 1$ and $\mathcal{P}'_{l_i} \in [0, 1]$, we have $\mathcal{P}'_{l_i} = 1$ which leads to the following conclusion:

$$\Delta\mathcal{P}_1 = \mathcal{P}'^{survive} - \mathcal{P}^{survive} = 0 - 0 = 0$$

To show statement $ii.$, we define event D_i as the trajectory l_i being infeasible. Accordingly, let $P(D_i)$ denote the probability that the trajectory l_i is infeasible and we have $P(D_i) = 1 - \mathcal{P}_{l_i}$. As a result, the mission survivability before and after including new trajectory l_{n+1} can be restated as following:

$$\begin{aligned}
\mathcal{P}^{survive} &= 1 - P(D_1 \cap D_2 \cap \cdots \cap D_n) \\
&= 1 - P(D_1)P(D_2|D_1)\cdots P(D_n|D_1 \cap \cdots \cap D_{n-1}) \\
&= 1 - P(D_1)P(D_2)\cdots P(D_n)
\end{aligned}$$

$$\begin{aligned}
\mathcal{P}'^{survive} &= 1 - P(D_1 \cap D_2 \cap \cdots \cap D_n \cap D_{n+1}) \\
&= 1 - P(D_1)P(D_2|D_1)\cdots P(D_n|D_1 \cap \cdots \cap D_{n-1})P(D_{n+1}|D_1 \cap \cdots \cap D_n) \\
&= 1 - P(D_1)P(D_2)\cdots P(D_n)P(D_{n+1}|D_1 \cap \cdots \cap D_n)
\end{aligned}$$

$$\begin{aligned}
\Delta\mathcal{P}_2 &= \mathcal{P}'^{survive} - \mathcal{P}^{survive} \\
&= P(D_1)P(D_2)\cdots P(D_n)\big(1 - P(D_{n+1}|D_1 \cap \cdots \cap D_n)\big) \\
&\geq 0
\end{aligned}$$

The $\Delta\mathcal{P}_2$ suggests that adding an independent new trajectory which fails with probability 1 does not improve mission survivability. In this case, we have $1 - P(D_{n+1}|D_1 \cap \cdots \cap D_n) = 1 - P(D_{n+1}) = 0$.

In addition, the mission survivability does not improve if the new added trajectory is

correlated with a subset of the feasible trajectories and is deemed infeasible based on the performance of the feasible trajectory set. In this case, we have $1 - P(D_{n+1}|D_1 \cap \cdots \cap D_n) = 1 - 1 = 0$.

□

Corollary 4.3.1.1. *If the current survivability of the mission is less than* 1*, improving the robustness of any existing trajectory or adding a new and uncorrelated trajectory to the current trajectory set increases the survivability of the mission.*

4.4 Compute Trajectory Flexibility via Backtracking Algorithm

A trajectory is feasible if it is free of conflicts, meets operational constraints such as the required RTA and safety radius, and adheres to the constraints on vehicle dynamics specified by the motion primitives per Definitions 4.3.1. To search for all feasible trajectories starting from the cell of interest to the goal landing site, we use a backtracking algorithm as illustrated in Figure 4.2.

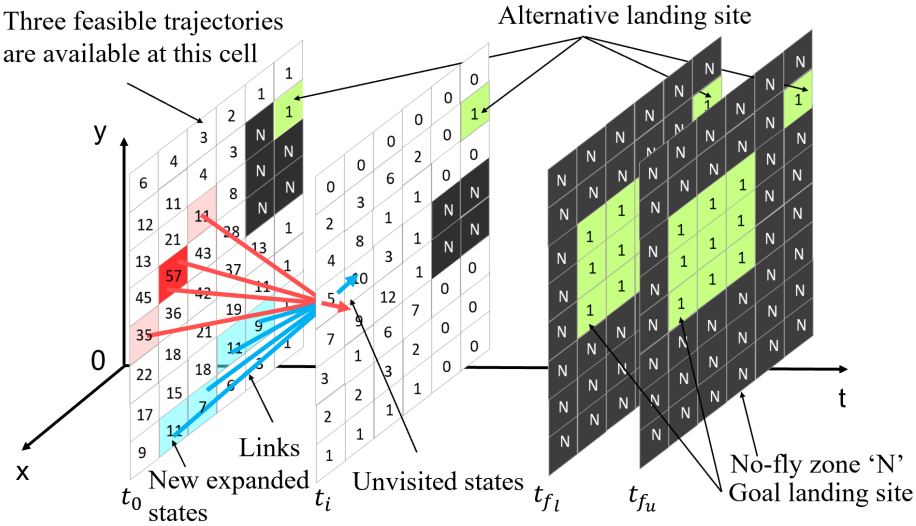


Figure 4.2: Compute the set of feasible trajectories for a cell in space and time.

We begin the search by marking the set of valid terminal states as unvisited and the corresponding cells as 1. We project intruder trajectories into space and time as blocked cells with the label 'N' to account for the motion of obstacles in the environment. We

generate a list of backward reachable states from each unvisited state under consideration and eliminate the set of states that cannot be part of a feasible trajectory. At the end of the iteration, only promising states will be mapped into cells and become unvisited states. It is possible that several states will be assigned to a single cell. Since each promising state corresponds to a feasible trajectory that leads to one of the valid terminal states, the total number of feasible trajectories in a cell equals the number of promising states in that cell. Denote by $c(x, y, t).N_f$ the total number of feasible trajectories and by $c(x, y, t).\bar{S}$ the set of unvisited states in cell $c(x, y, t)$. The main steps are summarized below:

- Initialization: As shown in Figure 4.2, the green squares represent valid landing cells, corresponding to goal-reaching site and alternative landing sites. For any valid terminal state $(x_f, y_f, h_f, v_f, t_f) \in \Omega(t_0)$, we set the number of feasible trajectories for a goal reaching cell $c(x_f, y_f, t_f)$ to one, i.e., $c(x_f, y_f, t_f).N_f = 1$. In addition, each valid terminal state becomes the only unvisited state in the corresponding goal reaching cell, i.e., $c(x_f, y_f, t_f).\bar{S} = (x_f, y_f, h_f, v_f, t_f)$. If a cell is blocked, we set $c(x_t, y_t, t).N_f = c(x_t, y_t, t).\bar{S} =$ 'N'. Otherwise, $c(x_t, y_t, t).N_f = 0$ and $c(x_t, y_t, t).\bar{S} = \emptyset$.

- Expand & Prune: At time frame t, given a non empty cell, any unvisited state (x_t, y_t, h_t, v_t, t) in the cell $c(x_t, y_t, t)$, called ancestor, should be propagated one step backward by applying all possible heading and velocity change $(-\delta_h, -\delta_v)$, where $(\delta_h, \delta_v) \in \mathcal{A}$. All new generated states $(x_{t-1}, y_{t-1}, h_{t-1}, v_{t-1}, t-1)$ should belong to the time frame $t-1$. (see red/blue arrow and its corresponding new generated states). A new generated state $(x_{t-1}, y_{t-1}, h_{t-1}, v_{t-1}, t-1)$ is forward-reachable if the minimum time required for the vehicle to move from the current state to the new generated state does not exceed $t-1-t_0$, as stated in Inequality (4.7).

$$\frac{||(x_t, y_t) - (x_0, y_0)||}{v_{max}} \leq t - 1 - t_0, \tag{4.7}$$

where (x_0, y_0, t_0) is the way point corresponding to a departure and v_{max} is the upper limit of vehicle's speed. In the pruning step, a newly generated state is eliminated if the segment, connecting itself to its ancestor, is infeasible; or if the newly generated state is not forward reachable; or if the vehicle is not within safety radius of any airport. Otherwise, we assign the newly generated state $(x_{t-1}, y_{t-1}, h_{t-1}, v_{t-1}, t-1)$ to the list of unvisited states in the corresponding cell $c(x_{t-1}, y_{t-1}, t-1)$, increase the number of unvisited states for cell $c(x_{t-1}, y_{t-1}, t-1).N_f$ by 1, and mark the ancestor state as visited.

- We repeat the steps described above for the previous time frame until $t - 1 = t_0$. Once the algorithm terminates, for each cell $c(x, y, t)$ in space and time, we obtain the number of feasible trajectories to the valid terminal cells by looking at its $c(x, y, t).N_f$ parameter.

The implementation details are provided in Algorithm 4, while the $ExpandPrune(\cdot)$ function is provided in Algorithm 5.

4.5 Estimate the Probability of Success for a Trajectory Segment via Monte Carlo Simulation

To plan a safe trajectory in presence of uncertainty, an autonomous vehicle should be able to first perceive all the safe states and then transition from one state to the next. A state is deemed as safe if this state provides feasible and safe trajectories to global safe states (i.e., valid terminal states at landing sites). The assessment of a safety level of a state involves identifying the total number of feasible trajectories emanating from the state, as discussed in section 4.4. Additionally, each feasible trajectory should be further evaluated regarding its robustness in uncertainty. The robustness of a trajectory relies on the robustness of each of its trajectory segments. Therefore, in this section, we develop a Monte Carlo (MC) simulation to quantify the safety level of a state by assessing the robustness of the trajectory

Algorithm 4: Compute feasible trajectory set via Backtracking

Input:
Origin:
(x_0, y_0, t_0)
Goal $\cup$ Alternative landing sites:
$(x_f, y_f, h_f, v_f, [t_{f_l}, t_{f_u}]) \cup (x_f, y_f, -, -, -)$
Flight dynamics: $\delta_h \in [\delta_{h_l}, \delta_{h_u}], \delta_v \in [\delta_{v_l}, \delta_{h_u}], v \in [v_{min}, v_{max}]$
BlockMap: Space and time with information on no-fly zones
Output: TrajMap: The estimate of total number of trajectories

$t = t_{f_u}$
for $t_f \in [t_{f_l}, t_{f_u}]$ **do**
    $c(x_f, y_f, t_f).N_f = 1$
    $c(x_f, y_f, t_f).\bar{S} \leftarrow (x_f, y_f, h_f, v_f, t_f)$
while $t > t_0$ **do**
    for $c(x_i, y_i, t) \in [X_l, X_u] \times [Y_l, Y_u]$ **do**
        if $BlockMap(x_i, y_j, t) == 0$ **then**
            $c(x_i, y_i, t).N_f = -\infty$
            $c(x_i, y_i, t).\bar{S} \leftarrow$ N
        else if $c(x_i, y_i, t).\bar{S} \neq \emptyset$ **then**
            for $s \in c(x_i, y_i, t).\bar{S}$ **do**
                $\mathcal{I}_{feasible} \leftarrow$ ExpandPrune(s, δ_t, $\mathcal{A}$)
                $c(:, :, t-1).\bar{S} \leftarrow \mathcal{I}_{feasible}$
                $c(:, :, t-1).N_f += |\mathcal{I}_{feasible}|$
    t = t - 1
for $c(x, y, t) \in [X_l, X_u] \times [Y_l, Y_u] \times [t_0, t_{f_u}]$ **do**
    TrajMap$(x, y, t) = c(x, y, t).N_f$
return TrajMap

segments (i.e., the probability of successfully transiting from one state to the other).

When operating in the airspace, an autonomous vehicle is vulnerable to a variety of internal and endogenous uncertainty. In this thesis, we are interested in exogenous uncertainty that come from dynamic obstacles as well as their motions. For instance, the airspace environment may have restricted zones for special purposes that prohibit the entry of vehicles. In addition, other participants in the airspace, such as weather systems as a specific case, may pose potential threats to the feasibility of the ownship's trajectory plan by violating the safe separation distance.

Algorithm 5: ExpandPrune(·) function

Function `Expand & Prune(`$s, \delta_t, \mathcal{A}$`)`**:**
  $(x_t, y_t, z_t, h_t, v_t, t) \leftarrow s$
  $\mathcal{I}_{feasible} \leftarrow \emptyset$
  for $(\delta_h, \delta_v) \in \mathcal{A}$ **do**
    $v_{t-1} = v_t - \delta_v$
    $h_{t-1} = h_t - \delta_h$
    $x_{t-1} = x_t - v_{t-1}\delta_t \cos(h_{t-1})$
    $y_{t-1} = y_t - v_{t-1}\delta_t \sin(h_{t-1})$
    $s_{new} \leftarrow (x_{t-1}, y_{t-1}, h_{t-1}, v_{t-1}, t-1)$
    if *segment connecting* $c(s)$ *and* $c(s_{new})$ *is feasible per Assumption 4.2.2* & s_{new} *is forward reachable per inequality 4.7* & $c(s_{new})$ *is within valid safety radius of at least one airport* **then**
      $\mathcal{I}_{feasible} \leftarrow \mathcal{I}_{feasible} \cup s_{new}$
  return $\mathcal{I}_{feasible}$

Essentially, dynamic obstacles have intrinsic uncertainties in their locations and shapes in terms of dimensions and volumes. The uncertainties associated with the dynamic obstacles propagate to others by interacting with the airspace environment, dynamically affecting the availability of the airspace. Some obstacles have simple dynamics, and so their behaviors and impacts on the airspace are well understood and easy to model. However, obstacles, such as intense thunderstorms, exhibit complicated dynamic natures which can only be understood partially. To one extreme, an obstacle may exhibit ultra-complicated behaviors, and thus its effects on airspace availability are unclear. Thus, the incomplete understanding of obstacles and the incapability of accurately modeling obstacles challenge the way we evaluate the impacts of an obstacle's uncertainty on the availability of the airspace and its propagated risks to other participants.

MC simulation is a type of simulation that performs statistical analysis based on repeated random sampling [57]. MC simulation can be viewed as a tool for conducting *"what if"* analysis and investigating the full spectrum of risks associated with predefined risky scenarios. Compared to other simulation methods, the MC simulation can be broadly applied to study any problem with randomness. Most importantly, MC simulation is suitable

for simulation of system dynamics with coupled degrees of freedom or with significant uncertainties. In our case of investigating the safety of an autonomous vehicle in presence of extreme uncertainties arising from obstacles whose motions exhibits strong spatio-temporal coupling, MC simulation provides an ideal tool to perform statistical analysis.

To this end, we propose a MC simulation based on mathematical models that describe the statistical properties of uncertain events (i.e., the motion of intruders). These events may be known, partially known, or unknown to the ownship. The MC simulation yields the robustness of a segment, which is then used to compute the robustness of a trajectory per Equation (4.4). We further demonstrate how it can be leveraged to measure mission success and survivability. The MC simulation built in this section with benefits the study in both chapter 5 which evaluates the impacts of uncertainties on how autonomous vehicles perceive the safe states in the airspace, and 6 that evaluates the performance of trajectory planning policies in mitigating uncertainties.

Trajectory model of intruders: We model the intruder's trajectory using a Bezier curve of degree k. Given k random control points $\{(x_i^B, y_i^B, t_i^B) \mid x_i^B \in [X_l, X_u], y_i^B \in [Y_l, Y_u], t_i^B \in (t_0, t_{f_u}), \forall i = 0, \ldots, k\}$, the uniquely defined Bezier curve l^B defines a continuous trajectory that enters the time and space network at cell $c(x_0^B, y_0^B, t_0^B)$ and exists the time and space network at cell $c(x_k^B, y_k^B, t_k^B)$. Adjusting the 0-th and k-th control points, one can specify the entrance and exit location of the intruder as well as its flight duration. By increasing the degree number k, the resulting curve has a greater number of twists and turns, mimicking the motion of an aggressive intruder. We generate a Bezier curve with degree number 1 - a straight line that occupies a fixed location for a time interval - to simulate the trajectory of a hovering intruder.

Uncertainty model of a trajectory Trajectory uncertainty models typically utilize either a probability density function (pdf) or bounded shapes [58]. We utilize a shape-based methodology to facilitate greater utility. With the assumption that the ownship is capable of tracking the prescribed trajectory, the possible trajectories can be bounded by geometric

volumes (i.e. sheared cylinders) [59]. Given an intruder trajectory l^B, we further define the trajectory uncertainty volume $V(l^B, r)$ as the union of disks with radius r that are centered at points along the trajectory l^B. A possible trajectory is a continuous curve starting at cell $c(x_0^B, y_0^B, t_0^B)$ and ending at cell $c(x_k^B, y_k^B, t_k^B)$ such that all intermediate points are within the trajectory uncertainty volume $V(l^B, r)$.

With this uncertainty model, we assume that if an intruder is only partially known to the ownship, the nominal trajectory l^B and the corresponding uncertainty level r are certain to the ownship, but the actual trajectory is unknown. Typically, if $r > 0$, the intruder may fly a trajectory that deviates at most r from the planned trajectory. If $r = 0$, the partially known intruder becomes fully known to the ownship. The known intruder will follow l^B exactly through space and time. In the extreme case where an intruder is unknown to the ownship, the ownship is unaware of the intruder's existence until the ownship encounters a conflict with the intruder.

