CONCEPTUAL-LEVEL ANALYSIS AND DESIGN OF UNMANNED AIR TRAFFIC MANAGEMENT SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Coline L. Ramée

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

Georgia Institute of Technology

December 2021

CONCEPTUAL-LEVEL ANALYSIS AND DESIGN OF UNMANNED AIR TRAFFIC MANAGEMENT SYSTEMS

Thesis committee:

Dr. Dimitri Mavris
Aerospace Engineering
Georgia Institute of Technology

Dr. Simon Briceno
Chief Commercial Officer
Jaunt Air Mobility

Dr. Tejas Puranik
Aerospace Engineering
Georgia Institute of Technology

Dr. Zohaib Mian
Head of Systems Engineering - Loon
Alphabet

Dr. Daniel Schrage
Aerospace Engineering
Georgia Institute of Technology

Mr. Mike Paglione
Aviation Research Division
Federal Aviation Administration

Date approved: August 16, 2021

ACKNOWLEDGMENTS

Going through the PhD has been a difficult, much longer than expected, but very formative process. Although thesis work is a solitary work it would not have been possible without the help and encouragements of many people.

I first want to extend my gratitude to all my committee members. I would like to thank my advisor Professor Dimitri Mavris for funding me and guiding me for the past seven years. I am grateful for the opportunities and freedom I had at ASDL, working on many different projects and learning a lot. I want to thank Dr. Simon Briceno, who was my RE at ASDL before he joined Jaunt. He really helped shape the thesis when it was in its early stage. Dr. Tejas Puranik provided very helpful feedback and encouragements. Thanks to Mr. Mike Paglione and Dr. Zohaib Mian for the feedback and suggestions they provided after my proposal. Finally, thanks to Professor Schrage for agreeing to join my committee.

This research was supported in part through research cyberinfrastructure resources and services provided by the Partnership for an Advanced Computing Environment (PACE) at the Georgia Institute of Technology, Atlanta, Georgia, USA.

The idea for this thesis originated during a 6-month internship in 2018. I am grateful to my boss at Staples Dr. Girish Kulkarni for his encouragements, kind words, and advice at the time. I am also lucky to have made great friends during my time at ASDL who I could always count on to discuss my thesis when I was stuck. To Pat Meyer, our weekly meetings to talk (mostly) about our theses have been incredibly helpful. To Dudu, thanks for providing great feedback when I was sending you things at the last minute.

I also want to thank Adrienne and Tanya for helping me navigate the administrative side of things and always being patient and encouraging.

I have had a lot of fun working with the Georgia Tech marine robotics group. The robotic competitions were always a highlight of the year, but I will miss most working in the lab with everyone, drawing ideas on the whiteboard, trying things out and watching

them fail then succeed after lots of tweaking. You made me a better and more confident engineer.

To my family, thanks for supporting me from afar, cheering me up when I needed, and not asking too often ”Alors tu finis quand?”.

Finally, to Raph, I would not have gone through it without your help and support. I am excited as we begin a next chapter of our life together. Now finish your thesis!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

4DT 4-Dimensional Trajectory

API Application Programming Interface

ATC Air Traffic Control

BVLOS Beyond Visual Line-of-Sight

ConOps Concept of Operations

DAA Detect and Avoid

eVTOL Electrical VTOL

FAA Federal Aviation Administration

FPV First-Person-View

HAR High Altitude Redesign

IFR Instrument Flight Rules

LAANC Low Altitude Authorization and Notification Capability

LoS Loss of separation

MVP Modified Voltage Potential

NMAC Near Midair Collision

NOTAM Notices to Airmen

ORCA Optimal Reciprocal Collision Avoidance

sUAS Small UAS

TFR Temporary Flight Restrictions

UAM Urban Air Mobility

UAV Unmanned Air Vehicle

USS Unmanned Service Suppliers

UTM Unmanned Traffic Management

VFR Visual Flight Rules

VLL Very Low-Level

VLOS Visual Line-of-Sight

VO Velocity Obstacle

VTOL Vertical Take-Off and Landing

SUMMARY

There have been multiple announcements by different companies in the past couple years of package delivery by drone and air taxi projects. However, there are still many barriers to the deployment of high densities of aerial vehicles in low-altitude airspace over urban areas. Current Air Traffic Control Systems cannot handle the high density of traffic being forecast. Integrating these new types of on-demand air mobility in the National Airspace requires a fundamental change to the traffic management system. Many different concepts of operations for unmanned traffic management (UTM) systems have been proposed, but there is no common framework to evaluate and compare alternatives at a conceptual design stage. This might cause a locally optimal system to be chosen, resulting in lower safety and economic performance than what would have been possible if a more systematic approach to the design of UTM system had been followed.

In this thesis, a systematic approach to the design of UTM systems is introduced. Based on the literature on conceptual design, a five step approach to the design of UTM systems is proposed. The steps of the approach are: define operating scenarios, generate UTM alternatives, select performance criteria, evaluate, and make decision. To generate UTM alternatives in a systematic manner, a matrix of alternatives should be created. However, this requires a system decomposition that does not currently exist for UTM systems. Here, a system decomposition into four subsystems is proposed: airspace structure, access control, preflight planning, and collision avoidance. For each subsystem, alternatives are identified using the literature. For the second step of the approach, operating scenarios for UTM are not well-defined. There are many external factors outside of the designer's control, and different studies make different assumptions. Three different external factors, or components of an operating scenario, are identified: demand, static obstacles, and priority traffic. The impact of the different subsystems and external factors on the performance of a given UTM architecture cannot be found in the literature. Many studies evaluate a point design or fix

assumptions to focus on a single subsystem. There is no available tool that allows to evaluate different UTM architectures while varying all the elements that have been presented here. To bridge that gap, an agent-based simulation was developed to allow the evaluation of the UTM systems generated using the matrix of alternatives in different operating scenarios. For the fourth step of the approach, performance criteria are selected from the aviation literature. To capture safety, the number of losses of separation and near-midair-collisions per flight hour are used. To measure the efficiency of the trajectories, a time and energy efficiency metrics are introduced. The capacity of the system is evaluated for a fixed overall density using the throughput, or number of vehicles completing a flight per minute. Finally, two simple multi-attribute decision making methods are selected from the literature. This allows to rank architectures based on their performance in a given scenario for a given set of weights representing a designer's preferences.

This thesis also proposes a novel 4D trajectory planning algorithm that relies on a local collision avoidance method. Experiments show that it performs well in terms of time efficiency and throughput when compared to a decoupled approach. The novel algorithm achieves a comparable performance to a global optimization algorithm in a nominal cruise scenario but is much more computationally efficient.

The impact of the inclusion of certain subsystems and external factors on the outcome of the conceptual design stage is systematically evaluated in a series of experiments. Performance of different architectures is evaluated with and without the subsystem or external factor of interest. The experiments show that there are significant interactions between agents' autonomous behaviors, airspace structure, and external factors such as demand, static obstacles, and priority traffic. The decision tables obtained with and without the element of interest are compared, and weights are found such that the architecture rankings are different. This shows that neglecting these interactions or making simplifying assumptions may change the outcome of the conceptual design stage and result in the selection of an architecture that underperforms in terms of safety, capacity or efficiency. This is validated

on two use cases, an air taxi scenario and a drone delivery scenario. In the air taxi scenario, using the proposed approach results in the selection of an alternative with a 25% higher score than the alternative selected with a baseline approach.

As a result of the work conducted in this thesis, the importance of including the autonomy, airspace structure, demand, static obstacles, and priority traffic in the early stage of UTM evaluation has been demonstrated. The necessity of including other subsystems or external factors can be evaluated by following the same process that was demonstrated in the thesis.

CHAPTER 1
INTRODUCTION AND MOTIVATION

Multiple companies announced drone package delivery and air taxi projects in recent years [1, 2, 3]. Several studies forecast a large deployment of such low-altitude air mobility technologies over urban areas by the end of the next decade if regulatory, acceptability, and technological barriers can be overcome [4, 5]. The focus of this thesis, the integration of a high density of new vehicle types in the low-altitude airspace, lays at the intersection of regulatory and technological barriers. This thesis introduces a flexible framework to study low-altitude traffic management systems in a systematic way. It evaluates the impact of the inclusion of different subsystems and external factors on the outcome of the conceptual design stage. It shows that there are many interactions between agents' autonomous behaviors, airspace structure, and external factors such as demand and static obstacles. Neglecting these interactions or making simplifying assumptions may result in the selection of an architecture that underperforms in terms of safety, capacity or efficiency.

This first chapter defines some of the terms and concepts that appear throughout the document and motivates the need for a low-altitude traffic management system. It then gives a brief overview of some of the systems that have been proposed to integrate Urban Air Mobility traffic into the National Airspace. At the end of the first chapter, the main research question guiding this thesis is introduced. The second chapter presents a systematic review of the literature to narrow the objective of the thesis and identify the gaps in the literature that prevent the main research question to be directly answered. The third chapter introduces the research questions that interrogate how the evaluation of UTM systems should be conducted. Hypotheses are formulated and experiments designed to validate or disprove them. The fourth chapter provides details on the agent-based simulation that was developed as part of the framework to conduct the experiments. The fifth and sixth chapter

present the results and conclusions of the experiments. The seventh chapter validates the proposed approach by comparing it to a baseline approach and shows that it results in the selection of better performing UTM architectures. Finally, the eighth and last chapter concludes the study, summarizes the main results, and introduces potential avenues for future work.

1.1 Urban Air Mobility

The term of Urban Air Mobility (UAM) is sometimes used in the literature to designate only passenger transportation using new vehicle types called eVTOLs (electric Vertical Take-Off and Landing). However, NASA defines UAM more broadly as a ”safe and efficient system for air passenger and cargo transportation within an urban area, inclusive of small package delivery and other urban Unmanned Aerial Systems (UAS) services, which supports a mix of onboard/ground-piloted and increasingly autonomous operations” [6]. Despite the very different scales of the vehicles involved, which may at first sight make them appear to be very different problems, drone package delivery and air taxi applications both entail a high-density of low-altitude operations over densely populated areas and therefore introduce similar challenges when it comes to their integration to the airspace.

Observation

Urban Air Mobility is comprised of both small unmanned air vehicles (UAVs) and large passenger-carrying vehicles.

1.1.1 Unmanned Air Vehicles and Package Delivery

The term Unmanned Air Vehicles (UAVs) covers a large variety of vehicle designs and scale. It can refer to any aerial drone ranging from small hobby quadrotors to large military drone. UAVs may be fixed-wing aircraft, multi-copters, or more unconventional designs that may transition between hover and fixed-wing flight. The term Unmanned Aircraft Systems (UAS) refers to the vehicle and the external systems required for its operation

that may for instance include the communication network, infrastructures (e.g. dedicated landing pad), and remote pilot. In practice, it is often used interchangeably with UAVs. Small UAS (SUAS) are small drones weighing less than 55 pounds that are regulated under a different set of rules by the FAA. The drones employed in the context of package delivery mostly belong to this category.

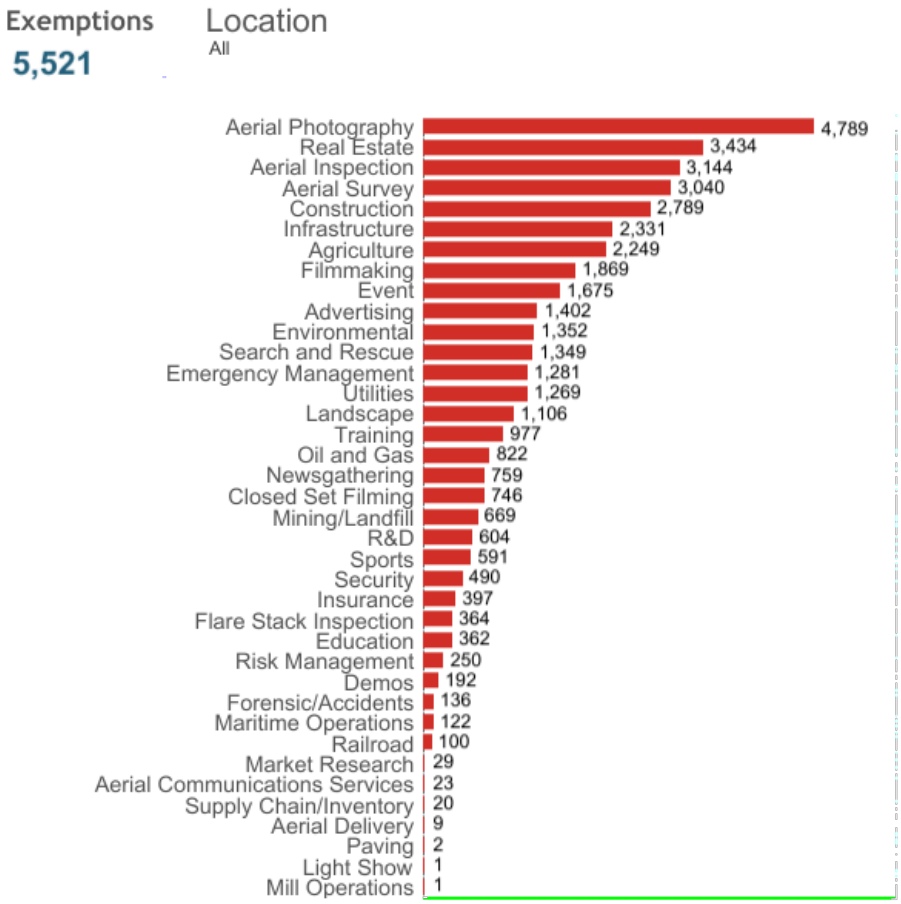


Figure 1.1: UAVs commercial applications from August 2014 to August 2016 as specified in the exemption requests submitted to the FAA (before part 107 was created) [7].

Recent advances in embedded electronics such as GPS, IMU, or flight controller, have made small UAVs easy to use and relatively inexpensive. In addition to their flexibility in terms of payload, this has led to a large acceptance by various industries and public actors.

Commercial UAVs have so far mostly been used to perform surveys. The vehicles are outfitted with various sensors such as cameras, and LIDARs, and are able to scan large areas

or difficult to reach zones. Examples of UAVs usage include observation of crowds for law enforcement, crop monitoring, scans of construction sites, building inspection, and video recording for the news or movie industry. As can be seen on Figure 1.1, most requests for commercial applications for UAVs from 2014 to 2016 concerned aerial photography, survey and inspection, and aerial delivery is very low on the list.

The integration of UAVs into these different industries has only just started. A study by PwC in 2016 showed that the addressable market for UAVs, i.e. the revenue opportunity for drones, could represent 127 billion dollars, with the infrastructure industry accounting for 35% of the total [8].

In the United States, the operation of SUAS has been regulated by the Federal Aviation Administration (FAA) under part 107 of the Federal Aviation Regulations (FAR) since August 2016. Before that, operators had to request the FAA for a waiver in order to operate a Unmanned Air Vehicle (UAV) commercially. The current legislation limits UAVs flights to good visibility conditions (day, 3 miles visibility) below 400 feet. Remote pilots are only allowed to fly their drones in visual line-of-sight (VLOS), i.e. they have to see their vehicles and its surrounding with their own eyes. Using a First-Person-View (FPV) system is not allowed, unless a safety pilot next to the remote pilot can monitor the drone in VLOS conditions. Flights are forbidden in the vicinity of airports and above crowds. The requirements can be waived on a case-by-case basis if the operator submits enough proof that it can operate safely outside the guidelines. For commercial operations, the operator must hold a Remote Pilot Certificate. NASA and the FAA have established several UAV test sites across the United States to test more complex operations such as Beyond-Visual-Line-Of-Sight (BVLOS).

The rules vary greatly between different countries and are quickly evolving as the technology matures. The current lack of a certified system allowing small UAVs to fly without human observers hinders the development of large-scale package delivery operations. Certifying autonomy is complex due to the requirements for robustness and to the large number

of situations that the vehicles must be able to handle safely. This leads to the following observation:

Observation

There is no scalable traffic management system for drone deliveries, and current operations are heavily limited due to the high uncertainty associated with autonomous capabilities.

Although the rules do not yet allow it in general, there is a huge interest from industry for package delivery using UAVs. To be economically viable, package delivery would require Beyond Visual Line-of-Sight (BVLOS) operations and one-pilot-to-many-drones operations as the cost of pilots and visual observers would be prohibitive at a large scale.

A study by McKinsey in 2018 suggests that drone delivery may become a viable market in the US as early as 2030 if all regulatory challenges are overcome. They forecast around 500 million deliveries at a cost of $4.2 per delivery in 2030 [4]. Surveys show that there is public interest for last-mile delivery with drones despite some concerns over safety and privacy [4, 9].

Some deliveries have already been conducted abroad or in dedicated UAVs test areas in the US. A few noteworthy initiatives are mentioned in the following paragraphs.

Figure 1.2: Zipline UAV (Stephen Shankland/CNET).

Figure 1.3: Project wing UAV [3].

Figure 1.4: Prime air UAV [2].

Zipline [1] delivers blood to remote hospitals in Rwanda and Ghana using autonomous fixed-wing aircraft. Doctors within 80km of one of the company's distribution centers may order blood at any time. Upon receipt of the order, technicians load and launch a UAV with

[1] http://www.flyzipline.com

the requested supply. The drone flies autonomously using GPS, parachutes its package, and goes back to its base [10].

Project Wing [2], originally a Google X project, has been testing deliveries of small packages by UAVs in a rural community in Australia since 2014. The testers can order small items such as medicine or snacks through a dedicated application. The drone delivers its package without having to land by slowly lowering a line to the ground. The company plans to expand its testing to Finland in Spring 2019. In addition to the design of the vehicle, Wing is also a UAS Service Supplier (USS) providing an interface between UAS operators and the FAA.

Amazon Prime Air [2] is developing vehicles and different concepts for drone deliveries. They advertised their first test delivery to a customer in the UK in 2016.

1.1.2 VTOLs and Air Taxi

FAA Order 8900.1 Volume 2, chapter 2 section 2-130 C. defines passenger air taxi operations as operations of an aircraft designed to carry no more than 60 passengers on an on-demand or limited schedule basis. These operations are regulated under Part 135. In the Urban Air Mobility literature, a distinction is sometimes made between air taxi, which is purely on-demand and unscheduled, and air metro, which consists of scheduled recurring flights. Air Taxi operations can be performed by conventional fixed-wing aircraft, short take-off and landing (STOLs) vehicles or Vertical Take-Off and Landings (VTOLs) vehicles. VTOLs are especially interesting for Air Taxi operations in urban settings as they do not require a lot of dedicated space on the ground, so they can operate from many different locations instead of just being limited to airport-to-airport operations.

Helicopters are an example of VTOL vehicle, they can take-off, hover, and land vertically. Currently a third of rotorcraft in the US are used primarily for medical transportation and aerial observations (mapping, patrol, search and rescue, surveillance,etc..). Air taxi

[2] https://wing.com/

and business travel only account for 10% of the vehicles as illustrated on Figure Figure 1.5 [11].

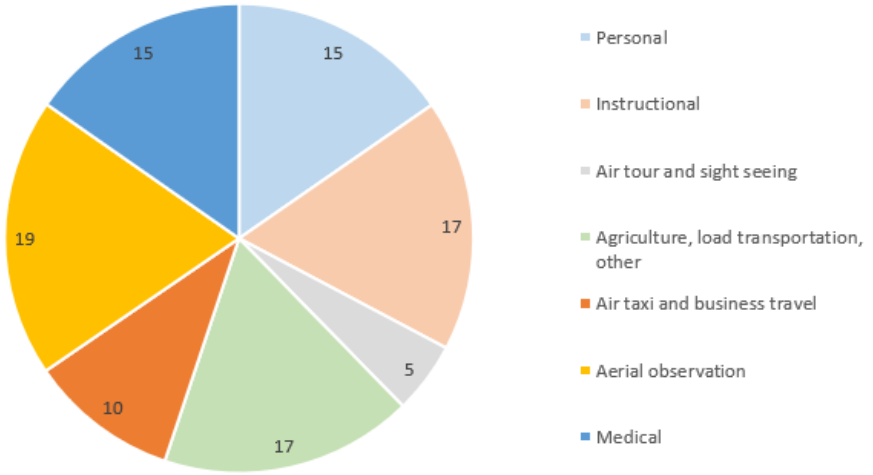


Figure 1.5: Repartition of primary use for rotorcraft in 2016.

There were attempts from the late 50s to 70s to develop the use of rotorcraft as air taxi in the United States. However, serious issues with noise, safety, and cost led to a reduction or termination of service [12]. A number of cities such as New-York and Los Angeles have taken ordinance to limit or prohibit helicopter flights due to past accidents and noise abatement concerns [13].

Despite those issues there is a lot of interest from the industry to expand air taxi services to allow users to escape the growing congestion of large metropolitan areas [1]. Some companies have already started to deploy ride share air taxi operations with helicopters. Airbus VOOM (https://www.voom.flights/) has deployed its service in Sao Paulo and Mexico as of February 2019. In Sao Paulo, an helicopter can be booked to go from Alphaville, a residential suburb, to Itaim Bibi, a business district, in 10 minutes for as low as US $70. The same ride would take an hour by car due to traffic, according to Google Maps. There are a number of specific conditions in these two cities that made them ideal for the deployment of a helicopter ride sharing service. In Sao Paulo criminality and the fear of kidnapping, poor public infrastructures, traffic congestion and spread out residential areas led to the

development of helicopter infrastructures early on [14]. Those factors offset some of the issues previously mentioned.

To overcome the noise, safety, cost, and efficiency barriers and allow air taxi and air metro to become more widespread, unconventional VTOLs concepts are being designed. A lot of attention is given to electrical propulsion and a new class of vehicle called Electrical VTOLs (eVTOLs) is being developed by several different companies. A few examples are illustrated on the following figures.

Figure 1.6: Airbus A^3 Vahana eVTOL www.vahana.aero.

Figure 1.7: Aurora Flight Science eVTOL www.aurora.aero.

Figure 1.8: Cora eVTOL www.cora.aero.

Of course there are still a number of concerns that cannot be addressed by vehicle design alone, such as the availability of landing sites, the impact of inclement weather on operations, the congestion of Air Traffic Control (ATC), and the integration in the national airspace. NASA launched in Fall 2018 a UAM grand challenge which goal is not only the development and certification of unconventional VTOLs, but also the development of a system to help integrate these new vehicles into the airspace [15, 16].

The integration of more on-demand operations in the airspace highlights current challenges in both Instrument Flight Rules (IFR) and Visual Flight Rules (VFR) operations. For IFR, large separation minimums strongly limit the density of operations. For instance in Boston, medical helicopters landing IFR at Mass General force Logan Airport to temporarily shutdown operations due to the proximity between the hospital and the airport (less than 3 miles). Moreover, controller workload also limits the number of aircrafts that can operate simultaneously in the same sector. Until recently, the helicontrol area over Sao Paulo, a city in which helicopter flights are especially developed, was limited to a maximum of 6

helicopters operating at the same time. If Urban Air Mobility (UAM) flights were to operate IFR under these conditions, many flights might get denied due to controller workload and airspace constraints.

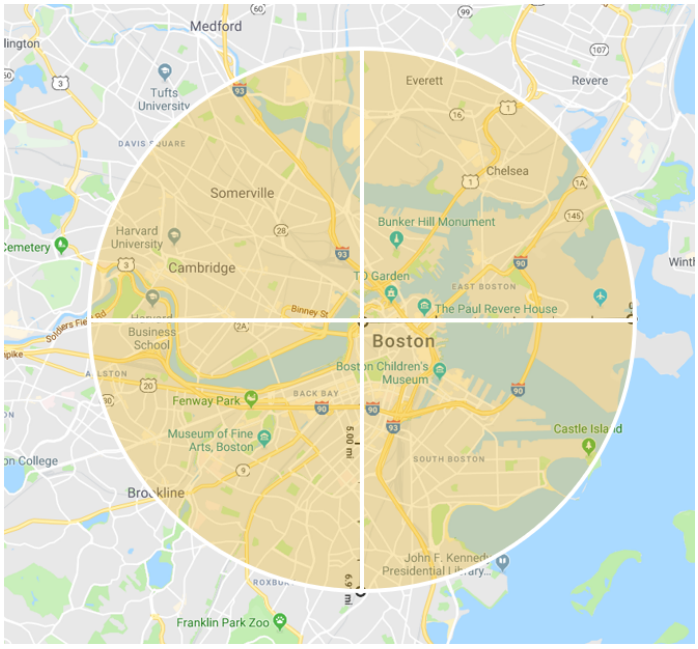


Figure 1.9: Visualization of a 3 nautical mile exclusion zone (the terminal area minimum separation under current IFR rules) over Mass general Hospital in Boston

VFR do not have such strict constraints, pilots simply need to remain well-clear of each other, but in high density operations this might come at the expense of safety guarantees. An example of area with high density of low altitude operations is the airspace over the Hudson river between the surface and 1300ft. The area is popular for its aerial views of New York City and is used by helicopter tour operators and general aviation aircrafts, but it is also right under the regulated airspace of large international airports, so there is only a limited altitude range accessible to uncontrolled VFR operations. In 2009, an accident over the Hudson river between a small general aviation aircraft and a helicopter prompted the FAA to create a Special Flight Rules Area. Operations in this area are still under VFR rules but additional requirements are added, one of which is that pilots have to broadcast their intentions on a UNICOM frequency and report their positions at specific points. Moreover, even if operating under VFR entering class B or C airspace require pilots to be in contact

with ATC. Class B and C airspace can be found around large airports, which are usually near large cities, hence they will affect UAM operations. If UAM flights were to operate under VFR rules, special rules might have to be added to ensure safe operations under high density, and flights would be impacted by controller workload [17].

This leads us to an observation similar to the one that was made for Small UASs (sUASs):

Observation

The current ATM system cannot safely accommodate high densities of air taxi operations.

1.1.3 Motivating Use Cases

The motivating use cases for this thesis were highly automated on-demand UAM operations with a large density of users and ground infrastructures. UAM operations will initially start at lower density, with significant involvement of human operators, and few points being connected by the system. As the technology matures and demand increases, highly automated operations will allow safe and economic operations [18]. Given the large density of flights being predicted by the reports [4, 5], the use cases focus on large scale operations rather than scenarios with few origin/destinations and routes which are more applicable in the near future but do not allow to measure scalability.

As explained in the previous sections, there are two main UAM applications are being considered: package delivery by sUAS, and Air Taxi using eVTOLs. The city of Atlanta, GA is used as the inspiration for the two scenarios. A density of up to 300 aircraft operating at the same time in an area of 20 by 20 km is used in both cases. This corresponds to more than 2.5 aircraft per square nautical mile. These use cases will be briefly presented here. Additional details on how they were modeled is available in Chapter 4.

In both use cases, the system should minimize delays, minimize energy costs, maximize the number of flights and guarantee a good level of safety.

Air Taxi

Air Taxi operations in Atlanta would allow users to bypass traffic and quickly reach their destination. There needs to be a high-density of landing zones (LZs) or vertiports so that passengers' origin and destination are within a short walk of a LZ making the air taxi convenient. Achieving a high density of LZs will take time, here we are considering that the infrastructure system is mature and that LZs are placed in a regular grid spaced by 1 km all over the area. This way any point in the city is within $500 * \sqrt{2} = 707m$ of a LZ, which, assuming a 5m/s walking speed, is less than a 10 minute walk. Demand at different LZs will depend on the surrounding population density.

The air taxi vehicles operate at an altitude of 800 feet similarly to helicopters, and must remain clear of controlled airspace.

Package Delivery

Package delivery by sUAS would allow customers to quickly receive goods without having to go to a store. The model chosen here is one similar to Amazon Lockers. UAVs might not be able to deliver all customers in a city at their exact locations due to constraints at the customer's residence. For instance, if there is no garden or if there is a high density of trees the UAVs would not be able to land or lower the package to the ground. By landing at safe locations these problems can be bypassed. Similarly to the Air Taxi use case there would be a need for a large number of delivery points. In this scenario, the goods come from a small number of warehouses placed near the periphery of the area, and goods are assumed to be available at all warehouses so that customers receive packages from the closest warehouse.

UAVs operate at a lower altitude of 200 feet, where there are few airspace users. These vehicles are assumed to be allowed to operate freely in the airspace that is currently regulated through Low Altitude Authorization and Notification Capability (LAANC) due to their low altitude.

1.2 Integration of UAM in the National Airspace

In June 2018, NASA flew a large remotely piloted drone in controlled airspace without a chase plane for the first time. Detect and Avoid (DAA) systems from multiple companies were onboard the aircraft and allowed remote pilots to see and avoid surrounding traffic. This demonstrated that DAA technology had reached a level of maturity sufficient to meet FAA standards. However, communication with ATC, such as clearances to enter the airspace, were still handled by remote pilots in a control room at NASA Ames. [19]

In the US, the current rules prevent commercial unmanned traffic operations BVLOS. Under part 107, every operations BVLOS require ad-hoc procedures to obtain waivers. Operators must show that they provide equivalent level of safety for any part of the standard regulation that must be waived [20]. For operations under part 91, a waiver of the right-of-way rules (91.113) is required to operate BVLOS . A tactical waiver can be obtained by first responders for limited BVLOS operations in emergency situations where UAVs might safeguard human life [21]. For UAM to deploy at large scale the regulations and current traffic management system must be adapted to interface with the new unmanned technologies and allow remote operations. However, this is not the only challenge. In 2017, the American ATC handled an average of 43 000 flights every day, with 5 000 IFR aircraft simultaneously in the air during peak operational times [22]. As seen in the previous sections, the UAM expected traffic volume is much larger. According to the director of Uber Elevate, by 2025 numerous cities across the United States could have 27 000 eVTOL flights every day per city [17]. In a market study commissioned by NASA, the number of operations forecast in 2030 is even higher: 500 millions UAV deliveries (more than 1.3 million a day) and 250 millions air metro operations (more than 650 thousands a day) [4]. These numbers are several order of magnitude larger than what the current system can safely accommodate.

Several companies have proposed to segregate SUAS flights from general traffic by reserving some layers of the airspace to small unmanned vehicles [23]. This would ensure

that they only interact with participating traffic of the same class. However, NASA has been pushing for truly integrating all traffic into the NAS. For larger vehicles flying exclusively below 400 ft seems unreasonable.

This leads to the following observation:

Observation

Integrating UAM in the national airspace is not only a question of adapting the system interface to work with new users, but requires a more fundamental change to adapt to the traffic increase.

As a result, many different agencies and research groups have started developing Concept of Operations (ConOps) to organize UAM traffic. A survey reviewing proposed concepts was published in 2021 [24]. Some of these concepts are discussed in the next subsection.