Traffic Flows: Intruders may enter and exit the airspace in a batch or individually. We propose two different schemes for introducing intruder traffic into regions of interest during a specified time interval. Let $N(t)$ represent the number of intruders in the environment at time step t.

The batch-entry-batch-leave scheme ensures that $N(t)$ is constant at all times (i.e., $t \in [0, T]$) in the environment. The batch-entry-batch-leave scheme starts by generating the entrance coordinate at $t = 0$ and exit coordinate at $t = T$ uniformly in 2D space for each intruder. With the entrance and exit coordinates, we can further specify the intruder's trajectory by generating Bézier curve with known end points. In Figure 4.3, we generate 50 intruders' trajectories using the batch-entry-batch-leave scheme in a $85km \times 85km$ region over a 6 unit time interval. Each dark line represents the mean motion of the intruder. The gray buffer centered on the dark line is proportional to the uncertainty radius. All 50 intruders enter the airspace at $t = 0$ and exit the airspace at $t = 5$. Therefore, we can observe that at each time step, the total number of intruders remains constant at 50. Figure

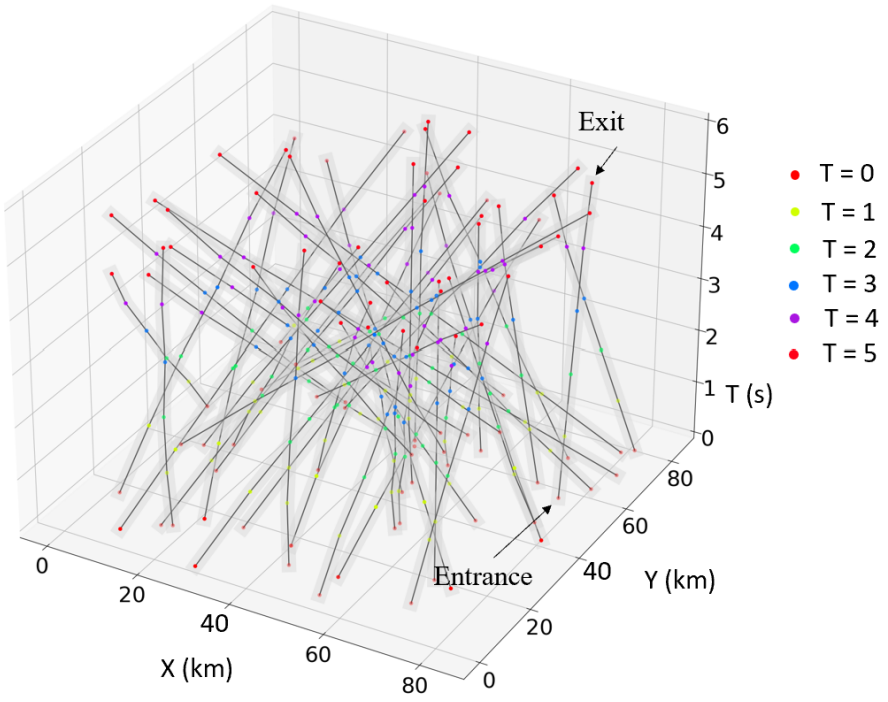


Figure 4.3: Generate 50 intruder trajectories using the batch-entry-batch-leave scheme. Notice that the total number of intruders is constant during the time interval from $t = 0$to $t = 5$. The trajectory model is represented by a Bézier curve of degree 2. The uncertainty radius is 5 km.

4.4 visualizes the same traffic flow using the trajectory model with a higher degree of freedom. Therefore, the intruders display relatively more aggressive turns and movements.

The one-entry-one-exit scheme attempts to simulate the occurrence of random intruders into the airspace. We assume that at any instance in time, intruders originate uniformly from each cell in airspace following a Poisson distribution with mean arrival rate λ per unit time. When shows up, the intruder's heading is uniformly distributed in all directions from 0^o to 360^o and its cruising speed is constant. The maximum flight endurance is user-defined. Figure 4.5 visualizes the intruder traffic generated by the one-entry-one-exit scheme. The mean arrival rate of the intruders is defined as 10 intruders per unit time. Notice that if an intruder shows up at $T = 6$, which is beyond the time interval of interest, the trajectory of the intruder is reduced to a point. A special scenario could involve an intruder entering in the airspace of interest, but then leaving the airspace in the middle of the flight. Therefore,

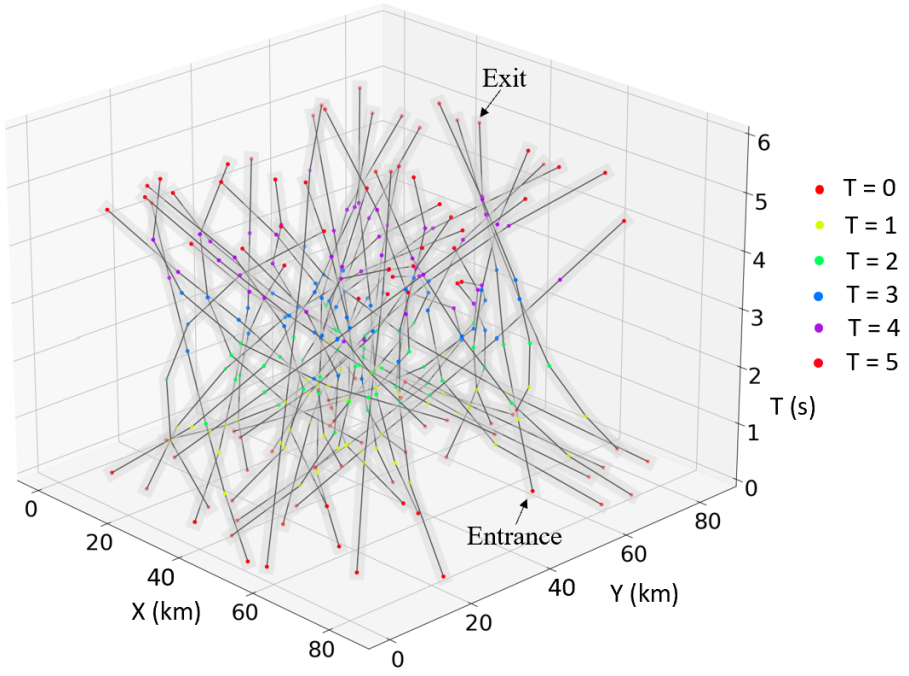


Figure 4.4: Generate 50 intruder trajectories using the batch-entry-batch-leave scheme. Notice that the total number of intruders is constant during the time interval from $t = 0$to $t = 5$. The trajectory model is represented by a Bézier curve of degree 3. The uncertainty radius is 5 km.

each intruder enters and exits the airspace individually. At each time instance, the total number of intruders $N(t)$ is inconsistent.

While the one-entry-one-exit scheme generalizes air traffic flows better, the randomness caused by the inconsistent number of occurrence of intruders is propagated to the availability of airspace by blocking out cells. Consequently, the proportion of available airspace varies dynamically at each time step.

Notice that the intruder traffic generated by batch-entry-batch-leave or one-entry-one-exit is not distributed randomly across the space and time grid. This is because not every cell in space and time is subject to traffic with the same probability. As shown in Figure 4.6, when projected on the 2D space, multiple traffic streams may twist together and add more strain on certain regions in the space and time grid.

Simulation of Unknown Traffic: Unknown traffic is, by definition, unknown, so it is impossible to make any assumptions regarding the occurrence and motion of unknown

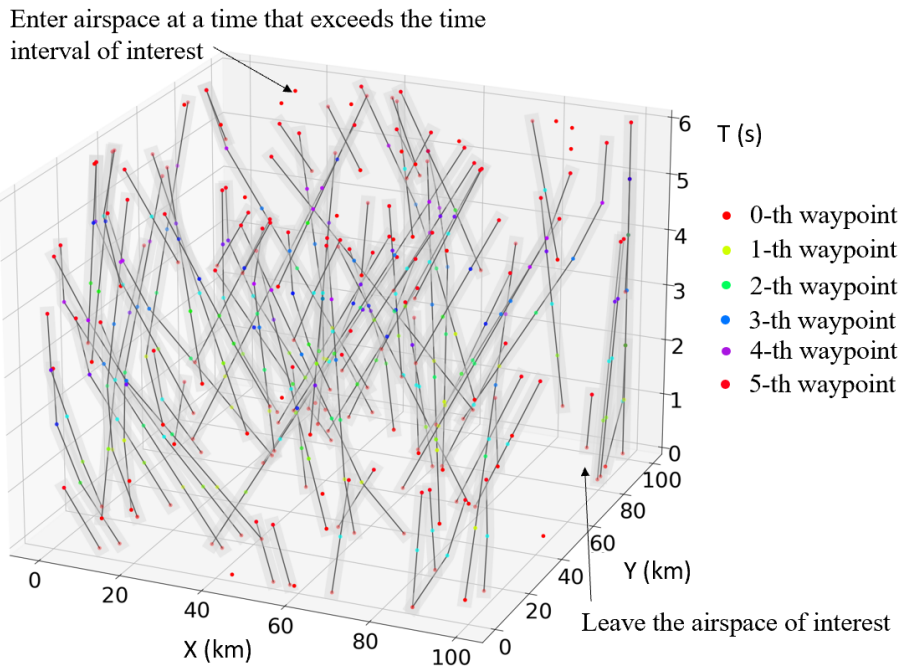


Figure 4.5: Generate intruder trajectories using the one-entry-one-exit scheme with the mean arrival rate at 10 intruders per second. Notice that the total number of intruders is inconstant during the time interval from $t = 0$to $t = 5$. The trajectory model is represented by a Bézier curve of degree 2. The uncertainty radius is 5 km.

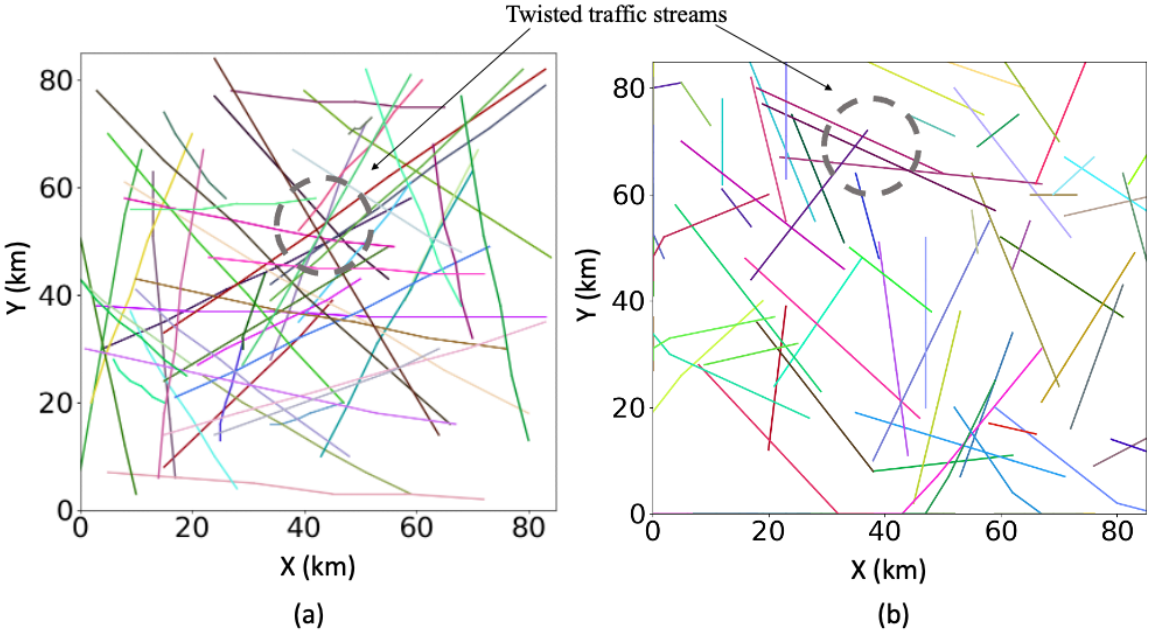


Figure 4.6: (a) Generate intruder trajectories utilizing the batch-entry-batch-leave scheme and project the trajectories in 2D space. (b) Generate intruder trajectories using a one-entry-one-exit scheme and project them in 2D space. We observe that certain areas in space and time are subject to a denser traffic.

intruders. It is possible that the unknown traffic could either completely block the airspace or have no influence at all. There is no universal method to simulate the unknowns. In this thesis, we argue that unknown traffic exhibits the same behavior as intruders. The presence of the unknown traffic is identified by the airspace and the simulator, but the decision maker is unaware of it (i.e., the ownship).

In the following, we describe the simulation process using the batch-entry-batch-leave scheme to simulate traffic flows. A traffic scenario is defined as a situation in which there are N_k known intruders, N_p partially known intruders, and N_u unknown intruders. The number of partially known intruders, the uncertainty threshold r of each intruder's trajectory, and the number of unknown intruders all influence the severity of the uncertainties in the environment. The proposed MC-based simulator takes into account a segment or set of segments related to the current trajectory planning decision all at once. The expected output for each input segment is the probability of the segment being feasible in the presence of the current traffic scenario. Figure 4.7 depicts the framework of the MC-based simulator which consists of four major processes:

- Initialization: During the initialization process, we simulate intruder trajectories by randomly generating $N_k + N_p$ Bezier curves of various lengths between $[t_0, t_{f_u}]$. A counter $N_{success}$ is used to record how many times a segment is feasible out of the N_{mc} different traffic realizations.

- Traffic Realization: The MC-based simulator randomly generates realizations of an intruder's trajectory for a given traffic scenario based on each intruder's nominal trajectory l^B and uncertainty threshold r (if applicable). When a new realization is generated, the timer advances by one. If the timer reaches the maximum number of runs N_{mc}, the simulation process terminates and outputs an estimate of a segment's robustness. Otherwise, the simulator proceeds to the next step.

- Feasibility Evaluation: A $BlockMap$ is created by blocking the cells traversed by

the realized trajectories l^B. Given a segment of interest, the simulator evaluates the feasibility of the segment via Assumption 4.2.2. The counter for the segment $N_{success}$ is increased by one if the segment is feasible. Otherwise, we do nothing.

- Termination: After N_{mc} runs, the simulation process terminates. The probability of the segment being feasible is measured by the ratio $N_{success}/N_{mc}$.

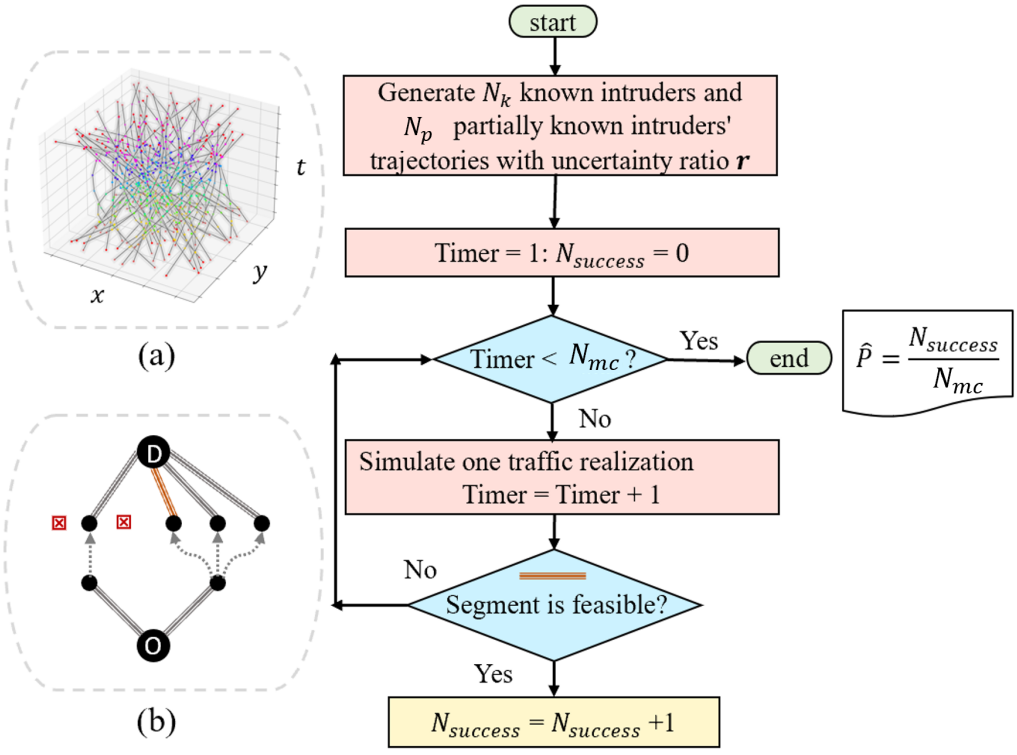


Figure 4.7: MC simulation framework. (a) Visualization of a traffic scenario in space and time. (b) Input trajectory segment.

ma

CHAPTER 5
A METHODOLOGY TO DEVELOP SURVIVABILITY MAPS FOR UNMANNED AERIAL VEHICLES

5.1 Overview

5.1.1 Introduction

Survivability is the guarantee that a system has at least one trajectory to a safe landing site. In the context of autonomous mobility, naturally, the autonomous vehicle has a set of basic *needs* to be fulfilled in order to survive as it is the case with living creature in the context of biology. In this chapter, we focus on a fundamental requirement for an autonomous vehicle: that to ensure survivability, it must be able to reach a safe landing site at any time and from any point during its flight. In other words, there must always have a robust set of feasible trajectories that lead to at least one valid landing site.