1.2.1 NASA's proposed UTM

NASA is proposing a system called Unmanned Traffic Management (UTM) to enable civilian low-altitude UAS operations BVLOS. This system was tested as part of the UTM Pilot Program (UPP) established by the Federal Aviation Administration (FAA) in several test areas around the US. The last phase of the project was concluded in 2019.

The term UTM can be found in the literature as a general name to describe any unmanned air traffic management system, however it is also the name given by NASA to their own implementation, which can be confusing: NASA's UTM is an instantiation of a UTM. In the rest of the thesis when writing UTM we will be referring to the general concept.

In NASA's UTM, the FAA provides an Application Programming Interface (API) so that third party Unmanned Service Suppliers (USS) act as an interface between unmanned operators and the airspace. The proposed structure is detailed in Figure Figure 1.10. The FAA provides a unified interface to the USS, called FIMS (Flight Information Management System). USS get additional data by collaborating with other USS and external data source.

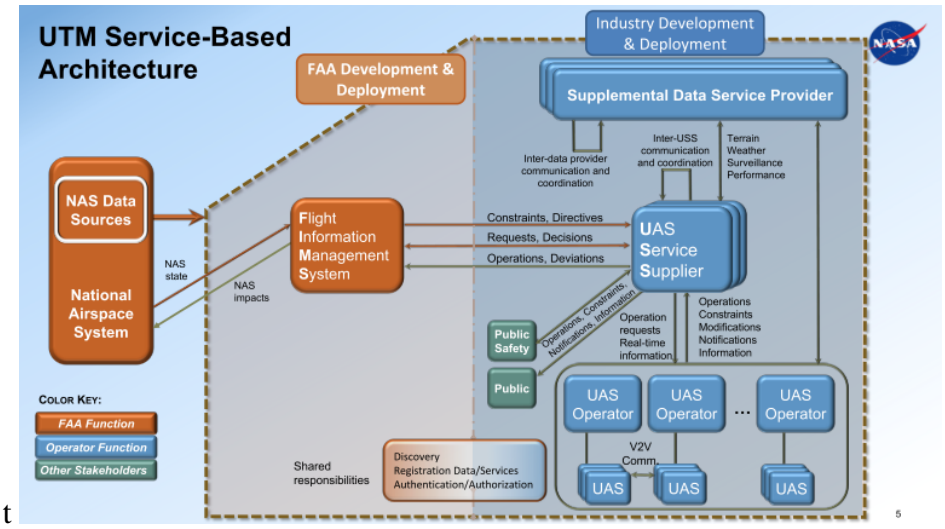


Figure 1.10: Proposed architecture for the UTM [26].

USS operators have already been authorized by the FAA as part of the LAANC initiative. LAANC allows UAV operators to operate closer to airports without having to go through the long process of requesting a waiver. Instead, they can install the USS application on their device and submit their flight plan to the USS. The USS checks that the flight plan does not break any rules by accessing FAA data such as Temporary Flight Restrictionss (TFRs) and Notices to Airmens (NOTAMs), it then approves the flight and notify the airport tower. The application displays flight restricted areas to help the operator plan their flight. As of June 5th 2021, 16 USS have been approved. USS companies are made up of traditional aerospace companies, tech companies and startups: UASidekick, AirMap, Google Project Wing, Airbus, etc. [25]. The LAANC initiative can be viewed as the first deployed step toward a UTM in the US.

Through the UTM, drone operators can reserve a section of airspace and share their operation intent. Operators can see what airspace has been reserved and schedule their flights to avoid conflicts. Emergency flights can use dynamic restrictions to seize the airspace, forcing other users to amend their flights by leaving the restricted area or by landing. The UTM can be used by operators in Visual Line-of-Sight (VLOS) conditions for safety and situational awareness. Participation in the UTM is mandatory to operate in BVLOS condi-

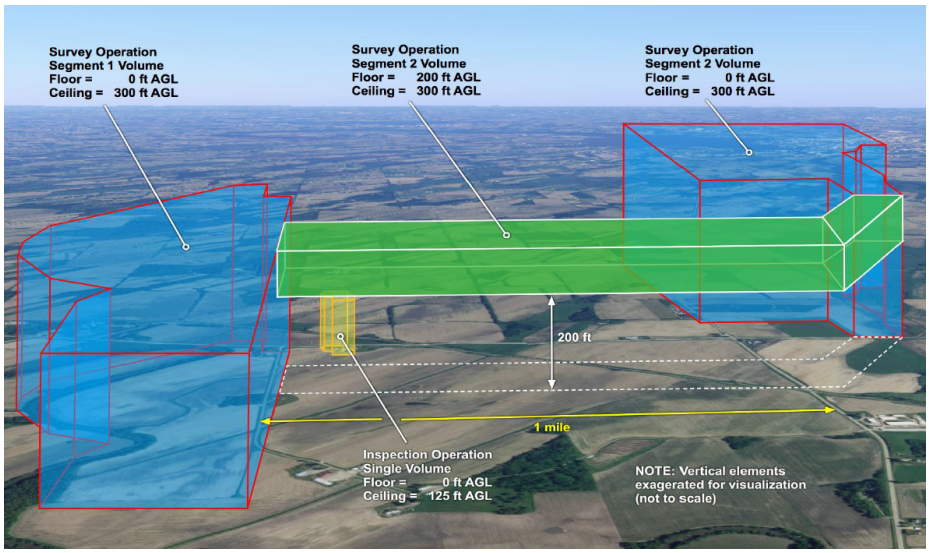


Figure 1.11: Illustration of volume reservation in the UTM (taken from FAA/NASA UAS Traffic Management Pilot Program (UPP), Industry workshop 1, March 2018).

tions.

The operational plan or flight geography is submitted by the operator to the UTM. That volume encompasses all the intended area of flight. The UTM then adds a conformance geography around it, which is a buffer to account for disturbances and control errors. If the vehicle exits the conformance buffer it is declared non conforming. The protected geography around the conformance buffer defines the limit beyond which other aircraft are allowed to operate. If the vehicle is operating outside its protected geography it is declared rogue [27].

The use cases provided by NASA for the UTM are inspection tasks below 400ft by sUASs in rural areas, which are adapted to the volume based system. The limitations of the UTM appear when one considers that to travel from a point A to a point B operators would have to reserve large portions of the airspace, blocking the whole trajectory when the vehicle would only be using a fraction of the airspace at a time and preventing it to be shared with other users. A paper from Iowa State University [28] studied a modification of the UTM in which the area reserved by the operator is dynamic with time and only depends on the initial and final time and the minimum and maximum speed of the vehicle.

The reserved area stretches along the planned trajectory and moves with time. The area is smaller than the full volume used by the UTM but it is still very large, only resulting in a minor improvement. In [18], the authors propose to extend UTM to handle On-Demand Mobility (ODM).

Although the UTM provides deconfliction capabilities with collaborating traffic, vehicles are still supposed to perform onboard DAA to avoid non-collaborating traffic such as drones flown VLOS, general aviation aircraft, birds or radio masts. The proposed architecture also allows for collaboration at the vehicles level through Vehicle-to-Vehicle (V2V) communication, although the protocols and type of collaboration allowed are still unclear. Several USS operators can provide their services in the same area, but they must collaborate by communicating flight plans. In effect, the proposed system is centralized.

This static volume-based reservation system ensures strong deconfliction but limits the capacity. No numerical analysis were found in the literature of its efficiency and throughput performance when the demand for point-to-point travel is large.

1.2.2 Europe's U-space

The Single European Sky ATM Research (SESAR) joint undertaking is a research program which has developed a ConOps for UTM called U-space.

In the U-space blueprint [29], the authors envision a scalable system which will evolve as automation and connectivity increase. Initially, the system might only provide a method to plan and broadcast the intended trajectory of a drone and track the vehicle during its flight. At later stages, the system would integrate DAA capabilities and allow drones to amend their flights to react to change in the environment such as priority flights.

In [30], the authors introduce three different classes of Very Low-Level (VLL) airspace that would have different deconfliction capabilities and rules of access. In X volumes, drones are remotely piloted and operate following current VLOS rules. In Y volumes, similarly to NASA's UTM, a deconflicted flight plan must be provided prior to take-off.

Finally, in Z volumes tactical or reactive collision avoidance is used to avoid collision without the need to deconflict before take-off. Each volume requires an increased level of onboard capabilities but should be capable of handling an increased amount of traffic under safety constraints. Areas must be evaluated to determine which type of airspace is most appropriate.

1.2.3 Other Concept of Operations

In addition to these official ConOps, many different researchers have proposed alternative systems. Some private companies involved in UAM have also proposed their own UTM systems. A few of them are quickly described here, a more in-depth comparison and discussion of the different systems components can be found in Chapter 2.

Sunil et al. presented different UTM concepts with different levels of airspace structure in [31]. In the Full Mix concept aircraft are free to fly anywhere. In the Layer concept vehicles' altitude are constrained based on their overall heading. In the Zone concept, illustrated in Figure Figure 1.12, radials and rings are defined around a city center and separate clockwise, anti-clockwise, inbound and outbound traffic. In the Tube concept, shown in Figure Figure 1.13, regularly spaced nodes are defined at each altitude and aircraft can only travel along edges that connect two nodes. Except for the tubes typology, aircraft separation is performed using one reactive collision avoidance method.The authors conclude the study by stating that "As a large number of conflicts and intrusions were found for all concepts, it is recommended to investigate novel conflict detection and resolution algorithms that cope with the limited maneuvering room available at extreme traffic densities " [32].

In a paper presented at SciTech in 2017, a concept of air highways was proposed with a system similar to traffic lights to handle conflicts at intersection. The concept is illustrated in Figure 1.14. Different types of routes and intersections were proposed with varying level of constraints for the vehicle on the route. The author argued that spacing between vehicles could be reduced as uncertainty on position was reduced. The concept was intended to

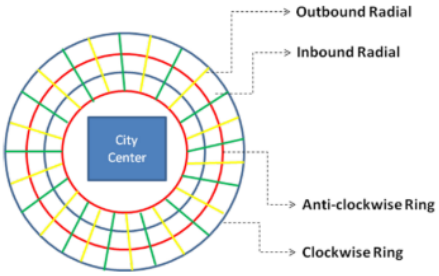


Figure 1.12: Metropolis Zones concept [32]

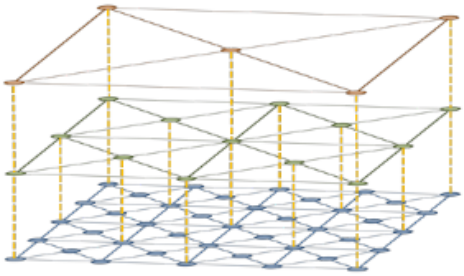


Figure 1.13: Metropolis Tubes concept [32].

handle traffic below the skyline of a large city [33]. An advantage of routes is that they can be designed to comply with other requirements: noise abatement, avoiding stadiums, schools or other highly densely populated areas. However, the intersection of routes are areas of high densities of traffic, making conflicts more likely to occur.

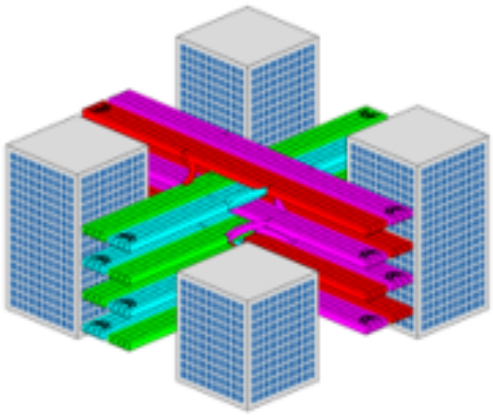


Figure 1.14: Sky highway concept [33]

In [34], the authors compare different systems that rely on changes in altitude to ensure deconfliction. They compare the number of layers that are required in a purely reactive case and in a case where UAVs' origin and destination are all known before-hand and trajectories can be deconflicted. The authors mention that dividing traffic by layers based on heading might not be required when a strategic assignment is performed.

For many of these ConOps there are no method or data available to compare them and

determine what could be the expected throughput and safety level for each of them. When studies do provide quantified metrics, they are often unique to that study. Moreover each study uses different assumptions which makes it even harder to do a meaningful comparison. No ConOps has been demonstrated to address all the desired capabilities in terms of capacity, safety and efficiency.

Observation

There are many different alternatives that have been proposed to organize low-altitude unmanned air traffic but there is no common framework to evaluate and compare them.

1.3 Conclusion

In the previous sections, the need for a low-altitude unmanned air traffic system that can accommodate a large quantity of UAM traffic under safety and efficiency constraints was outlined. A number of ConOps have been proposed but there is a lack of a common framework to compare them which would allow an informed decision to be made. Looking at the Air Taxi and Package delivery scenarios that have been proposed, there is no way to know if the ConOps that have been proposed would work at these densities and which one would provide the best performance. There is also no information on which elements of the use cases are important to model when making the decision.

The main research question that guides the thesis is then:

Research Question

How can unmanned low-altitude traffic management systems be systematically and quantitatively assessed and compared at a conceptual design stage?

A summary of the observations that were made in this Chapter and how they motivate the guiding question of the thesis is provided in Figure 1.15.

In the next chapter, a structured approach to conceptual design is introduced and is

used to guide the literature review and identify the elements required to answer the main research question.

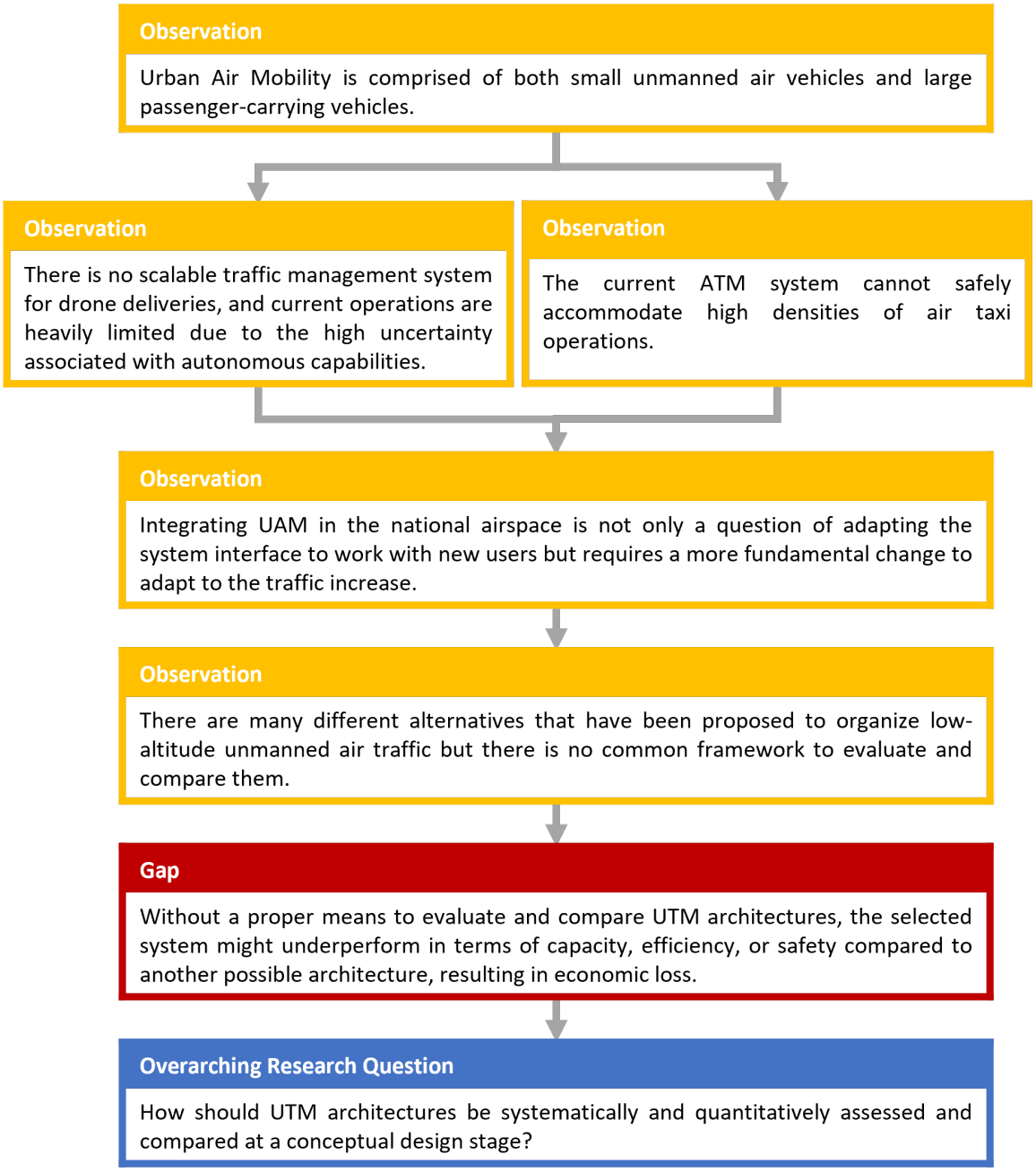


Figure 1.15: Summary of the motivation section

CHAPTER 2
LITERATURE REVIEW AND APPROACH

2.1 Objective and High-Level Approach

The previous chapter introduced the need for new Unmanned Traffic Management (UTM) systems. A rapid survey of proposed UTM architectures recognized the lack of a consensus on the process by which a UTM system should be designed and the difficulty to compare different concepts of operations to one another. This led to the research question guiding this thesis:

Research Question

How should UTM architectures be systematically and quantitatively assessed and compared at a conceptual design stage?

The literature review conducted in this chapter aims at identifying how to properly answer this question.

Employing systematic decision-making processes in the conceptual stage has been shown to maximize performance when designing a new system [35]. A top-down decision-making process at the conceptual design stage should follow these steps: (1) establish the need, (2) define the problem, (3) establish value objectives, (4) generate feasible alternatives, (5) evaluate alternatives, and (6) make decision. A literature review uncovered no application of such a systematic approach to the design of UTM systems. There are challenges to applying this methodology to UTM systems because, as explained in the previous chapter, requirements and objectives for UTM systems are not all clearly defined yet.

Some desired qualities of the system are already known. It should safely and efficiently coordinate a large number of low-altitude air traffic over urban areas in order to accommodate the forecast UAM traffic. It should also allow for a high-level of automation since

manual deconfliction performed by human operators does not appear to be realistic at such a large scale, if nothing else due to cost reasons. These aspects are however not quantified.

Some other constraints or parameters of the system are still unknown due to the rapidly evolving legislation and technology. For example, vehicle performance, the environment in which the system will be deployed, and the actual demand for the system are susceptible to change. Additionally, UAM encompasses very different types of operations and vehicles, ranging from passenger-carrying eVTOLs performing Air Taxi missions to small package delivery UAVs. Although they are grouped together under UAM, these vehicles operate under different constraints, desired performance and will operate in different environments. Since defining the need is not straightforward and there are good arguments to keep some flexibility at that stage, a good conceptual design framework to assess UTM alternatives should be adaptable. As outlined in [36], the understanding and knowledge of the design problem increases during the initial design phase, which is why a static approach is inferior to a dynamic parametric trade environment.

As a result, the following observation is made:

Observation

A structured top-down decision-making approach should be followed when designing a complex system under uncertain assumptions and constraints.

As a result, the objective of this thesis is to develop a framework that enables the generation, evaluation and comparison of UTM architectures in a systematic way under flexible assumptions.

Research Objective

Develop a framework that enables the generation, evaluation, and comparison of UTM architectures in a systematic way under flexible assumptions.

The top-down decision making approach should contain the following steps:

- Define Operating Scenarios: a set of external constraints, assumptions, or conditions

in which the UTM will operate.

- Generate UTM Alternatives: a method to generate feasible alternatives.
- Performance Criteria: a list of metrics measuring the performance of a UTM architecture.
- Evaluate: a method to get a quantitative estimation of the performance of a UTM architecture in a specific scenario.
- Make Decision: a method to rank alternatives based on the designer's preferences.

A literature review is conducted to identify what should be included at each step. As will be explained in the rest of the chapter, for some elements there is sufficient information in the literature while for others gaps are identified.

The remainder of the chapter is organized as follows. Each section focuses on one of the approach's step. They are presented in a different order than the decision-making approach so that concepts that are important when discussing the literature are introduced before they are referenced in other sections. The first section focuses on performance criteria. The second gives a quick overview of decision-making. These two steps are mostly addressed by the existing literature. The third section goes into details into UTM alternatives, it introduces a system decomposition, uses it to analyze previously proposed UTM ConOps and introduces different options for each subsystem. The fourth section focuses on the operating scenarios definition for UAM. The fifth and final section reviews evaluation method for UTM architectures.

2.2 Performance Criteria

In the background section, three main aspects were identified as important when designing a UTM system: capacity, safety, and efficiency. Indeed, an ideal UTM system would accommodate a large number of flights while being safe and minimizing delays and cost of

operations. Each of the next subsections focuses on one aspect. A review of the literature is performed and new metrics are presented where required.

2.2.1 Capacity

The traffic volume or density alone is not sufficient to evaluate the capacity of an airspace. Similarly to what is done when analyzing car traffic, the throughput, i.e. the number of aircraft exiting the airspace per minute, should also be evaluated [37]. In [13], the authors propose to use practical capacity to measure the scalability of airspace systems. Practical capacity is defined as the number of aircraft that the airspace can accommodate before the average delay becomes greater than a certain threshold.

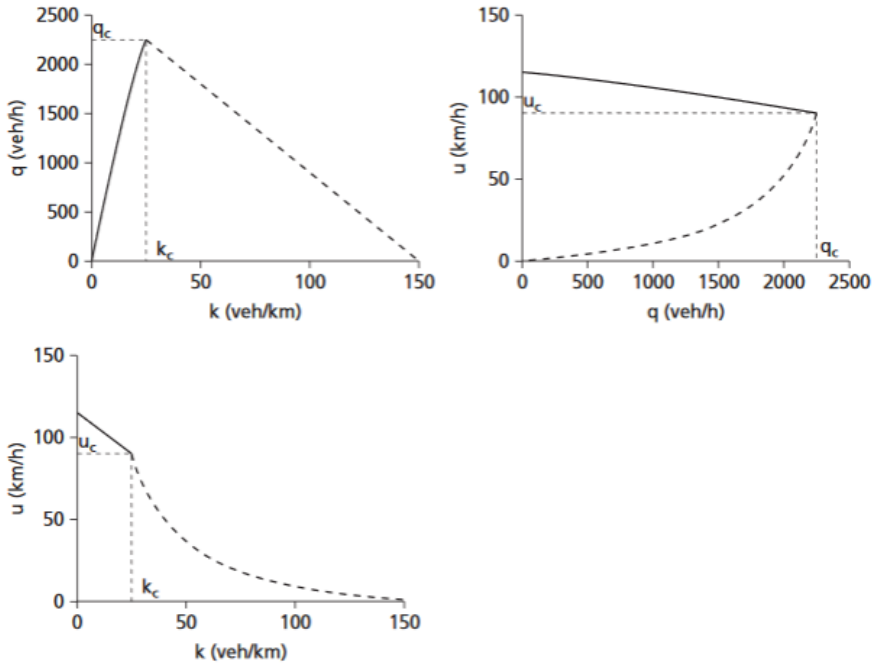


Figure 2.1: Fundamental traffic flow diagrams for road vehicles [38].

For road vehicles, traffic is often analyzed using density and throughput. The fundamental diagram of traffic flow in Figure 2.1 shows that as the density of vehicles, k increases, the average velocity, u decreases. Indeed, as density gets higher, vehicles must slow down to maintain separation with other vehicles, and the traffic becomes congested.

The model shows that velocity decreases faster than linearly with respect to the density. The throughput, q, is the product of the average velocity and the density. As a result it starts by increasing with the density but reaches a maximum and decreases if the density of vehicles is increased further.

In the proposed framework, the evolution of throughput and delays in function of agents density will be evaluated.

Throughput is related to the number of agents in the simulation, the average ideal travel time and the time efficiency. Trends in time efficiency will be reflected in the throughput, all other things being equal.

2.2.2 Safety

To measure the capability of an algorithm or a system to keep vehicles safe, a definition of safe operations, conflict and collision is necessary.

For manned aviation, different definition of safe operations exist. The code of federal regulations pertaining to general operating and flight rules 14 CFR 91.113b states that "[...] vigilance shall be maintained by each person operating an aircraft so as to see and avoid other aircraft. When a rule of this section gives another aircraft the right-of-way, the pilot shall give way to that aircraft and may not pass over, under, or ahead of it unless well clear". However, the definition of well-clear is not specified and remains at the appreciation of the pilot. For autonomous vehicles, RTCA introduced a well-clear definition based on the time to closest approach, the range and altitude differential which can be found in the minimum operating standards (MOPS) [39, 40, 41]. For IFR traffic, more straightforward separation minimums are used. Minimum separation distances define a protected volume around a vehicle (the ownship) in which other vehicles (the intruders) are not allowed to penetrate. A Loss of separation (LoS) or intrusion happens when an intruder penetrates this protected volume. The IFR vary depending on the area where the aircraft operate (terminal or en route) and the sensors used (type of beacon and surveillance system). The values

usually in application for commercial aircraft in cruise are 1000ft vertical separation and 5nm horizontal separation [42]. Different separation distance should be used for urban air mobility applications due to the difference in scale.

As explained in [43], there is an important distinction between conflict and intrusion. The intrusion or LoS is the actual breach of the protected area. The conflict is the predicted intrusion, hence it depends on the algorithm used to propagate the trajectory. If no action is taken a conflict will result in an intrusion, except for false positives.

The number of LoS per flight hour, n_{LoS}/h, is one of the metrics used for measuring safety. However, using the number of LoS per flight hour as the only measure of safety is not sufficient. As explained in [44], a conflict severity metric should be used to express the degree to which a loss of separation was close to a collision. In concrete terms, when comparing two scenarios where the same LoS happens, the scenario where the LoS leads to a lower minimal distance between ownship and intruder should be penalized more. The Horizontal Intrusion Parameter (HIP) accounts for this. It is defined as:

$$HIP = 1 - \min_{t \in [t_{SOC}, t_{EOC}]} \left(\frac{\Delta s(t)}{S_{std}} \right) \tag{2.1}$$

Where t_{SOC} and t_{EOC} are the times at which the LoS starts and ends, $\Delta s(t)$ is the horizontal distance between the aircraft at time t, and S_{std} is the minimum horizontal separation distance required. A HIP value of 0 means that the two vehicles are at the appropriate horizontal distance. A HIP of 1 means that the two vehicles are at the same horizontal coordinates. This however does not necessarily mean that there is a collision since the vehicles could be at different altitudes (with the difference in altitude being less than the required vertical distance).

As will be seen in the results chapters, most LoS that occur in the alternatives that are studied have a low severity. As a result, the average HIP value is not very informative. Looking at the distribution of HIP is more interesting as it shows outliers and their severity.

Moreover, although a LoS is a safety incident it is not a real threat if the minimum separation distance is very large with respect to the vehicles' size. To better capture the likelihood of a severe LoS, the number of Near Mid-Air Collisions (NMAC) is also measured. NMAC occur when two aircraft get within 500 feet of each other [45]. Here, the number of NMAC per flight hours, n_{NMAC}/h is used to measure how common severe intrusions are in the different alternatives.

2.2.3 Efficiency

Efficiency can be defined with respect to time or energy, and alternatives leading to good energy efficiency may not lead to good time efficiency. For example, the energy efficiency is maximized in [28] by always flying the shortest path, but this can result in long ground delays for the vehicle. In other instances, neither time nor energy is directly minimized. In the tube topology proposed in [32], the ground delay is minimized by selecting the first available path found by the shortest path algorithm even though it might result in a path that is longer and might even arrive later than a delayed start. In free flight approaches, the initial planned path is the shortest path and local changes are made in case of conflicts, so the paths are not optimized globally.

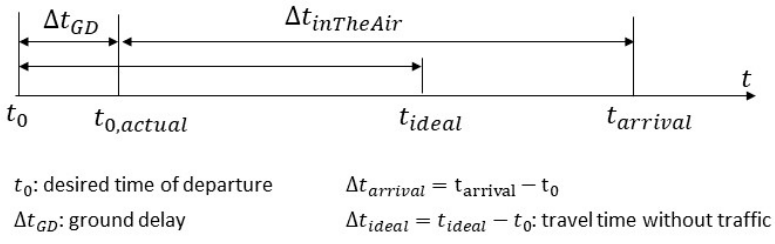


Figure 2.2: Definition of the time intervals of interest.

Figure 2.2 introduces the different time intervals that are used to define the efficiency metrics. $t_{0,actual}$ is the actual take-off time of the vehicle. Two different metrics are introduced to represent the time and efficiency metric.

The time efficiency is defined as:

$$\eta_{time} = \frac{\Delta t_{ideal}}{\Delta t_{arrival}} \tag{2.2}$$

The energy efficiency is defined as:

$$\eta_{energy} = \frac{\Delta t_{ideal}}{\Delta t_{intheair}} \tag{2.3}$$

Here time in flight is used as an approximation for the energy required for the flight. Time in flight was selected rather than path length due to the fact that hovering is part of some of the strategies being evaluated. In [32], the authors propose to use Work obtained by integrating the forward thrust along the path as the energy efficiency metric, but this requires a model of the vehicle and will not work well if vehicles can hover.

More efficient paths will lead to efficiency metrics values close to 1. As paths become less optimal, their efficiency metrics values will tend toward 0. The time efficiency is inferior or equal to the energy efficiency because $\Delta t_{arrival} > \Delta t_{inTheAir}$ by definition.

Note that rather than being expressed as efficiencies, some studies look at the same type of information and express it as a percentage extension in travel time or percentage extension in travel length or travel time [46]. This is equivalent to the efficiency as $A = 100 \times \frac{\Delta t_{ideal}}{\Delta t_{arrival}} = \frac{100}{\eta_{time}}$, where A is the percentage extension in travel time (including ground delays).

2.2.4 Conclusion

Authors have introduced many different metrics and definitions for safety, efficiency and capacity. A subset of metrics that are of interest to the designer and capture the different aspects can be selected from the literature. In the experiments that will be introduced in the next chapter the metrics that were used to measure UTM architecture performance are: Q the throughput, η_{time} the time efficiency, η_{energy} the energy efficiency, n_{LoS}/h the number

of Loss of Separation per agent flight hour, and n_{NMAC}/h the number of Near Mid-Air Collisions per agent flight hour.

Observation

There are many different metrics that have been proposed in the literature to measure safety, efficiency, and capacity.

2.3 Decision-Making

As explained in the objective and approach section, the final step of the conceptual design process is decision-making. The role of the conceptual design stage is to select the most promising architecture or subset of architectures. Indeed, the next stage of the design (preliminary design) requires more detailed analyses and cannot be realistically conducted on a large set of alternatives. The designers must choose from all potential architectures which ones are the most likely to result in the best system for their use case [36]. However as seen in the previous section, UTM architectures have multiple performance criteria that can be conflicting. For instance, a very energy efficient architecture might not be time efficient or a very safe architecture might have a limited throughput. In situations where there exists several architectures that are non-dominated in a Pareto sense, a method to rank architectures that are on the Pareto frontier is required. Hence, the decision-making problem that must be addressed here is a Multiple Criterion Decision Making (MCDM), and more specifically a Multiple Attribute Decision Making (MADM) since the architectures that are being evaluated can be viewed as a set of discrete choices rather than continuous variable to be optimized as will be shown in the next section.

A MADM problem starts with a normalized decision table or decision matrix that presents the value of the attributes for each alternative and a set of weights indicating the relative importance that the designer assigns to each attribute [47].

Table 2.1 presents a notional decision matrix with N alternatives and M attributes. $m_{i,j}$ is the value of attribute j for alternative i.

Table 2.1: Notional Decision Matrix

	Attribute 1	...	Attribute j	...	Attribute M
Alternative 1	$m_{1,1}$	...	$m_{1,j}$	...	$m_{1,M}$
...	...	...	...	...	...
Alternative i	$m_{i,1}$	...	$m_{i,j}$	...	$m_{i,M}$
...	...	...	...	...	...
Alternative N	$m_{N,1}$	...	$m_{N,j}$	...	$m_{N,M}$
Weights	w_1	...	w_j	...	w_m

The weights are positive for a desirable attribute and negative for an undesirable attribute. The units of the different attributes are usually different and the decision table must be normalized. This can be done by dividing each column by the largest elements so that $m_{i,j}^{norm} = m_{i,j} / \max_i m_{i,j}$ (normalization using ℓ^∞ norm). The weights are also typically normalized so that the ℓ^1 norm of the weight vector is equal to 1 (taxicab norm or sum of the absolute value of the vector elements). We can remark that due to the normalization of the decision matrix if all alternatives were to have the value of an attribute change by a common positive k-factor k_j such that $m_{i,j}^{new} = k_j m_{i,j}$, the normalized decision matrix would not change whichever norm is used for normalization. Indeed, the norm has an absolute homogeneity property $||k\mathbf{V}|| = |k|.||\mathbf{V}||$.

There are many different decision-making methodologies that have been proposed that can help a designer rank alternatives once they have been evaluated [48]. They have different benefits and drawbacks but determining which method is the most appropriate at this stage of the design is out of scope for this thesis. Here we present two simple MADM techniques that will be used in the thesis.

SAW. In the simple average weighted (SAW) methodology, the score of each alternative is simply the weighted sum of the performance metrics for one alternative. Assuming the decision matrix and weights have already been normalized, the score of one alternative S_i

can be expressed as:

$$S_i = \sum_{j=1}^{M} w_j m_{ij}$$

The alternatives are ranked from the highest to the lowest score. This method is very straightforward to understand.

TOPSIS. The Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) method scores alternatives based on their euclidean distance to ideal solutions. The decision matrix is normalized by the 2-norm (euclidean norm) of the attribute vector $Att_j = [m_{1,j}, ..., m_{N,j}]^T$. The weighted and normalized alternative vector is then:

$$R_i = [w_1 m_{i,1}/||Att_1||_2, ..., w_i m_{i,N}/||Att_N||_2,]$$

A positive ideal solution is defined as $R^+ = [w_j \text{ best}(Att_j) \text{ for} j = 1, ..., M]$, and a negative ideal solution is defined as $R^- = [w_j \text{ worst}(Att_j) \text{ for} j = 1, ..., M]$. Where best is the maximum of Att_j if Attribute j is a desirable attribute and the minimum if Attribute j is an undesirable attribute, and worst is the opposite. The weighted distance to the positive and negative distance are then respectively defined as $D_i^+ = ||R_i - R^+||_2$ and $D_i^- = ||R_i - R^-||_2$. The relative distance to the negative ideal solution of one alternative can then be expressed as $S_i = D_i^-/(D_i^- + D_i^+)$ which would be 1 for the positive ideal solution and 0 for the negative ideal solution. The alternatives are ranked from the highest to the lowest S_i value. Compared to SAW, if two values are similarly close to the positive ideal, TOPSIS will give a higher weight to the one that is furthest from the negative ideal.

Observation

There are many different decision-making framework in the literature that can be tailored to the designer's needs.

The impact of the choice of decision-making methodology will not be evaluated in this work. A decision-making methodology will be taken directly from the literature and applied to the results of the alternative evaluations.

2.4 Generate UTM Alternatives

The role of the UTM system is to provide a framework that allows operations to occur in a safe and efficient manner at scale. There are multiple ideas that have been proposed in the literature to achieve this core function. Different subsystems are considered and combined to achieve the function. Some rely on a centralized authority and detailed flight plans to ensure separation, some rely on onboard autonomy, some rely on airspace rules such as routes, and some rely on a combination of these elements. There is no formal structure or classification that is followed in the definition of a ConOps for a UTM system.

To generate feasible alternatives, a common approach is to breakdown the system into subsystems and to identify what are the potential options for each of them. This allows to create a morphological matrix or matrix of alternatives [49]. When different options are not compatible, a compatibility matrix can be created. With this approach all alternatives can be generated in a systematic manner.

Observation

A matrix of alternatives must exist to ensure that a systematic review of all feasible architectures is conducted.

As stated before, in the literature the analysis of UTM systems is usually conducted for a specific ConOps. No system decomposition for UTM systems or systematic alternative generation method could be found. This results in the first gap:

Gap

There is no existing system decomposition for UTM architectures, which prevents a systematic exploration of the design space. Potential UTM architectures might be neglected.

As a result some work is required to identify what subsystems should be included to define UTM alternatives.

The following subsection looks at the literature in the commercial aviation domain to

find inspiration for a UTM system decomposition. Four subsystems are identified and discussed in more details in the following subsections.

2.4.1 System Decomposition

For manned ATM an ontology for the traffic management system was presented in [50]. Keller introduces a breakdown of the system in the following subsystems:

- airspace components (sectors, routes, etc.),
- departure and arrival routes,
- flights (flight plans and flight paths),
- control facilities (towers, air route traffic control centers, etc.),
- traffic management initiatives (ground delay programs, miles-in-trail, flow constrained areas, etc.),
- airlines,
- aircraft (vehicle type and performance),
- infrastructure (airports and their subsystems such as terminal, gates, runways, etc.),
- weather.

This ontology was developed for data modeling of the existing ATM system, but such a decomposition would be useful to generate and classify feasible alternatives for unmanned traffic management systems.

However, this ontology is not directly applicable to generate alternatives for UTM systems. First, it is tailored to the current commercial air traffic management system and some subsystems such as traffic management initiatives are very specific. Second, it has a high level of detail which is not manageable for a conceptual design step of a system-of-system.

Third, it mixes subsystems that can be designed by the decision-maker, such as airspace components or traffic management initiatives, with subsystems that are outside of the system designer control, such as weather or airlines.

This is why we propose a simple decomposition of UTM systems into four subsystems, loosely based on the ATM ontology, to generate alternatives:

- Airspace structure,
- Access control,
- Preflight planning, and
- Collision avoidance.

The infrastructure, weather and airlines subsystems are not modeled in the preliminary framework and only a simple model for the aircraft is implemented in order to keep the complexity manageable at the conceptual design stage.

This proposed system decomposition is also useful to analyze the literature in an organized manner as illustrated in Table 2.2.

The next four subsections discuss in more details the alternatives that have been proposed for each subsystem. This allows to build a matrix of alternatives for the UTM system. Finally, the last subsection introduces the first research area that stems from the observations and gaps that are identified while reviewing the literature.

2.4.2 Airspace Structure

We define airspace structure as the rules that regulate airspace navigation. There can be many different types of structures:

- Free-Flight: Aircraft can fly their preferred path to their destination (also called free routing)

Table 2.2: Proposed Alternatives for Unmanned Traffic Management System in Literature

Architecture		Airspace Structure	Access Control	Preflight Planning	Collision Avoidance	Simulation Framework
Full Mix	[32]	Free	None	None	Decentralized (MVP)	TMX
Layers	[32]	Layers	None	None	Decentralized (MVP)	TMX
Zones	[32]	Sectors with heading range	None	None	Decentralized (MVP)	TMX
Tubes	[32]	Route Network	4DT contract at the nodes	A* (first available)	None	TMX
NASA UTM	[27]	Free	Volume	Volume Reservation (manual)	None	N/A
Iowa UTM	[28]	Free	Volume	Static A*	None	Custom 2D
Sky Highways	[33]	Routes	None	None	1D velocity adjustment	Custom 1D
DLR Delivery Network	[51]	Free	4DT contract	Decoupled Scheduled (ground delay heuristics, altitude maneuver)	None	Custom 3D
Onera VLL Operations	[52]	Free	4DT contract	Decoupled with ground delays	Swarm Algorithm	N/A
Altiscope	[53]	Free	4DT	Local collision avoidance maneuver	Static iterative method	Custom 2D
Ames Reactive	[37]	Free	None	None	ICAROUS-based, hover, potential field	Custom, Fe3

Table 2.2 – continued on next page

Table 2.2 – continued from previous page

Architecture		Airspace Structure	Access Control	Preflight Planning	Collision Avoidance	Simulation Framework
Linköping distributed	[34]	Layers	None	None	Hovering, Layer Change	Custom 3D
Linköping centralized	[34]	Layers	4DT Contract	Ground delay, layer assignment	None	Custom 3D
U-space Y volume	[30]	Free	Volume	Volume Reservation (manual)	None	N/A
U-space Z volume	[30]	Free	None	None	Automated (Zu) Manual (Za)	N/A
TM-UAS	[54]	Sectors	4DT	N/A	None	Custom using AGI STK

- Routes: Aircraft can be constrained to fly on a network of airways defined by specific waypoints
- Layers: Aircraft altitude can be constrained based on the type of operation and/or direction of travel
- Sectors: The airspace is divided into areas that can restrict access or impose additional rules

The current National Airspace is a mix of all these structures. Papers that proposed alternatives for unmanned air traffic management have often focused on a single airspace structure, with the notable exception of the Metropolis project [32].

The free routing concept, which was introduced in the late 90s, aimed at improving flight efficiency for commercial aviation by allowing aircraft to fly directly to their destination once at cruise altitude instead of having to follow airways or routes. Indeed, the development of GPS made it easy to follow any route instead of traveling from beacon to beacon and relying on dead reckoning. The FAA started implementing some free-routing areas at or above FL390 as part of the High Altitude Redesign (HAR) initiative. The aircraft can enter and exit the HAR airspace through pitch and catch points that make the interface with regular airspace. Waypoints in the HAR airspace are regularly spaced by latitude and longitude to form a grid, which allows a lot of flexibility. Phase 1 of the initiative was implemented in 2003 but it looks like it was not extended or evaluated further [55]. Similarly, Eurocontrol has been deploying Free Route Airspace (FRA) over Europe since 2008 and is expecting to save as much as 45,000 tons of fuel a year once the system is completely deployed [56].

Although horizontally the airspace is unstructured, there is usually a vertical structure in free routing concepts. A common vertical structure, that will be referred to as **layers** in the rest of the thesis, is to segregate aircraft by altitude based on their heading. This is already the case for VFR or IFR flights. For instance, according to CFR 14.91.159, VFR

aircraft with a heading between 0 to 179 degrees should be flying at an odd thousand foot altitude plus 500 feet (e.g. 7500 feet MSL), whereas aircraft flying in the other direction are supposed to fly at an even thousand foot altitude plus 500 (e.g 4500 feet MSL).

Different papers tried to evaluate the impact of free routing and layers on conflicts number and severity. In a paper by Bilimoria and Lee from 2001, traffic was analyzed over 24 hours and compared to direct free-routing trajectories using the FACET simulator developed by NASA. Aircraft's altitudes were chosen according to FAA requirements, i.e. it was implementing a layer structure. It was shown that 29% of aircraft were involved in a conflict for structured routing whereas that number was only 27% for the free routing case. Moreover, there were more conflicts with a high intrusion rate (i.e. the aircraft were predicted to come closer to each other if no avoidance was performed) in the structured routing case [44].

The positive impact of layers were similarly demonstrated for for Urban Air Mobility applications in [32] and [53]. According to a study by Altiscope (now Airbus UTM), some structures, such as layers, are required to ensure safe operations: "Relying only on simple preflight deconfliction rules and allowing flights to operate without having to use traditional airspace structures (e.g. altitude separation, one-way routes or charted arrival procedures) simply does not provide a path to safe airspace usage" [53].

On the other hand heavily structured airspace such as routes and sectors were shown to increase local density and result in more conflicts. In a high-level study by Vascik et al., a management of low-altitude traffic using tactical control by airspace sector was deemed "unlikely to be feasible for the densities of flights anticipated for UAS and UAM networks" [17].

Table 2.3 summarizes the alternatives that were identified for the Airspace Structure subsystem.

Table 2.3: Airspace Structure Alternatives

Subsystem	Alternatives
Airspace Structure	Free Layers Sectors Routes

2.4.3 Access Control

Access Control is the system that regulates traffic flow by enforcing who can access sections of the airspace.

Free-access. Current VFR traffic in uncontrolled airspace operates without access control, any aircraft can take-off if there are no immediate conflicts. There are some collaboration and rules for take-off and landing at non-towered airports but in cruise separation is ensured through ”see and avoid”. In the late 1990s to early 2000s there were proposals to combine free-flight structures (i.e. no routes) with moving the separation insurance responsibility to the aircraft rather than relying on air traffic controllers [57]. Free-access alternatives have been proposed as part of several UTM ConOps [58, 34, 59, 30]. The Metropolis study was a European project in which Urban Air Mobility was studied in a fictitious city that concluded in favor of a free routing free-access approach compared to more rigid structures [59]. The U-space's Z volumes would also rely on free-access for high density of operations [30]. However, other studies have also identified strong limitations of the free-routing system. In dense constrained cases conflicts might be unavoidable without a centralized decision maker [60].

Capacity. For commercial traffic ATC can use capacity to regulate traffic by sectors in order to keep workload manageable for air traffic controllers.

Volumes and 4DT Contract. In the NASA UTM access control is regulated by reserving volumes of airspace, the protected volume is called a geofence. The volume is reserved to a single vehicle from take-off to landing. An alternative is to have the protected volume dynamically follow the vehicle. This is also known as 4-Dimensional Trajectory (4DT). In

4DT, an aircraft flight plan not only states its trajectory in space but also in time, which would allow to coordinate trajectories strategically rather than reactively, while keeping the blocked volume of airspace to a minimum. 4DT is a key element of Trajectory Based Operations (TBO) which is being studied as part of the FAA NextGen initiative [61]. If the 4DT trajectory is perturbed beyond its tolerance due to disturbances or disrupted by higher priority or non-compliant vehicles, it would have to either try to re-plan or fallback to a decentralized approach [52]. In a study conducted on a medium term 4DT ConOps for commercial aircraft in cruise, simulation showed that the probability of a loss of separation is greatly reduced by the use of medium-term 4-Dimensional Trajectory (4DT) planning. Results also indicated that as traffic increases the distance traveled by the aircraft increases as well, resulting in a loss of efficiency [62].

Table 2.4 summarizes the alternatives that were identified for the Access Control subsystem.

Table 2.4: Access Control Alternatives

Subsystem	Alternatives			
Access Control	Free	Volume Reservation	Capacity-based	4DT-contract

2.4.4 Collision Avoidance

Collision avoidance methods are reactive actions that aim at maintaining an appropriate distance between agents. The agents' paths are updated as they go along based on the surrounding traffic. Reactive strategies can be :

- Centralized: agents are in communication with a central authority which is in charge of providing deconfliction capabilities. The current ATC system for IFR flight is centralized.

- Decentralized: agents follow sets of rules to avoid each other with only basic information about surrounding traffic (velocity, position). For instance for VFR traffic, pilots must ”see and avoid” each others. In [37], the authors compare several decentralized collision avoidance methods. Intruders can either be cooperative, i.e. they also try to avoid the ownship, or uncooperative, i.e. they do not alter their path to avoid the imminent loss of separation. Decentralized collision avoidance methods can take advantage of cooperative intruders to reduce the ownship collision avoidance effort without the need for communication.

- Collaborative: agents can communicate to decide on avoidance maneuvers. For instance on commercial airplanes the Traffic Alert and Collision Avoidance System (TCAS) of two aircraft involved in a conflict communicate to coordinate their resolution advisories [63].

In the case of automated detect and avoid (DAA) systems on-board vehicles, collision avoidance algorithms are often referred to as conflict detection and resolution (CD&R) algorithms in the aerospace community. Collision avoidance algorithms have also been extensively studied in the mobile robotics community. The next subsections will go over some concepts and proposed algorithms for collision avoidance.

ACAS Xu

In December 2020, RTCA published the ”Minimum Operational Performance Standards for Airborne Collision Avoidance System Xu (ACAS Xu)” [64] paving the way for certifiable UAVs that could operate BVLOS without human supervision.

The ACAS X program was developed as part of FAA’s NextGen to overhaul the existing TCAS II. It integrates new sensors such as ADS-B (Automatic Dependent Surveillance - Broadcast) and is compatible with small aircraft or unmanned vehicles that could not use TCAS. The fundamental logic of the algorithm is completely modified to provide more optimized responses and fewer false alerts. ACAS X models the collision avoidance problem

as a Markov Decision Process (MDP), which is a probabilistic representation of the problem. The MDP is solved offline using a dynamic programming algorithm. The solution is used to generate a lookup table that is used onboard to select the most appropriate action for the vehicle [65]. ACAS Xu is a flavor of ACAS X designed for unmanned vehicles. Compared to ACAS Xa (for large aircraft), ACAS Xu includes the possibility of fusing onboard sensor to detect uncooperative traffic, tailored resolution tables to adapt to varying vehicle performance, and horizontal resolutions in addition to vertical maneuvers [66].

ACAS Xu standards are not freely available and the algorithm is very complex.

Velocity Obstacles

In robotics a lot of methods have been developed for obstacle avoidance with static obstacles. Although these methods can be adapted to work with moving obstacles, especially if the obstacles move slowly with respect to the ownship, their performance is degraded and they must be run iteratively. To address moving obstacles different algorithms have been developed using the concept of Velocity Obstacle (VO).

Figure 2.3 shows a conflict viewed in a ground reference frame, with a circle showing the protected area around the intruder at a time during the conflict. By using relative velocities and considering the conflict in the intruder reference frame as shown in Figure 2.4, the protected area that the ownship must avoid is fixed. From there it is easy to see that any relative velocity in direction of that protected area would eventually result in a conflict with the intruder. This set of all relative velocities that would result in a collision is shown in blue on Figure 2.5, and is called a VO.

As can be seen in Figure 2.5 the VO is a cone whose sides are tangent to the disk centered around the intruder at $p_i - p_o$ where p_o and p_i are the respective positions of the ownship and the intruder. The disk radius r is equal to the sum of the radius of the protected area of the ownship r_o and the radius of the protected area around the intruder r_i. In most cases the separation minimums are the same for all aircrafts, so $r = r_o = r_i$. If planning

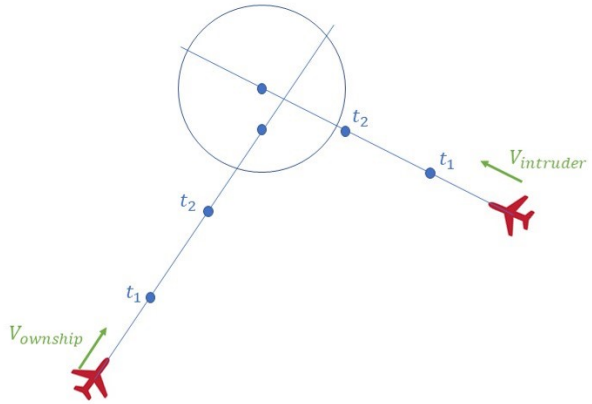


Figure 2.3: Conflict viewed in a ground reference frame

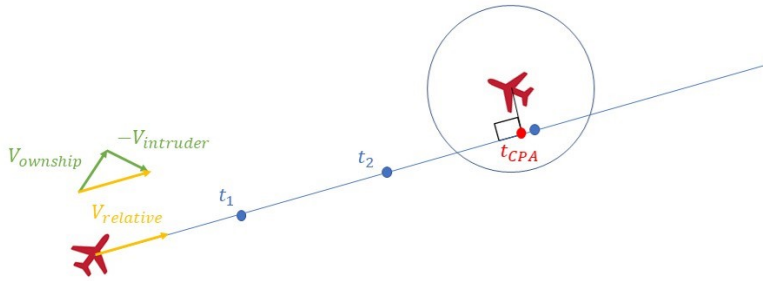


Figure 2.4: Conflict viewed in the intruder reference frame

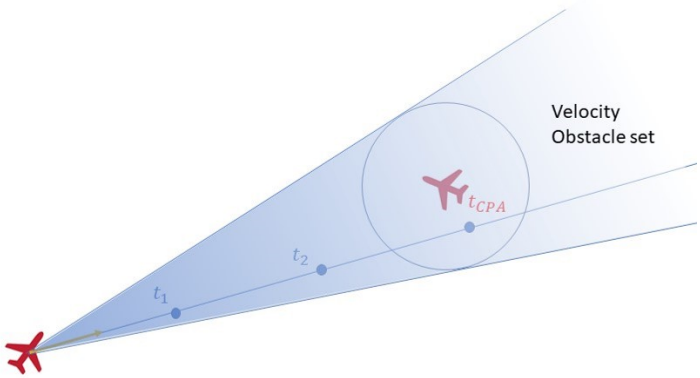


Figure 2.5: Velocity Obstacle set illustration

over a finite horizon τ, the cone is truncated by a circle centered at $(p_i - p_o)/\tau$ whose radius is r/τ.

By definition picking a relative velocity inside the VO will lead to a loss of separation, so VO algorithms try to find which relative velocities would bring the ownship closer to its goal while being outside of the collision cone.

In the geometric optimization method presented by Bilmoria in [67], different strategies are proposed to choose a relative velocity outside the VO: a heading change, a speed change or both. The goal of each of these strategies is to minimize the velocity change required from the ownship. However, the method is developed for conflicts that only involve two vehicles. If more aircraft are involved in the conflict, the author proposes to solve conflict sequentially based on the time to first loss of separation.

Modified Voltage Potential

The Modified Voltage Potential (MVP) algorithm was originally developed by Eby in [68] based on a geometric heuristic but was subsequently improved by Hoekstra as illustrated in [69] using VO concepts.

MVP only considers intruders that are in direct conflict with the ownship and not all neighboring intruders. MVP works by finding the time of closest approach between the ownship and an intruder. A derivation of the formula for the time of closest approach is provided in Appendix A. If there is a conflict, i.e. if the distance between the intruder and the ownship is projected to be less than the minimum separation distance, then the algorithm computes the intrusion vector and identifies the velocity change required for the ownship to avoid the conflict. In cases where there are multiple conflicts, the velocity changes required are summed. The method provides an exact solution when there is only one intruder and its velocity is constant, and yields a usually satisfying approximation otherwise. Note that the resulting velocity vector might not be feasible (e.g. the desired velocity might be larger than the maximum velocity).

The MVP algorithm is used in several studies and simulators [32, 70, 71].

ORCA

The previous method is only exact when there is a single intruder. A better strategy is to consider all the VOs created by each intruder and to pick a velocity outside of the union of all VOs [72, 73]. However, as pointed out in [74], one issue that arises when all agents are using a VO-based method for collision avoidance at the same time is the potential development of oscillations. This can also occur with MVP. The Optimal Reciprocal Collision Avoidance (ORCA) method was developed to address the issues that occur with collaborative VO-methods.

A brief overview of the ORCA algorithm is given here, for a detailed explanation the reader is directed to the paper by Berg et al. that presents the algorithm [75]. ORCA formulates the VO using a time horizon, meaning that conflict that would arise after a certain time will not be considered. This time horizon manifests itself by rounding the tip of the VO cone. ORCA simplifies the problem of finding the optimal velocity outside of the union of the VO sets by representing the constraint created by an intruder as a single constraint tangent to the VO created by that intruder. The constraint is different between cooperative and non-cooperative intruders or static obstacles. When the intruder is cooperative the avoidance burden can be shared in half which moves the linear constraint further away. Thanks to the linearization, a linear programming method can be used to find a solution to the constraints created by all the intruders quickly. The algorithm is very fast and has been used in crowd simulations where it was able to handle hundreds of conflicts at a time. It can handle a combination of cooperative and non-cooperative agents. However, there is no guarantee that a solution can be found in dense conditions, i.e. all vehicles might come to a standstill.

The method has been adapted to handle non-holonomic vehicles and take into consideration the dynamics of the vehicle. In [60], the authors apply the ORCA algorithm to

aircrafts with velocity constraints. They show that for aircrafts entering conflicts with a shallow difference in heading, the ORCA algorithm tends to increase the length of the conflict by making aircrafts take parallel tracks rather than resolving the conflict. A centralized evolutionary algorithm is shown to perform better in dense constrained scenario than the ORCA algorithm. They later proposed a slight modification of the algorithm for constant speed aircrafts that limit the converging track problem [76].

Other approaches

Some methods rely on changes in altitude for collision avoidance (similarly to TCAS). By having the lowest priority agent either descend to another altitude and continue its flight there or having the lowest priority agent descend and hover at a dedicated ”safe” altitude while the higher priority agent continue its flight [34], conflicts can be resolved using simple rules. However, in order to keep the simulation relatively simple, it was decided to focus on 2D operations.

Discrete methods discretize the possible actions that the vehicle could take over an horizon and estimates the best series of actions required to exit the conflict. A few discrete methods are described in the next paragraphs.

A team at NASA Langley developed a software called DAIDALUS (Detect and AvoID Alerting Logic for Unmanned Systems) which provide collision detection and resolution notifications for a UAS remote pilot. The algorithm provides ranges of maneuvers that will result in a loss of well-clear, and if the vehicle is already breaching the minimum separation it provides the ranges of maneuvers that will result in the earliest resolution. To determine these ranges, feasible commands for the vehicle either in terms of longitudinal velocity, vertical velocity, or heading are generated and propagated over a finite time horizon [77].

In [78], a Monte-Carlo Tree Search (MCTS) approach is used to evaluate potential trajectories. The vehicle is limited to a discrete set of actions: it can turn to the left by two degrees, keep flying straight, or turn two degrees to the right. The actions are propagated

for 1 second, then a new action is chosen. This process is carried out for a few time steps, building a tree of potential trajectories. To evaluate a trajectory, a random trajectory is generated from the leaves of that tree and scored based on 1) whether it resulted in a collision and 2) the proximity to the goal. The scores of the leaves are back-propagated to their parents. The action that is most likely to result in a high score is selected.

A final option is to not have collision avoidance. This means that the agent either follows its flight plan or fly straight to its goal at constant speed. In cases where flight trajectories were deconflicted before take-off and agents are assumed to be perfectly able to follow their 4DT contract there is no need to model a collision avoidance behavior. On an actual system, there would still be a collision avoidance system as backup. In cases where agents are not deconflicted prior to take-off and simply fly straight to their goal, this is not a realistic option but it can help gain insight into the system. This approach has been used in the literature to roughly estimate the number of conflicts that can be expected given a certain demand pattern and density of operation. The main advantage of this approach is that it can be modeled very quickly since it does not require a step-by-step modeling of agent behavior. Instead an event-based approach using time of departure and arrival of agents can be sufficient to identify conflicts. This approach is used in [44] to make the case for free routing versus routes.

Summary

As can be seen from the partial survey above, there are many different algorithms that have been proposed for collision avoidance. The algorithms vary in terms of complexity, public availability, constraints considered, and types of resulting maneuvers. Usually papers that focus on collision avoidance compare a new algorithm to one or several previously proposed algorithms in a given set of conditions. There is rarely enough information how sensitive these algorithms might be to external factors or change in conditions.

Table 2.5 presents some of the alternatives that were identified for the Collision Avoid-

ance subsystem.

Table 2.5: Collision Avoidance Alternatives

Subsystem	Alternatives						
Collision Avoidance	None	Manual	MVP	Hover	ORCA	ACAS Xu	...

2.4.5 Preflight Planning

Preflight Planning is the algorithms used to strategically plan a path before take-off. If there is an access control system then there must be rules and algorithms in place for agents to request access.

As can be seen in Table 2.2, a number of ConOps are formulated with the operator manually planning the trajectory. However, we are interested in systems with a high-level of automation.

The problem of finding the shortest path while avoiding static obstacles has been studied in the robotics community. Many of the methods developed for static obstacles can theoretically be applied to moving obstacles by replacing a spatial representation, or configuration, with a configuration-time representation, i.e. by finding a path in 4 dimensions. An example of configuration-time representation is provided in Figure 2.6. Moving obstacles are fixed in the configuration-time representation, so static methods could be used. For instance, the classic shortest path algorithm A* could be used to plan a path in 4D [79, 80]. However, this approach can be impractical due to the the increased dimensionality of the solution space.

To reduce the cost associated with trajectory planning, some papers have proposed a decoupled strategy in which the spatial path is first determined and then the velocity required to avoid conflicts along that path is determined. In [28], the 3D path is fixed to be the shortest path and a ground delay is applied until the path is clear of conflict. In

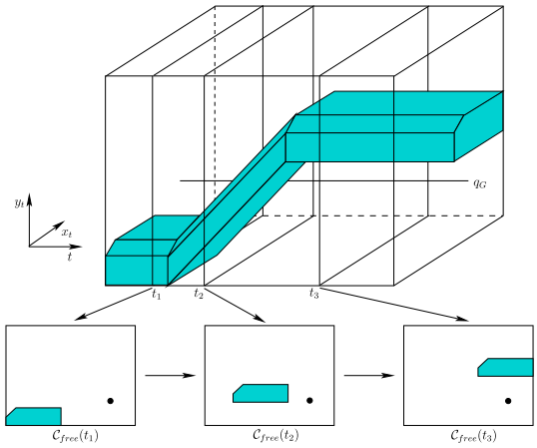


Figure 2.6: State-time representation of a moving obstacle in 2D [80].

[51], the vehicle's initial strategy is a straight path at the maximum altitude and the paper compares and combines three heuristics to perform conflict resolution. Two of them act on ground delays while the third considers local avoidance maneuvers. In [52], the authors fix the 3D path and propose to use simulated annealing to find a ground delay that minimizes a weighted sum of conflict duration and delays. In [52] and [51], the analysis is done for scheduled flights, which allow to optimize the overall traffic as opposed to just one flight.