The survivability at different points in an airspace is illustrated in Figure 5.1. We observe that the original flight path of an autonomous vehicle is about to pass through a low-survivability yellow zone. Therefore, the vehicle must deviate from its current path in order to pass through areas with higher probabilities of survival.

A key challenge in determining whether an autonomous vehicle is survivable in its current state, i.e., determining whether it is in a *survivable state*, is determining the number of feasible trajectories that are available and the likelihood of a trajectory being robust given the current *situation* of the vehicle. According to [60], the *situation* of an autonomous vehicle typically includes the following: (1) the current power and capability of the vehicle, (2) stationary and dynamic obstacles present in the environment, (3) legal regularization/operational constraints, and (4) flight intent and mission. Since the *situation* can change dynamically over time, the autonomous vehicle must continuously assess the like-

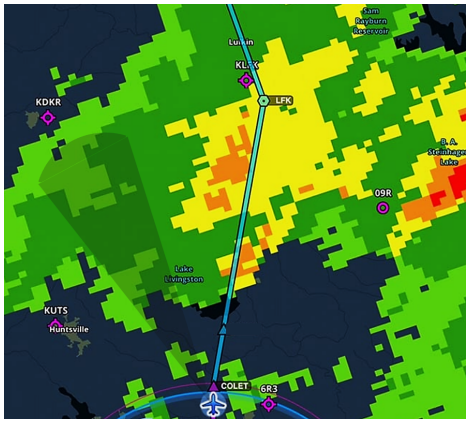


Figure 5.1: The visualization of the survivability map concept in the airspace. An autonomous vehicle modifies its trajectory to escape from a low-survivability region to a high-survivability region. The color red indicates areas with zero survivability, while the color dark green represents areas with absolute survivability.

lihood of survival given the current situation and monitor global survivable states. When a performance decline is detected, the autonomous vehicle will transition from a threat state to a survivable state. Clearly, the survivable state plays a central role in the decision-making process of an autonomous vehicle and provides the flight mission with the bare minimum safety guarantee.

In this chapter, we propose a method for automatically generating a survivability map (i.e., a visual representation of survivable states) for an autonomous vehicle. The methodology is comprised of two primary components. Initially, a back-propagation algorithm is developed to compute feasible trajectories that satisfy novel operational constraints. Second, a Monte Carlo Simulation is designed to evaluate the robustness of a feasible trajectory. A state's survivability is determined by the combination of the number and robustness of feasible trajectories at that state. The novelty of this study lies in its sensitivity analysis of the effects of airport density, vehicle lifetime and maneuverability, and air traffic on the determination of survivability. The insights are valuable for both strategic and tactical contingency planning.

5.1.2 Related Work

There is little research on the survivability of autonomous vehicles. Nonetheless, research into the safety of autonomous vehicles, which is highly relevant to the notion of survivability, has a long history. The safety of autonomous vehicles can be improved by enhancing their power capability or physical model, such as by incorporating two engines/power supplies and advanced structure materials. A large body of literature focus on the decision-making capability of autonomous vehicle. Specifically, an autonomous vehicle must continually make decisions about how to behave in a dynamically changing environment in a manner consistent with the programmed goals and constraints. This is dependent on an accurate assessment of the level of safety/risk in the surrounding environment. The enhancements to vehicle design and decision-making algorithms show promise for achieving a higher level of safety in the vast majority of risk situations. Nonetheless, unforeseen circumstances, such as engine failures, may still occur. Consequently, one stream of work investigates contingency planning that may begin before a flight mission and must be revised during the mission.

A fundamental requirement of the contingency planning is to continuously identify safe states in the environment so that the autonomous vehicle can preserve an acceptable level of safety by moving to a safe state when an emergency occurs. Therefore, the quantitative and qualitative assessment of the availability of airspace and its risks due to terrain and obstructions attracts extensive research interest. In [61], Murca developed a data-driven methodology for identifying and forecasting available airspace to support emerging urban air mobility operations. The data-derived understanding of spatial traffic patterns is used to develop a stochastic model that forecasts active traffic patterns and their spatial confidence regions. Thus, 3D airspace availability can be derived in a dynamic manner. Similar work [62] utilized aircraft tracking data to assess the spatial traffic distribution in the vicinity of the airport. Additionally, we observe a shift in emphasis from continuous to more structured airspace (i.e., network). In [63], Salleh et al. evaluated how the urban infrastructure's

topography influences the determination of the optimal route network for autonomous vehicle operations. Likewise, Vascik and Hansman [64] leveraged aircraft tracking data to statistically define lateral and vertical containment boundaries for airport arrival and departure trajectories and to assess airspace regions where a potential air network could be established. Some research studies highlight the importance of raising the awareness of risk by predicting potential conflicts in the environment. For example, Zou et al. [65] proposed a computationally efficient method for estimating the collision probability of an autonomous vehicle, taking into account the position error uncertainties of other vehicles. The proposed method can be utilized to detect potential conflicts in real time and derive dynamic safety limits in space.

Surviving a mission requires safely touching down at a valid landing site. This requires precise reasoning regarding the accessibility and risk level of the airspace. In addition, the autonomous vehicle necessitates a sophisticated algorithm for trajectory planning in order to construct reliable trajectories to valid landing sites. Extensive research has been conducted on algorithms for planning trajectories in the presence of uncertainty. For example, Bry and Roy [66] proposed a graph-search based algorithm–the Rapidly-exploring Random Tree algorithm–where, given a nominal trajectory, distributions for future states of the vehicle are first predicted to assess whether the probability of collision, given a state, is bounded below a threshold value. Next, a set of trajectories is incrementally constructed while efficiently searching for the best candidate path. Paths are evaluated based on their probability of being realized by a closed-loop controller. Similar work that focuses on graph-based search algorithm can also be found in [67] and [68]. Separately, roadmap-based approaches that rely on an understanding of the safe states in the environment (or the configuration space in general) have attracted significant research interest. Typically, they utilize one of three techniques: visibility graphs [69], Voronoi diagrams [70], or potential fields [71]. In the context of contingent planning, [72] proposed a path-planning formulation that incorporates probabilities of obstacle predictions to enable efficient gener-

ation of a safe set of contingency paths in a dynamic environment. Recently, a new stream of research has grown up around the theme of designing trajectory planning algorithm in presence of extreme uncertainties [73].

Individual research into how traffic, weather, and generalized dynamic obstacles affect the risk/availability of airspace and trajectory planning has been vigorous. Most works conduct short-term analysis and disregard long-term planning that could enhance the survivability design. For instance, the generation of safe trajectories in response to a disruptive event only takes existing airports into account. The opportunity to construct new airports is disregarded. Consideration of long-term planning, specifically the geometrical density and distribution of airports, becomes a research gap when evaluating survivable states and calculating safe alternatives. Additionally, the maneuverability of autonomous vehicles is absent from the assessment of their survivability. There is a lack of research on autonomous vehicle maneuverability's effect on the marginal improvement of safety. The answer to this research question will benefit future dense unmanned traffic management operations, in which autonomous agents should operate closely and be able to maneuver agilely. Multiple agents can safely collaborate by demonstrating how two vehicles with distinct maneuverability perceive the environment's safety. Last but not least, a new operational concept inspired by the Extended-range Twin-engine Operational Performance Standards (ETOPS) should be considered further to reflect the current state of aviation development.

The remainder of this chapter is organized as follows. In section 5.2, we review the notion of survivability and establish the theoretical foundation for identifying survivable states in the environment. At the same time, based on new introduced operational constraint, we modify the backtracking algorithm for counting the number of feasible trajectories in section 4.4 as well as a Monte Carlo simulation to estimate the robustness of a trajectory segment in section 4.5. In section 5.3 we set up computational experiments for validating the proposed methodology. The results and analysis of hypothetical cases and a real scenario based on the San Francisco Bay Area are presented and further discussed in

section 5.4.

5.2 Methodology

In this section, we will first present a few scenarios that demonstrate the need of taking into account aspects like airport locations, vehicle mobility, remaining lifetime, and air traffic while developing a survivability map.

5.2.1 Unsurvivable Scenarios

Airport Location and Safe Radius

Airports are ground facilities that offer operatically open surface for an vehicle to take off and to land. According to the Extended-range Twin-engine Operations Performance Standards (ETOPS), an airplane should not deviate more than a certain distance from the nearest airport [74]. This safe distance is determined by the number and reliability of engines aboard the airplane. As a result, airports provide an opportunity for an airplane to extend its route in the airspace, resulting in safe, sustained, and efficient operations.

The determination of a safe radius for AAM vehicles is still in its early stages. The AAM includes the implementation of revolutionary vehicles powered by clean energy, such as electricity and hydrogen. To ensure the safety and reliability of the AAM flight, we anticipate that new operational guidelines inspired by ETOPS will be implemented. Specifically, AAM vehicles should always stay within a safe radius of at least one valid landing site. One of the main factors that determines safe radius is the maturity of power supply technology for AAM vehicles, such as the steady state power density of a hydrogen fuel cell. Unlike large commercial planes, which require operational runways for landing, AAM vehicles typically have vertical landing/takeoff capability and require relatively small landing space. As a result, another factor influencing an airport's safety radius is the availability of landing spots near the airports as well as those in the city.

In this thesis, we investigate the effects of the safe radius on vehicle survivability and

offer recommendations for siting airports as well as determining the safe radius for AAM operation. To follow the ETOPS inspired requirements, a vehicle should always stay r_{safe} within of at least one of the valid airports. Figure 5.2, consider a vehicle with unlimited fuel traveling from airport A to airport B in clear traffic. We assume that the survivability of the vehicle declines as the distance with respect to the airport increases. To better illustrate the concept, we utilize the exponential decay function with decay constant 1, that is $P^{\text{survive}} = e^{-d} \in [0, 1]$, where $0 \leq d \leq r_{\text{safe}}$ is the Euclidean distance between vehicle and airport of interest. In particular, consider a single airport, the survivability of the vehicle reaches critical value $e^{-r_{\text{safe}}}$ and then drops to 0 when it flies across the safe radius of the airport. In Figure 5.2, during flight leg $l_A(l_B)$, the vehicle can only access to airport A(B). However, when traveling during flight leg l_{AB}, the vehicle has two options, and it can either turn around and land at airport A or continue to land at the airport B. Figure 5.3 depicts the situation that the safe radius is not large enough to cover the entire flight path. Therefore, the middle segment has absolutely no chance of surviving the mission. In this case, we observe that as the safe radius increases, the length of the unsurvivable segment decreases. When the safe radius is large enough that the flight path can be covered fully as shown in Figure 5.4, the vehicle has a greater chance to survive the mission due to the increased number of landing options.

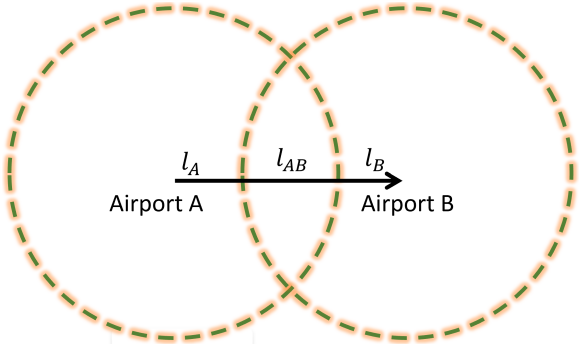


Figure 5.2: Visualization of a simple flight mission.

In summary, the airport influences the survivability map by limiting the number of valid landing sites, which is determined by airport locations as well as the predetermined safe

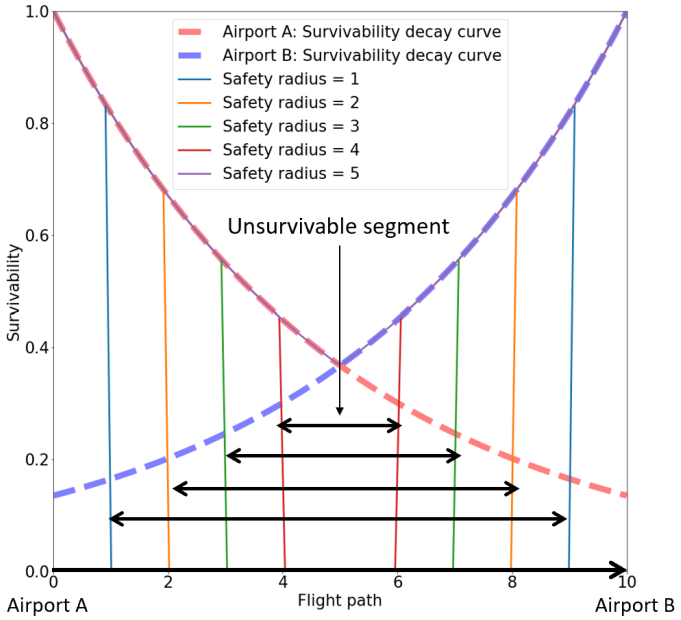


Figure 5.3: As the safe radius expands, the unsurvivable regions gradually diminish until they reach a point.

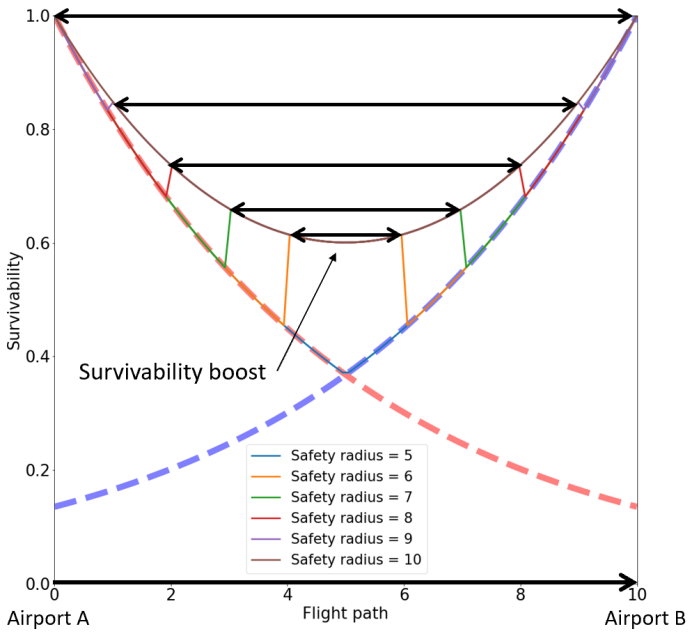


Figure 5.4: As the safe radius expands, the middle flight segment offers a greater chance of survival because there are more landing options at both ends.

radius around the airport.

Vehicle Maneuverability and Remaining life

In this thesis, we discretize the dynamics of a vehicle so as to constrain the feasible trajectories to the family of time-parameterized segments that may be constructed by connecting motion primitives. The motion primitives of a library (i.e., action plans) are defined by the coupling of heading and velocity changes and enable the design of complicated actions. Two vehicles with varying degrees of agility are preparing to exit the corridor in Figure 5.5 as an intruder approaches them directly. Four of Vehicle A's seven motion plans are able to guide the vehicle out of the corridor. On contrast, vehicle B has just three motion plans, two of which are infeasible due to the airspace's availability and one of which is feasible but has a high collision probability. In the example, the limitations on the vehicle B's turn angle and speed adjustment result in a low chance of survival.

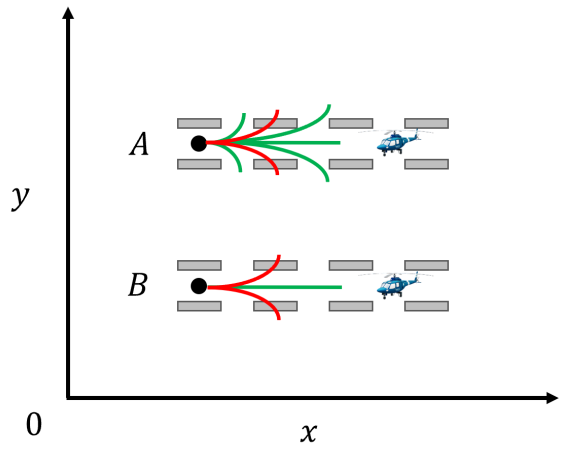


Figure 5.5: Initially, the vehicle is flying east. An intruder enters the corridor from the west direction. The vehicle must exit the corridor to avoid a possible collision.