In the tubes topology presented in [31], the authors use A* with a fixed start time to find a path in a route network with dynamic agents. If no solution can be found, the start time is incremented by a fixed amount and the search is rerun. However, this can result in very long path. I.e. there can be an available path that takes large detours and arrives later than another path that would have departed later but arrived sooner. This strategy is shown to perform poorly when compared to strategies without preflight planning in a less constrained environment. However, rather than fixing the start time, the time could be considered like an additional dimension of the search problem. This would ensure that the best trajectory in terms of arrival time can be found, rather than the trajectory resulting from the first available path in the search. The reason why the author did not include the time dimension in the A* search but rather externally incremented the time of departure independently is that A*'s computational cost increases strongly with the dimensionality

of the search problem. Looking at the robotics literature showed that a method called Safe Interval Path Planning (SIPP) had been developed to partially overcome the curse of dimensionality incurred when including the time dimension [81]. However, the SIPP algorithm has not been evaluated in a UTM and UAM context.

In [53], the authors use a local collision avoidance algorithm to create a trajectory that does not conflict with previously filed flight plans. The collision avoidance algorithm used does not take into account the speed of the intruder. It solves the conflict iteratively by applying a heuristic considering the position of ownship and intruder at a given time. It can result in non-optimal paths or fail to find a solution when there are multiple intruders. When the method fails to find a valid path the flight is either rejected or the agent still goes through with it, which results in poor safety. This idea of using a local method to plan a global path is interesting, but there is a lot of room for improvement in the method proposed by Altiscope. Using a better local collision avoidance method that does take into account the velocity of the intruder, and handling cases where the local planner fails to find a solution would result in a more robust solution.

In [82], finding a path through moving obstacles is formulated as a constrained linear optimization problem. In order to write the problem under a linear canonical form, i.e. a linear cost function and linear constraints, time is discretized and the vehicle dynamics are linearized. To represent constraints introduced by convex obstacles at every time step, **OR** and **AND** statements must be introduced. These statements can be converted to a linear form through the use of binary slack variables. Under this formulation the problem is a large mixed integer-linear program (MILP). The number of variables increases quickly with the number of time step and this approach is not really scalable.

Observation

Proposed 4DT planning algorithms in the UTM literature have shortcomings.

Table 2.6 presents some of the alternatives that were identified for the Preflight Planning subsystem.

Table 2.6: Preflight Planning Alternatives

Subsystem	Alternatives					
Preflight Planning	Static Shortest Path	Decoupled	Altiscope Local	MILP	SIPP	

2.4.6 Research Area 1

The alternatives identified in the previous subsections are then used to build a matrix of alternatives, shown in Table 2.7.

Table 2.7: Matrix of Alternatives

Subsystem	Alternatives						
Airspace Structure	Free	Layers	Sectors	Routes			
Access Control	Free	Volume Reservation	Capacity-based	4DT-contract			
Collision Avoidance	None	Manual	MVP	Hover	ORCA	ACAS Xu	...
Preflight Planning	Static Shortest Path	Decoupled	Altiscope Local	MILP	SIPP		

The four proposed subsystems have proven useful to analyze the literature and highlight elements that have not been evaluated in-depth. As can be seen in Table 2.2, many studies have proposed a single UTM architecture or have varied one or two subsystems while keeping the others fixed. In the context of a conceptual design stage, were these studies justified in the assumptions they made for certain subsystems or is it important to consider all subsystems in the design? At a conceptual level, the design space to analyze can be large. If some simplifications can be made or some subsystems can be neglected until later in the design process, it would reduce the cost of the analysis and of the conceptual design stage. At the same time, we wish to make a decision that will maximize the satisfaction

of the customer. If these simplifications lead to a different choice that does not satisfy the customer as well, then they should not be made.

This is summarized by the following high-level research question:

Research Question 1

Which subsystems should be included when evaluating architectures at a conceptual design stage?

The question is not straightforward to answer because this system decomposition has not been proposed before. Each ConOps presented in Table 2.2 is evaluated using different simulations and under different assumptions. When different options for subsystems are considered in a study, the others are kept fixed which prevents an evaluation of potential interactions.

Observation

UTM architectures are evaluated using a wide range of tools under different assumptions and the impact and interactions of the proposed subsystems have not been measured.

The different tools that have been used to evaluate UTM architectures' performance will be discussed in more details in section 2.6.

In the study presented in [32], the airspace structure is varied but the collision avoidance method is kept constant so there is a lack of information on whether the same change in performance can be expected when other algorithms are used. Other studies have looked at the impact of layers on conflict count in isolation of other architectures elements [83]. However, there is a lack of information on how conflict count exactly impact airspace performance. The higher the conflict count is, the more one can expect the paths to lose efficiency since more maneuvers are required and the more likely it is that there would be a loss of separation, but this is not quantified.

This leads to these two gaps:

Gap

The impact of the autonomy/behavior at the agent level on the airspace performance overall has not been evaluated.

Gap

The impact of airspace structures has not been evaluated across a range of architectures.

2.5 Define Operating Scenarios

In the different studies that are available in the literature different assumptions are made about the environment in which the UTM system is deployed. A few external factors that are relevant to the two use cases that were described in the introduction are described in the next subsections.

2.5.1 Demand

Some studies assume that the demand is uniform in their analysis [83, 33]. However, the demand for low-altitude air mobility is unlikely to be uniform across the area controlled by the air traffic management system. The type of demand pattern will also depend on the use case.

For package delivery, a common assumption is that deliveries will be done from a distribution center to a customer [51, 28, 52]. Hence, a hub-and-spoke traffic flow pattern can be expected, with high densities of traffic near the distribution centers (hub) and more uniformly distributed densities in the rest of the area. The studies mentioned above fix the demand and the UTM architecture and analyze the results.

Air taxi or air metro might operate with a similar flow pattern as commuting flows between business centers and residential suburbs [53]. In an analysis of UAM mission constraints presented in [12], demand for UAM services is estimated using a combination

of commuting information, household income, existing on-demand air services and points of interest such as stadiums or transportation hubs. The study presented in [84] uses census data to generate origin and destination pairs proportionally to the population density. In the metropolis study different demand patterns were created and the results presented in [31] are averaged over different demand scenarios, which prevents analyzing how each proposed architecture is affected by changes in the demand patterns.

2.5.2 Static Obstacles

The low-altitude airspace above an urban area is scattered with obstacles. The obstacles can be restricted airspace such as controlled airspace around airports or temporary flight restrictions around stadiums. Obstacles can also be man-made constructions such as radio towers or tall buildings. Figure 2.7 shows how cluttered the low-altitude airspace above a large city like Atlanta can be. On this terminal area chart, multiple airports represented by small blue circles and blue rectangles can be seen around around the city center, and tall constructions are shown as elongated blue triangles.

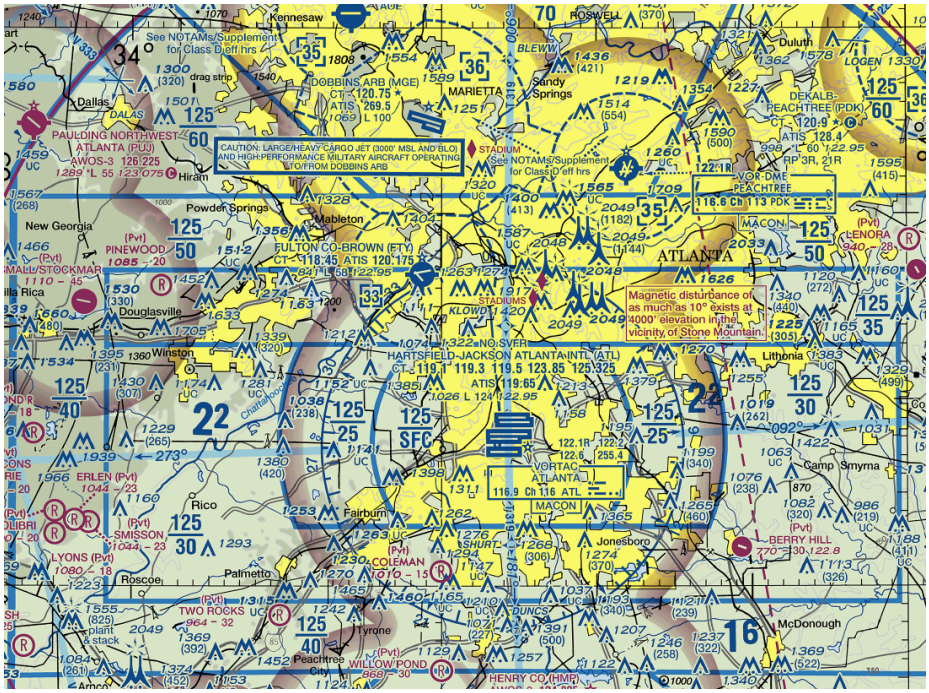


Figure 2.7: Section of Atlanta VFR chart, taken from the FAA.

Some studies do not consider static obstacles in their analysis [51, 34, 84, 32]. In [28], three rectangular no fly zones are defined over the map, but the impact of static obstacles on performance is not evaluated. In [33], the obstacles are considered implicitly since routes or sky highways are defined around buildings. However, the analysis is only performed on a section of road so there is no direct information on how changes in building density might affect performance. In [85], an analysis is conducted on the available airspace volume over the San Francisco Bay area using different types of airspace restrictions. This study presents different metrics such as percentage accessible population and percentage available airspace and evaluates the impact of different ATC access rules on these metrics. This allows a comparison between cities and the evaluation of different airspace restrictions but it cannot be used to estimate safety, capacity and efficiency.

The size and types of static obstacles depend on the altitude. EVTOLs should not be strongly affected by man-made constructions at the altitude they operate, but will be more strongly affected by airspace reserved for other NAS users, such as approach paths to airports' runways. SUAs for package delivery on the other hand will encounter more constraints due to buildings and communication towers at low altitude, but they should be able to operate closer to airports since they will be under the approach paths.

2.5.3 Priority Traffic

The traffic management system must work even when perturbed by non-participating traffic. Emergency flights or non-conforming flights might impact planned trajectories.

In the NASA UTM ConOps, operators are responsible to clear the area when an emergency vehicle requires access. A number of studies do not include an evaluation of the impact of non-participating traffic on their proposed ConOps.

In [51] an emergency helicopter perturb a package delivery scenario in which agents are controlled by a centralized authority, all agents successfully avoid the conflict by performing vertical avoidance maneuvers. In this study the helicopter fly straight from its base to

an accident site in the middle of the simulation. In [32], rogue aircraft that fly haphazardly through the airspace are added to the simulation and the effect on different UTM architectures that use different airspace structures is evaluated. Contrary to [51], in this study agents are shown to suffer from intrusions due to the rogue aircraft with the tubes structure being the most affected. The discrepancy between the two studies comes from different assumptions on the behavior and number of the non-participating traffic, the differences in access control method and collision avoidance algorithms.

Emergency helicopters operate at a similar altitude as eVTOLs so it is likely that there will be interactions between air taxis and emergency helicopters. Below 400 feet, interactions with emergency vehicles are more unlikely. Some ConOps assume that this airspace could be reserved to participating UAV traffic [23].

2.5.4 Research Area 2

Observation

Operating at low-altitude in urban environments presents challenges due to static obstacles, demand patterns and priority traffic. These external factors depend on the application (air taxi/drone delivery) and are not well defined yet.

As can be seen from the subsections above, different studies have made different assumptions and used different evaluations methods. Not all studies result in quantified measurements and when they do they do not all use the same metrics. Although, these studies are enough to get an idea of trends caused by different factors, such as non-uniform demand patterns increase the density locally which results in decreased efficiency, it is difficult to estimate precisely how these factors impact individual UTM alternatives exactly. Since some studies neglect some of these factors in their evaluation of alternatives it is reasonable to wonder whether they are justified in doing so.

Gap

The impact of external factors on the overall performance of each architecture has not been systematically evaluated. There is no information on whether they should be included at the conceptual design stage.

If external factors could be neglected, the analysis would be cheaper to perform and the results of the conceptual design stage would be valid for a range of use cases. If on the other hand these factors strongly impact the decision that is made at the conceptual design stage, then they should not be ignored.

From there the second research area and high-level research question arises naturally:

Research Question 2

Which external factors should be included when evaluating architectures at a conceptual design stage?

2.6 Evaluate

From the two sections above, it can be seen that UTM ConOps that have been proposed in the literature have been evaluated under very different assumptions and using very different methods. The lack of a common framework is an issue as it prevents comparing different architectures in a meaningful way.

As shown in Table 2.2, a number of researchers have developed their own custom simulation and modeling system for the purpose of their research, but they do not make it available to the rest of the community. Different types of analysis have been conducted. Most detailed studies rely on a discrete-time agent-based simulation approach. In discrete-time agent-based analyses, the simulation marches through time using a fixed time step, computes the behavior of all the agents at each step, and propagates the behavior until the next time step. Agent-based simulations allow to capture the interactions between agents and to model collision avoidance algorithms. Some other studies have used a more high-level

approach, and have introduced different metrics to measure airspace complexity, available airspace volume or conflict count [84, 83, 85]. These approaches cannot be used to estimate quantitatively the metrics that were presented earlier in this chapter.

Observation

UTM architectures are evaluated using a wide range of tools under different assumptions and the impact and interactions of the proposed subsystems have not been measured.

This is because of this observation that the three gaps on the lack of information on the impact of different subsystems and external factors on different ConOps are really gaps that cannot be easily bridged.

A brief overview is given here of agent-based simulations that have been used to study UTM ConOps and for which information is available.

In the Metropolis study, the Traffic Manager (TMX) simulator is used. It was originally developed by NASA Langley and the National Aerospace Laboratory of the Netherlands (NLR) to study free flight concepts [86]. However, it is not available to the general public. Professor Hoekstra at TU Delft, who worked on TMX, later developed an open-source Python-based simulator named Bluesky [71]. The code is readily available on GitHub. The project is still under active development, and not quite mature yet. Several functionalities were missing from the master branch, such as recording of collisions, when the thesis was started.

NASA Ames developed a simulator called FACET (Future Air Traffic Management Concepts Evaluation Tool) in the early 2000s to study advanced air traffic management concepts [70]. The simulator was implemented in Java, and is available through an academic license. However, it is geared towards commercial aviation, its documentation is somewhat lacking, there is no direct access to the source code, and it is no longer under active development.

NASA is developing a new simulator for unmanned traffic management called FE3,

however it has not been publicly released yet [87] [1].

Private companies are also developing simulators that could be interesting for this problem. Airtop and AgentFly were contacted. A NDA was signed with AgentFly and the possibilities to use it for the research were investigated. However, it proved to be too much of a blackbox to allow for the experiments that will be presented in the next chapter.

2.7 Conclusion

In this chapter, a structured top-down approach was proposed to address the research objective, and the different steps of the approach were studied using the literature. For the two last steps, *Performance Criteria* and *Make Decision*, metrics and methods can be taken directly from the literature. For the three other steps, gaps were identified that prevent them to be fully defined without more work. Two research areas were identified: which subsystems should be included in a UTM architecture, and which external factors should be considered when evaluating the architectures? To answer these high level research questions a flexible evaluation method will have to be developed.

The top-down approach, observations and gaps are summarized on Figure 2.8. The next chapter goes into more details into each of these research areas, introduces more detailed research questions and hypotheses, and presents experiments designed to validate or reject the hypotheses.

[1] An author of the paper just cited was contacted in February 2019, and he stated that NASA is currently working to release a Software User Agreement for academics.

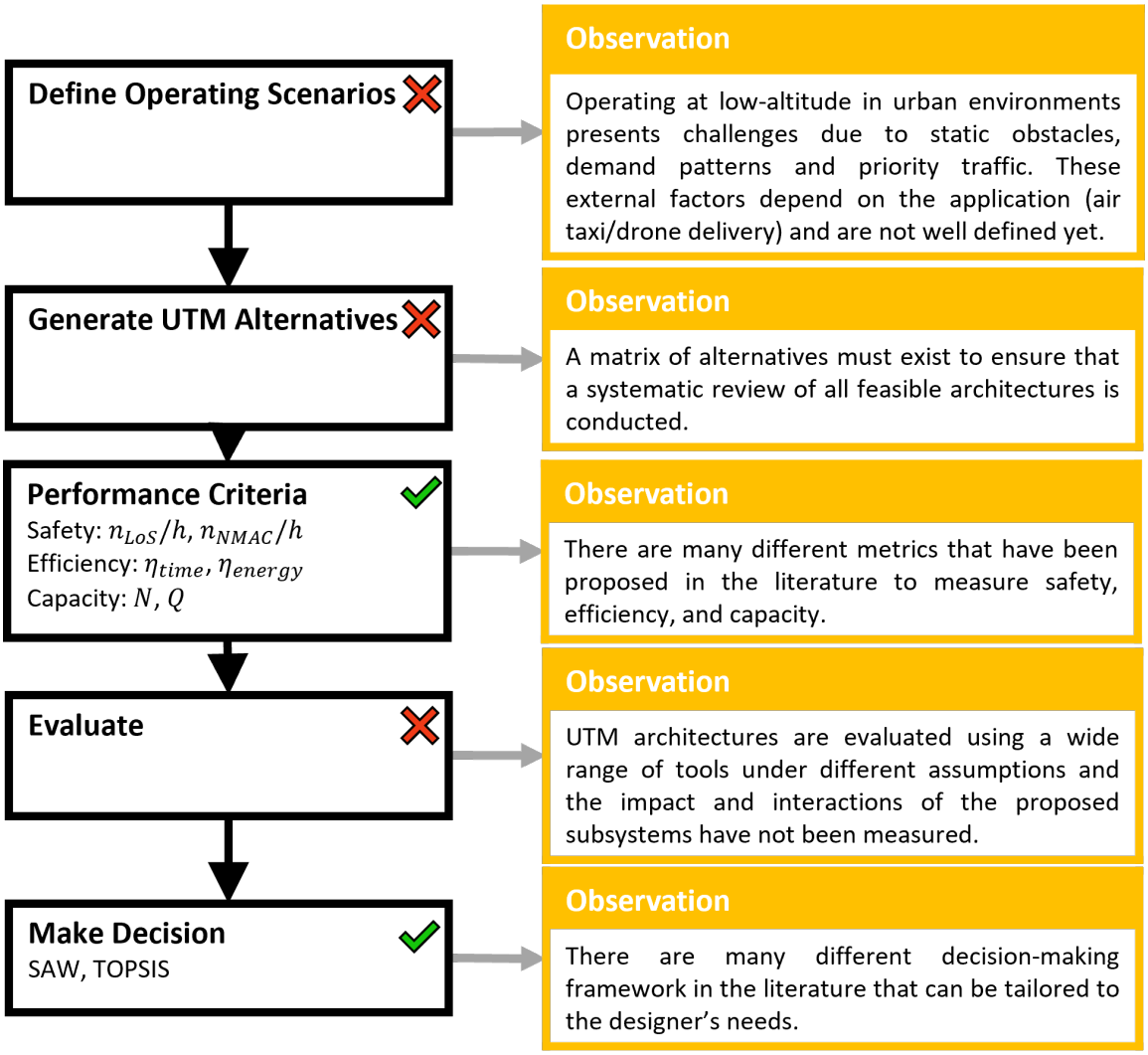


Figure 2.8: Summary of the Literature and Approach section

CHAPTER 3
RESEARCH FRAMING

In the first chapter, the need for a new low-altitude unmanned traffic system that could accommodate new types of airspace users, such as eVTOLs for air taxi or small UAVS for package delivery, was outlined. A preliminary literature review showed that many different alternatives had been proposed but there was a lack of a common framework to evaluate them. This motivated the research objective:

Research Objective

Develop a framework that enables the generation, evaluation, and comparison of UTM architectures in a systematic way under flexible assumptions.

In the second chapter, it was observed that a systematic top-down decision making approach should be followed when designing a complex system. A review of the literature revealed that some steps of the process such as performance criteria and decision-making had been extensively covered but that others had gaps.

Two main research areas were identified in the previous chapter. The first section of this chapter focuses on the gap affecting the ”generate UTM alternatives” step of the top-down conceptual design approach. The second section presents the second research area which addresses the gap related to the definition of operating scenarios. In each of these section, research questions are formulated to address the gaps that were identified, and hypotheses that answer the research questions are formulated based on the literature and intuition. Experiments are designed to provide the data required to confirm or reject the hypotheses. The final section returns to the overarching question that guides the thesis and formulates the overarching hypothesis based on the hypotheses presented in the previous sections. An experiment is designed to show the value of the proposed approach and validate the hypothesis.

3.1 Research Area 1: System Decomposition

3.1.1 Research Questions and Hypotheses

In the previous chapter, a gap was identified. No system decomposition exists for UTM architectures, preventing a systematic exploration of the design space. Based on work done for manned commercial aviation a system decomposition was proposed. This proved useful to analyze the literature and the different ConOps in an organized manner. However, there is no information on whether the proposed subsystems should indeed be considered early in the design or if their impact is sufficiently small to be neglected until further down in the design process. This prompted the first high-level research question:

Research Question 1

Which subsystems should be included when evaluating architectures at a conceptual design stage?

The subsystems that were identified in the previous chapter significantly change the rules and behaviors of the agents in the UTM system. As a result, it is reasonable to believe that airspace structure, access control, preflight planning and collision avoidance should all be evaluated at the conceptual design stage. Since the conceptual design stage ends by a decision-making step, in order to validate that the subsystems should indeed be included, changes in rankings obtained at the end of the conceptual design stage are a valid method to measure the impact of the inclusion or not of some subsystem. As a result the following high-level hypothesis is formulated:

Hypothesis

If neglecting or making simplifying assumptions for airspace structure, access control, preflight planning and collision avoidance leads to change in the preference ordering of UTM architectures, then these subsystems should all be considered at the conceptual design stage.

Although there have been studies of different algorithms and comparisons between them, there have not been studies on the interaction between the different elements that consitute a UTM architecture. When evaluating an algorithm for collision avoidance for instance, the structure is usually kept fixed. On the other hand, in the Metropolis study the structures and rules of access were changed but the algorithms used were fixed.

This can be summarized by the following gap:

Gap

The impact of the autonomy/behavior at the agent level on the airspace performance overall has not been evaluated.

Due to this gap, the hypothesis that the preflight planning and collision avoidance subsystems should indeed be included at an early stage of the design cannot be proven. This prompts the following Research Question:

Research Question 1.2

Should preflight planning and collision avoidance be explicitly considered when comparing architectures with different access control alternatives?

The reason why this Research Question is numbered 1.2 rather than 1.1 will be explained shortly. Preflight planning and collision avoidance algorithms will have an impact on the performance of the overall airspace. Including them in the evaluation requires more work than if they could be just considered fixed or approximated. There would be a strong argument to include them at an early stage if choices of other subsystems depended on the preflight planning and collision avoidance subsystems. If a 4DT contract architectures is always preferable to a free-access architecture, no matter what method is used then it would not be necessary to model them in details. On the other hand, if the choice of autonomy impacts how well an access control method performs compared to another, then it should be included early on.

Hypothesis 1.2

If the ranking of architectures that use different access control alternatives depends on the choice of algorithms for preflight planning and collision avoidance, then autonomy algorithms should be explicitly part of the decomposition.

However, when a survey of preflight planning methods for UTM 4DT contract architecture was conducted in the previous chapter we remarked that they had a lot of shortcomings. A lot of ConOps relied on manual planning by the UAV operator. In the Metropolis study, the first available path is used which combined with the constraining route airspace structure results in high density and low throughput. In [28], the path is planned in static and agents must wait until their desired path clears up before taking-off. In the study presented in [53], there are more degree of freedom but no guarantee that the planner will converge to a safe solution and the algorithm used to resolve the conflict is not really adapted to dynamic collision avoidance. Other studies optimize the time along a trajectory but the path is constrained to be the shortest path. Other only let agent change their altitude to avoid conflicts, the path in the latitude-longitude plane is fixed.

Observation

4DT-contract planning algorithms proposed or evaluated in the UTM literature have shortcomings.

This is an issue because if 4DT planning algorithms unduly constrain the path or make too many assumptions and simplifications this might make them appear as poor alternatives when compared to free-access alternatives.

This must be addressed before trying to answer Research Question 1.2. The first research question that should be answered is then:

Research Question 1.1

Which options should be added to the matrix of alternatives for the preflight planning subsystem?

Looking beyond the UTM literature, the problem of planning trajectories in an environment with known moving obstacles has been addressed in part in the robotics literature. Planning a path in a 4D known-environment is very similar to planning a path in a static environment if time is considered as just an additional dimension, with an added constraint that motion can only go in one direction along that dimension. However, adding one dimension to some common path planning algorithms such as A* comes at a computational cost. The Safe Interval Path Planning algorithm was developed to alleviate some of that cost. Many path planning algorithms are evaluated in situations where agents motion are heavily constrained such as mazes. In the UTM environment even if there are many agents in the air at a given time the solution space is not as heavily constrained. This is why a local approach like the one proposed Altiscope's technical report [53] has some merit. It is likely that there are many solutions to the planning problem, many of them quite similar, and that solutions that do not veer too far from the ideal path would be close to optimal. Since the main issue with the approach proposed by Altiscope was the algorithm used for computing collision avoidance, this should be a focus when developing an alternative algorithm. In looking at collision avoidance methods in the previous chapter, VO were mentioned as a commonly used representation of the problem of avoiding moving agents. An approach that combines the idea of using local collision avoidance for 4D trajectory planning but makes it more robust appears as a promising idea.

In subsubsection 2.4.4 the concept of Velocity Obstacles was presented. VO methods define sets of velocities for the ownship that would eventually result in a collision if the intruder were to maintain its current velocity. Collision avoidance methods based on VO then aim at finding a velocity outside of the union of all VOs. Methods to find a velocity outside of these collision velocity sets often use a heuristic. Typically methods based on VO either use sampling or computing the intersection of the sets to find feasible velocities [88]. In the ORCA algorithm that was presented in the previous chapter, VOs are simplified to be expressed as a single constraint, allowing the problem to be solved using a linear

programming algorithm with a quadratic constraint on the norm [75].

In our approach finding a velocity outside the union of the VO set was formalized as a mixed-integer quadratically constrained program (MIQCP) problem. The ownship velocity is expressed as $V = [v_x, v_y]$.

The VO set can be defined as all v_x, v_y such that $s_{i,1}(v_x, v_y) \geq 0$ **and** $s_{i,2}(v_x, v_y) \geq 0$, where $s_{i,1}(x, y) = 0$ and $s_{i,2}(x, y) = 0$ are the equations of the lines forming the VO cone generated by intruder i, as illustrated in Figure 3.1.

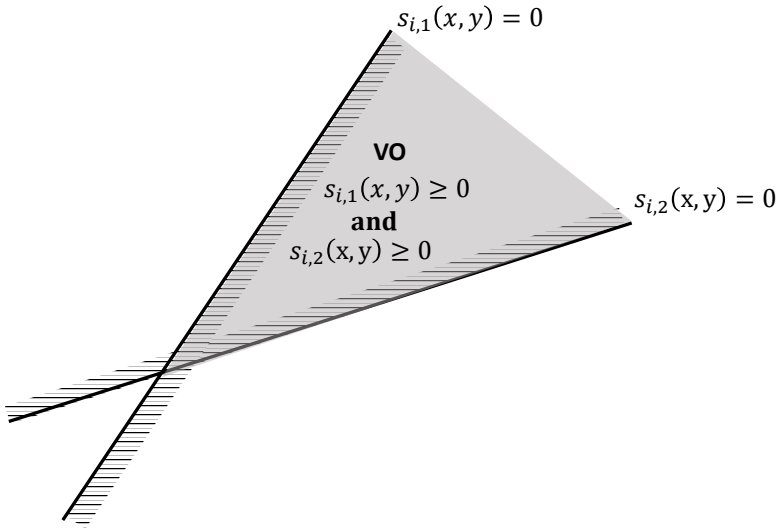


Figure 3.1: Expressing the Velocity Obstacle as two linear constraints

In order to get velocities outside the VO, the complement of that set must be obtained. The constraints from an intruder i can be expressed as:

$$s_{i,1}(v_x, v_y) \leq 0$$

$$OR$$

$$s_{i,2}(v_x, v_y) \leq 0$$

To represent the OR constraint in a linear framework, integer variables are introduced sim-

ilarly to what is done in [82].

$$s_{i,1}(v_x, v_y) - K * a_{i,1} \leq 0$$
$$s_{i,2}(v_x, v_y) - K * a_{i,2} \leq 0$$
$$a_{i,1} + a_{i,2} \leq 1$$

Where $a_{i,1}$ and $a_{i,2}$ are binary variables. K can be chosen to be arbitrarily large. However selecting a value too large for K can create issues for the solver, making some elements poorly conditioned and creating issues with relative tolerance values. Hence for a practical implementation the value of K is selected based on the maximum value of $s_{i,1}$ and $s_{i,2}$ for v_x and v_y in the range $[-v_{max}, v_{max}]$. A final quadratic constraint comes from defining a maximum velocity for the agent. This is the only constraint that is quadratic.

$$v_x^2 + v_y^2 \leq v_{max}^2$$

Finally, the objective of the optimization problem is to minimize the difference between the chosen velocity and the desired velocity. The desired velocity is simply a vector pointing towards the goal that has an intensity of v_{max}, this is the velocity that would be optimal if there were no intruders.

$$\underset{v_x, v_y}{\text{minimize}} (v_{desired,x} - v_x)^2 + (v_{desired,y} - v_y)^2$$

This optimization problem is solved using the Gurobi software python interface [89].

Algorithm 1 illustrates how the velocity obstacle solver is used in a centralized strategic method. First, the algorithm finds the first suitable take-off time that occurs after the agent's desired time of departure. This is performed by checking that there is no agent within the minimum separation of departure at the start time and if there is adding ten seconds to the start time and continuing to check until a solution is found. Once a suitable start time has

been found, the agent's trajectory is constructed by solving a MIQCP problem every ten seconds. If at some point the solver cannot find a solution then it means that there is no way to avoid the conflict given the state of the trajectory. The trajectory is reset and the start time used to generate this trajectory is incremented. This gets repeated until a valid trajectory is found. A valid trajectory will always exist due to the first come first serve priority scheme: if the agents wait long enough there will be no planned aircraft in the air.