The maneuverability of a vehicle is essential for resolving urgent conflicts in challenging traffic conditions or exploiting and learning the environment. Given that every trajectory is a combination of motion primitives, it is evident that the remaining life of the vehicle determines the maximum length of a single trajectory and the dimension of the motion combination. Specifically, the length of a trajectory indicates the maximum deviation a vehicle could have when avoiding obstacles in the airspace while still achieving its objec-

tive. In conclusion, the maneuverability and remaining life of a vehicle have a significant impact on its survivability.

Traffic

Traffic (i.e., other airspace participants besides the ownship) generates the most environmental uncertainty. First, the presence of traffic physically impedes a proportion of the total airspace volume. Insufficient space could prevent a vehicle from performing a life-saving maneuver. Second, the level of uncertainty associated with the trajectory of traffic would affect the vehicle's confidence in its safety near traffic streams. For a conservative vehicle with a high threshold for survivability, the unpredictability of traffic streams reduces the vehicle's safety options.

5.2.2 Survivability Map Construction

The computation of survivability map for a vehicle relies on the theoretical foundation as discussed in chapter 4. We compute the mission survivability $p^{survive}$ of the vehicle for each cell, assuming that the vehicle flies initially in that cell and could potentially land at any valid landing sites under consideration. The vehicle is given a remaining lifetime of T minutes and obeys the assumptions on its maneuverability, terminal constraints, air traffic, and the predetermined safe radius centered around airports.

The computations process follow two steps: (1) For each cell in the space, compute the total number of feasible trajectories to any valid landing site at that cell via the proposed back tracking algorithm introduced in section 4.4 and (2) For each trajectory, estimate the robustness of the trajectory in presence of air traffic. The chance of the vehicle survives a mission given in a cell is then computed via section 4.3.3. The survivability visualizes space and time cells with $p^{survive} \geq 1 - 10^{-8}$.

5.3 Experiments Setup

In the previous section, we introduce the methodology to identify potential safe regions for autonomous vehicles. As we discussed previously, the safety level of a cell in the space and time is determined by the total number of feasible trajectories available at that cell and the robustness of each trajectory. The choice of trajectories as well as the quality of the trajectories are dependent on the geographical distribution of the valid landing sites, air traffic and the maneuverability of the autonomous vehicle. In this section, we setup an experiment environment featured on the San Francisco Bay Area. Results will be present and discussed in the next section.

5.3.1 Airport Location Dataset

The safe states analysis around local airports are deeply influenced by the city pattern, density, and type of airports as well as local traffic. To select the best area for our study, we use the open dataset from [75] which provides detailed information on the name, locations in terms of latitude and longitude, elevation, types of airports etc. After carefully investigating the dataset, we consider a case study of analyzing the safe states around airports in the San Francisco Bay Area. The reasons are in three-folds: (1) The Bay Area is the hub of airports in diverse size. As shown in Figure 5.6, the Bay Area has two major airports, San Francisco International Airport (SFO) and Oakland International Airport (OAK), one medium airport, Norman Y. Mineta San Jose International Airport, and fourteen airports in active daily operations. Other than that, there are more than forty closed small airports which may offer potential benefits in providing additional safety buffer once open. (2) The Bay Area has attracted a great amount of interest in the study of general aviation, for example [76, 77, 78, 79, 80]. (3) A lot of eVTOL manufactures envision the future of UAM operation in the highly urbanized environment of the San Francisco Bay Area for alleviating the traffic congestion and housing affordability crisis.

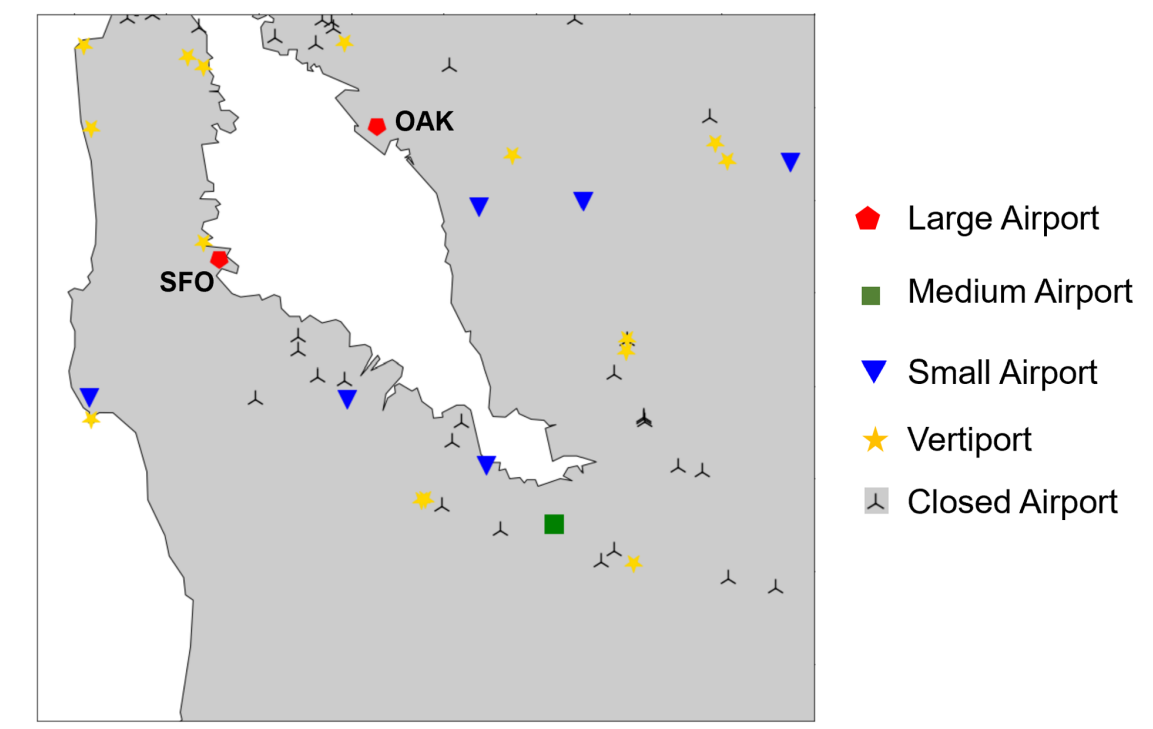


Figure 5.6: Visualization of the airports in the San Francisco Bay Area.

As a summary, we are interested in flight mission in the San Francisco Bay Area which is specified as an $85 \times 85\ km^2$ area. The area of interested is discretized into $1 \times 1\ km^2$ cells and the time is discretized into 1 minutes. The time and space grid consists of cells with dimensions ($1km$, $1km$, $1min$). For simplicity, we assume that each airport offers four landing sites with headings in $h_f \in \{0^o, 90^o, 180^o, 270^o\}$.

5.3.2 Vehicle Maneuverbility

We apply general assumptions on a helicopter-like dynamics model with cruising and hovering capability. Let the maximum flight duration/battery lifetime as $T_b = 60$ minutes and the accumulated flight time as t, the vehicle then associates with a remaining lifetime T_b - t minutes. Time is discretized into equally five minutes windows. At each planning time stamp, the vehicle can change its heading by $dh \in [-30^o, 30^o]$, in five equally spaced discrete increments and its speed by $dv \in [-10, 10]\ m/s$, in five equally discrete increments. During flight in each time window, While there is no limitation on the heading of

the vehicle, the cruising speed of vehicle should be in the range from $[0, 40]$ m/s.

5.3.3 Air Traffic

In this experiment, we only consider partially known intruders with uncertainty radius $r = 1km$. The Monte Carlo simulation used to evaluate the robustness of trajectories apply the batch-entrance-batch-leave scheme. Each simulation generates $N_{max} = 1000$ traffic samples.

5.4 Results and Discussion

In this section, the results of the empirical study are presented and discussed. Our analysis focuses primarily on the connectivity of the map of survivability. First, because the survivability map has the inherent property that once the survival regions are no longer connected, connectivity is permanently lost as the vehicle's remaining life runs out. Second, the continuity of survivability has a considerable impact on whether a vehicle can move from one survivable state to another. In the following sections, we will begin with some hypothetical scenarios and investigate key properties of the survivability map. Then we interpret the findings from the San Francisco Bay Area case study. The target safety threshold for all experiments is $1 - 1e - 8$.

5.4.1 The Continuity of the Survivability Map

The objective of the first experiment is to examine the continuity of the survivability map. In Figure 5.7, five airports are evenly distributed along the reverse diagonal, and each airport has a 10 km safe radius. Figure 5.7 depicts how an autonomous vehicle with 40 minutes of remaining flight time perceives the survivability in the airspace on a clear day (i.e., no traffic). Compared to figure (a), figure (b) depicts the same scenario with 100 partially known intruders. According to the results, two factors contribute to the discontinuity of the survivability map: (1) the lack of overlap between airport safety zones and (2) airspace

traffic flows (e.g., the white lines shown in figure (b)). In this scenario, survivable states are clustered within the safety zones of a single airport but divided up between airports. Due to a decreasing remaining lifetime, the autonomous vehicle's trajectory flexibility decreases over time. As a result, the survivability map's coverage will decrease. This motivates the second set of experiments.

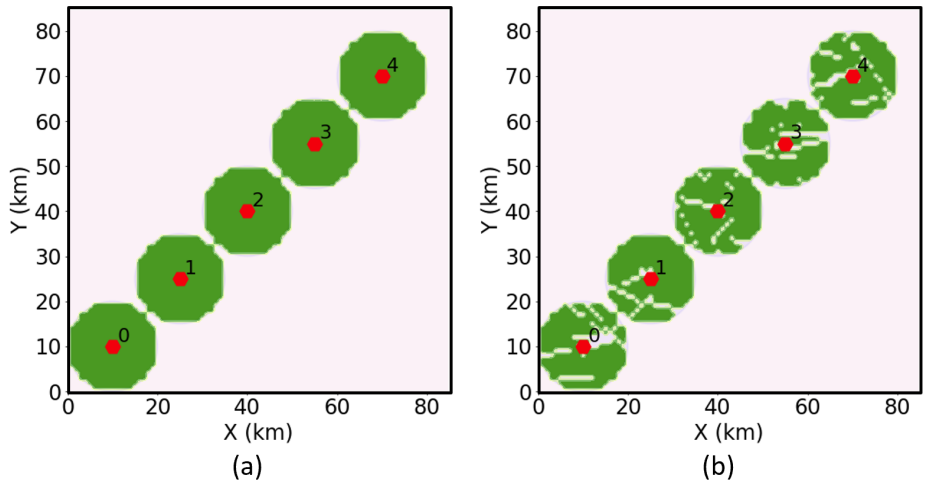


Figure 5.7: The survivability map is perceived by an autonomous vehicle with a remaining life of 40 minutes. The safe radius is 10 km. (a) The airspace contains no intruder traffic streams. (b)Consider 100 partially unknown intruders.

In the second set of experiments, we investigate factors that improve the coverage and connectivity of survivability map. First, we expand the airport's safe radius from 10 to 20 kilometers. As shown in Figure 5.8 (a), since the safety regions of five airports overlap between two consecutive airports, the resultant survivability on a clear day is also connected. The white lines between the survival zone and the airspace affected by intruders' traffic are still visible. The results of the last two sets of experiments indicate that, during the strategic planning phase, the allocation of airports in a city and the legal policy on determining the safe radius have a significant impact on the airspace's continuity of survivability. However, increasing the safe radius offers no obvious benefits on mitigating the disruption caused by air traffic. This leads to the next question, which is how to enhance the continuity of the survivability map caused by intruder traffic.

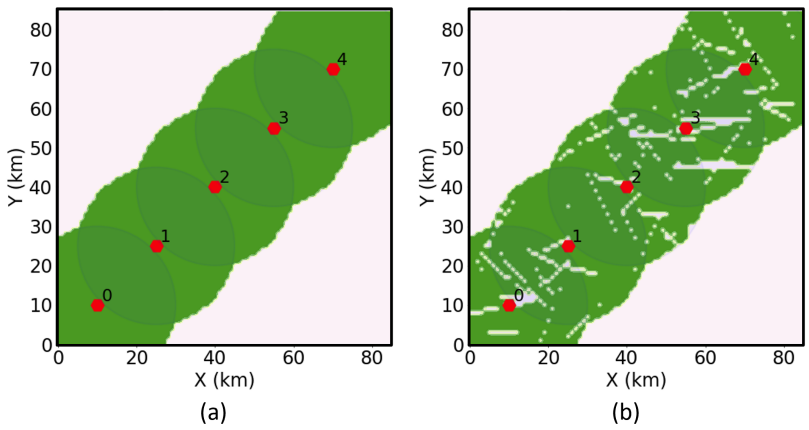


Figure 5.8: The survivability map perceived by an autonomous vehicle with a remaining life of 40 minutes. The safe radius is 20 km. (a) The airspace contains no intruder traffic streams. (b) Consider 100 partially unknown intruders.

The third set of experiments examines the influence of vehicle maneuverability and remaining lifetime on the autonomous vehicle's perception of airspace safety. In Figure 5.9 (a), we consider a scenario in which 500 partially known intruders are presented in a dense air traffic environment. A vehicle with a 40-minute battery life observes a glaring disconnect between airports 2 and 3. This observation indicates multiple air traffic streams whose trajectories are twisted in space and time and this twist has a profound effect on the disconnected region by blocking certain volume of airspace. Instead of increasing the safe radius of airports to improve the discontinuity of the survivability map, we increase the remaining life of autonomous vehicles, which is equivalent to "charging" the vehicle en route. The increase in remaining lifetime directly contributes to the increase of trajectory flexibility. Specifically, the total number of trajectories in each cell in space and time increases exponentially with time. This observation is consistent with the reality that, given more time, the vehicle is able to devise more diverse responses (i.e., trajectories) to airspace disturbances. By theory, the new set of plans should not be inferior to those created with a quicker response time.

The final set of experiments examines the effect of maneuverability on how the autonomous vehicle perceives the survivability of airspace. Consider two autonomous vehi-

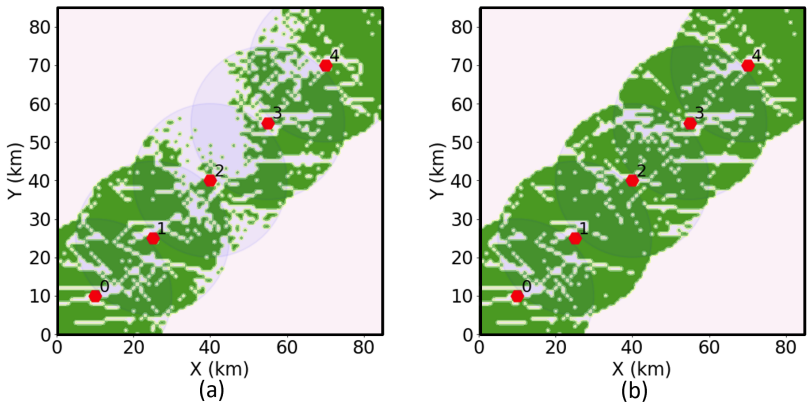


Figure 5.9: The survivability map perceived by an autonomous vehicle when 500 intruders are presented in the airspace. The safe radius is 20 km. (a) Remaining life of the vehicle is 40 minutes. (b) Remaining life of the vehicle is 50 minutes.

cles with identical remaining lifetimes of forty minutes. Nevertheless, the maneuverability of two vehicles is different. In particular, agent 1 can change its heading by $dhin[-30o, 30o]$ and its speed by $dvin[-10, 10]m/s$ in five discrete increments, totaling a 25-sized set of motion primitives. Comparatively, to agent 1, agent 2 can adjust its heading and velocity by the same bonds, but in eight equally spaced discrete increments, which constitutes a motion primitive set of 64. Figure 5.10 contrasts the survivability of two vehicles in the presence of 500 intruders. The color green represents agent 1 and the color red represents agent 2. The green map is superimposed on top of the red map (i.e., a red cell under each green cell). First, we note that the red and green maps are largely aligned, with the red map covering a slightly larger area. Figure 2 depicts a detailed zoom-in inspection of the area surrounding the airport (b). We observe that the red area provides greater coverage in the vicinity of the airport. This suggests that if two vehicles approach the same airport, the more maneuverable vehicle will have a greater chance of landing safely because it get access to a more survivable state. The same analysis can be applied to situations in which a vehicle with increased maneuverability has a greater chance of resolving conflicts with vehicles with lower maneuverability due to more choices of survivable states.

Observe that improving the discontinuity of the survivability map by increasing the vehicle's maneuverability has negligible effects in comparison to adding airports or increasing

the safe radius around airports. Alternatively, enhancing the maneuverability of vehicles may be more cost-effective than developing airport infrastructure or conducting scientific research on determining and certifying the safe radius, which could take longer and require more resources.