Algorithm 1 Local VO

```
while NOT FoundSafeTrajectory do
    startTime = manager.FirstAvailableTakeOffTime(startTime, startPosition)
    time=startTime, position=startPosition, trajectory=[(time,position)]
    while position ≠ goal AND NOT noSolution do
        intruders = manager.getPlannedIntruders(position, time)
        desiredVelocity = getOptimalVelocity(position, goal, maxSpeed)
        model = setupMIQCP(intruders, position, desiredVelocity)
        model.solve()
        if model.hasSolution() then
            velocity = model.solution
            position += velocity * dt
            time += dt
            trajectory.append((time,position))
        else
            noSolution = True
    if noSolution then
        startTime +=dt
    else
        FoundSafeTrajectory = True
return trajectory
```

We formulate the following hypothesis:

Hypothesis 1.1

If the SIPP and a locally optimized algorithms perform better in terms of capacity and efficiency than the decoupled approach, then they are viable alternatives for the preflight planning subsystem and will open the design space for 4DT-contract architectures.

If indeed these two algorithms perform as hypothesized, then they can be used as op-

tions when evaluating 4DT access control alternatives as required by Hypothesis 1.2.

Once it has been shown that preflight planning, collision avoidance and access control have to be evaluated together, then the last subsystem, airspace structure, should be considered. Airspace structures such as layers have been evaluated in some studies and shown to have a positive impact on airspace performance [31]. However in those studies the access control or autonomy algorithms are fixed, hence there is no information on how different architectures might react to layers. This can be summarized by the following gap:

Gap

The impact of airspace structures has not been evaluated across a range of architectures.

Due to this lack of information the hypothesis that airspace architecture should be explicitly included at an early stage of the design cannot be confirmed. As will be detailed below there is also a lack of information on whether airspace architecture needs to be modeled within the agent-based simulation or whether the simplified approach proposed in the literature capture its effects with enough accuracy. Hence the following research question is formulated:

Research Question 1.3

Should airspace structures be explicitly considered when comparing architectures and how should they be modeled?

If all architectures were impacted similarly by airspace structures then the evaluation and choice of subsystems could be decoupled. First, the three other subsystems could be evaluated and the best partial architecture chosen according to the designer preferences. Then the effect of airspace structure could be evaluated and one type of airspace structure selected. This would lead to a complete architecture selection. However, if there is an interaction between airspace structure and the other subsystems then they should be evaluated concurrently to capture that interaction. This does prompt the following question: is it really necessary to capture the interaction? If the interaction between subsystems is small

it might not impact the rankings of the partial architectures (access control, preflight planning and collision avoidance) and could be ignored at the conceptual design stage. On the other hand if the interaction is significant then it might change the ordering of the partial architectures and it should not be ignored as it could change the outcome of the conceptual design stage.

There have also been studies that looked at the impact of airspace structures on conflict count in an airspace where all vehicles are considered to travel straight to their goal [83]. Conflict count has been used in different studies as a proxy for airspace complexity [84, 44, 90]. Indeed, the more conflicts there are the more a vehicle will have to maneuver to resolve them. This measure is an approximation since once vehicles maneuver they might create or remove conflicts that were predicted when all agent flew straight (this is measured by the domino effect parameter [32]) but it can be computed very quickly since it does not require to model agents' behaviors. If this measure of complexity could be used to predict performance of different architectures it would be useful at the conceptual design stage. This prompts the following question: can a conflict count-based model accurately capture the performance of different architectures when the airspace structure is changed? Once again, the significance of the error introduced by the model can be evaluated by looking at the outcome of the conceptual design phase.

As a result, the following hypothesis is formulated:

Hypothesis 1.3

If the ranking of architectures can change due to difference in performance when the airspace structure is included or when a simplified model is used, then it should be explicitly part of the decomposition and modeled at an early stage.

Figure 3.2 summarizes the research questions and hypotheses of the first research area.

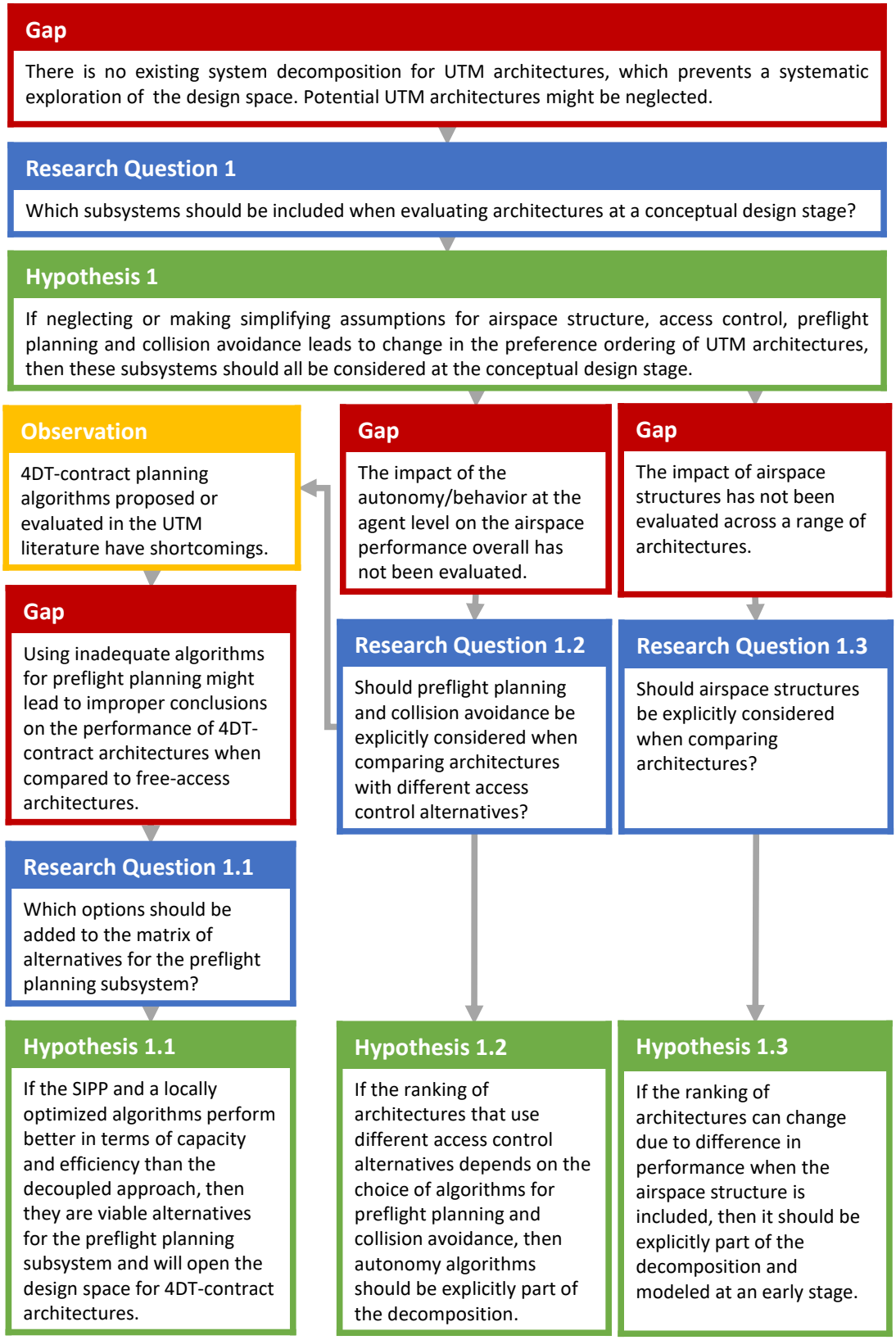


Figure 3.2: Summary of the first research area

3.1.2 Formulation of Experiments

The three research questions that were presented above arose because there was a lack of available data to address the gaps that had been identified. The role of the experiments is to generate the missing data in order to be able to confirm or reject the hypotheses that were formulated.

The first experiment's role is to evaluate the two methods for 4DT preflight planning that have been proposed against a method commonly proposed in the literature. For this experiment an agent-based simulation with a centralized authority that can store flight plans is required. The algorithms that are proposed are 2D, so a relatively simple simulation can be used. The details of the implementation of the different 4DT planning algorithms are given in the next chapter along with information on the simulation. The experiment will generate performance data (throughput, time efficiency, and energy efficiency) and the planning time will be recorded. This will allow to compare algorithm performance and verify that the algorithms are feasible in terms of computation power when several hundred of agents are in the simulation at the same time.

It will be interesting to see how the algorithms scale with the number of agents. The differences between methods can be expected to increase as density increases as the planning problem becomes more complex. At a low number of agents, a decoupled method that heavily constrain the path should perform correctly. As the number of agents increase so does the number of conflicts, which should mechanically decrease performance.

Experiment 1.1

Factors: Preflight planning algorithms (SIPP, Local VO, Decoupled), number of agents

Metrics: Q, η_{time}, η_{energy}, runtime.

Results: Comparison of performance of the two proposed algorithms against the approach from the literature.

The second experiment's goal is to evaluate whether the algorithms should indeed be included early in the design. If hypothesis 1.1 is correct, the previous experiment will have shown that the proposed algorithms are appropriate for 4DT planning and perform better than a method taken from the literature, providing a range of potential performance for 4DT-contract architectures. The question is then whether this range overlaps with the range provided by free-access architectures. This experiment will evaluate two free-access architectures. The first one will use MVP as the collision avoidance subsystem. MVP is a commonly used algorithm that has been used by other simulators and is simple to implement. MVP does not rely on agents cooperation when computing the next velocity vector for the ownship. The second architecture will use ORCA. ORCA relies on VO concepts and the cooperation of other agents in the simulation. It has been used in the robotics literature for large number of agents. Although the implementation is more complex than MVP, it is available open-source as part of the RVO2 library. The different collision avoidance algorithms should result in different performance, with ORCA providing better safety performance than MVP. Similarly to what has been done in the previous experiment, the evaluation will be conducted in 2D. The simulation parameters will be the same as the one that were used in the previous experiment to allow for comparisons. This will allow to rank architectures for a high number of agents in the simulation. As in the previous experiment the number of agents in the simulation will be varied to better understand how different architectures scale.

Experiment 1.2

Factors: Collision avoidance algorithms (MVP, ORCA), number of agents

Metrics: Q, η_{time}, η_{energy}, n_{los}/h, n_{NMAC}/h.

Results: Rank free-access and 4DT-contract architectures using TOPSIS methodology.

The third experiment aims at evaluating whether the last proposed subsystem, airspace structure, should indeed be included as part of the evaluation, and how it should be eval-

uated. One type of airspace structure will be modeled: layers. Layers were selected because they are simple, used in aviation today, and commonly proposed as part of UTM ConOps. Rules for assigning traffic to layers can easily be customized (heading range between $[0°, x°]$, inbound traffic, ...), hence layers give a lot of flexibility to the designer and are a good first step to model airspace structures. Moreover layers can be modeled through a 2D or 2.5D simulator, limiting the complexity of the required analysis. The agent-based simulation is modified for this experiment to allow for the modeling of layers. In addition to analyzing all the airspace architectures that have been considered in the previous experiments (free-access MVP and ORCA, 4DT-contract decoupled, SIPP and local VO) with a layered airspace structure, this experiment will also count the number of conflict that would occur if agents flew straight to their destination with and without airspace structure. Two models will be built and evaluated as part of the experiment, the first one will assume the impact of layers is uniform across all alternatives, while the second will be based on conflict count. The error introduced by each model will be evaluated and the impact of the error on the preference ordering of architectures will be evaluated.

This experiment will also vary the layer definition to better understand how reducing the range of heading impact performance. The study will be conducted over a range in the number of agents to get insights on how the systems scale and allow comparison with the results from the previous experiment.

Experiment 1.3

Factors: Airspace Structure (layers), Access Control, Preflight Planning, Collision Avoidance, number of agents

Metrics: Number of conflict, Q, η_{time}, η_{energy}, n_{los}/h, n_{NMAC}/h.

Models: Impact of airspace structure as uniform over all architectures, impact of airspace structure as a function of the number of conflicts

Results: Rank architectures: 1) when neglecting airspace structure, 2) when modeling airspace structure as uniform, 3) when modeling airspace structure using conflict count, and 4) when modeling airspace structure in the agent-based simulation.

These experiments will also provide general insights on UTM systems and different architectures. This additional knowledge can be helpful at an early stage of the design to better understand capabilities and limitations of the different system.

3.2 Research Area 2: External Factors

3.2.1 Research Questions and Hypotheses

In the previous chapter external factors that are dependent on the application and not directly under the designer's control were mentioned. Different studies made different assumptions on the demand, presence of static obstacles, and priority traffic. However, information on the impact of these factors on the performance of individual architectures could not be found in the literature. This prompts the second high-level research question, which guides the second research area:

Research Question 2

Which external factors should be explicitly considered when designing a UTM architecture?

In the previous chapter, three candidates external factor were mentioned: demand, static obstacles and priority traffic. Other factors could be added to the list, but for the purpose

of this thesis only these three will be considered.

As mentioned in the previous chapter in the section on decision making, if a factor that affect all alternatives can be modeled as k-factors, i.e. all alternatives have the same sensitivity to that factor, then the normalized decision matrix would remain identical whether or not that factor is included in the analysis. Hence decisions made using the SAW or TOPSIS methodologies would not change. As a result there would be no reason to include them in the analysis at this stage as the addition of more factors adds complexity to the analysis. Factors should only be included if they impact the outcome of the conceptual design stage. If factors cannot be modeled as a uniform impact then, unless architectures dominate each other in a Pareto sense which would make the ranking insensitive to weights, it is likely that there exists a set of designer preference such that the ranking of alternatives changes based on the inclusion of the factor.

The following high-level hypothesis is formulated to answer the research question:

Hypothesis 2

If different architectures have different sensitivities to demand, static obstacles, and priority traffic, then this will cause the preference ordering of architectures to change when these external factors are included.

The impact of these three external factor is then the subject of their own research question and hypothesis.

Demand is commonly included as part of UTM analyses, however the results are often presented with the demand being fixed or the results are averaged over different demand patterns. Hence, there is no information on how demand might affect different architectures. This prompts the first research question:

Research Question 2.1

Should demand be included at the conceptual design stage?

Due to the nature of the UTM system with agents in different architectures following different logic, it can be expected that there will be interactions between the demand and

agents' behaviors. Demand should have a non-uniform effect on the architectures and it is likely that there is a set of designer preferences such that the rankings when demand is included is different from the ranking when demand is not considered. The hypothesis is then:

Hypothesis 2.1

If the ranking of architectures can change due to difference in performance when demand is included, then it should be explicitly modeled at an early stage.

If the hypothesis is proved correct then demand should be included when evaluating architectures at the conceptual design stage.

The next external factor that was considered is static obstacles. As explained in the previous chapter the airspace above an urban area is busy. Airports' protected airspace usually extend over part of the city even when placed at the periphery. At the low altitude considered for package delivery, constructions will represent significant obstacles as well. These static obstacles will impact UTM in different manners. First they reduce the available volume to operate. Second they constrain where landing zones (LZ) can be placed. This in turn will impact the demand as origin and destination pairs will no longer be placed uniformly over the simulation area. Previous studies have usually not considered static obstacles effect in detailed analyses. As a result, we formulate the following gap:

Gap

Static obstacles constrain demand by limiting the locations where LZ can be placed. However, static obstacles are not usually included as part of UTM evaluation so there is no information on whether modeling them beyond their impact on demand will affect different architectures differently, potentially leading to a different choice.

Modeling static obstacles will increase the complexity of the autonomy algorithms used at this stage as they will have to handle non-cooperating obstacles and static obstacles can create local minima that can trap agents. If the effect of static obstacles could be modeled as just their impact on demand, and assuming hypothesis 2.1 has been proven correct, then

the analysis would be simplified.

Research Question 2.2

Should static obstacles be included at the conceptual design stage beyond their effect on demand?

Different algorithms should result in different behaviors, which should in turn cause varied impact on airspace performance. As a result, the following hypothesis is formulated:

Hypothesis 2.2

If the ranking of architectures can change due to difference in performance when static obstacles are included compared to when only the impact on demand is modeled, then static obstacles should be explicitly modeled at an early stage.

Finally, the last external factor that should be considered is priority traffic. In NASA's UTM ConOps, a provision is made that if an emergency vehicle requires to move through the airspace that has been reserved by a UTM participant, the operator should land to avoid interfering with priority operations. Over an urban airspace, Helicopter Air Ambulance flights or law enforcement operations are common and should be considered when designing the UTM system. However, the number of emergency vehicles in the air over one city at a given time is relatively low.

Research Question 2.3

Should priority traffic be included at the conceptual design stage?

Even if the number of priority agents is low, at high density adapting the 4DT plan should be challenging.

Hypothesis 2.3

If the ranking of architectures can change due to the change in performance when priority traffic is included, then priority traffic should be included at an early stage.

Figure 3.3 summarizes the research questions and hypotheses of the second research area.

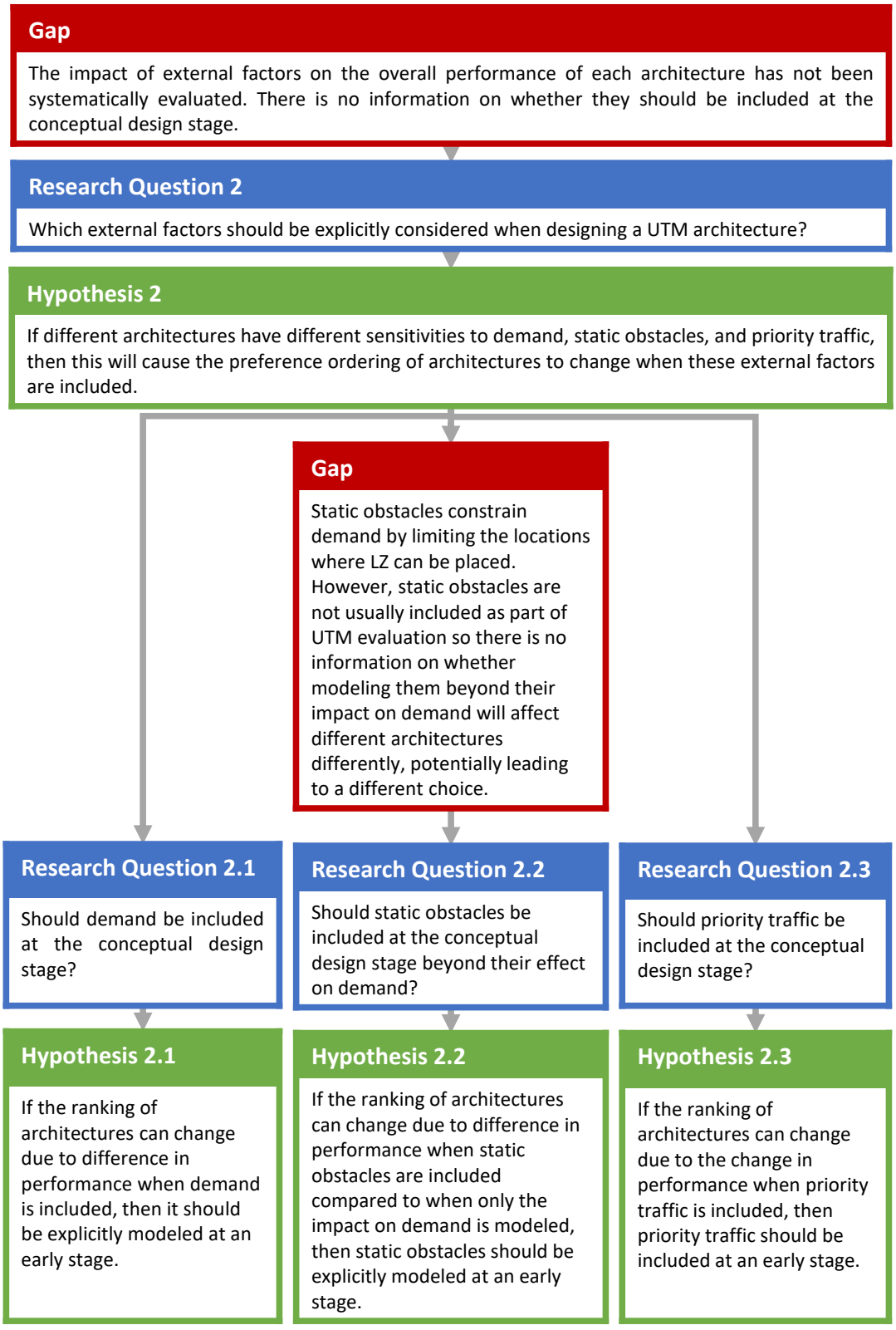


Figure 3.3: Summary of the second research area

3.2.2 Formulation of Experiments

One experiment per hypothesis is designed to provide the missing data and allow to draw conclusions. As stated at the beginning of the previous subsection, if an external factor can be modeled as a k-factor identical across architectures then the decision matrix would not change. Hence there would be no point in modeling it at the conceptual design stage. In all three experiments of this research area, a k-factor model is fitted to the results. The fit error is measured using the R^2 value. If the error is negligible, then the hypothesis would be rejected and the external factor would not need to be considered. If the error is shown to be significant, potential designer preferences are explored to find whether there exists a set of weights such that the inclusion of the external factor can result in a change in the ranking of architectures (see Appendix E on how this problem can be formulated as a Linear Programming problem and solved systematically). If such weights exist the hypothesis is confirmed.

The first experiment requires to model demand and measure performance for different architectures. To limit the number of cases to run in the simulation, only architectures with a free airspace structure will be evaluated. As will be presented in the results chapters, these architectures prove sufficient to validate the hypothesis. Two different types of demand will be considered in the experiment: the first will be based on population density, similarly to what is done in some studies from the literature, while the second will be based on a hub-and-spoke pattern. The first case corresponds to an Air Taxi scenario with higher demand in dense areas. The city of Atlanta will be used as an inspiration. The second case corresponds more to a drone delivery scenario with a limited number of warehouses serving customers around them. More details on the implementation of demand in the simulator can be found in the next chapter. A k-factor model will be built based on the performance of the free-access MVP architecture since it is the most commonly evaluated in the literature. Architectures will be ranked in the case where the demand is uniformly distributed over the area and this will be compared to the ranking obtained when considering population

density. This will allow to confirm the hypothesis.

Experiment 2.1

Factors: Demand, UTM Architectures, number of agents

Metrics: Q, η_{time}, η_{energy}, n_{los}/h, n_{NMAC}/h.

Results: Rank architectures with and without demand using the SAW methodology.

In the second experiment we wish to identify how much of the impact of static obstacles is due to the constraints they cause on demand and how much is not captured when only the demand is considered. As will be seen in the results chapter, the previous experiment proves that demand has a strong impact on architecture performance and that this impact is non-uniform across architectures. Since static obstacles impact demand, this must be accounted for when evaluating static obstacles otherwise the answer would be the same. Two different static obstacles maps are defined based on the city of Atlanta. The first one corresponds to a low-altitude where there are buildings and the restrictions due to airports are smaller. The second one corresponds to a higher altitude where there are few buildings but airports are a larger constraint. Additional details on how the maps were defined, how algorithms were modified to handle static obstacles, and some of the challenges encountered can be found in the next chapter. For each of these maps two simulation are run: one when the map is only used when creating origin-destination pair, and one where agents in flight also try to avoid the static obstacles on the map. As in the previous experiment the k-factor model is built based on the free access MVP architecture. The rankings obtained after both simulations can be compared to evaluate whether the impact of static obstacles beyond what can be captured by demand is really important at the conceptual design stage. This will confirm the hypothesis and answer the research question.

Experiment 2.2

Factors: Static Obstacles, Demand, UTM Architectures, number of agents

Metrics: Q, η_{time}, η_{energy}, n_{los}/h, n_{NMAC}/h.

Results: Rank architectures using the SAW methodology 1) with static obstacles 2) with demand such that there are no vehicles taking off or landing in restricted areas.

In the third experiment the impact of priority traffic is evaluated. This experiment requires the development of mitigation techniques for 4DT-contract alternatives. Indeed, if a priority agent perturb their flight the ownship must amend its flight or default to a reactive collision avoidance method. Free-access alternatives can be easily modified to handle non-cooperative traffic. More details on the implementation and logic used by agents is available in the next chapter. The number of priority agents is kept low (1 and 5) as there are rarely many emergency helicopters operating at the same time over a given city.

Experiment 2.3

Factors: Number of priority agents, UTM Architectures, number of agents

Metrics: Q, η_{time}, η_{energy}, n_{los}/h, n_{NMAC}/h.

Results: Rank architectures with and without priority traffic using the SAW methodology.

As in the previous set of experiments, these experiments also allow to gain general knowledge about the system which might be helpful to the designer.

3.3 Verification of the Approach

As a result of the investigation conducted in the two research areas presented above, a proposed approach will have been developed. Whether to include the proposed subsystems and external factors in the evaluation and how to model them at a conceptual design stage will have been determined. The next step is to verify that this proposed approach performs better than what existed in the literature previously.

The overarching research question that motivated the thesis is reminded here:

Overarching Research Question

How should UTM architectures be systematically and quantitatively assessed and compared at a conceptual design stage?

The two previous research areas will have allowed us to formulate a proposed approach. This will be compared to a method representing what existed before, which will be referred to as the baseline approach. There could be different ways of formulating the baseline given that a systematic approach to evaluate UTM architectures has not been proposed before. For the purpose of this thesis, the baseline approach is defined as a method that considers access control and airspace structure subsystems but makes simplifications. First, the baseline approach fixes the algorithm used with different access control options. It uses MVP for free-access architectures and the decoupled algorithm that was implemented for this thesis for 4DT-contract architectures. Second, the baseline approaches uses a conflict count model to estimate how limiting the heading range per layer impacts performance. Finally, it does not consider any external factor when making the evaluation.

The overarching hypothesis can be formulated as follows:

Overarching Hypothesis

If both the proposed and baseline approaches are used to select UTM architectures under the same assumptions and result in different rankings, then the oversights and simplifications introduced by the baseline approach would effectively lead to the selection of a worse architecture and the proposed approach should be used instead.

In the previous experiments, each element was evaluated individually. One element was varied while the others were kept constant and the results were observed. In this experiment, multiple elements are combined to create scenarios that are representative of UTM use cases. As mentioned in the first chapter, UAM consists of both air taxi and drone package delivery applications. The two use cases presented in the introduction are used to

test the proposed approach. Vehicles operating as part of these applications will operate at different altitude and face different demand. Moreover, designers might have different objectives when choosing a UTM architecture for different applications. For instance, in a drone delivery use case, time efficiency might be less of a concern than for air taxi.

Final Experiment

Factors: UTM Architectures, Scenario, Approach

Metrics: Q, η_{time}, η_{energy}, n_{los}/h, n_{NMAC}/h.

Results: Using the SAW methodology, rank architectures with the proposed approach in the two scenarios and rank architectures with the baseline approach.

3.4 Conclusion

This chapter has presented the research framing of the thesis. It has introduced questions that stemmed from the gaps that were identified during the literature review and formulated hypotheses to answer them. Experiments to provide the data missing to confirm or reject these hypotheses have been designed.

The experiments all require a simulation environment. The next chapter provides technical details on how the simulation was implemented, the assumptions that were made and the study limitations. The following three chapters present the results of the experiments.

CHAPTER 4
EXPERIMENT IMPLEMENTATION

This chapter gives technical details on how different elements of the simulation were implemented, the assumptions that were made in the different experiments, and how different parameters were chosen. The first section presents the scope of the experiments. The second section gives an overview of how the agent-based simulation was implemented. The third and fourth sections respectively focus on collision avoidance and preflight planning algorithms implementation. Sections 4 to 7 each present a different aspect that is the focus of an experiment, e.g. layers, demand patterns, static obstacles, and priority traffic.

4.1 Scope

In this section the scope of the analysis that was implemented for the experiments is detailed.

In the previous chapter, multiple metrics to measure capacity, safety, and efficiency were introduced. In the analysis, the following metrics were measured to evaluate the performance of the alternatives: Q the throughput, η_{time} the time efficiency, η_{energy} the energy efficiency, n_{LoS}/h the number of Loss of Separation per agent flight hour, and n_{NMAC}/h the number of Near Mid-Air Collisions per agent flight hour. This set of metrics was selected because it captures the different performance aspects (capacity, efficiency, safety) and allows to have a somewhat nuanced understanding of the potential trade-offs within these aspects. For instance, there might be a trade-off between time and energy efficiency, or a trade-off between the number of LoS and the number of severe LoS. However, the UTM designer could be interested in different measures of performance such as the average ground delay, the percentage of agents having an extension of travel time greater than 15%, etc. In that case, the proposed approach could be adapted for these metrics.

When the experiments evaluate the existence of a set of weights such that the inclusion of a subsystem or external factor can change the ordering of the UTM architecture, they assume that there is no information about the designer's preferences. If additional information is available, the study could be conducted for the range of weights or specific weights of interest to the designer. This might change the conclusion of some of the experiments.

Not all ConOps that were found during the literature review have been implemented in the simulation environment. In the matrix of alternatives only a few options for each subsystem were implemented, this is shown on Table 4.1. This matrix of alternatives is smaller than the one defined in Table 2.7. The options that were implemented proved sufficient to validate the hypotheses. With more time and resources it would be interesting to expand the simulation further by implementing some of the options that were left aside. More details on the implementation of each option presented in this matrix of alternative are provided in the following sections.

Table 4.1: Matrix of Alternatives for UTM system generation and evaluation

Airspace Structure	Free	Layers		
Access Control	Free	4DT Contract		
Preflight Planning	None	SIPP	LocalVO	Decoupled
Collision Avoidance	None	MVP	ORCA	

Throughout the experiments several parameters (simulation size, agents' maximum velocity, and minimum separation distance) were kept constant although they were implemented as variables that could be easily changed. A UTM system is only responsible for coordinating local traffic at a city-level scale. Hence the simulation area was limited to a size of 20 by 20 kilometers, which roughly corresponds to the size of a city like Atlanta. Atlanta was used throughout the experiments as an inspiration to model external factors. As explained in the previous chapter, the minimum separation distance is not really defined for UAM use cases. Different studies have used very different values: 5-10 m [37], 30 m

[54], 45 m [90], 50-300 m [34], 150 m [53], 250m [32], 5 NM [83, 60]. There have been studies in the literature looking at how changes in minimum separation distance might impact throughput [37] or efficiency [34]. In [91], the authors propose a well-clear definition for sUAS of 2000 feet (610 m). Arbitrarily, the minimum separation was set at 500 m. Similarly, for vehicle velocity the assumptions done in studies from the literature vary greatly: 11 m/s [54], 15-20 m/s [37], 10-20 m/s[90], 25 m/s [34]. Arbitrarily, the maximum velocity was set at 20 m/s.