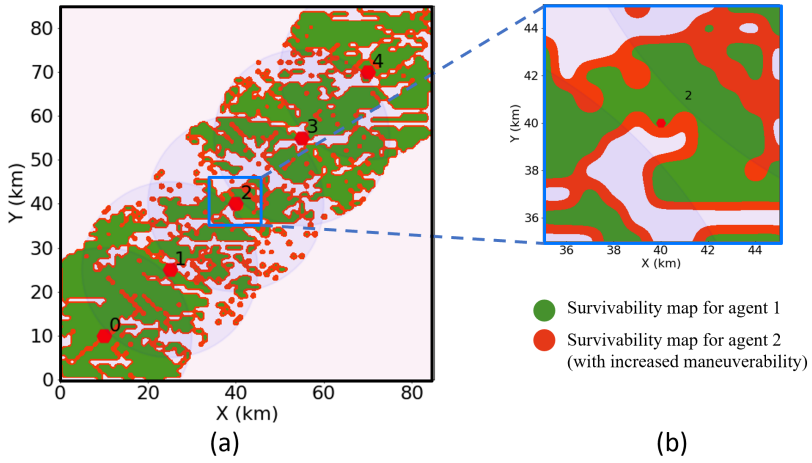


Figure 5.10: (a) The survivability map is perceived by two autonomous vehicles with different maneuverability when 500 intruders are presented in the airspace. The safe radius is 20 km and the remaining life of the vehicle is 40 minutes. Compared to agent 1, agent 2 has increased maneuverability.

5.4.2 Case Study of the San Francisco Bay Area

This section investigates the effects of airport locations and intruder traffic on vehicle survivability based in the San Francisco Bay Area. We consider a vehicle with standard dynamics as outlined in 5.3.2. Figure 5.6 shows the airport topology in the San Francisco Bay Area. Based on the type of airport, we further categorize the airports into three groups based on their daily operations:

- The 9 major airports:{large airports, medium airports, small airports}
- The 24 active airports:{large airports, medium airports, small airports, *vertiports*}
- In total 55 airports:{large airports, medium airports, small airports, vertiports, *closed airports*}

- The 16 pseudo-airports. In the San Francisco Bay Area, these airports do not exist physically. They are generated randomly and distributed uniformly in 2D space. Due to their comprehensive coverage of the area of interest, the resulting survivability map considering only uniform airports can be used as a benchmark against which other survivability map can be compared.

We assume that each airport, regardless of its type, has a maximum of four landing strips. The radius of safety is 20 km. Regarding intruder traffic, only partially known intruders with an uncertainty radius of 1 km are considered. Based on the number of intruders, we classify intruder traffic into three different congestion levels: (a) light traffic with 200 intruders, (b) moderate traffic with 400 intruders, and (c) dense traffic with 600 intruders.

Figure 5.11 depicts a variation of the survivability map for the San Francisco Bay Area as a function of the number of airports and the level of traffic congestion. In a particular row, we maintain the same traffic density and only consider major airports only, active airports only, all airports and uniform pseudo-airports. In the vertical direction, we keep the same topology of airports and increase the traffic density from the light traffic to dense traffic. Notice that the purpose of presenting column (b) is to provide a reference map where the geometries of airports including its safety zones are fully connected.

First, in the first column of figures, we confirm our observation that increasing the number of airports significantly improves the map's coverage. It is interesting to note that even after opening all 55 existing airport facilities, the bottom left corner of the vehicle still lacks survivability. By contrast, we can fully cover the airspace by uniformly distributing 16 pseudo-airports throughout the space. This indicates that the geometrical distribution of airports has a significant effect on the allocation of safe states in the airspace, in addition to the number of airports. With a well-designed airport topology, a vehicle can maximize its survivability over a greater distance by maximizing its use of valid landing space.

Second, by referring to figures column-wise, we further confirm our previous observation that as the intruder traffic density increases, the vehicle perceives the airspace with

reduced coverage of survivability. In the extreme case where 600 vehicles are presented in the environment, the airspace suffers from excessive access and extensive uncertainty coming from vehicle trajectories, which leads to a substantial decrease in the number of feasible and robust trajectories to any valid landing site. Even if all 55 airports were to open their facilities, the lack of safety in the airspace would be difficult to restore.

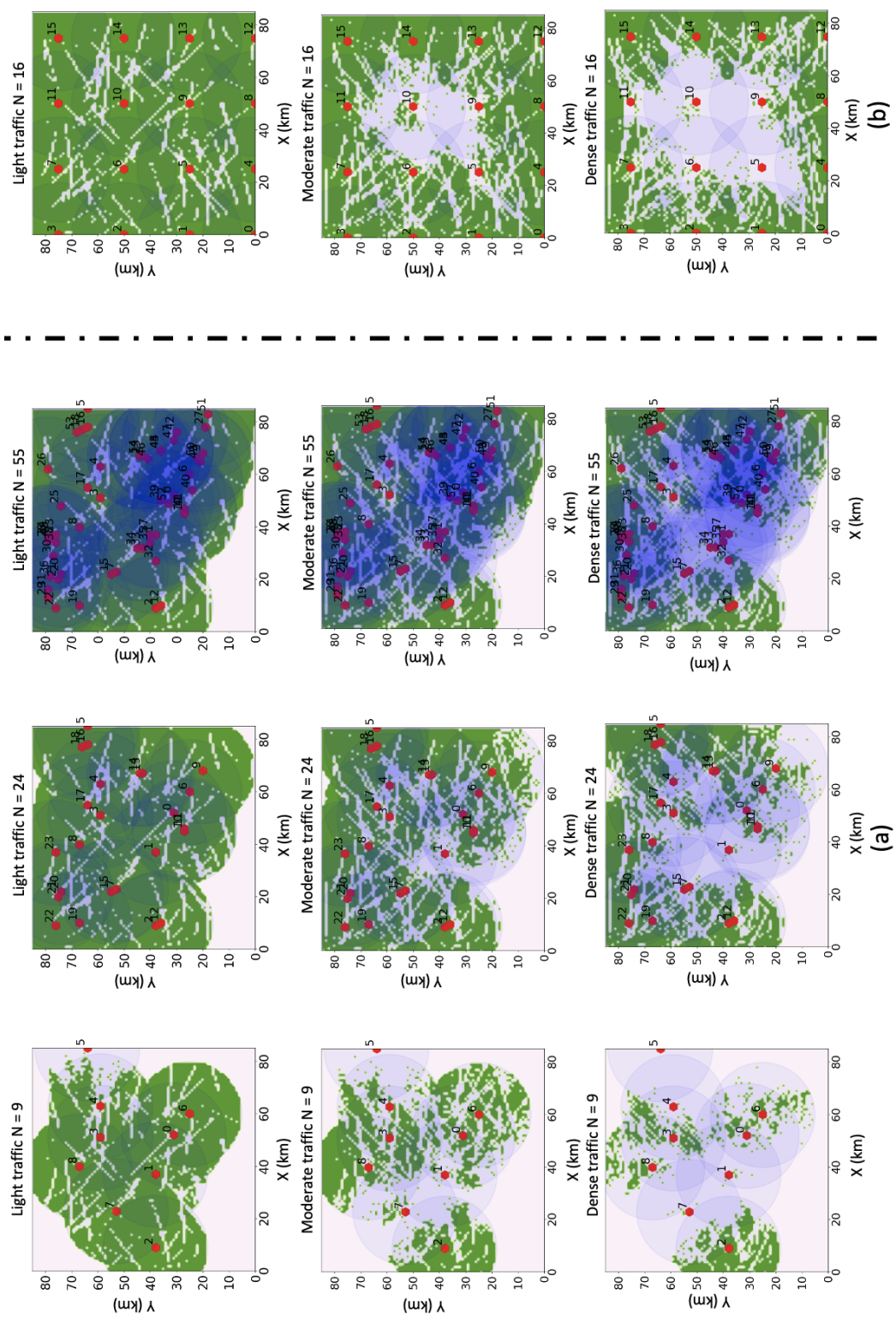


Figure 5.11: Survivability map depending on the number of open airports and the level of traffic congestion.

CHAPTER 6
TRAJECTORY PLANNING FRAMEWORK

6.1 Overview

Trajectory planning is a particularly challenging task for autonomous vehicles when there are moderate to extreme uncertainties in their operating environment, i.e., where the trajectories of hazards are *partially known* to *completely unknown*. In this chapter, we propose a receding horizon control strategy with novel trajectory planning policies that enable dynamic updating of the planned trajectories of autonomous vehicles. The proposed policies utilize two metrics: (1) the number of feasible trajectories; and (2) the robustness of the feasible trajectories. We measure the effectiveness of the suggested policies in terms of mission survivability, which is defined as the probability that the primary mission is accomplished or, if that is not possible, the vehicle lands safely at an alternative site. We show that a linear combination of both metrics is an effective objective function when there is a mix of partially known and unknown uncertainties. When the operating environment is dominated by unknown disturbances, maximizing the number of feasible trajectories results in the highest mission survivability. These findings have significant implications for achieving safe aviation autonomy.

6.1.1 Related Work

Several methods have been proposed to characterize and/or bound the uncertainty associated with dynamic obstacles and thereby account for their occurrence and motion. However, it is difficult to characterize or for that matter bound the behavior of highly unpredictable occurrences, or model unobserved or complex dynamics (such as non-cooperative intruder traffic or a flock of birds crossing the path of a vehicle) [81]. In this study, we

focus on the uncertainty caused by the trajectories of uncertain intruder traffic.

To date, researchers who have studied the problem of trajectory planning for highly autonomous vehicle in uncertain operating environments have focused on scenarios where the uncertainty can be characterized statistically. This typically involves two steps: (1) predicting and estimating the potential impacts of the uncertainty on the feasibility of a trajectory; and (2) determining the trajectory that minimizes the predicted risk. For example, Bry and Roy [66] proposed a graph-search based algorithm–the Rapidly-exploring Random Tree algorithm–where, given a nominal trajectory, distributions for future states of the vehicle are first predicted to assess whether the probability of collision, given a state, is bounded below a threshold value. Next, a set of trajectories is incrementally constructed while efficiently searching for the best candidate path. Paths are evaluated based on their probability of being realized by a closed-loop controller. Similar work that focuses on graph-based search algorithm can also be found in [67] and [68]. Separately, roadmap-based approaches that rely on an understanding of the safe states in the environment (or the configuration space in general) have attracted significant research interest. Typically, they utilize one of three techniques: visibility graphs [69], Voronoi diagrams [70], or potential fields [71].

The trajectory flexibility metrics proposed by Idris et al. [82] provide the basis for a promising approach to improving the ability of autonomous vehicles to adapt to unknown disturbances. Specifically, research work [83] demonstrated that self-separation and self-organizing behaviors may be induced among autonomous agents, and traffic complexity reduced by maximizing trajectory flexibility. Further, building on that research, the authors leveraged adaptability, one of the trajectory flexibility metrics, to estimate airspace capacity under different control schemes [84].

6.1.2 Contribution

In this chapter, we contribute to the literature by proposing a receding horizon control strategy with a set of novel trajectory planning policies that enable the autonomous vehicle to dynamically update its planned trajectory in environments where potential conflicts are, from a statistical perspective, either *partially known* or completely *unknown*. Most importantly, we demonstrate that maximizing the total number of feasible trajectories is effective in mitigating the consequences of extreme uncertainty.

6.2 Methodology

6.2.1 Receding Horizon Control

An autonomous vehicle using a fixed horizon control scheme optimizes a sequence of control actions $a_1, a_2, \ldots, a_T$ over T steps. If, during the T steps, unexpected events occur or the system behaves differently than was expected during the design of the control scheme, the controller will not be able to account for them. This shortcoming can be addressed through Receding Horizon Control (RHC) where control actions are repeatedly optimized over a moving time horizon. Specifically, the controller generates the optimal control inputs over M time steps and executes the first control action. At the next time step, a new control problem with the most recent environmental information will be solved for the remaining M-1 time steps [85]. We summarize the general RHC optimization problem as follows.

$$\min_{z} \quad F(s_t, a_t) + \beta \sum_{k=0}^{T} V(c_{t+k}) \tag{6.1}$$

$$\text{s.t.} \quad s_{t+k+1} = f(s_{t+k}, a_{t+k}) \tag{6.2}$$

$$s_t = (x_t, y_t, h_t, v_t, t) \tag{6.3}$$

$$c_{t+k} = c(x_{t+k}, y_{t+k}, t+k) \tag{6.4}$$

$$c_{t+k} \notin \mathcal{B}(t) \tag{6.5}$$

$$s_{t+T} \in \Omega(t) \tag{6.6}$$

$$a_{t+k} = (dh_{t+k}, dv_{t+k}) \in \mathcal{A} \tag{6.7}$$

$$s_{t+k} \in \mathcal{R}^3 \times [v_l, v_u] \times [t, t_{f_u}] \tag{6.8}$$

where we optimize over control commands $z = \{a_t, \ldots, a_{t+T-1}\}$ for the remainder of the mission given the current knowledge of the environment stored in $\mathcal{B}(t)$ at time t. However, only the first command a_t is executed. This process is then repeated until the vehicle reaches the goal, or terminated if no feasible solution was founded. The vehicle dynamics are prescribed in Constraint (6.2) with the initial state given by constraint (6.3). The vehicle state is mapped to a cell in discrete space and time in Constraint (6.4). In addition, we ensure that a vehicle does not cross a blocked region $\mathcal{B}(t)$ via constraint (6.5). Constraint (6.6) ensures that the end state is one of the valid terminal states in the set $\Omega(t)$. Note that set $\Omega(t)$ consists of only goal-landing sites at the start of the flight. However, if no feasible trajectory to the goal-landing site is available, the set $\Omega(t)$ will be changed to include alternative landing sites. This ensures that priority is given to the completion of a mission over its survival. Constraint (6.7) limits actions to the set of motion primitives $\mathcal{A}$. The state-space is specified by constraint (6.8) where the speed of the vehicle is explicitly bounded by $[v_l, v_u]$ and the mission duration is upper bounded by t_{f_u}.

The objective function (6.1) is made up of two parts. The first term $F(s_t, a_t)$ represents the robustness of the first trajectory segment, which is the outcome of applying control a_t in state s_t. The second term $V(c_{t+k}), k = \{0, \ldots, T-1\}$ measures the quality of the k-th way point by evaluating the goodness of its corresponding cell. A discount coefficient $\beta \in [0, 1]$ is applied to the second term to adjust the weighting between instant and future response. Generally, the discount can be a function of time, which is not considered in this paper.

6.2.2 Trajectory Planning Policies

A data preparation process includes computing the following parameters:

- $c_t.\mathcal{L}$: Feasible trajectory set available at cell c_t via the Backtracking algorithm intro-

duced in section 4.4.

- $c_t.N_f$: Cardinality of the feasible trajectory set $c_t.\mathcal{L}$.
- $p_{c_t c_{t+1}}$: The robustness of a segment connecting cell c_t to cell c_{t+1} (probability of segment being feasible) using the MC-simulation introduced in section 4.5.
- $\mathcal{P}_{l_i}, \forall\, l_i \in c_t.\mathcal{L}$: Robustness of trajectory l_i.
- $\mathcal{P}^{survive}(c_t.\mathcal{L})$: Survivability of the mission.

The first policy is to maximize the robustness of the resulting trajectory to partially known disturbances.

$$(\pi_R) \quad p_{c_t c_{t+1}} + \beta \sum_{k=1}^{T} P^{succeed}(c_{t+k}.\mathcal{L})$$

Motivated by a situation where there are only unknown events or estimates of segment robustness are unavailable, the second policy considers the maximization of the number of trajectories only.

$$(\pi_{N_f}) \quad c_{t+1}.N_f + \beta \sum_{k=2}^{T} c_{t+k}.N_f$$

The third policy takes into account both the robustness of a trajectory and the total number of alternative trajectories available at each way point to maximize mission survivability. The resulting trajectory attempts to provide at least one feasible trajectory to guide the vehicle to a safe landing at any point along the trajectory.

$$(\pi_S) \quad p_{c_t c_{t+1}} + \beta \sum_{k=1}^{T} \mathcal{P}^{survive}(c_{t+k}.\mathcal{L})$$

The fourth policy serves as an uninformed baseline against which other policies can be measured, where the vehicle chooses one of the feasible paths at random. The objective function maximizes the chances of surviving the mission under this assumption.

$$(\pi_{Avg(R)}) \quad p_{c_t c_{t+1}} + \beta \sum_{k=1}^{T} \left(\frac{1}{c_{t+k}.N_f} \sum_{l_i \in c_{t+k}.\mathcal{L}} \mathcal{P}_{l_i} \right)$$

6.3 Experiment Setup

Our computational experiments had two goals: First, to show how the proposed trajectory planning framework can be used to direct a mission. Second, to compare the effectiveness of our four planning policies in mitigating varied levels of uncertainty. The experiment is set up as follows.

Mission We are interested in a mission in the San Francisco Bay Area which is specified as an $85 \times 85\ km^2$ area as shown in Figure 6.1. A vehicle takes off from the Santa Clara Towers Heliport and heads northwest to land at the UCSF Helipad. We consider two alternative landing sites: San Francisco International Airport and Stanford Hospital Heliport. The flight mission includes a succession of three phases: takeoff, cruising, and landing. All three phases may involve hovering.

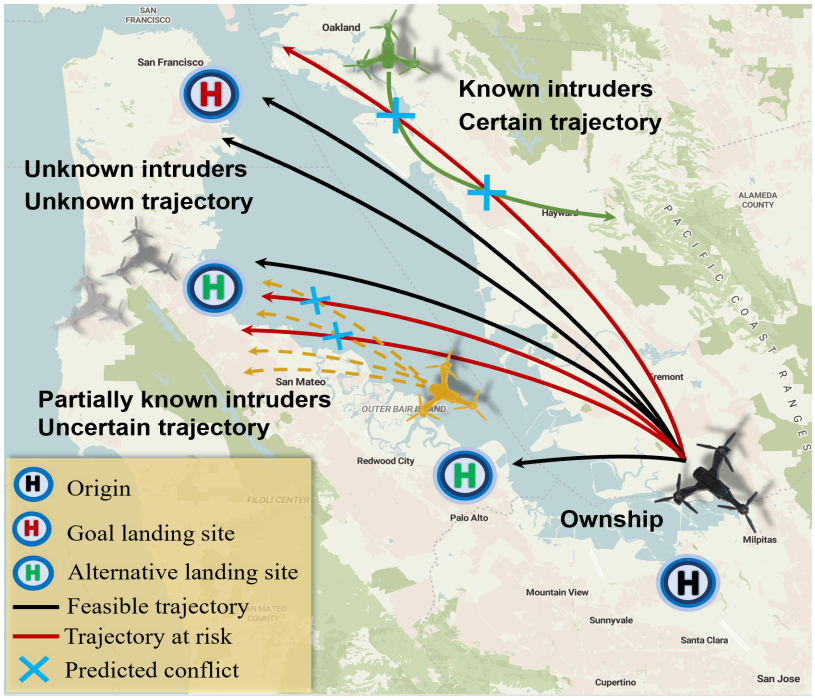


Figure 6.1: Illustration of the three types of intruders present in the operating environment.

The departure time is $t_0 = 00:00$ minutes, and the required arrival time at the goal landing site is $t_f = [00:40, 00:55]$ minutes. We assume that the vertical take-off and hover, vertical landing and hover phases each take 5 minutes. In addition, the vehicle enters and exits in the cruising phase with a speed of 0 m/s. Due to the topology of the destination helipad, the vehicle must maintain its headings at $h_f \in \{90^o, 135^o, 180^o\}$ upon arrival at the goal landing site before initiating a vertical descent. When the vehicle arrives at an alternative landing site, the heading requirement changes to $h_f \in \{90^o, 180^o, 270^o\}$.

During the cruising phase, we assume that the vehicle's cruise speed is in the range $[0, 40]\ m/s$. At each replanning window, the vehicle can change its heading by $dh \in [-30^o, 30^o]$, in 5 equally spaced discrete increments, and its speed by $dv \in [-10, 10]$ m/s, in 5 equally spaced discrete increments. Therefore, the motion primitive set is composed of 5×5 different combinations of heading-velocity changes. We divide time and space into cells of size $(1\ km, 1\ km, 1\ minute)$. As a result, the mission entails 6, 7, 8, or 9 decision-making windows of 5-minute duration. We assume that the maximum flight endurance is 60 minutes, which is equivalent to 12 decision-making windows.

Environment Intruders are of two types: (1) Observed intruders with uncertainty radius $r = 5\ km$ whose trajectories are partially known to the ownship; and (2) Unobserved intruders whose presence is unknown to the ownship. Our experiment assesses 36 different traffic scenarios. Each scenario is characterized by the total number of partially known intruders $N_p \in \{50, 100, 150, 200, 250, 300\}$, and the total number of unknown intruders $N_u \in \{50, 100, 150, 200, 250, 300\}$.

Simulation Setup For each traffic scenario characterized by (N_p, N_u), we generate $N_{instance}$ different traffic instances. For each instance, N_p trajectories are randomly generated with different entrance and exit points in space and time. In addition, we generate a set of N_u trajectories that are unknown to the ownship during trajectory planning but actually exist in the simulated environment. Given a traffic instance, we simulate the real traffic between

t_0 and t_{f_u}. The ownship plans its trajectory by employing each of the policies π_{N_f}, π_S, π_R, and $\pi_{Avg(R)}$. The computation of π_S, π_R, and $\pi_{Avg(R)}$ involves measuring the robustness of feasible trajectory segments via MC simulation introduced in section **??**. The three possible outcomes of a mission are: success by arriving at the goal landing site (G), survival by arriving at one of the two alternative landing sites (A), or failure, which involves the more risky situation of an emergency landing at an unplanned location (L). The survivability of a mission under traffic scenario (N_p, N_u) when employing a specific policy is estimated by the ratio of the number of outcomes (G) and (A) out of $N_{instance}$. Similarly, the success of the mission is measured by the ratio of the number of outcomes (G) out of $N_{instance}$.

Convergence Analysis The convergence analysis is based on the principle of utilizing sample proportion to estimate an unknown population proportion, as well as quantifying uncertainty in a population proportion estimate. First, we investigate the case of utilizing direct MC simulation to estimate the robustness of a trajectory segment, i.e., the likelihood of a trajectory segment being feasible. Given a segment of interest, let E denote the event that the segment is feasible. The robustness of a segment p is defined as the expected value of the indicator function $\mathbb{I}_E$, that is $p = Exp(\mathbb{I}_E)$. The variance of $\mathbb{I}_E$ is $p(1-p)$. To estimate p, we generate n independent samples $x_1, x_2, \cdots, x_n$ of $\mathbb{I}_E$ using MC simulation and compute the sample mean.

$$\hat{p}_n = \frac{1}{n}\sum_{1}^{n} x_i$$

In this example, we refer to p as the population proportion and $\hat{p}_n$ as the sample proportion. We can further use sample proportion $\hat{p}_n$ to approximate the sample variance of $\mathbb{I}_E$, that is $s_n^2 = \hat{p}_n(1-\hat{p}_n)$. To quantify the uncertainty/error associated with the estimator $\hat{p}_n$, we leverage the central limit theorem to construct a confidence interval associated with $\hat{p}_n$, with the chance of p falling inside this interval equal to $1-\alpha$. The confidence interval is given as follows [86].

$$\hat{p}_n \pm z_c \frac{s_n}{\sqrt{n}} = z_c \frac{\sqrt{\hat{p}_n(1-\hat{p}_n)}}{\sqrt{n}}$$

Where $z_c = 1.96$ for a common choice of confidence level $1 - \alpha = 95\%$. Notice that when $\hat{p}_n = 0.5$ the half uncertainty band $h = z_c \frac{\sqrt{\hat{p}_n(1-\hat{p}_n)}}{\sqrt{n}}$ attains its maximal value and therefore is upper bounded by $h_{max} = z_c \sqrt{\frac{0.25}{n}}$. In our experiment, given the computation time for obtaining one data sample and the desired accuracy of estimator $\hat{p}_n$, it is sufficient to limit h_{max} within 2% with confidence level 95%. To achieve this goal, in theory, we need at least $n = 2401$ samples to obtain a point estimate $\hat{p}_n$ of p. Therefore, in the case of using MC simulation to evaluate the robustness of a trajectory segment, we set the number of MC simulation runs to $N_{mc} = 2500$. As an example, we consider measuring the robustness of three[1] independent trajectory segments in an environment with 100 partially known intruders. As shown in Figure 6.2, for each segment, the 95% confidence interval associated with the trajectory robustness estimate shrinks as N_{mc} increases. Moreover, when $N_{mc} \geq 2500$, the half uncertainty band is smaller than 2% for all three segments.

Given a traffic scenario, we further assess the convergence of a policy's performance measure, such as mission survival rate, with respect to the number of samples $N_{instance}$ used to estimate performance metrics. Consider measuring the mission survival rate when the vehicle employs policy π_S in a traffic scenario parameterized with (N_p, N_u). In this example, let $E^{'}$ indicate the occurrence in which the vehicle survives a mission. The mission survival rate $p^{'}$ is defined by the expected value of the indicator $Exp(\mathbb{I}_{E'})$. With n independent samples of $\mathbb{I}_{E'}$, we can then use the sample average $\hat{p^{'}_n}$ to estimate mission survival rate $p^{'}$. Following the same logic as the MC simulation, we can theoretically obtain a half uncertainty band $\leq 2\%$, centered on the mission survival rate estimate $\hat{p^{'}_n}$, with a 95% confidence level by utilizing at least 2401 samples of $\mathbb{I}_{E'}$. Therefore, to assess the performance of a trajectory planning policy given a traffic scenario, we set the number of traffic instances to $N_{instance} = 2500$. As shown in Figure 6.3, we consider employing policy π_S in five different traffic scenarios where the ratio of N_p to N_u is kept constant at one while the total number of intruders in the system grows linearly from 100 to 500.

[1]Three segments are surrounded by relatively light, moderate, and severe traffic, respectively.

When $N_{instance} \geq 2500$, the half uncertainty band associated with the mission survival rate estimate is smaller than 2% across all traffic scenarios.

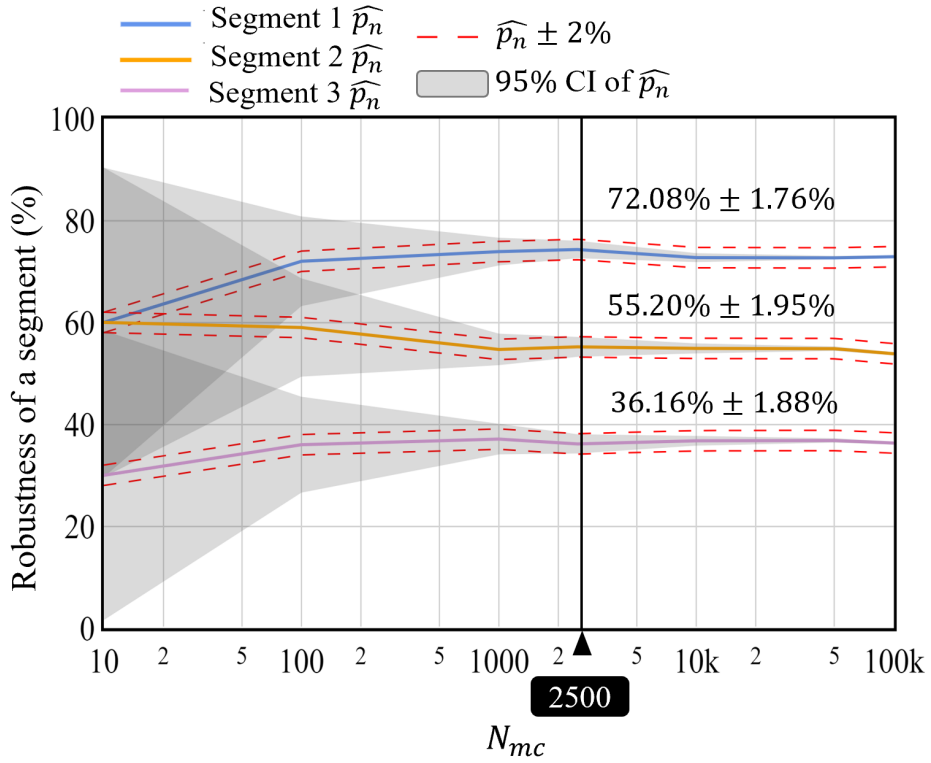


Figure 6.2: The estimate of the robustness of three independent trajectory segments via MC simulation.

6.4 Results and Discussions

6.4.1 Comparison between Mission Success and Survival Rates

In the first set of analyses, we examine how the traffic volume affects mission robustness and survivability. For this purpose, we focus on two critical performance indicators: mission success rate, which indicates mission robustness, and mission survival rate, which implies mission survivability. The results are summarized in Figure 6.4 for five traffic scenarios $(N_p, N_u) = \{(50, 50), (100, 100), \ldots, (250, 250)\}$ where the ratio of partially known and unknown intruders is constant at 1. Along the x-axis, the total number of intruders N_{total} in the environment increases from 100 to 500 corresponding to the five traffic

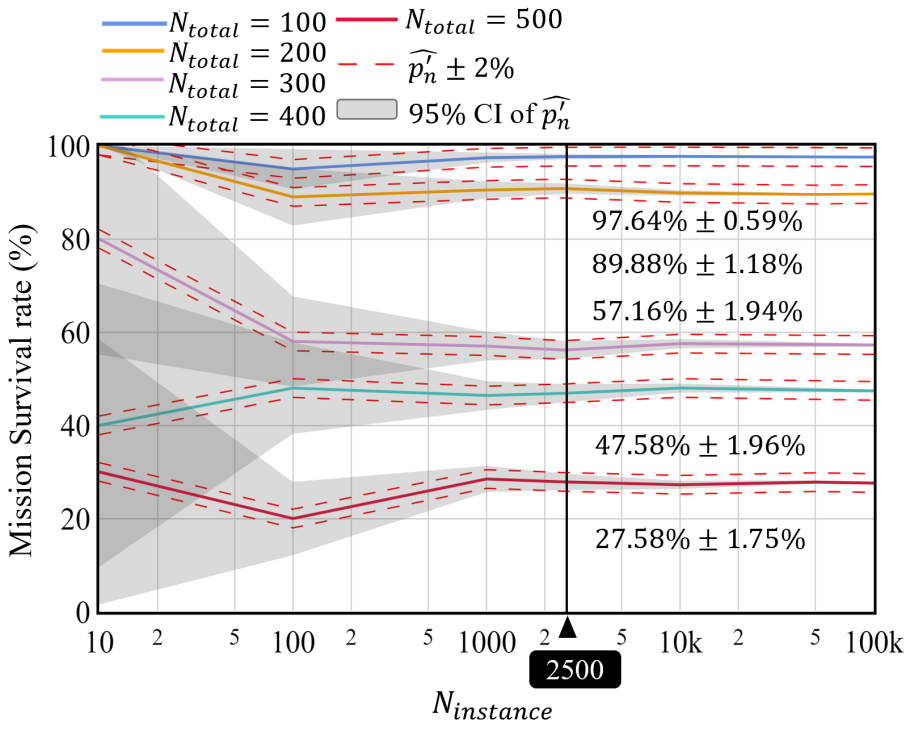


Figure 6.3: The estimate of mission survivability when employing policy π_S in five different traffic scenarios.

scenarios. Each traffic scenario is associated with a group of bars, and each bar in a group reports the vehicle's survival and success rates employing one of the four proposed planning policies including π_{N_f}, π_S, π_R, and $\pi_{Avg(R)}$ as introduced in section 6.2.2.

First, as expected, the mission success and survival rates decline as more intruders are introduced into the environment regardless of the policy that has been employed in trajectory planning.

Second, allowing a vehicle to land at alternative landing sites increases flight safety significantly. This is shown by positive improvements in mission survival rate compared to mission success rate across all columns. In the first set of grouped bars, for example, the orange bar indicates that when the vehicle flies through the airspace that contains 100 actively operating intruders while enforcing policy π_S, the vehicle has a 28.24% higher probability of landing safely with only two additional landing sites provided.

Third, policy π_R may be sufficient to maximize mission success rate at low traffic vol-

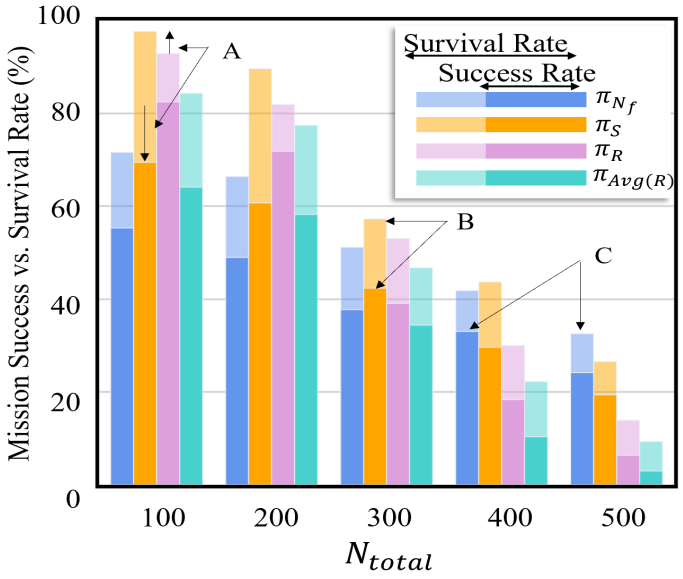


Figure 6.4: Mission Success Rate vs. Mission Survival Rate.

ume, but this is achieved at the expense of the survival rate, which is improved by also considering the number of trajectories per policy π_S (see A). Considering the number of feasible trajectories becomes more effective as the traffic volume increases, as indicated by the fact that π_S is the best policy for both success and survival at 300 (see B), and the policy π_{N_f} is best for mission success at 400 and for survival at 500 (see C). These observations motivate us to further explore in section 6.2.2 the policies π_S and π_{N_f} in terms of their ability to mitigate uncertainty that is either partly known or unknown; and to concentrate on mission survival for the remainder of the study.