The vehicles' kinematics were ignored for the purpose of the experiments. Agents were assumed to be able to change heading and velocity instantaneously, and to be capable of zero-velocity hovering. The effect of wind was not modeled. Agents are assumed to have perfect communication with a centralized authority when a 4DT-contract architectures is implemented and to be able to adhere to their contracts without deviation unless there are priority agents. In Free-access architectures, agents are assumed to have access to all surrounding agents' velocity and position.

The focus of the experiments is on cruise operations. To keep the simulation simple operations were modeled in 2D, agents' climbs and descents were ignored. The interactions between layers when multiple altitudes are available were also neglected.

This section presented the assumptions and limitations of the analysis that was implemented to perform the experiments. The goal of the experiments is to show interactions between subsystems and external factors, the approach used in this thesis could be replicated with a different more detailed simulation framework. Since what the thesis is trying to demonstrate is the value of the framework rather than the value of a particular solution, the focus is more on the validity of the process than on the actual performance results, which is not impacted by the model accuracy.

4.2 Simulation Overview

A discrete-time agent-based modeling approach was used to implement the analysis of UTM architectures. This section presents some elements of the simulation that are important to understand how the analysis is performed. First, it gives an overview of what happens during a simulation time step. Then, it presents how a steady-state was defined and which agents are counted when measuring performance.

4.2.1 Simulation Steps

A time step of 1 second is used when the experiments involve collision avoidance (Free-access architectures, or 4DT-contract architectures that must react to the presence of priority traffic). The time step is set at 10 seconds otherwise.
At each time step, the simulation performs the same list of actions:

1. It looks through all agents that are on the ground and evaluate whether they can safely take-off. Once an agent is cleared for take-off it is added to the list of agents in the air.

2. It looks through all agents that are in the air and makes them compute their next move.

3. It looks through all agents that are in the air and makes them update their position.

4. It adds agents to the simulation if necessary.

5. It removes agents that are within the tolerance distance of the goal.

6. It performs a collision check on agents currently in the air.

The first action is only necessary if agents have not submitted a 4DT-contract. Agents in free-access architectures must perform this pre-takeoff check to ensure that there is no agents in the air within a separation distance of them. To check if it can take-off, the

agents query agents in the air and agents that have been cleared to take-off that are within a separation distance of their current position. The time to climb is neglected and the agent is directly assumed to be at the appropriate altitude if it is cleared for take-off. For 4DT-contract architectures, agents are assumed to be in the air if the simulation time is greater than the start time of their flight plan and they have not yet been removed from the simulation.

The second action iterates through all agents that are in the air and makes them compute their next move. If the agent is using collision avoidance, it computes its new velocity vector based on other agents position and velocity according to the chosen algorithm (MVP or ORCA). If the agent is following a flight plan, it computes its next position according to its 4DT contract.

The third action iterates through all agents and makes them execute their planned move. The simulation time is incremented by the time step value. The planning and execution stage are separated to ensure that all agents plan with the information available at time t. Once all agents have planned and executed their move, the time is now $t + dt$.

The fourth action consists in adding new agents to the simulation if necessary. The logic of when agents are added to the simulation is explained in more details in the next subsection. In simulations of Free-access architectures, new agents are added to the list of agents on the ground. In simulations of 4DT-contract architectures, each new agent performs a pre-flight planning step.

The fifth action removes agents that are within a given distance, called the tolerance, from their goal. When an agent is removed metrics relative to its flight are recorded, and the agent record is added to the simulation logs. To compute the metrics such as efficiency despite that premature removal from the simulation, the agent is considered to have traveled in a straight line at maximum speed from the point where it has been removed from the simulation to its goal. There are different reasons why a tolerance must be added to the simulation. First, numerical errors can cause the position of the agent to be slightly different

from the goal. Without a tolerance, the agent could hover over its goal without ever being removed. A small tolerance could be sufficient to handle these cases. Second, in reality agents would have to follow approach procedures to land and would start to descend before being over their goal but this was not part of the modeling effort for this thesis. A tolerance of 1km was used in all the experiments. The agent records contains its flight status, ideal time of departure, actual time of departure, ideal time of arrival, and actual time of arrival. These are sufficient to measure the throughput, time efficiency, and energy efficiency.

The sixth action checks whether any loss of separation occur at $t + dt$. Note that agents are removed before the collision check is performed due to edge cases in which the agents reached their goal between time steps, which created a loss of separation even though one of the agent was technically at its destination and should have been removed. Before checking for LoS, a kd-tree is built containing the position of all agents in the air at this time. A kd-tree is a data structure allowing to efficiently query the nearest neighbors or the neighbors within a given radius of a position. The scikit-learn library provided an off-the-shelf implementation of the kd-tree structure [92]. A LoS is identified using the set containing the IDs of the two agents that are involved in the LoS. A LoS between two agents can last multiple time steps, and the simulation ensures that it is only counted once and that the metrics of interest (start time, end time, minimum separation distance) are recorded.

4.2.2 Defining a Steady-State

The simulations were conducted for a constant number of agents in the simulation area. The reason why this was selected, rather than at a constant rate of addition of new agents as is sometimes done in the literature, is because at high densities in some conditions the airspace is not capable of absorbing all the demand and a steady-state cannot be reached. The agents would queue on the ground and the length of the queue would continue to grow if the inflow happened to be greater than the outflow. In such cases, the time efficiency

will keep decreasing with time as the ground delays get increasingly larger. Keeping the number of agents constant allows to get a steady-state value for all metrics in all conditions. Note that not all agents in the simulation are actively flying. The total number of agents in the simulation is the sum of active agents and agents that are waiting on the ground due to ground delays. Figure 4.1 illustrates the different phases of the simulation. The simulation

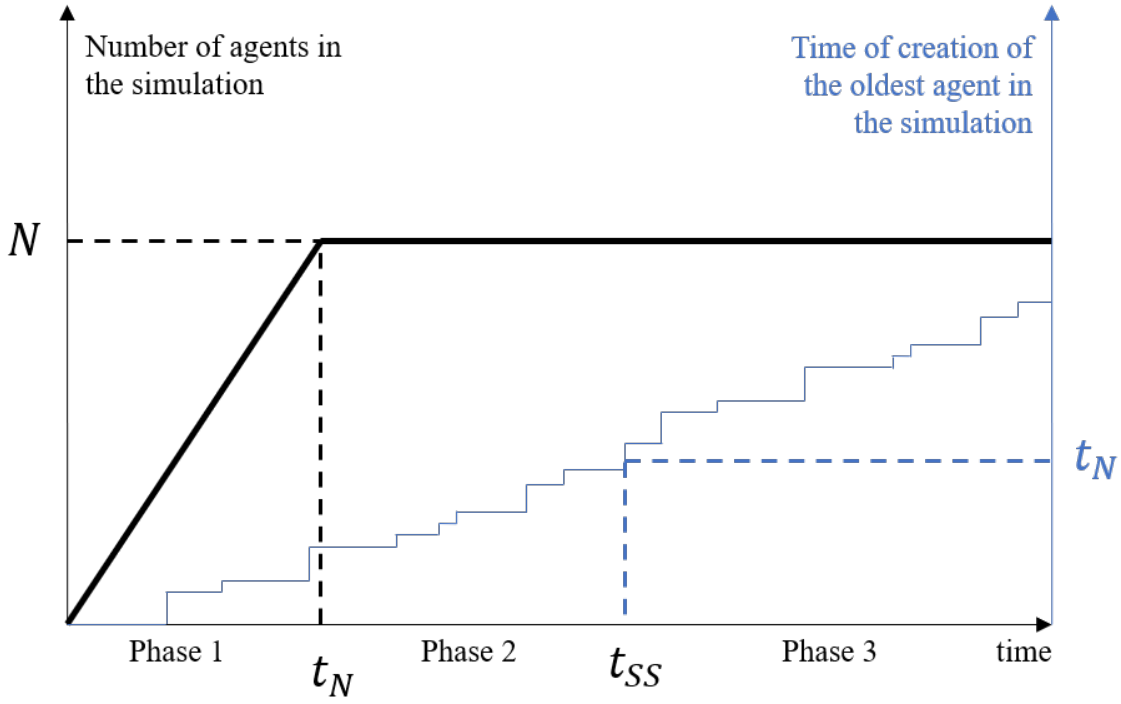


Figure 4.1: Notional sketch of the simulation phases, showing the number of agents in the simulation and the time the oldest agent in the simulation was created.

area starts completely empty. In the first phase agents are added to the simulation environment at a constant rate while also compensating for all agents that exit the simulation. This means that the total number of agents in the simulation increases at a constant rate during this phase.

Once the desired number of agents has been reached at time t_N, the simulation enters a second phase. In the second phase, a new agents is only added when another agent exits, so that the number of agents in the simulation is kept constant.

Once all the agents in the simulation have been created after time t_N, the simulation

reaches phase 3. This means that all agents that are currently present in the simulation have been added when the target density had been reached. This last phase is considered to be a steady state for the simulation, and the time at which it starts is noted t_{ss}.

The simulation ends after 100 agents that were created after t_{ss} have exited the simulation. Agents that exit the simulation after t_{ss} are considered *valid agents* and are used when computing the performance of the airspace. If the metrics were measured only for agents that entered the simulation after t_{ss}, the results would be biased toward faster agents at high densities, e.g. agents with shorter routes and fewer conflicts than the average. The ramp-up approach of the simulation results in many agents that are simulated but not directly exploited in the results. All agents that exit the simulation in phase 1 and phase 2 are ignored. The reason why a slow ramp-up was selected rather than adding all agents at once was to prevent the formation of artificial traffic patterns. Indeed, in some of the experiments the agents are generated on the edges of the simulation area. If all N agents were added at the same time there would be a lot of agents on the edges and none in the center, the density wave would propagate through the simulation area and this pattern might take a long time to subside. The rate at which agents are added in phase 1 was selected arbitrarily based on the average time of travel through the simulation in a uniform cruise case, so that the first agent created would roughly exit the simulation as the target density was reached.

4.2.3 Running the Analysis

The simulation was implemented using Python. The source code is freely available under a MIT license [1]. It can be run with a simple visualization that relies on the python Tk interface tkinter. Tk is an open source graphical user interface (GUI) toolkit. The GUI is optional and the simulation can be run in a headless mode when a visualization is not required. The simulations were set up to be run either on a local machine or on the Georgia Tech PACE cluster, a High Performance Computing (HPC) cluster. The cluster allows to

[1] https://github.com/colineRamee/UTM_simulator.git

run multiple simulations at once, greatly reducing the time required for each experiment.

The simulation creates one log for each run and saves it as a JSON file. One log contains information about the input settings of that run (including the random seed), when the simulation reached the different phases mentioned above, a list of all the losses of separation, and a list of the agents that were created with the characteristics of their run. The analysis of several runs and the computation of the metrics are done in a post-processing step.

4.3 Collision Avoidance

The collision avoidance algorithms that were implemented came from the literature and detailed information was readily available.

4.3.1 MVP

The implementation in this framework is based on the implementation provided in the Bluesky simulator [2]. A 10% safety factor is added to the minimum separation distance to compensate for floating point errors and reduce the number of LoS. Oscillations can occur with head-on conflicts or when a cluster of agents form.

4.3.2 ORCA

ORCA code is available open-source in the RVO2 library [3]. Since that code is written in C++, the code was transcribed to Python for the purpose of this thesis. The minimum separation distance was slightly inflated to avoid floating points error. ORCA requires some user-specified settings. The range, i.e. the maximum distance at which intruders are being considered, was set to 5km similarly to the local VO method. The time horizon was set to be equal to the range divided by the max velocity of the agents which yielded 250 seconds or 4.2 minutes. Following examples of the ORCA method and indications given by the

[2] https://github.com/bluesky/bluesky
[3] http://gamma.cs.unc.edu/RVO2/

authors in the paper and code, the maximum number of intruders that are considered was set to 10.

4.3.3 Baseline: straight flight

Assuming that all agents move in a straight line at constant speed has been used in the literature to measure the number of conflicts in an unorganized airspace. This can give a measure of the complexity of the airspace. In the experiments, this model is referred to as "Baseline, straight" and is used for comparisons and to build models.

The baseline was implemented similarly to other architectures. The main difference with a Free-access architecture is that for the straight baseline, agents on the ground are immediately cleared for take-off and do not need to check whether other agents are close by. When agents in the baseline compute their next move they always return the same velocity vector pointing toward their goal at the maximum allowable velocity. Although an alternative analysis, for instance an event-based analysis, might be faster to evaluate the baseline, using the discrete-time agent-based analysis was preferred for three reasons. First, it did not require the development of an additional analysis, the agent-based simulation could be used with only minor modifications. Second, although it is less efficient than an event-based analysis, the agent-based simulation ran sufficiently fast that those cases could be run on a local machine. As a result improving the performance was not important. Finally, this provides a sanity check that the agent-based simulation was correctly implemented.

4.4 Preflight Planning

This section presents some information on the preflight planning algorithms that were implemented in the simulation. First a brief overview of the A* algorithm is given. A* is a well-known algorithm to find the shortest path in a graph. It is used extensively in the SIPP algorithm and in the Decoupled algorithm that were implemented as preflight plan-

ning algorithms for 4DT-contract planning. If the reader is not already familiar with this algorithm this subsection will be helpful to understand how the other algorithms work. The specific implementation selected for the Decoupled Algorithm is presented in the following subsection. A brief overview of the SIPP algorithm and some details about choices that were made in its implementation are presented next. Finally, some considerations on the Local VO algorithm that was proposed in the previous chapter are discussed.

4.4.1 A*

The A* algorithm can find the shortest path in a graph without exploring the whole graph. Its basic principle is the same as the Dijkstra algorithm but with an added heuristic that favors the exploration of nodes in the direction of the goal. The heuristic is a function that estimates the minimum cost-to-go to the goal from the current node. For instance, the heuristic can be the straight line distance from the node to the goal, ignoring obstacles and graph constraints. The A* algorithm can be shown to be optimally efficient. There does not exist another algorithm which given the same heuristic can guarantee that it will visit fewer nodes in the graph [79].

The pseudo-code for the A* algorithm is shown in Algorithm 2. The algorithm starts by

Algorithm 2 A* Algorithm

```
 1: priorityQueue.push(start,0)
 2: while current!=goal do
 3:     current=priorityQueue.pop()
 4:     for neighbor in current.getNeighbors() do
 5:         g=current.costToGo+travelCost(current,neighbor)
 6:         h=heuristic(neighbor)
 7:         f=g+h
 8:         if not neighbor.open OR neighbor.costToGo>g then
 9:             priorityQueue.push(neighbor,f)
10:             neighbor.open=True
11:             neighbor.costToGo=g
12:             neighbor.parent=current
```

finding all the neighbors to its current node. For each of the neighboring nodes, it computes

the cost-to-go and the heuristic. The cost-to-go of a node A is the cost to go from the start node to the node A. For the neighbor node it is equal to the cost of the current node plus the cost to travel along the edge from the current node to the neighbor node. The neighbor is said to have been opened. If the neighbor has never been opened or if the cost-to-go computed by going through the current node is smaller than the previous cost-to-go, it updates the neighbor's cost-to-go and it adds it to a priority queue with a priority equal to f, the cost-to-go plus the heuristic. To select the next node to visit, the algorithm takes the node with the smallest f value from the priority queue. It then iterates until the goal is reached.

The algorithm prioritizes the most promising nodes, i.e. the nodes with the lowest f value. This biases the search in the direction of the goal. The search is widened automatically when obstacles are on the shortest path. In order for the algorithm to return the optimal path on the graph, the heuristic must be admissible, i.e. it must be a lower bound on the remaining cost-to-go. Using a non-admissible heuristic can help find a path quicker but there is no guarantee on its optimality. The path found is optimal only with respect to the discretized space or graph.

Many variants of the algorithm have been developed over the years to relax some constraints or tailor the algorithm to a specific use case [93, 94].

4.4.2 Decoupled

The decoupled approach, as indicated by its name, decouples the spatial and temporal planning. First, the algorithm identifies a valid path in space. Then, timing of the vehicle along this path is optimized to avoid dynamic obstacles. When no static obstacle is present, a straight line is the best path in space. There have been different heuristics or optimization techniques that have been proposed to compute the timing of the agent along its predetermined path but detailed information on the implementation is often lacking. Hence, a custom decoupled algorithm based on time discretization and optimization using A* is

introduced here.

When static obstacles, such as buildings or flight restricted areas, are present, a path that avoids those static obstacles must be determined in the first step of the decoupled algorithm. Here, a simple approach was taken to find this path. First, the space is discretized into an 8-connected grid and A* is applied to find the shortest collision-free path on the grid. Then, the path obtained with A* is smoothed to remove the grid constraint. Note that the resulting path is not guaranteed to be the shortest path in space as A* returns the shortest path **on the grid**, the smoothing improves the efficiency of the path while still guaranteeing safety but does not guarantee optimality. When there are no static obstacles, this method will return a straight line from the start to the goal. To further improve this path the θ^* method, a variation on A* introduced in [94], could be used.

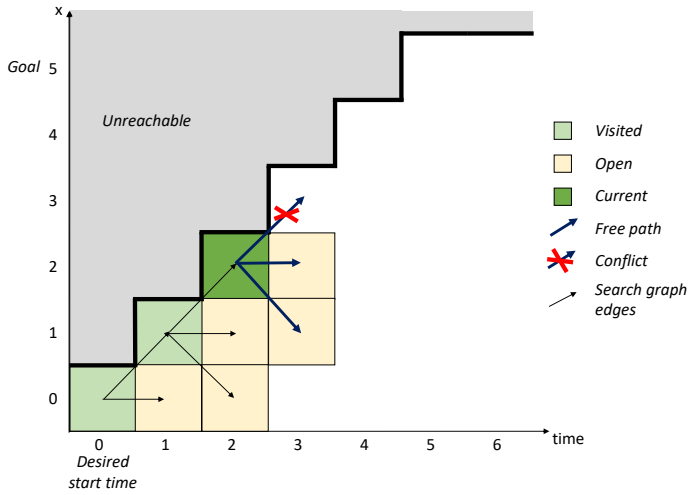
Algorithm 3 Path Smoother

```
1: function SMOOTH_PATH(path, grid)
2:     path_smoothed = [path[0]]
3:     for i in [1,length(path)) do
4:         if grid.is_path_blocked(smoothed_path[-1], path[i]) then
5:             smoothed_path.append(path[i-1])
6:     smoothed_path.append(path[-1])
7:     return smoothed_path
```

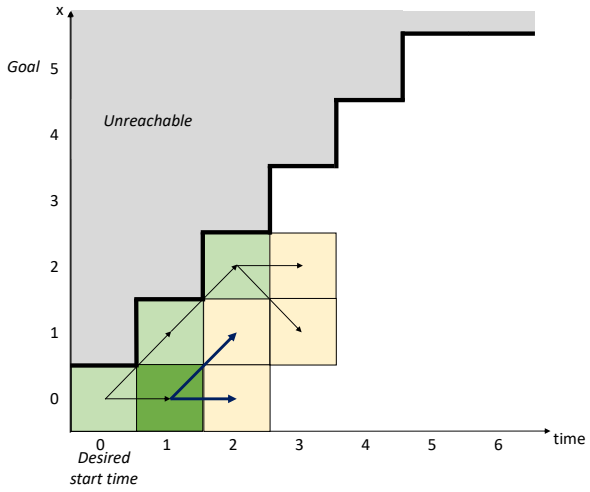
The smoother, presented in Algorithm 3, assumes that the path it is given as input is collision-free. It starts by trying to connect the origin point of the path to the furthest possible point on the path that does not result in a collision. If a connection would result in a collision, then the previous point which did not result in a collision is added to the smoothed path, and it becomes the new start point for connection attempts. The start and end point of the path are conserved. If there is no static obstacle, the path is never blocked and the smoothed path is simply the start and end point.

The second step consists in optimizing the agent motion along this path while considering dynamic obstacles to create a trajectory without loss of separation. The smoothed path is discretized into a line graph. That line is discretized into cells of equal path length.

The length of the cell is chosen as the largest distance smaller than the desired separation distance (500 m) such that there are an integer number of cells in the path. As a result of having cells of constant length to represent a potentially not straight path, the path within a cell is not necessarily straight. Having cells of constant path length, allows to have constant travel time between connected elements of the graph. This allows to represent the time dimension by an integer. As a result every cell in the graph can be represented by two integers, one representing the position along the path and one representing the ”position” in time. Dealing with integers rather than floating point numbers avoids numerical error build up. Agents can move into neighboring cells or wait in place. Hence, each cell has three neighbors in space-time (with the exception of the cells at the start and end of the path). The A* algorithm is used to find the trajectory that results in the earliest arrival time. A secondary objective is to minimize time spent in the air. To do so three priorities are used to pop elements from the priority queue: the estimated total time, which corresponds to the time to reach a node plus the heuristic, the estimated total time in flight, which corresponds to the time spent in the air plus the heuristic, and finally the heuristic. All metrics are expressed in terms of time. The heuristic is equal to the remaining distance along the path divided by the agent maximum speed. As the heuristic is only dependent on the space coordinate, it can be pre-computed to speed up the algorithm. The algorithm starts with a node at the desired start time at the start of the path. If the agent waits in place at the start node, the time in the air is 0, otherwise the time in the air is equal to the time of travel. If the secondary objective was not implemented, a trajectory in which the agent takes-off, wait one minute for another agent to go through, then goes straight to the goal, would be equivalent to a trajectory in which the agent waits one minute on the ground then takes-off and goes straight to the goal. Although in both cases the agent reaches its goal at the same time, in the latter case the agent expends less energy. Having this secondary objective slows down the computation, as more nodes must be explored. Since the runtime of the algorithm with the secondary objective was acceptable for our use cases, it was used in our



(a) Expanding the current cell



(b) New current cell

Figure 4.2: Illustration of the decoupled approach search in time and 1D space

implementation to get more energy efficient paths.

The planning problem is 2D as illustrated on Figure 4.2. The x-axis represents time, while the y-axis represents the distance along the path. All cells at the same y coordinate have the same heuristic, $h = y_{goal} - y$. All cells on a diagonal have the same f-value, since the f-value is $x + (y_{goal} - y)$. The f-value increases with $x - y$. The search will proceed along a diagonal until it becomes blocked because of a conflict with another agent as illustrated on Figure 4.2a. If there is a conflict and the search cannot progress on the

diagonal, the next explored cell will be the cell with the smallest f-value. However, there are several cells in that case that have the same f-value. The cells$(1, 0)$, $(2, 1)$ and $(3, 2)$ all have a f-value of 6. To break the ties, as explained above, the estimated time in the air is considered. The cell $(1, 0)$ has an estimated time in the air of 5, while the two others have an estimated time in the air of 6. Indeed, the energy or time in the air, does not increase on the bottom row which corresponds to the agent waiting on the ground, and waiting on the ground is always possible (i.e. no conflict). This is why as illustrated on Figure 4.2b shows that the new current cell is $(1, 0)$. This also shows how the search graph is reordered due to the energy minimization objective. The cell $(2, 1)$ which had been opened by $(1, 1)$ can be opened at a lower energy cost and same time cost by cell $(1, 0)$. The search graph can be reorganized for open cells, but the search algorithm guarantees that it indicates the lowest cost path for visited cells.

4.4.3 SIPP

The Safe Interval Path Planning method was introduced in [81]. It is a variation of the A* algorithm, shown in Algorithm 2, that reduces the dimensionality of the time dimension by reasoning on time intervals instead of discretizing the time dimension in fixed increments like the A* implementation used above. This idea of using time intervals was first introduced in [95] to find paths on a spatial graph (called a roadmap in the article). The time intervals represent continuous time segment where a point on the grid is free of conflict. The algorithm assumes that it is always advantageous to move forward and that the vehicle can wait in place. The SIPP method maximizes the time efficiency.

The main difference between the A* algorithm and SIPP is a change at line 4 of the Algorithm 2. The getNeighbors() method is replaced by a getSuccessors method that query accessible intervals instead of neighboring grid cells. As can be seen in Algorithm 4, this method requires to find the first time at which the agent can arrive on the new interval. This requires to determine what is the earliest time at which the agent can start traveling

Algorithm 4 getSuccessors(current)

```
1:  successors = []
2:  for neighbor in current.getNeighbors() do
3:      delta_t = travelTime(current,neighbor)
4:      start = current.time +delta_t
5:      end = current.end +delta_t
6:      for interval in neighbor.getSafeIntervals() do
7:          if [start,end]∩interval ≠ ∅ then
8:              t = getFirstArrivalTime(current, interval)
9:              if t ≠ ∅ then
10:                 successors.append((interval, time))
    return successors
```

toward the new interval without losing separation with other agents. The details of how this function was implemented along with a geometric explanation are available in Appendix A.

The Safe Interval Path Planning requires the discretization of the space. The area was divided into square cell of size 500 by 500 m (the minimum separation distance between agents). The grid was setup to be 8-connected, meaning that each cell (except the ones on the border) has 8 neighbors and that agents can travel along diagonals.

The SIPP centralized manager updates its free intervals when it receives a new flight plan. First, a variation on a digital differential analyzer (DDA) algorithm is used to identify all the cells along the flight plan trajectory that might be occupied (the algorithm is simplified if the flight plan follows grid points). The interval at which the center of a cell is within the minimum distance is found by solving a quadratic equation. The pseudo-code for updating the free intervals using the new occupied interval is attached in Appendix B.

Static obstacles can be preprocessed to speed up execution. When the spatial grid is initialized, the algorithm iterates over each cell of the grid and evaluates whether the center of the grid is covered by an obstacle and if it is not covered then evaluates the connections to neighboring grid cells. If the center of the cell is covered by a static obstacle it has no free intervals. If the center of the cell is free of conflict then its list of free intervals is initialized as $[[0, \infty)]$, and the connections to neighboring cells are evaluated. Available connections are stored in a sparse matrix of size n^2 by n^2 where n^2 is the number of cells

in the square grid used to discretize the airspace. In our implementation $n = 40$.

Early in the thesis work, an hexadecimal grid was first implemented to run the A* algorithm. The advantage of an hexadecimal grid is that each neighboring node is at a constant distance, this makes it efficient to explore in time assuming constant speed, it can also make collision checks faster when all agents travel along grid points. Indeed on an hexagonal grid there is no need to check for collisions when the agent moves to a previously unoccupied grid point. On an 8-connected grid, agents traveling along diagonals might cross paths. However, the same reason that makes it more efficient makes it less flexible since it relies on assumptions to speed things up. The approach chosen in the end is not optimized in terms of computation cost but is flexible which is highly desirable at an initial research stage. Moreover, the less connected the grid is (i.e. 6 neighbors instead of 8) the more constrained the path is, which can result in a loss of path efficiency. There is once again a trade-off between the optimality of the solution and computational expenses.

4.4.4 Local VO

The MIQCP problem that was presented in Chapter 3 is solved using the Gurobi software [89]. The solver's python interface is used.

Since this formulation is based like ORCA on the Velocity Obstacle concept, many elements are similar to ensure that the linear programming problem is not too constrained. Those include limiting the sensing radius to consider nearby intruders only and limiting the maximum number of constraints by only considering the nearest agents. For agents in the simulation, the sensing radius was set to 5km and only the 10 nearest agents are considered. ORCA also introduces a time horizon, which is not included in the Local VO approach. For static obstacle this means that agents cannot fly straight in the direction of a static obstacle. To limit this issue, the range at which static obstacles are considered is smaller, only 1 kilometer. Note that the distance to a static obstacle is defined as the distance to the obstacle's boundary, not to the static obstacle center. The closest 5 static obstacles

are considered, unless there are more than 5 static obstacles within a planning distance of the current position, in which case all static obstacles within that distance are considered. The planning distance is the planning time step multiplied by the vehicle maximum speed, here $10sec \times 20m.s^{-1} = 200m$. If static obstacles within that distance were ignored it might result in an invalid velocity vector. This can occur when there are many small static obstacles near each others.

Because this method is local it can get trapped in local minima, as illustrated on Figure 4.3. This is not an issue when there are no static obstacles as there is always a time such that there is no other agent in the simulation, hence the algorithm is complete although there is no guarantee on delays. With static obstacles however, there can be "pockets" or"dead-ends" formed by static obstacles that will locally trap the agent. Indeed, the cost function that the MIQCP minimizes is $|V_{desired} - V|^2$. If only constraints on angles are considered (which is the case if there are only static obstacles), for a given direction θ relative to the desired velocity, the maximum intensity of velocity returned by the optimizer will be $|V_{desired}| \cos\theta$ if θ is in $[-90°, 90°]$, and 0 otherwise. Concretely, the agent will slow down if there is an obstacle in the direction of the goal. If there is no available direction within $[-90°, 90°]$ the optimal velocity will be 0 and the planning will be stuck. The first heuristic developed to try to alleviate this issue was to detect when a local minima was occurring and change the desired velocity to be in the first available static directions. However, as illustrated on the Figure, there are instances where one time step is not enough to escape the local minima "sink". This causes the agent to be stuck in one area.

To remediate this issue, a global preflight planning step was added. This first step is identical to the method implemented for the decoupled planning. A* finds the shortest available path on the grid, which is then smoothed. The main change to the localVO method is then in the way the desired velocity is defined. Instead of being a vector that points to the goal, it is a vector that points to a lookahead point on the path. The lookahead distance was set to 500 meters, which corresponds to a distance that the agent can cover in 2.5 time

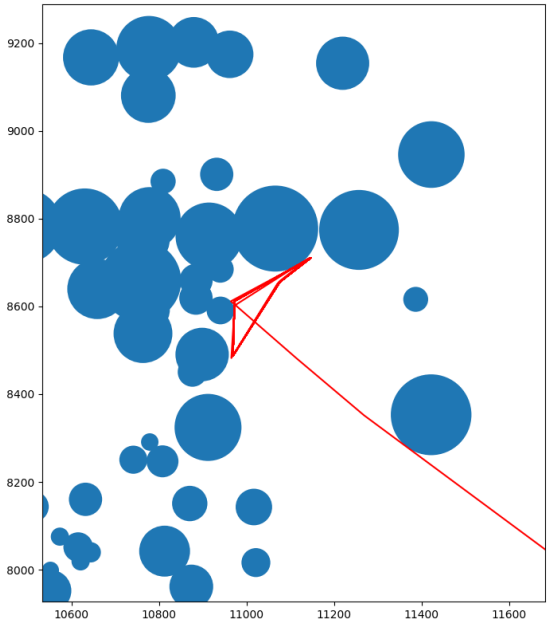


Figure 4.3: Illustration of a local minima that traps the local VO method (with an added local heuristic) if no global planning step is added. The static obstacles are shown in blue, while the agent planned trajectory is shown in red.

steps. To find the lookahead point, first the closest point on the path to the position of the agent is found by iterating through the path segments. Then by following the path from that closest point, a point that is at the lookahead distance can be found. If the closest point is closer than the lookahead distance from the goal, the goal is returned.