6.4.2 Effectiveness of Policies π_S and π_{N_f}

We observed that policy π_{N_f} offers no significant benefit in protecting the ownship from partially known uncertainty. The is borne out by the comparison in Figure 6.5 of the mission survival rate of the four policies when the number of unknown intruders in the environment is kept constant at 100 and the number of partially known intruders is increased from 50 to 300. It is apparent that employing policy π_S results in the highest mission survival rate across all 5 traffic scenarios. This promising result can be viewed as a by product

of the Theorem 4.3.1. In particular, maximizing the total number of feasible trajectories and improving the robustness of a trajectory increase the likelihood that the ownship will maintain at least one trajectory that will lead to a safe landing.

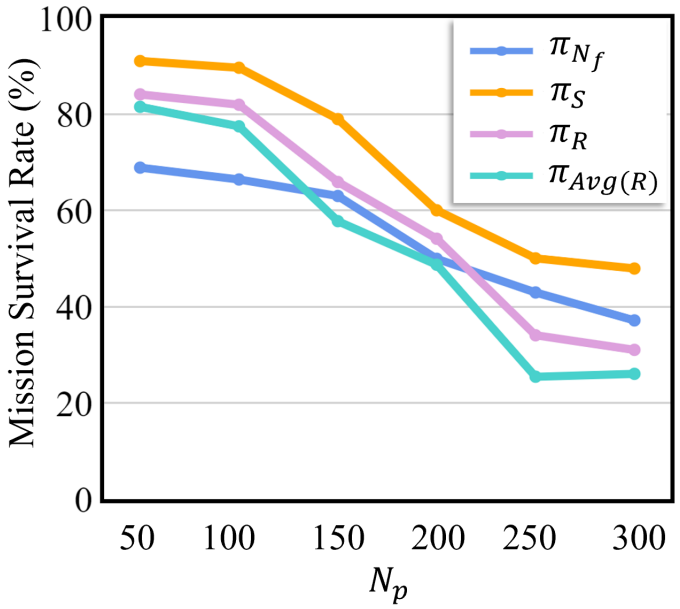


Figure 6.5: Performance of four trajectory planning policies as a function of the number of partially known intruders. Note that the number of unknown intruders is kept at 100.

Second, the policy π_{N_f} is shown to be the most effective policy in mitigating extreme uncertainty - the unknown. As shown in Figure 6.6, where we present the performance of the proposed policies while keeping the number of partially known intruders at 100 and increasing the number of unknown intruders from 50 to 300, we observe a gradual decline in the mission survival rate as more unknown intruders are introduced into the environment no matter which policy is applied.

The phenomenon that occurs when the blue curve (i.e., policy π_{N_f}) and the yellow curve (i.e., policy π_S) intersect at the red dot corresponds to a traffic scenario with around 229.9[1] unknown intruders. This suggests that given a fixed number of partially known intruders in the environment N_p, policy π_{N_f} provides marginal protection against unknown uncertainty over all other policies when the number of unknown intruders exceeds the threshold value

[1]For comparison purpose, we keep the number to the nearest tenth. In real application, this number should be an integer.

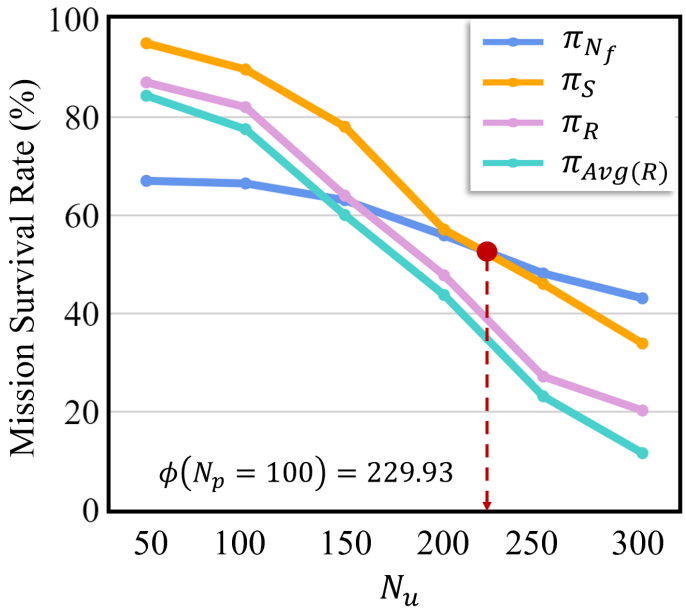


Figure 6.6: Performance of proposed trajectory planning policies as a function of the number of unknown intruders. Note that the number of partially known intruders is kept at 100.

$\phi(N_p)$. Therefore, it is beneficial to switch to the policy π_{N_f} at that threshold. As shown in Figure 6.6, $\phi(N_p = 100) = 229.9$.

The intersection of the blue and yellow curves was identified for values of N_p between 100 and 300, and the corresponding policy switching threshold $\phi(N_p)$ for the different values of N_p are plotted (see red line) in Figure 6.7. As may be seen, policy π_{N_f} outperforms policy π_S for traffic situations in the blue area where $N_u > \phi(N_p)$. This suggests that policy π_{N_f} is highly effective when the level of uncertainty is moderate to extreme (i.e., when there are more unknown intruders than partially known intruders). We posit that this is due to increased difficulty measuring mission robustness and survivabiity in a highly unpredictable environment (because it only accounts for partially known intruders), and that maintaining the number of trajectories is a more effective strategy when accounting for unknown risks.

Additional evidence for the effectiveness of policy π_{N_f} is provided in Figure 6.8. In this study, we set the total number of intruders in the environment to $N_{total} = 300$ while adjusting the ratio of unknown intruders in $[\frac{50}{300}, \frac{100}{300}, \cdots, \frac{250}{300}]$, corresponding to traffic scenarios

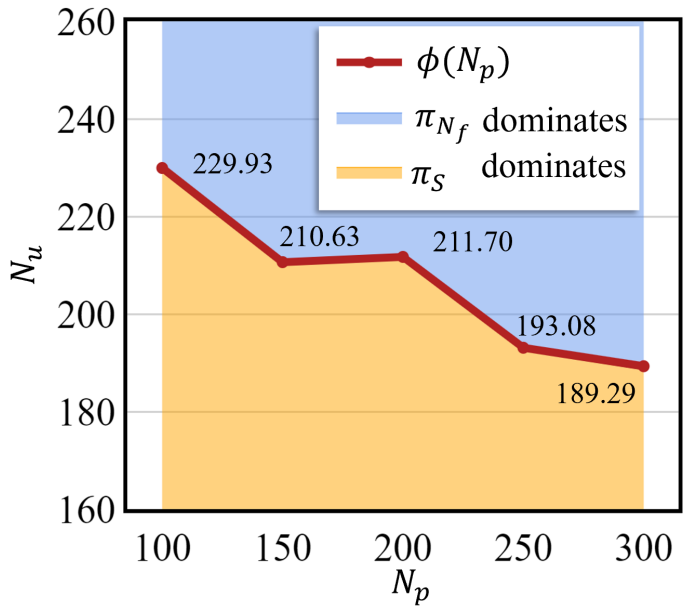


Figure 6.7: Policy switching boundary as a function of the number of unknown intruders.

$(N_p, N_u) = \{(50, 250), (100, 200), \cdots, (250, 50)\}$. We ignore the impact of traffic congestion and focus exclusively on the relative severity of the uncertainty. As the ownship's environment becomes more uncertain, policy π_S, π_R, and $\pi_{Avg(R)}$ become less effective, as evidenced by a lower mission survival rate. What stands out in this figure is the increasing trend in mission survival rate associated with policy π_{N_f}. The rising blue curve intersects the orange curve at a point corresponding to when 64.4% intruders in the environment are unknown. Hence, the policy π_{N_f} performs better than policy π_S when more than 64.4% intruders in the environment are unknown. These observations imply that policy π_{N_f} is highly competitive in mitigating extreme uncertainty caused by an increased ratio of unknown events.

The critical percentage $\eta(N_{total})$ of unknown intruders beyond which policy π_{N_f} becomes the most effective is shown in Figure 6.9 as a function of traffic volume (see red line). As may be seen, $\eta(N_{total})$ decreases with increasing value of N_{total}, which (as the red line) indicates that as traffic becomes more congested, policy π_S becomes less tolerant of unknown intruders – $\eta(N_{total})$ drops from 64% to 38%. The blue region constitutes traffic scenarios with a large percentage of unknown intruders in the environment. We observe

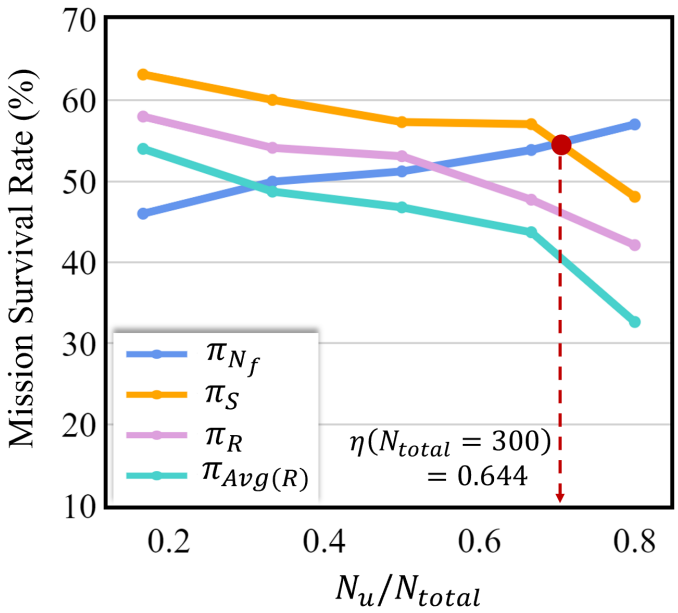


Figure 6.8: Performance of proposed trajectory planning policies as a function of the ratio of unknown intruders in the environment. Note that the total number of intruders is kept at 300.

that policy π_{N_f} results in the highest mission survival rate for traffic scenarios falling in the blue region.

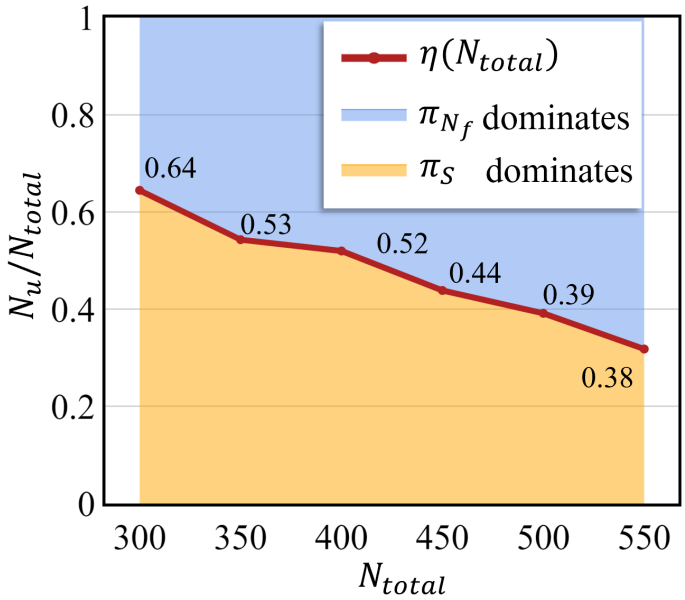


Figure 6.9: Policy switching boundary as indicated by the ratio of unknown intruders.

6.5 Conclusion

In this chapter, we studied the trajectory planning problem under uncertainty due to partially known and unknown intruders. We demonstrated that the success and survivability of a mission from a given point in space and time are dependent on two metrics: (1) the number of feasible trajectories available at that point, which can be computed using the proposed backtracking algorithm; and (2) the robustness of each of the trajectories, which can be evaluated using a Monte-Carlo simulation. We performed experiments for 36 traffic scenarios with varying numbers of partially known and unknown intruders. Our findings indicate that a policy π_S that combines the two metrics by maximizing the probability of having at least one feasible trajectory, outperforms all other policies when the vehicle is exposed to partially known uncertainty and moderate levels of unknown uncertainty. When uncertainty is dominated by unknown intruders, however, policy π_{N_f}, which maximizes the first metric, yields the highest success and survival rates.

In summary, part II of the thesis discusses how unmanned aerial vehicles (UAVs) can manage complex missions safely and autonomously in an uncertain environment. We establish a rigorous understanding of the qualitative and quantitative understanding of the survivability of an UAV. Moreover, the impacts of airport geometry, vehicle maneuverability and its remaining lifetime, and air traffic are empirically investigated. Finally, we propose four trajectory planning policies and whose effectiveness in mitigating uncertainty are carefully assessed.

Part III

Conclusions

CHAPTER 7
CONTRIBUTIONS AND FUTURE WORK

In this chapter, we conclude the thesis by highlighting our major contributions to the research community and outline potential future research directions.

7.1 Summary of Contributions

The purpose of this thesis is to improve the scalability, efficiency, and safety of future AAM systems through the development of core decision-making tools for the routing and trajectory planning of unmanned aerial vehicles.

Part I of the thesis focuses on a specific type of vehicle routing problem known as the Coordinated Vehicle Routing Problem. The problem's complexity stems from its combinatorial nature.

In chapter 2, we carefully assess the operational conditions for real-world practice and develop a Mixed Integer Quadratically-constrained programming model known as Nested-VRP. In the following ways, we contribute to the literature and fill a research gap:

- To the best of our knowledge, we are the first to provide answers to these questions via a single formulation that incorporates the following real-world considerations: (a) non-zero surveillance times at locations; (b) flight endurance limitations; (c) the requirement that the truck must arrive at the rendezvous location before the drone battery charge has expired; (d) the truck is allowed to perform a battery swap while shipping the drone from one location to the other; and (e) a non-zero battery swapping time.

- We have shown that the proposed Nested-VRP model is more compact than the state-of-the-art model.

In chapter 3, we investigate both exact and heuristic approaches to solving the Nested-VRP problem.

- We employ linearization and constraint strengthening techniques to enhance the model's performance further. We are able to expand our knowledge of how traditional model strengthening techniques can accelerate the process of solving an advanced routing model.

- We offer novel insights into the complexity of the Nested-VRP model with and without prior knowledge of drone routing. We provide an absolute lower bound on the objective function of the Nested-VRP model as a benchmark for evaluating the quality of heuristic solutions.

- We propose an effective Neighborhood Search (NS) heuristic to solve the Nested-VRP. Although the NS heuristic is widely studied in solving combinatorial optimization problems, the proposed heuristic includes innovations in evaluating the goodness of local geometry by measuring how efficiently the drone battery can be used in nested units.

- We conduct extensive computational experiments from which we empirically examine the improvement of the Nested-VRP model by applying linearization and constraint strengthening techniques, demonstrate the effectiveness and efficiency of the proposed NS heuristic, and extract valuable insights for practitioners.

In part II of the thesis, we study a trajectory planning problem. The difficulty of the problem stems from the need for UAVs to mitigate partially known and unknown uncertainty in the environment.

In chapter 4, we introduce the concepts of robustness and flexibility and explain how their combination can provide autonomous vehicles with the resilience to withstand adverse conditions. We also propose a set of design and performance metrics to enhance a

vehicle's safety and survivability awareness. Methodologies for computing these metrics are proposed. This chapter makes the following contributions to the literature:

- We establish a qualitative and quantitative understanding of flexibility and robustness in terms of its ability to reduce uncertainty. The concept of "flexibility" originated in economics and was subsequently introduced to other fields, such as management science and industrial engineering. We are one of the first authors to define trajectory flexibility in the context of trajectory planning and investigate its potential for mitigating uncertainty.

- We propose a backtracking algorithm to compute trajectory flexibility metrics. This backtracking algorithm keeps the trajectory information in a continuous state during the back-propagation steps, which improves the accuracy of the trajectory flexibility measurement.

- We develop a Monte Carlo simulation to measure the robustness of a trajectory segment in the presence of deeply coupling factors, such as air traffic flows and the uncertainty of intruders' trajectories.

With the conceptual tools developed in the earlier chapter, an immediate byproduct is a methodology to develop a survivability map that depicts all survivable states for an autonomous vehicle in the chapter 5.

- We are one of the first authors to investigate the effects of vehicle maneuverability, remaining lifetime, airport locations and safe radius, and air traffic on the vehicle's survivability awareness.