This issue of the local VO method becoming trapped does show that it is necessary to consider static obstacles when evaluating a method.

The local VO assumes that all agents maintain their velocity over a planning time step. When the simulation time step and planning time step are equal, that assumption holds. However, in some cases, such as when priority traffic is added, the simulation time step is smaller than the planning time step. Moreover, the VO solver only considers the nearest agents and static obstacles when looking for a solution. To ensure that the solution returned is indeed valid a sanity check is performed using the solution returned by the solver. If the solution is not valid when taking all the constraints into account then the trajectory planner proceeds as if the solver had failed to find a solution. This is acceptable in the use cases

that are considered in this thesis as the situation arises rarely.

4.5 Layers

Since agents do not change layers in flight in any of the architectures being considered in this thesis, simple 2D approaches were use.

The first approach was implemented for experiment 1.3. In that experiment the demand is uniform To restrict the range of headings, a simple check is used when generating the start and goal pair for one agent. If the pair falls in the heading range being considered then that pair is kept. If the heading from start to goal is outside the range, then a new start and goal pair is redrawn until a satisfying pair is found. The different ranges of heading considered were $[0, 180]$, $[0, 120]$ and $[0, 90]$, which respectively correspond to a minimum of 2, 3 and 4 layers to cover all possible flights.

For the final experiment, layers are used in conjunction with demand and obstacles, so multiple layers must be modeled simultaneously. A 2.5D approach is used, where multiple 2D environment are created but operated independently. Agents are created in a common environment and then assigned to a layer, either following rules if a structure was specified (e.g. heading range for each layer) or following a random uniform drawing if no rules are specified. Note that the number of agents per layer is not controlled, only the total number of agents in the simulation. Adding layers reduces the number of agents per layer. For instance, if running a simulation with $N = 300$, two layers and no airspace structure (i.e. random assignment of agents to the layers), on average there will be 150 agents per layer. The results in terms of time and energy efficiency and safety will reflect this reduced density and will be similar to what could be observed for one layer with $N = 150$. The throughput however should be about twice as high.

4.6 Traffic Patterns

Two types of traffic patterns were implemented. The first one generates origin destination pairs based on the local population density, the second one is a hub and spoke pattern.

4.6.1 Population Density

To model a non-uniform traffic demand, population density data was used similarly to what was done in [84, 53]. Here population density by census tract for the city of Atlanta developed by the Atlanta Regional Commission Open Data and Mapping Group was used [96]. Potential start and goal for the agents were placed in a regular grid with points spaced

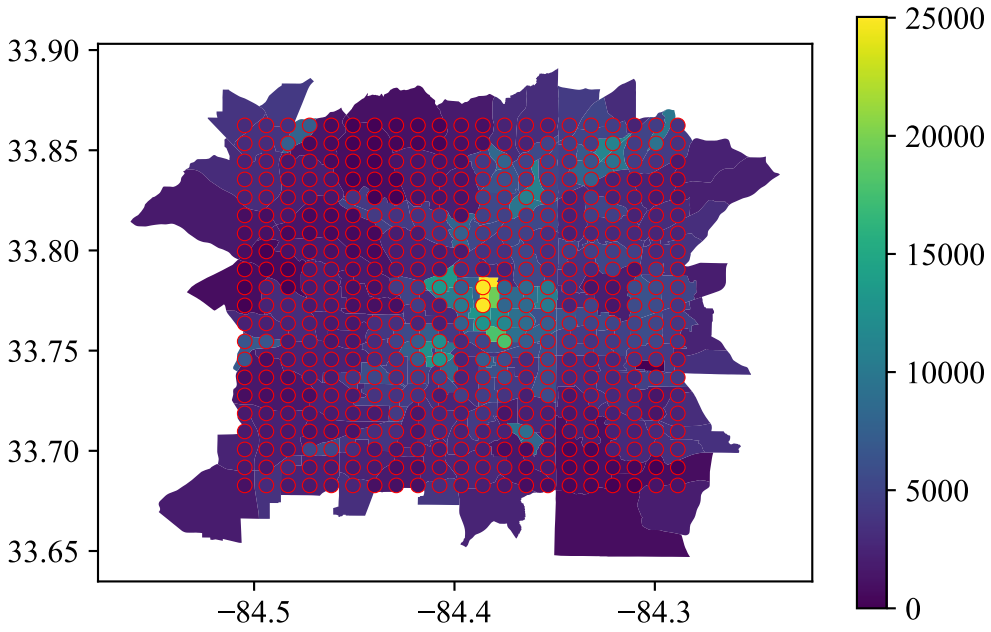


Figure 4.4: Population density in people per square mile in a 20 by 20 km area of Atlanta centered around Georgia Tech with potential start and end positions of vehicles shown as red circles

by 1 km. The grid is a square with sides of 20km length, which means that there are 421 grid points. The Weber III building at Georgia Tech was selected to be the center of the grid

(latitude:33.772571N, longitude: 84.396622W). To find the latitude and longitude of the grid points, a simple equirectangular projection centered on the center of the grid was used. The population density at each of the grid points was retrieved using the geopandas library which allows to read shapefiles and perform geographic operations such as joining points and polygons table using the 'within' operation. The code used to process the geographic information can be found in Appendix D.

Figure 4.4 shows the population density and the grid points that are used. The resulting densities at each point was normalized by the sum of densities over the grid. This way the sum of the normalized densities over the grid is equal to one. This allowed to quickly sample points proportionally to their densities using a uniform random variable between 0 and 1 combined with an array containing cumulative densities.

- Draw a random number a from a uniform distribution between $[0, 1)$
- Search the cumulative density array for the index i such that:
  $cumulative\_density[i-1] < a \leq cumulative\_density[i]$
- Return the coordinates corresponding to this index

4.6.2 Hub and Spoke

In the hub and spoke model all traffic originates from a few centralized locations and agents go to their destination and then come back to the hub. As illustrated on Figure 4.5, 4 distribution centers that each can serve 109 potential destinations were considered.

To generate a random origin destination pair, first a random distribution center is selected, then a destination is selected at random from the distribution center customer list. When the agent is close to its destination, it gets removed from the simulation, if it was on its first leg (i.e. if it was traveling from the distribution center to the customer), a new agent is added to the simulation with opposite origin and destination (i.e. from the customer to the distribution center).

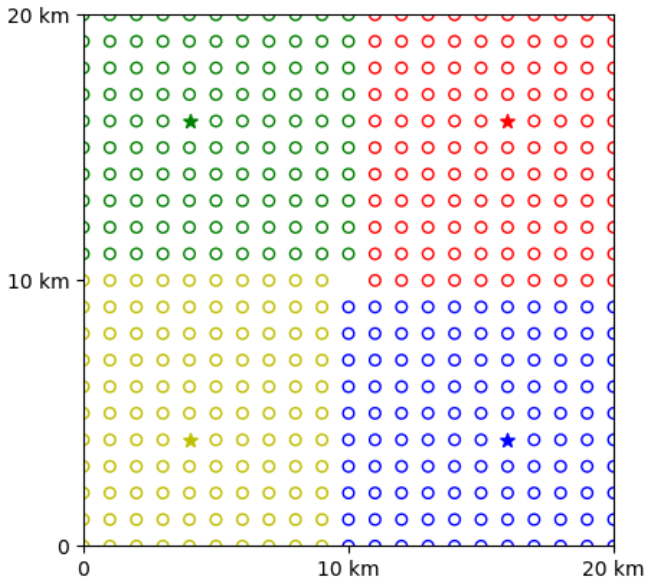


Figure 4.5: Hub-and-spoke distribution network with four warehouses (star) and their designated delivery points (circle).

4.7 Static Obstacles

To reduce the complexity all static obstacles were approximated to be circular. The study focused on obstacles at two altitude levels: 200 feet and 800 feet. The altitude levels were assumed to be defined above ground level, i.e. the altitude levels follow the terrain, so the impact of terrain can be safely ignored.

4.7.1 Airspace

The Atlanta terminal area was used as the inspiration for static obstacles due to airspace constraints. Similarly to what was done for the population density the map was centered on the Weber III building at Georgia Tech and an equirectangular projection was used to convert between cartesian coordinates and latitude and longitude. There are four airports close to the Atlanta city center: Hartsfield-Jackson International (ATL), Peachtree-Dekalb (PDK), Fulton County (FTY), Dubbins Airforce Base (ARB).

Except ARB, all these airports are active participants in LAANC. Maps of the ceilings of the area surrounding the airports are available online [4]. Manually these limits were used to define the obstacles for the 200 feet altitude layer. Note that in reality operators must still apply to operate in controlled airspace with LAANC, but here we assume that these areas would be freely accessible to UAM operators. For the 800 feet layers we considered that vehicles would only operate in uncontrolled airspace and the controlled airspace limits surrounding the four airports were used.

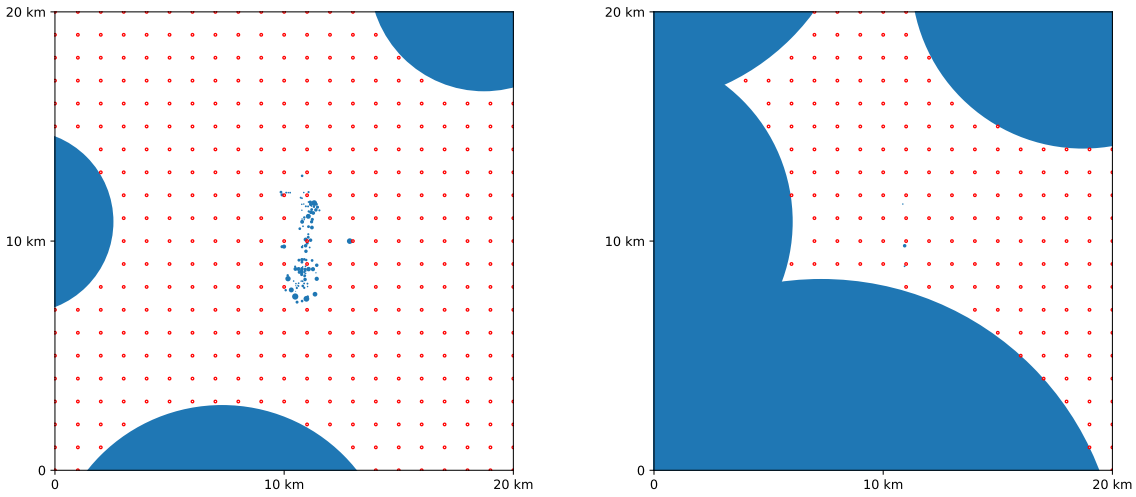
4.7.2 Buildings

Microsoft published an open-source dataset of building footprint and heights for a large number of US cities [5]. An area of Atlanta of approximately 12 by 12 km is covered by this dataset. Data for the rest of the city is not available. For simplicity all obstacles considered were approximated as circles in the simulation. The center of the circle was set at the centroid of the polygon representing the building footprint and its radius was set to be equal to the maximum distance from the centroid to a vertex of the polygon. When looking at a specific altitude only buildings that have a height superior or equal to the altitude were considered.

4.7.3 Results

Two maps were created using the data presented in the two subsections above. One at an altitude of 200ft (very low altitude, might be used by package delivery UAVs) and one at an altitude of 800ft (low altitude, might be used by eVTOLs for air taxi applications). Two files per altitude were created, the first one contains the list of obstacles, characterized by their center and radius, the second file lists the LZ available. The LZ list is in the same format as the file used for the population density study, with a uniform density across all LZ. To make sure that each LZ was reachable, LZ that were covered by a static obstacles

[4] https://www.faa.gov/uas/commercial_operators/uas_facility_maps/
[5] https://www.arcgis.com/home/item.html?id=3b0b8cf27ffb49e2a2c8370f9806f267



(a) Altitude 200 feet

(b) Altitude 800 feet

Figure 4.6: Maps of the static obstacles (filled blue circles) and Landing Zones (red circles) at different altitudes

were removed and A* was used to try to connect a handpicked point in a free area to all other LZ to guarantee that no LZ was surrounded by static obstacles.

The map at 200 feet has some large obstacles on its sides due to airports, and lots of small obstacles in the center due to buildings. The map at 800 feet on the other hand has larger obstacles due to airports on the sides of the planning area but very few obstacles due to buildings. The first map presents challenges for trajectory planning due to the presence of local minima as illustrated on Figure 4.3 in section subsection 4.4.4, while the second map presents challenges due to the large areas that are restricted, increasing the local density of flights.

4.8 Priority Traffic

Priority traffic has been modeled as straight flights at constant velocity between an origin and destination which has priority over all the other agents. Priority traffic start and end points are created outside the simulation area to simplify the logic associated with taking-off in a potentially busy airspace. The priority traffic is assumed to participate in the UTM

system although it does not abide by all the rules that regular agents must follow. This corresponds to emergency flights such as Helicopter Air Ambulance flights that might fly at low altitudes and must be given emergency priority through the airspace and can be in contact with air traffic control to tell them their needs.

For the strategic cases that means that the priority agents send their flight plan to the centralized authority as it takes-off. A specific logic was written for strategic agents that are perturbed by the new priority agent plan. When a priority agent adds its flight plan to the centralized manager, it iterates over all previously submitted flight plans by non-priority agents that are not finished and finds the ones that will conflict. Priority is given to agents that are already in the air. Agents that are already in the air must switch to a reactive behavior. For the SIPP centralized manager, the occupancy grid is reset each time a priority agent enters the simulation.

There are many different ways the perturbation to the ownship's flight plan caused by a priority agent could be designed. First, the ownship could try to replan a valid 4DT with the updated information. This would require the ownship to have good communication with the centralized manager and the planning to be relatively quick. If multiple agents need to replan at the same time, the planning must be coordinated. There might be cases where there is no valid solution found by the centralized manager to a replanning problem and an emergency plan should exist to handle this possibility. Second, the ownship could default to a reactive behavior. Third, the ownship could be given some flexibility in its 4DT contract to account for those types of perturbations, this might require the planning to be more conservative. This would come at a cost in terms of area reserved for one agent while planning. Here, a mixed method was chosen. For agents that are still on the ground when the priority agent perturbs their plan, we considered that they could easily replan using the centralized manager since there would be no constraint in terms of time required for planning or communication issues. For agents in the air we considered that a default to a reactive method was a natural failsafe that would be required on unmanned

vehicles. However, we considered that switching to a reactive method would only happen once the ownship was close to the priority agent (within 2km), until that condition is met the ownship can continue following its original flight plan. Agents that are in the air are treated in priority by the centralized manager, agents that are on the ground can replan once the agents in the air have been updated.

For reactive cases, the priority agent does not need to communicate, other agents simply have access to its velocity, position and priority status, the reactive algorithms were modified when needed to handle the priority agent. Since MVP does not rely on collaboration when defining the change in velocity required to avoid a conflict, the algorithm did not need to be changed. In ORCA, the avoidance burden is shared by collaborative agents so priority agents must be handled differently. A condition was added to handle non-participating agents similarly to the way obstacles are treated.

4.9 Conclusion

This chapter presented in details the choices that were made when implementing the simulation and the reasoning behind these choices. The next chapters present the results of the experiment that were conducted using the simulation.

CHAPTER 5
EVALUATION OF SUBSYSTEMS FOR UTM ARCHITECTURES

In this chapter the results from Experiment 1.1, Experiment 1.2 and Experiment 1.3 are presented in three separate sections. As explained in Chapter 3, these experiments are used to validate or disprove the hypotheses formulated as part of the research framing. Each experiment focuses on different subsystems of the UTM system decomposition and evaluates its impact on the overall performance and interactions with other subsystems. Each section first gives a reminder of what the experiment is trying to achieve and a quick overview of how it was setup. Then, results from the experiment are presented and general remarks are made. Finally, the results are used to validate or disprove the hypothesis and answer the research question conclusively.

5.1 Experiment 1.1: Evaluation of Preflight Planning Algorithms

5.1.1 Goal and Setup

As a reminder, the first research question concerned the implementation of preflight planning algorithms and their relative performance.

Research Question 1.1

Which options should be added to the matrix of alternatives for the preflight planning subsystem?

While performing the UTM literature review in Chapter 2, decoupled algorithms were found to be commonly used to plan 4D trajectories in 4DT contract ConOps. Decoupled algorithms constrain the path in order to reduce the planning complexity, but this comes at a cost for efficiency. Agents might have to wait a long time before the path becomes available. The idea of using a method introduced in the robotics literature called the Safe

Interval Path Planning (SIPP) and a novel method relying on local collision avoidance was introduced in that chapter.

Hypothesis 1.1

If the SIPP and a locally optimized algorithms perform better in terms of capacity and efficiency than the decoupled approach, then they are viable alternatives for the preflight planning subsystem and will open the design space for 4DT-contract architectures.

Since no information is available on using SIPP in a UTM context, and the Local VO is novel and has never been implemented, a dedicated experiment is required to evaluate them and verify the claim that they perform better than a decoupled approach. Another constraint stems from the agent-based evaluation, the algorithm must be run thousands of times per simulation. If it yields good performance but takes hours to plan a trajectory it will not be compatible with the time constraints of conceptual design.

The prerequisites for this experiment are an agent-based simulation that can simulate 4DT contracts, and the implementation of the three preflight planning algorithms. Detailed information on the simulation and algorithms can be found in Chapter 4.

In this experiment algorithms are evaluated in a nominal cruise case. In following experiments, the nominal cruise will be perturbed by external factors or additional factor, but the nominal cruise case gives a baseline for the performance of the algorithms. Agents origin/destination pairs are generated at random so that the distribution of heading is uniform. Agents take-off or land from the edges of the simulated area. The metrics are measured once the simulation has reached steady-state as explained in Chapter 4. The simulation stops once a sufficient number of agents that were created in the steady-state phase have finished their flights.

The simulation is run 10 times for different values of N, where N is the total number of agents in the simulation in the steady-state phase, including agents waiting on the ground and agents in the air. This allows to see how algorithms scale with the number of agents

and might potentially nuance the results of the experiment. An upper bound of $N = 300$ was selected somewhat arbitrarily. Uber Elevate claimed that there could be up to 27 000 eVTOLs flights per day per city by 2025, which assuming 18 hours of operation per day corresponds to a throughput of 25 flights per minute [17]. Using the 2D agent-based simulation in a nominal cruise case to model straight constant speed flights, at $N = 300$ an average throughput of 28 agents per minute is obtained.

5.1.2 Results

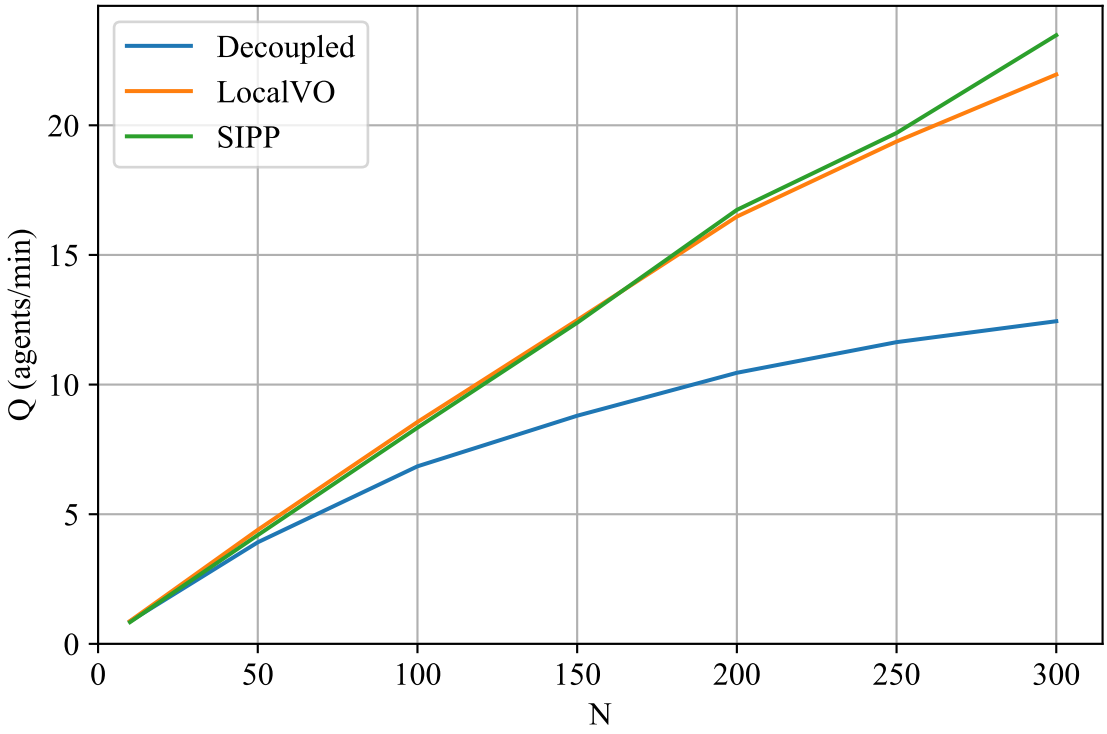


Figure 5.1: Evolution of throughput Q in function of agents density N for different preflight planning algorithms

Three different preflight planning algorithms were compared. They all use a priority system: flight plans are awarded on a ”first-come, first-served” basis. The decoupled algorithm fixes the agent path to be the shortest path to the goal without considering dynamic obstacles. It then schedules the aircraft motion along this path to avoid losses of separation. This method simplifies the planning problem by reducing its dimension. As can be seen on

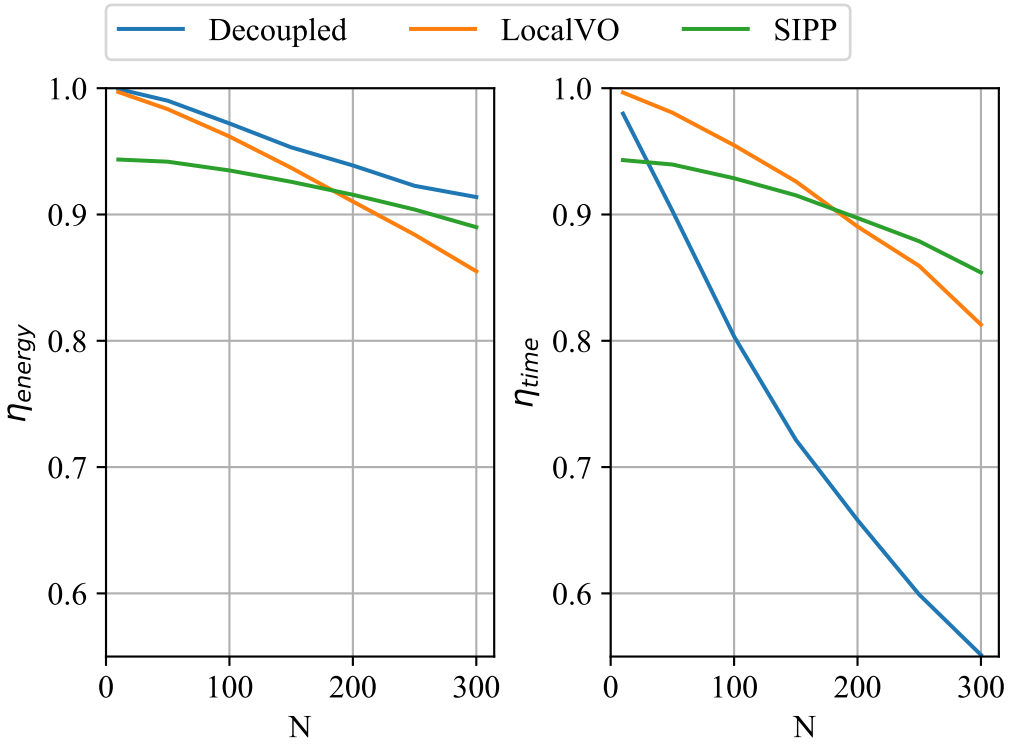


Figure 5.2: Evolution of energy (left) and time (right) efficiencies in function of agents density N for different preflight planning algorithms

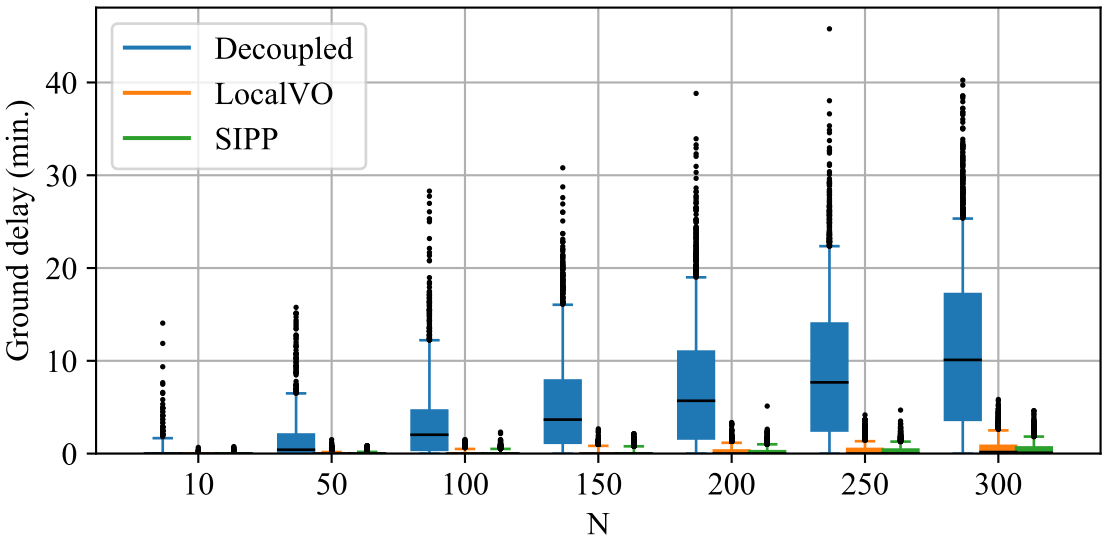


Figure 5.3: Box plot showing the distribution of ground delays across all valid agents for different preflight planning algorithms in function of agents density N. The whiskers indicate the 5th and 95th percentiles.

Figure 5.2, the paths planned by this strategy are efficient in terms of energy, agents always fly along a straight path. Note that η_{energy} is not 1 because they might have to hover in place to avoid a loss of separation (LoS). However, the trajectories are very inefficient in terms of time because it creates large ground delays as can be seen on Figure 5.3. Moreover, observing the distribution of ground delays for all valid agents shows a very large spread. This means that some agents have to wait a long time while other can take-off very quickly and as such the system is quite unfair. Because of the low time efficiency η_{time}, the throughput that can be obtained with this method is strongly limited as illustrated on Figure 5.1.

To some extent SIPP can be viewed as imposing an airspace structure. Indeed, since the algorithm relies on the discretization of the airspace, agents are constrained to progress along the edges of the graph from one waypoint to the next. This is somewhat similar to the Tubes topology shown in [32] or an airways/routes network. The SIPP algorithm returns the optimal path in terms of time efficiency on this grid/network. Because it adds a grid constraint, there might be a more efficient trajectory in the non-discretized space. This becomes evident when considering a case when there are no intruders. Let's consider an agent trying to go from point A $(0, 0)$ to point B $(6, 3)$. An agent which plans its path with the decoupled method can go from A to B in a straight line and travel a distance of $\sqrt{3^2 + 6^2} = 6.7$. On the other hand an agent which plans its path with SIPP is restricted to move on the grid, i.e. it can only do 45 degree turns. This means that the shortest trajectory to go from A to B is $[(0, 0), (3, 3), (6, 3)]$, which has a total length of $\sqrt{3^2 + 3^2} + 3 = 7.2$, 8% longer than the shortest path. In this case the energy efficiency of the SIPP trajectory is $\eta_{energy} = t_{ideal}/t_{actual} = (d_{ideal} * V)/d_{actual} * V = 0.93$. This can be seen on Figure 5.2 at low density. This disadvantage of the SIPP algorithm is most visible at low densities. As density increases all agents begin to have to maneuver significantly or wait on the ground to avoid intruders which causes their time efficiency to drop quicker than what can be observed for SIPP. At high densities SIPP has the highest throughput and time efficiency. Note that in the implementation presented here an 8-connected grid network was used but

the SIPP algorithm could be applied on a different graph structure.

The local VO approach simplifies the problem by performing maneuvers that are locally optimal, i.e. they find a velocity which avoids intruders while minimizing the deviation to the desired velocity. As shown on Figure 5.2, it is more efficient than SIPP at low and medium densities. Between 150 and 200 agents the SIPP algorithm starts to outperform the local VO algorithm in terms of efficiency. This shows that there is an advantage to optimize for time globally when density is high. However that advantage is relatively small and the locally optimal method performs satisfactorily even at high densities.

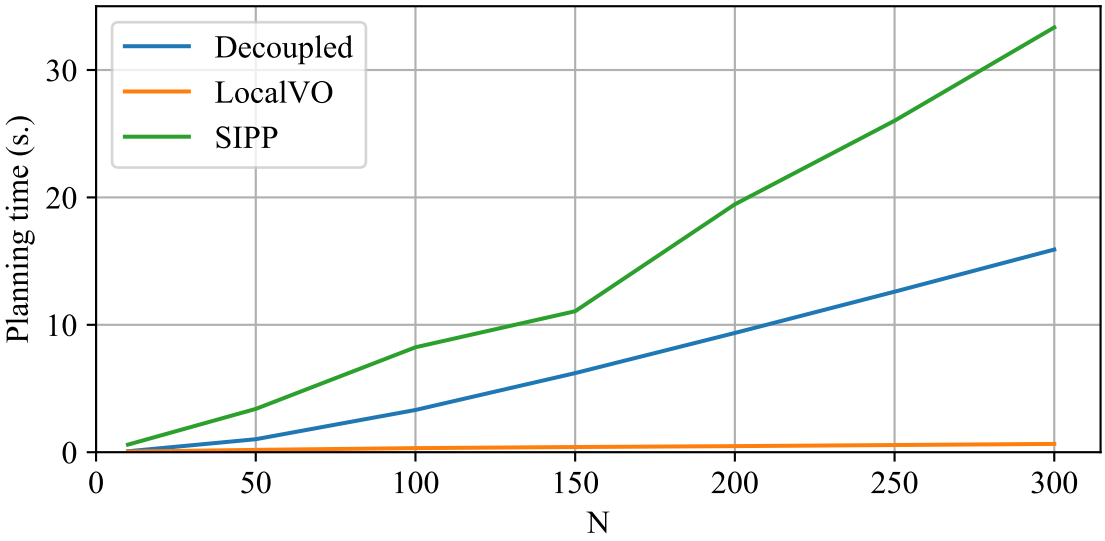


Figure 5.4: Average planning time per agent in function of agents density N for different preflight planning algorithms.