- We offer a novel perspective on airport siting that takes the survivability of vehicles into account. From a demand-supply standpoint, it is recommended that airports be constructed in regions with relatively dense populations. Our findings indicate that if

airports were uniformly dispersed across the geographical area under consideration, they could potentially provide greater survivability coverage.

Last but not least, in chapter 6, we propose four trajectory planning policies and empirically evaluate their efficacy in mitigating diverse levels of uncertainty through extensive computational experiments.

- We demonstrate that when the operating environment is dominated by unknown disturbances, maximizing the number of feasible trajectories results in the highest mission survivability.

7.2 Future Work

This thesis's concepts and methodologies inspire multiple research directions. Separately, we discuss research ideas that may appeal to researchers in the communities of vehicle routing and trajectory planning.

7.2.1 Potential research work inspired by solving the coordinated vehicle routing problem

- Develop advanced modeling techniques to better represent the unique features of the Drone-Truck surveillance problem (e.g., cyclic vehicle operation).
- Examine a more sophisticated model involving the coordination of multiple trucks and drones to complete a surveillance mission. Or using the one-truck-one-drone model to solve wider problems.
- Consider the ground traffic/node service time as a stochastic component of the model, so that transit/observation delays will be accounted for in the planning.
- Investigate how the drone's energy consumption in observation mode and flying mode would affects routing decisions.

- Utilize the Nested-VRP model as a component to study more social-oriented concepts, for example, the fairness among victims.

7.2.2 Potential research work inspired by solving the trajectory planning problem

- Design new support AAM ground facilities considering the coverage and continuity of the survivability map.
- Incorporate the proposed metrics and trajectory planning policies to NASA research in support of increasingly autonomous airspace operations.
- Examine the impact of the proposed metrics on collective autonomous behaviors among multiple agents.
- Investigate the computational efficiency of the proposed methodology for practical applications.
- Explore the impact of spatial and temporal quantization on metrics estimation.

As a final observation, research on integrated routing and trajectory problems is ambitious but still fragmented. Extensive research effort is required to advance our understanding of integrated routing and trajectory problems in order to realize innovative new applications in the era of AAM.

REFERENCES

[1] S. G. Gupta, M. M. Ghonge, P. M. Jawandhiya, *et al.*, "Review of unmanned aircraft system (uas)," *International journal of advanced research in computer engineering & technology (IJARCET)*, vol. 2, no. 4, pp. 1646–1658, 2013.

[2] *Faa aerospace forecast fiscal years 2021–2041*, https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/unmanned_aircraft_systems.pdf, (Accessed on 12/06/2021).

[3] C. Rego, D. Gamboa, F. Glover, and C. Osterman, "Traveling salesman problem heuristics: Leading methods, implementations and latest advances," *European Journal of Operational Research*, vol. 211, no. 3, pp. 427–441, 2011.

[4] 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.

[5] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations Research*, vol. 21, no. 2, pp. 498–516, 1973.

[6] P. Toth and D. Vigo, *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia., 2002.

[7] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 3, pp. 345–358, 1992.

[8] R. Kulkarni and P. R. Bhave, "Integer programming formulations of vehicle routing problems," *European Journal of Operational Research*, vol. 20, no. 1, pp. 58–67, 1985.

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

[10] N. Agatz, P. Bouman, and M. Schmidt, "Optimization approaches for the traveling salesman problem with drone," *Transportation Science*, vol. 52, no. 4, pp. 965–981, 2018.

[11] C. C. Murray and R. Raj, "The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones," *Transportation Research Part C: Emerging Technologies*, vol. 110, pp. 368–398, 2020.

[12] R. Daknama and E. Kraus, "Vehicle routing with drones," *arXiv preprint arXiv :1705.06431*, 2017.

[13] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Ha, "On the min-cost traveling salesman problem with drone," *Transportation Research Part C: Emerging Technologies*, vol. 86, pp. 597–621, 2018.

[14] J. C. De Freitas and P. H. V. Penna, "A randomized variable neighborhood descent heuristic to solve the flying sidekick traveling salesman problem," *Electronic Notes in Discrete Mathematics*, vol. 66, pp. 95–102, 2018.

[15] M. Dell'Amico, R. Montemanni, and S. Novellani, "Drone-assisted deliveries: New formulations for the flying sidekick traveling salesman problem," *Optimization Letters*, pp. 1–32, 2019.

[16] P. Kitjacharoenchai, M. Ventresca, M. Moshref-Javadi, S. Lee, J. M. Tanchoco, and P. A. Brunese, "Multiple traveling salesman problem with drones: Mathematical model and heuristic approach," *Computers & Industrial Engineering*, vol. 129, pp. 14–30, 2019.

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

[18] D. Schermer, M. Moeini, and O. Wendt, "Algorithms for solving the vehicle routing problem with drones," in *Asian Conference on Intelligent Information and Database Systems*, Springer, Cham., 2018, pp. 352–361.

[19] P. Bouman, N. Agatz, and M. Schmidt, "Dynamic programming approaches for the traveling salesman problem with drone," *Networks*, vol. 72, no. 4, pp. 528–542, 2018.

[20] S. Poikonen, B. Golden, and E. A. Wasil, "A branch-and-bound approach to the traveling salesman problem with a drone," *INFORMS Journal on Computing*, vol. 31, no. 2, pp. 335–346, 2019.

[21] S. Poikonen and B. Golden, "The mothership and drone routing problem," *INFORMS Journal on Computing*, vol. 32, no. 2, pp. 249–262, 2020.

[22] P. L. González-Rodríguez, D. Canca, J. L. Andrade-Pineda, M. Calle, and J. M. Leon-Blanco, "Truck-drone team logistics: A heuristic approach to multi-drop route planning," *Transportation Research Part C: Emerging Technologies*, vol. 114, pp. 657–680, 2020.

[23] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 70–85, 2017.

[24] C. Cheng, Y. Adulyasak, and L.-M. Rousseau, *Formulations and Exact Algorithms for Drone Routing Problem*. CIRRELT, Interuniversity Research Center on Enterprise Networks, Logistics and Transportation, 2018.

[25] X. Wang, S. Poikonen, and B. Golden, "The vehicle routing problem with drones: Several worst-case results," *Optimization Letters*, vol. 11, no. 4, pp. 679–697, 2017.

[26] S. Poikonen, X. Wang, and B. Golden, "The vehicle routing problem with drones: Extended models and connections," *Networks*, vol. 70, no. 1, pp. 34–43, 2017.

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

[28] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM (JACM)*, vol. 7, no. 4, pp. 326–329, 1960.

[29] T. R. P. Ramos, M. I. Gomes, and A. P. B. Póvoa, "Multi-depot vehicle routing problem: A comparative study of alternative formulations," *International Journal of Logistics Research and Applications*, vol. 23, no. 2, pp. 103–120, 2020.

[30] R. Bazrafshan, S. Hashemkhani Zolfani, S. M. J. Al-e-hashem, *et al.*, "Comparison of the sub-tour elimination methods for the asymmetric traveling salesman problem applying the seca method," *Axioms*, vol. 10, no. 1, p. 19, 2021.

[31] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & Sons, New York., 1998.

[32] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. John Wiley & Sons, New York., 1999, vol. 55.

[33] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, Association for Computing Machinery, New York., 1984, pp. 302–311.

[34] R. C. Jeroslow, "There cannot be any algorithm for integer programming with quadratic constraints," *Operations Research*, vol. 21, no. 1, pp. 221–224, 1973.

[35] M. Desrochers and G. Laporte, "Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints," *Operations Research Letters*, vol. 10, no. 1, pp. 27–36, 1991.

[36] H. D. Sherali and P. J. Driscoll, "On tightening the relaxations of Miller-Tucker-Zemlin formulations for asymmetric traveling salesman problems," *Operations Research*, vol. 50, no. 4, pp. 656–669, 2002.

[37] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," in *International Conference on Principles and Practice of Constraint Programming*, Springer, 1998, pp. 417–431.

[38] *Flexible — definition in the cambridge english dictionary*, https://dictionary.cambridge.org/us/dictionary/english/flexible, (Accessed on 06/12/2022).

[39] J. M. Adam, F. Parisi, J. Sagaseta, and X. Lu, "Research and practice on progressive collapse and robustness of building structures in the 21st century," *Engineering Structures*, vol. 173, pp. 122–149, 2018.

[40] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre, "Collaborative recommendation: A robustness analysis," *ACM Transactions on Internet Technology (TOIT)*, vol. 4, no. 4, pp. 344–377, 2004.

[41] H. Kitano, "Biological robustness," *Nature Reviews Genetics*, vol. 5, no. 11, pp. 826–837, 2004.

[42] J. V. Bradley, "Robustness?" *British Journal of Mathematical and Statistical Psychology*, vol. 31, no. 2, pp. 144–152, 1978.

[43] R. A. Jones and J. M. Ostroy, "Flexibility and uncertainty," *The Review of Economic Studies*, vol. 51, no. 1, pp. 13–32, 1984.

[44] A. G. Hart, *Anticipation, Uncertainty and Dynamic Planning.* University of Chicago Press, 1940.

[45] T. C. Koopmans, "On flexibility of future preference," 1962.

[46] G. Stigler, "Production and distribution in the short run," *Journal of Political Economy*, vol. 47, no. 3, pp. 305–327, 1939.

[47] B. William, *Economic Dynamics.* The Macmilian Company, New York, 1959.

[48] B. Gersonius, R. Ashley, A. Pathirana, and C. Zevenbergen, "Climate change uncertainty: Building flexibility into water and flood risk infrastructure," *Climatic change*, vol. 116, no. 2, pp. 411–423, 2013.

[49] S. C. Graves and B. T. Tomlin, "Process flexibility in supply chains," *Management Science*, vol. 49, no. 7, pp. 907–919, 2003.

[50] A. Das, "Towards theory building in manufacturing flexibility," *International journal of production research*, vol. 39, no. 18, pp. 4153–4177, 2001.

[51] R. Dubey, A. Gunasekaran, S. J. Childe, S. Fosso Wamba, D. Roubaud, and C. Foropon, "Empirical investigation of data analytics capability and organizational flexibility as complements to supply chain resilience," *International Journal of Production Research*, vol. 59, no. 1, pp. 110–128, 2021.

[52] D. Ivanov, A. Das, and T. Choi, "New flexibility drivers in manufacturing, service, and supply chain systems," *International Journal of Production Research*, vol. 56, no. 10, pp. 3359–3368, 2018.

[53] R. Beach, A. P. Muhlemann, D. H. Price, A. Paterson, and J. A. Sharp, "A review of manufacturing flexibility," *European journal of operational research*, vol. 122, no. 1, pp. 41–57, 2000.

[54] S. K. Ider and O. Korkmaz, "Trajectory tracking control of parallel robots in the presence of joint drive flexibility," *Journal of Sound and Vibration*, vol. 319, no. 1-2, pp. 77–90, 2009.

[55] M. Good, L. Sweet, and K. Strobel, "Dynamic models for control system design of integrated robot and drive systems," 1985.

[56] R. F. Jacobus and M. A. Serna, "Modal analysis of a three dimensional flexible robot," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, IEEE, 1994, pp. 2962–2967.

[57] S. Raychaudhuri, "Introduction to monte carlo simulation," in *2008 Winter simulation conference*, IEEE, 2008, pp. 91–100.

[58] R. Lange *et al.*, "On a generic uncertainty model for position information," in *International Workshop on Quality of Context*, Springer, 2009, pp. 76–87.

[59] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain, "Managing uncertainty in moving objects databases," *ACM Transactions on Database Systems (TODS)*, vol. 29, no. 3, pp. 463–507, 2004.

[60] M. Ghasri and M. Maghrebi, "Factors affecting unmanned aerial vehicles' safety: A post-occurrence exploratory data analysis of drones' accidents and incidents in australia," *Safety science*, vol. 139, p. 105 273, 2021.

[61] M. C. R. Murça, "Identification and prediction of urban airspace availability for emerging air mobility operations," *Transportation Research Part C: Emerging Technologies*, vol. 131, p. 103 274, 2021.

[62] P. J. Burke, "Small unmanned aircraft systems (suas) and manned traffic near john wayne airport (ksna) spot check of the suas facility map: Towards a new paradigm for drone safety near airports," *Drones*, vol. 3, no. 4, p. 84, 2019.

[63] M. F. B. Mohamed Salleh *et al.*, "Preliminary concept of adaptive urban airspace management for unmanned aircraft operations," in *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018, p. 2260.

[64] P. D. Vascik and R. J. Hansman, "Assessing integration between emerging and conventional operations in urban airspace," in *AIAA Aviation 2019 Forum*, 2019, p. 3125.

[65] Y. Zou, H. Zhang, G. Zhong, H. Liu, and D. Feng, "Collision probability estimation for small unmanned aircraft systems," *Reliability Engineering & System Safety*, vol. 213, p. 107 619, 2021.

[66] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *2011 IEEE international conference on robotics and automation*, IEEE, 2011, pp. 723–730.

[67] J. Miura and Y. Shirai, "Probabilistic uncertainty modeling of obstacle motion for robot motion planning," *Journal of Robotics and Mechatronics*, vol. 14, no. 4, pp. 349–356, 2002.

[68] G. S. Aoude, B. D. Luders, D. S. Levine, and J. P. How, "Threat-aware path planning in uncertain urban environments," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 6058–6063.

[69] L. Lulu and A. Elnagar, "A comparative study between visibility-based roadmap path planning algorithms," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2005, pp. 3263–3268.

[70] O. Takahashi and R. J. Schilling, "Motion planning in a plane using generalized voronoi diagrams," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 2, pp. 143–150, 1989.

[71] T. Khuswendi, H. Hindersah, and W. Adiprawita, "UAV path planning using potential field and modified receding horizon A* 3D algorithm," in *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, 2011, pp. 1–6.

[72] J. Hardy and M. Campbell, "Contingency planning over probabilistic obstacle predictions for autonomous road vehicles," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 913–929, 2013.

[73] Z. Fanruiqi, I. Husni R., and J.-p. Clarke, "Trajectory planning for mission survivability of autonomous vehicles in moderately to extremely uncertain environments," in *Fourteenth USA/Europe Air Traffic Management Research and Development Seminar*, 2021.

[74] R. W. Taylor, "Extended range operation of twin-engine commercial airplanes," *SAE Transactions*, pp. 959–970, 1985.

[75] *Open data @ ourairports*, https://ourairports.com/data/, (Accessed on 06/02/2022).

[76] B. German, M. Daskilewicz, T. K. Hamilton, and M. M. Warren, "Cargo delivery in by passenger evtol aircraft: A case study in the san francisco bay area," in *2018 AIAA Aerospace Sciences Meeting*, 2018, p. 2006.

[77] P. D. Vascik, J. Cho, V. Bulusu, and V. Polishchuk, "Geometric approach towards airspace assessment for emerging operations," *Journal of Air Transportation*, vol. 28, no. 3, pp. 124–133, 2020.

[78] V. Bulusu, R. Sengupta, and Z. Liu, "Unmanned aviation: To be free or not to be free?" In *7th International Conference on Research in Air Transportation*, 2016.

[79] D. Locascio, M. Levy, K. Ravikumar, B. German, S. I. Briceno, and D. N. Mavris, "Evaluation of concepts of operations for suas package delivery," in *16th AIAA Aviation Technology, Integration, and Operations Conference*, 2016, p. 4371.

[80] K. R. Antcliff, M. D. Moore, and K. H. Goodrich, "Silicon valley as an early adopter for on-demand civil vtol operations," in *16th AIAA Aviation Technology, Integration, and Operations Conference*, 2016, p. 3466.

[81] H. A. Blom, S. H. Stroeve, and T. Bosse, "Modelling of potential hazards in agent-based safety risk analysis," Tenth USA/Europe Air Traffic Management Research and Development Seminar, 2013.

[82] H. Idris, N. Shen, and D. Wing, "Complexity management using metrics for trajectory flexibility preservation and constraint minimization," in *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, including the AIAA Balloon Systems Conference and 19th AIAA Lighter-Than*, 2011, p. 6809.

[83] H. Idris, D. Delahaye, and D. Wing, "Distributed trajectory flexibility preservation for traffic complexity mitigation," in *Proceedings of the 8th USA/Europe Air Traffic Seminar (ATM'09)*, Citeseer, 2009.

[84] H. Idris and N. Shen, "Estimating airspace capacity based on risk mitigation metrics," in *Tenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2013), Chicago*, 2013.

[85] J. Mattingley, Y. Wang, and S. Boyd, "Code generation for receding horizon control," in *2010 IEEE International Symposium on Computer-Aided Control System Design*, IEEE, 2010, pp. 985–992.

[86] D. C. Montgomery and G. C. Runger, *Applied statistics and probability for engineers*. John Wiley & Sons, 2010.