Finally, the runtime of the algorithms is shown on Figure 5.4. The more expensive an algorithm is, the more inconvenient it will be to do large parametric studies with it. This gives an idea of how each algorithm compares to the others. Note that the code for each algorithm was not profiled or optimized and so it is likely that better performance could be obtained if the implementation was improved. Moreover, Python can be much slower than lower-level language like C++. With these caveats stated, it can be still observed that the Local VO method performs much faster than the two methods that rely on A* (the decoupled and SIPP strategies). SIPP is more expensive to compute than the Decoupled

algorithm, however on average each trajectory takes less than a minute to compute which makes it acceptable for the agent-based simulation.

5.1.3 Conclusion

Using the results generated by the experiment the hypothesis can now be validated. The average performance of each algorithm is summarized in Table 5.1, Table 5.2, and Table 5.3, with the best performance for each metric shown in bold.

Table 5.1: Comparison of preflight planning algorithms for 50 agents

Algorithm	Q	η_{time}	η_{energy}	runtime
Decoupled	3.9	0.9	**0.99**	1
LocalVO	**4.4**	**0.98**	0.98	**0.18**
SIPP	4.2	0.94	0.94	3.4

Table 5.2: Comparison of preflight planning algorithms for 150 agents

Algorithm	Q	η_{time}	η_{energy}	runtime
Decoupled	8.8	0.72	**0.95**	6.2
LocalVO	**12**	**0.93**	0.94	**0.41**
SIPP	**12**	0.92	0.93	11

Table 5.3: Comparison of preflight planning algorithms for 300 agents

Algorithm	Q	η_{time}	η_{energy}	runtime
Decoupled	12	0.55	**0.91**	16
LocalVO	22	0.81	0.86	**0.65**
SIPP	**23**	**0.85**	0.89	33

The three algorithms are non-dominated in a Pareto sense, for any algorithm there is no other algorithm among the ones studied that would improve or be equivalent on all of the

metrics of interest for all densities.

The variation in performance and the trends are clear from Figure 5.1 and Figure 5.2. The hypothesis can be formally tested using a statistical test on the average performance of each run at a given density. Simulations for each preflight planning algorithm were run 10 times with different random seeds. Using a one-tailed Welch's unequal variances t-test with a p-value threshold of 5%, we can test the null hypotheses that $H_{0,1} : \mu_{decoupled} > \mu_{SIPP}$ and $H_{0,2} : \mu_{decoupled} > \mu_{LocalVO}$ at $N = 300$ for the throughput and time efficiency. The statistical tests clearly show that the null hypotheses should be rejected. Note that the statistical test is performed on the metric average value per run and not on the metric value per valid agent as the distribution of the efficiency metric per valid agent is not normal.

Table 5.4: Welch's t-test (cross for rejecting the null hypothesis, checkmark for rejecting the inverse of the null hypothesis H_1, minus if undetermined)

H_0	Q			η_{time}			η_{energy}		
$\mu_{decoupled} > \mu_?$	t	P		t	P		t	P	
SIPP	-51	2.10^{-20}	✗	-67	2.10^{-17}	✗	10	$1 - 3.10^{-8}$	✓
LocalVO	-39	3.10^{-17}	✗	-53	5.10^{-19}	✗	23	$1 - 2.10^{-14}$	✓

As had been hypothesized, the throughput and time efficiency of the proposed algorithms are better than the decoupled algorithm. The decoupled algorithm however outperforms both in terms of energy efficiency. Although the SIPP algorithm is roughly twice as expensive as the decoupled algorithm (in their current implementation), this is not prohibitive. As a result the hypothesis is validated and all three algorithms are considered as viable options for the preflight planning subsystem.

Conclusion 1.1

4DT planning algorithms that do not over-constrain the trajectory are computationally feasible and yield solutions that are better in terms of capacity and time efficiency than a decoupled algorithm. These algorithms open the design space for 4DT-contract alternatives.

5.2 Experiment 1.2: Impact of Autonomy Algorithms on the Choice of Access Control Method

5.2.1 Goal and Setup

The previous experiment has shown that the two algorithms that had been proposed for preflight planning are valid options that provide a range of performance, and that the Decoupled algorithm also has some advantages if the priority of the designer is energy efficiency. This yields a non-dominated set of three alternatives in the Pareto sense.

These three alternatives assumed that access to the airspace was controlled through 4DT contract enforced by a centralized authority. In the literature many approaches have proposed ConOps that instead rely on free-access to the airspace. As mentioned in chapter 2, there have been very few studies comparing free-access to 4DT contract alternatives, and when such studies were performed the collision avoidance and preflight planning algorithms were fixed. It is true that the type of airspace control can have a direct impact on some of the metrics of interest, for instance 4DT contract alternatives in a nominal case will have perfect safety, $n_{LoS}/h = 0$, while such guarantees cannot generally be made for free-access alternatives. So if the only metric of interest is safety it might not be worth the effort to model the algorithms at this stage. However, if the designer has more nuanced preferences then we hypothesize that it is important to model the algorithms for these two subsystems. Indeed, in the previous experiment we have seen that there can be large variations in performance among 4DT alternatives. From the literature, we know that different

collision avoidance also have very different performance. The role of this experiment is to demonstrate that the preflight planning and collision avoidance must be explicitly considered as part of the alternative evaluation.

Research Question 1.2

Should preflight planning and collision avoidance be explicitly considered when comparing architectures with different access control alternatives?

Hypothesis 1.2

If the ranking of architectures that use different access control alternatives depends on the choice of algorithms for preflight planning and collision avoidance, then autonomy algorithms should be explicitly part of the decomposition.

Since studies that have considered free-access alternatives used different assumptions, different models, and did not compare their results to the 4DT alternatives proposed here, an experiment is required to evaluate free-access alternatives in the same framework used to evaluate the 4DT contract alternatives. This was missing from the previous literature.

This experiment will simulate two free-access alternatives in a nominal cruise case without airspace structure. For collision avoidance, the first alternative will implement the commonly used MVP algorithm while the other will implement the ORCA algorithm. The ORCA algorithm explicitly relies on cooperative behaviors which might yield better performance than MVP. The same experiment setup that was used for experiment 1.1 will be used here.

Agents take-off and land from the edges of the simulated area and are generated randomly with a uniform distribution of heading. The number of agents in the simulation in the steady state is varied between 10 and 300, and each run is repeated 10 times with different random seeds.

In the next section, the performance of the free-access alternatives is commented and contrasted to the performance of the 4DT contract alternatives. These results are then used to rank the alternatives and validate or reject the hypothesis.

5.2.2 Results

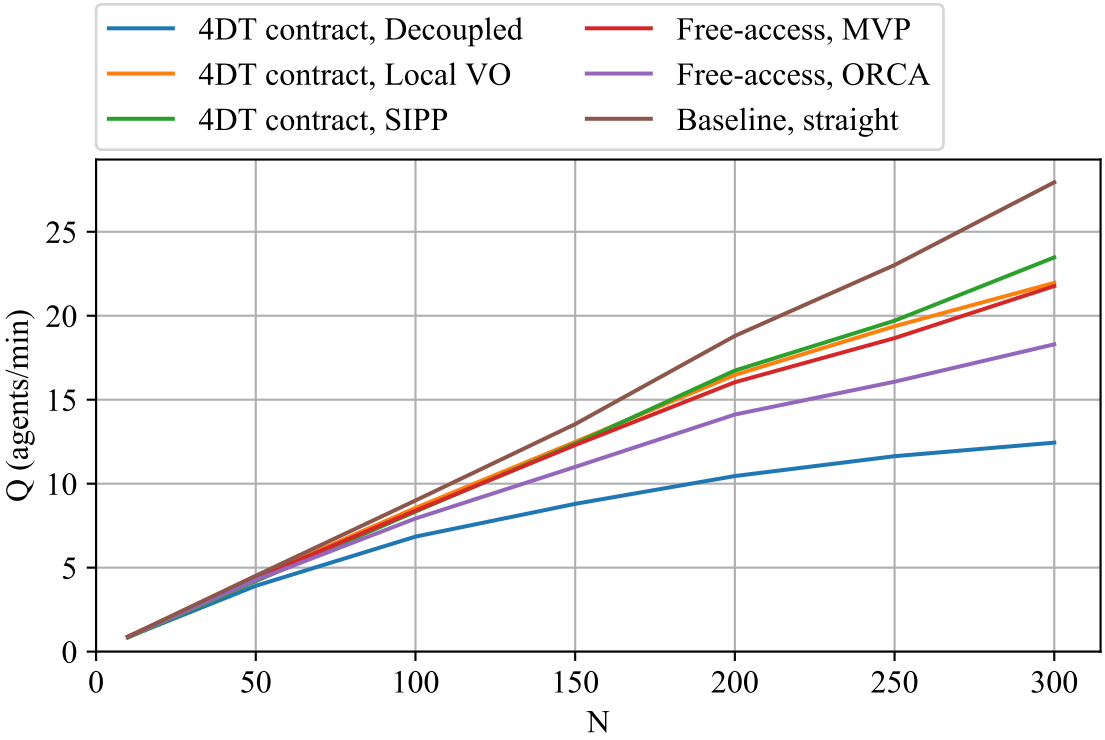


Figure 5.5: Evolution of throughput Q in function of agents density N for different architectures

In Figure 5.5 and Figure 5.6 the baseline was added to show what would happen if agents did not perform avoidance maneuvers and flew straight at constant speed to their goal. It gives an upper bound on the throughput for each density N. The efficiencies for the baseline are equal to 1 since agents follow their ideal straight trajectories. The baseline throughput is roughly linear in function of N, there are small variations due to the randomness of the origin-destination pairs created.

The trends from the free-access alternatives are similar to what was observed for 4DT contract in Experiment 1.1. As can be seen on Figure 5.5, throughput increases with agents density, while on Figure 5.6 it can be seen that both measures of efficiency decrease with density due to avoidance maneuvers.

The two collision avoidance methods rely on VO concepts similar to the preflight plan-

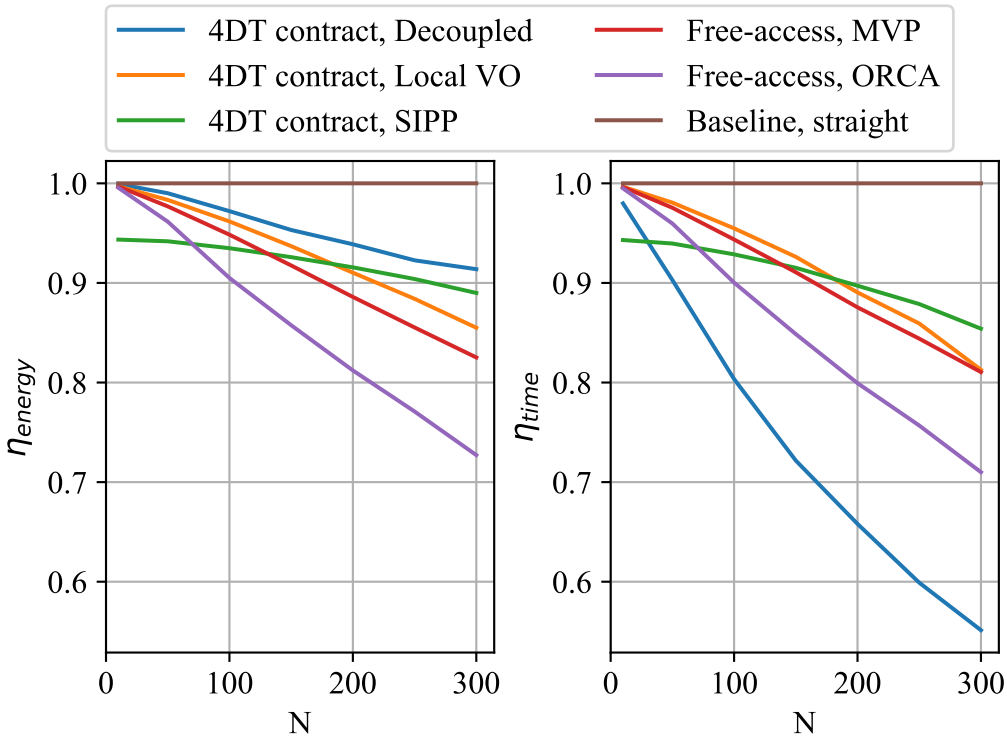


Figure 5.6: Evolution of energy (left) and time (right) efficiencies in function of agents density N for different architectures

ning local VO method. Both methods take-off as soon as the airspace above their origin is clear of conflicts. So although there can be a ground delay it is usually quite small. The ORCA method is collaborative, each agent is responsible for half the collision avoidance burden. ORCA does not attempt to solve the conflict and the solution that is selected can end up prolonging the conflict [60]. This explains why the energy efficiency decreases strongly. When comparing the distribution of energy efficiencies among agents in Figure 5.7 the ORCA method shows a wider spread than the local VO method, which indicates that with this method some agents have trajectories that are much longer than the shortest path.

The MVP method on the other hand does not account for other agents' actions. This avoids artificially prolonging the conflict and the efficiency remains relatively high. Its performance is comparable to the 4DT contract with Local VO alternative.

The main difference between 4DT contract and free-access alternatives showed in this

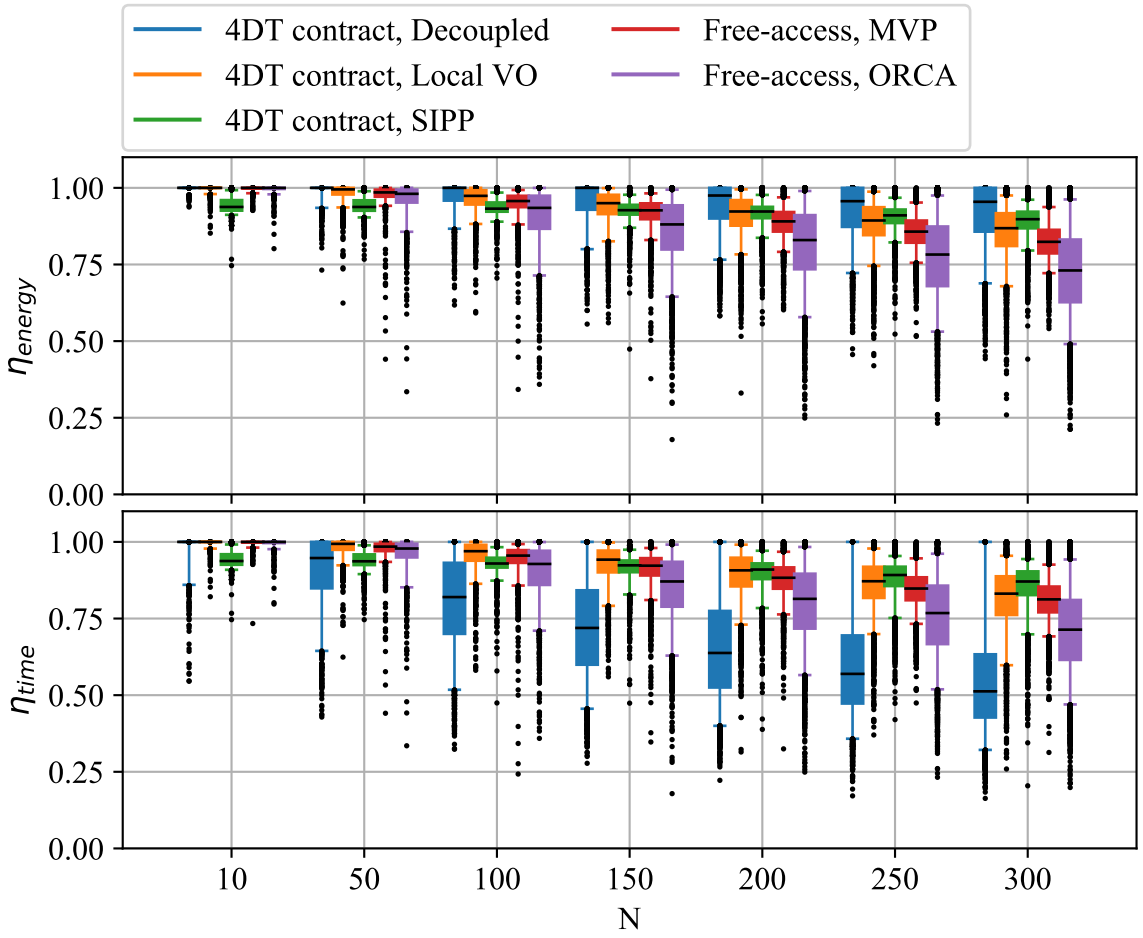


Figure 5.7: Box plot showing the distribution of efficiencies across all valid agents for different architectures in function of density. The whiskers indicate the 5th and 95th percentiles.

experiment is loss of separation. By design 4DT-contract architectures avoid all loss of separation, whereas free-access architectures fail to avoid losses of separation. This is why on Figure 5.8 only the free-access architectures and the baseline are shown.

The baseline method shows the average number of conflict per flight hour in an unorganized airspace.

The average number of losses of separation per flight hour encountered by an agent increases with the density for the free-access architectures and the baseline. The MVP algorithm is based on a geometric method which yields the exact avoidance velocity vector only when there is a single intruder and its velocity is constant. When there are multiple

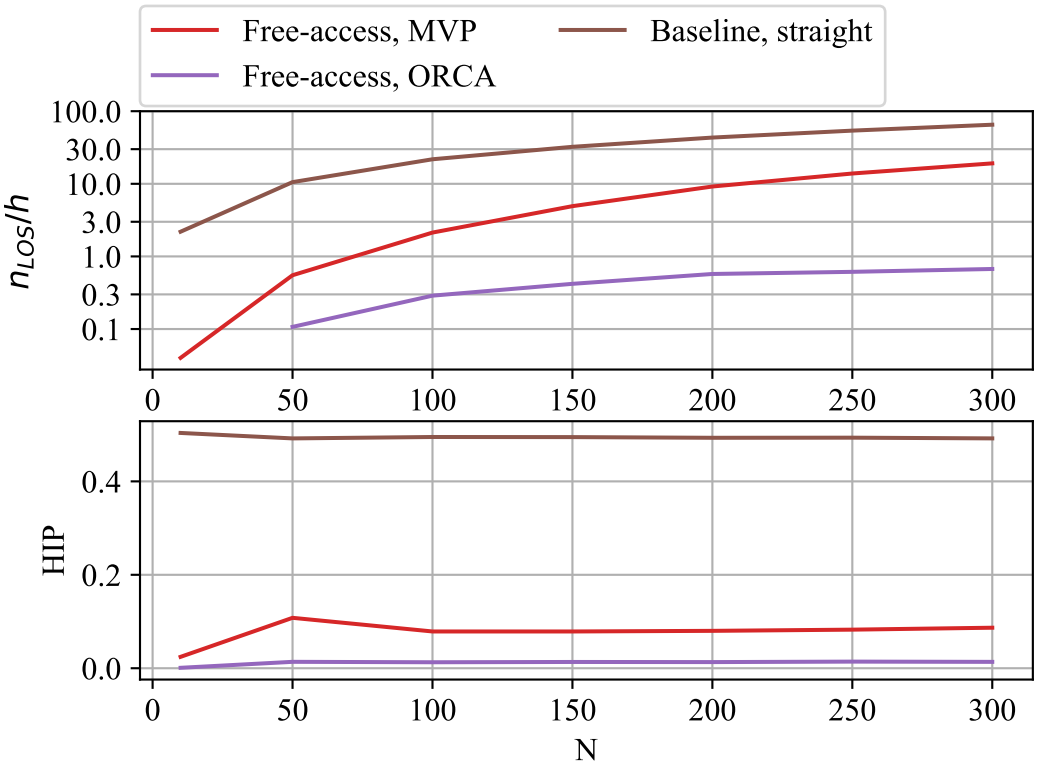


Figure 5.8: Evolution of average number of losses of separation per agent flight hour (top, shown on a log scale) and HIP (bottom) in function of agents density N for different free-access architectures

intruders that create a conflict, it sums the avoidance velocity vector which can lead to non valid solutions.

The ORCA algorithm on the other hand finds velocity vectors that will avoid losses of separation considering its k-closest neighbors. At high densities there can still be no solution, but because it considers more agents when planning it is able to limit losses of separation much more efficiently than MVP.

When looking at intrusion severity on Figure 5.8, an interesting result is that for a given architecture the average intrusion severity remains roughly constant with density. The ORCA-based architecture has on average less severe intrusions than the MVP one.

For the baseline method the average intrusion severity is equal to $1/2$. This result can be derived analytically. Remember that HIP is defined as $HIP = 1 - \min_{LoS} r/d_{min}$. Consider an agent at a fixed position, the point $\min_{LoS} r$ occurs on the line leaving the agent that is

perpendicular to the trajectory of the intruder. To find the average HIP, we can do the analysis for a set of parallel intruder trajectories and extend the results by symmetry. Since all intruders have a parallel trajectory the analysis can be restricted to a segment of length d_{min} leaving the intruder and perpendicular to the intruder trajectories. We integrate HIP over this segment and average it by dividing by the length of the segment.

$$\begin{aligned}\int_0^{d_{min}} HIP(r)dr &= \int_0^{d_{min}} (1 - \frac{r}{d_{min}})dr \\ &= d_{min} - \frac{1}{d_{min}}\frac{d_{min}^2}{2} \\ &= \frac{d_{min}}{2}\end{aligned}$$

This yields an average intrusion severity of $1/2$ no matter what the intruder density is, which is what is observed experimentally. If we were interested in the average intrusion value at any point and not just at the minimum distance, a similar approach by integrating the HIP parameter over the area $rd\theta$ would show an average value of $1/3$.

The average severity is zero for 4DT-contract architectures since there are no loss of separation and low for both free-access architectures. Given that the average HIP is small and that by definition its value is bounded by zero, it means that most losses of separation have the intruder's and the ownship's separation volumes grazing each other. This is confirmed by Figure 5.9 which shows the distribution of HIP. The upper quartiles of the distribution are close to zero for the two free-access alternatives, meaning that three quarters of loss of separation have the agents remain relatively far from each others. The baseline HIP appears to be uniformly distributed between 0 and 1, as could have been expected from the explanation of the average HIP value. ORCA has notably fewer losses of separation than MVP, and those who do occur are less severe. Looking at the distribution of Near Midair Collisions (NMACs) per flight hour confirms that ORCA manages to avoid NMAC entirely, while MVP suffers from some NMAC starting at medium/high densities (150 agents). Note that during an early implementation of this experiment, agents were

not deconflicted at take-off (i.e. agents did not check whether other agents would be taking off at the same time in the vicinity). This led to a number of conflicts and NMACs for all the free-access architectures. This highlighted the need for some sort of centralized deconfliction at take-off.

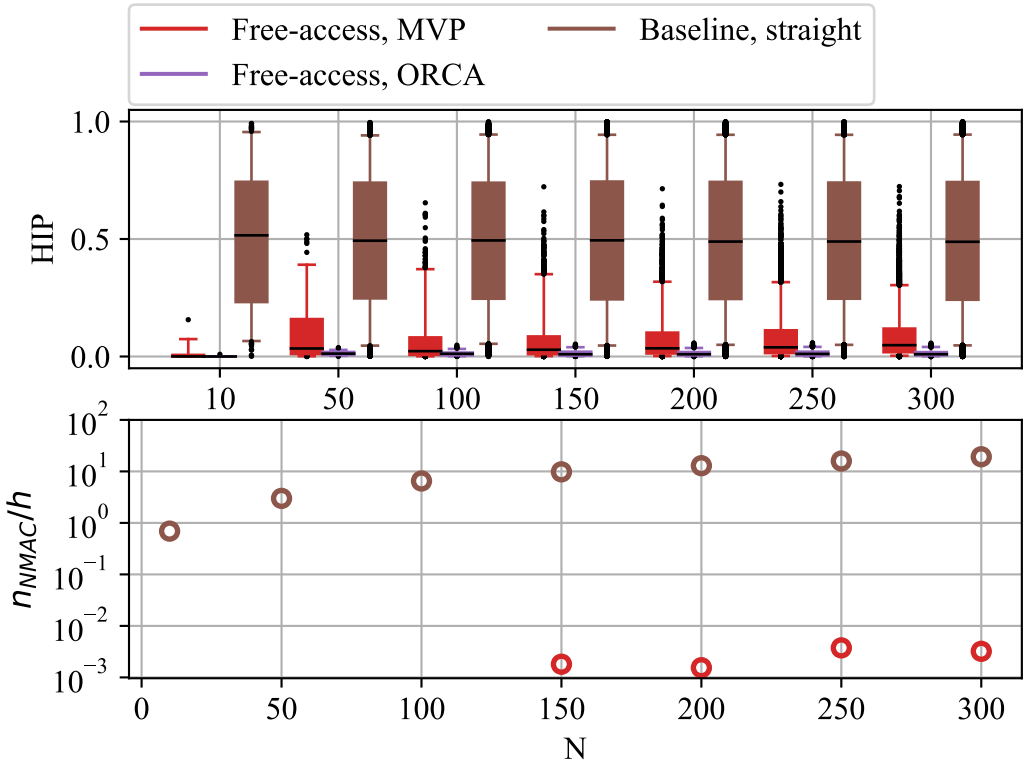


Figure 5.9: Distribution of loss of separation severity as measured by the Horizontal Intrusion Parameter (top) and number of NMAC per agent flight hour shown in log scale (bottom) in function of agents density N for different free-access architectures. The whiskers of the bounding boxes indicate the 5th and 95th percentiles.

This experiment illustrates that the algorithms chosen for collision avoidance or preflight planning significantly impact airspace metrics such as throughput, efficiency and safety. Airspace access rules guarantee safety but performance among the 4DT planning methods vary greatly on the other metrics of interest. Some free-access architectures can also have a good safety record. If infrequent, non-severe LoS are tolerable to the designer, a free-access architecture using ORCA as a collision avoidance method might be chosen.

5.2.3 Conclusion

The results show that performance of the free-access alternatives fall in-between the performance of the different 4DT alternatives that were studied in Experiment 1.1 in terms of throughput and efficiencies. For safety, the two alternatives perform worse than the 4DT alternatives as could have been expected, although ORCA manages to avoid severe loss of separation. Since all the 4DT alternatives do not dominate the free-access alternatives on all the metrics of interest, it stands to reason that the ranking of the alternatives is not only dependent on the airspace control system chosen but also on the choice of collision avoidance and preflight planning algorithm. This can be quickly confirmed by ranking the alternatives using a TOPSIS methodology for $N = 300$ as shown in Table 5.5.

Table 5.5: TOPSIS

Attributes	Alternatives					Weights	Objective
	4DT contract			Free-access			
	Decoupled	LocalVO	SIPP	MVP	ORCA		
Q	12.4	22.0	23.5	21.8	18.3	7	Maximize
η_{time}	0.55	0.81	0.85	0.81	0.71	1	Maximize
η_{energy}	0.91	0.86	0.89	0.83	0.73	1	Maximize
n_{LoS}/h	0	0	0	19.1	0.67	1	Minimize
n_{NMAC}/h	0	0	0	0.0032	0	1	Minimize
Score	0.45	0.90	0.99	0.51	0.67		
Rank	5	2	1	4	3		

As can be seen from that ranking the two free-access alternatives are sandwiched between 4DT alternatives. With this set of weights the Decoupled approach is last. This shows that it is important to consider the algorithm at this stage. Indeed, studies from the literature often used a decoupled approach and MVP for preflight planning and collision avoidance respectively. If the analysis was performed with only these two alternatives and

without considering the impact that the autonomy algorithms might have, one might conclude erroneously that a free-access alternative outperforms the 4DT-contract alternative and should be chosen.

By comparing the performance of alternatives with various algorithms, the nuance becomes evident. The hypothesis can be validated: autonomy algorithms should be explicitly part of the alternative.

Conclusion 1.2

Agents' autonomous behaviors must be explicitly considered in the architecture selection as it might change the decision on another subsystem.

5.3 Experiment 1.3: Evaluation of Airspace Structures

5.3.1 Goal and Setup

As has been previously explained, segregating traffic by layers has been shown to reduce the number of conflicts and loss of separation by different studies. However, the sensitivity of different traffic separation methods to airspace segregation has not been evaluated.

Research Question 1.3

Should airspace structures be explicitly considered when comparing architectures and how should they be modeled?

In previous papers, the impact of segregating traffic by layers on the number of conflicts was modeled [83]. There have also been a number of studies that have looked at the number of conflict as a proxy for complexity [84, 44, 90] without considering collision avoidance. In [31], the impact of layers is compared to a free airspace but the collision avoidance algorithm is fixed (MVP). This prompts several questions, can the number of conflicts obtained in the uncontrolled case be used to predict performance for the controlled case? Can the performance of layers on one alternative be extrapolated to the performance of the other alternatives?

Due to the complex interaction between agents with autonomous behaviors, the hypothesis is that simple rules like the one stated above will fail to capture some "emergent" aspects that can be captured using the 2D agent-based simulation.

Hypothesis 1.3

If the ranking of architectures can change due to difference in performance when the airspace structure is included or when a simplified model is used, then it should be explicitly part of the decomposition and modeled at an early stage.

For this second experiment, the baseline setup of the simulation remains the same as in Experiment 1.1 and Experiment 1.2. Since agents do not move between layers in any of the avoidance methods evaluated and to keep the complexity manageable, the simulation was kept 2-dimensional, with only one layer simulated at a time. The demand is still assumed to be uniform in terms of heading, hence the results from one layer can be assumed to be similar for the other layers and only one layer needs to be simulated. Three ranges of heading $[0°, 180°]$, $[0°, 120°]$, and $[0°, 90°]$ corresponding respectively to separating the airspace in 2, 3 or 4 groups, were evaluated. Note that to extrapolate the results to multiple layers based on the results for only one simulated level and ignoring vertical conflicts, the only metric that should change is the throughput, it should be multiplied by the number of levels. All the other metrics of interest are normalized by agent or flight hour.

The following section starts by analyzing the performance of each alternative similarly to what has been done in the previous experiments. Then, two different models of the effect of layers are compared to the results obtained in the agent-based simulation. These models are based on approaches that were used in some studies found in the literature. Based on the experiment's results we can conclude that the simple models do not appropriately capture all the effect of a structured airspace on the hypothesis.

5.3.2 Results

Effect of Structuring the Airspace with Layers

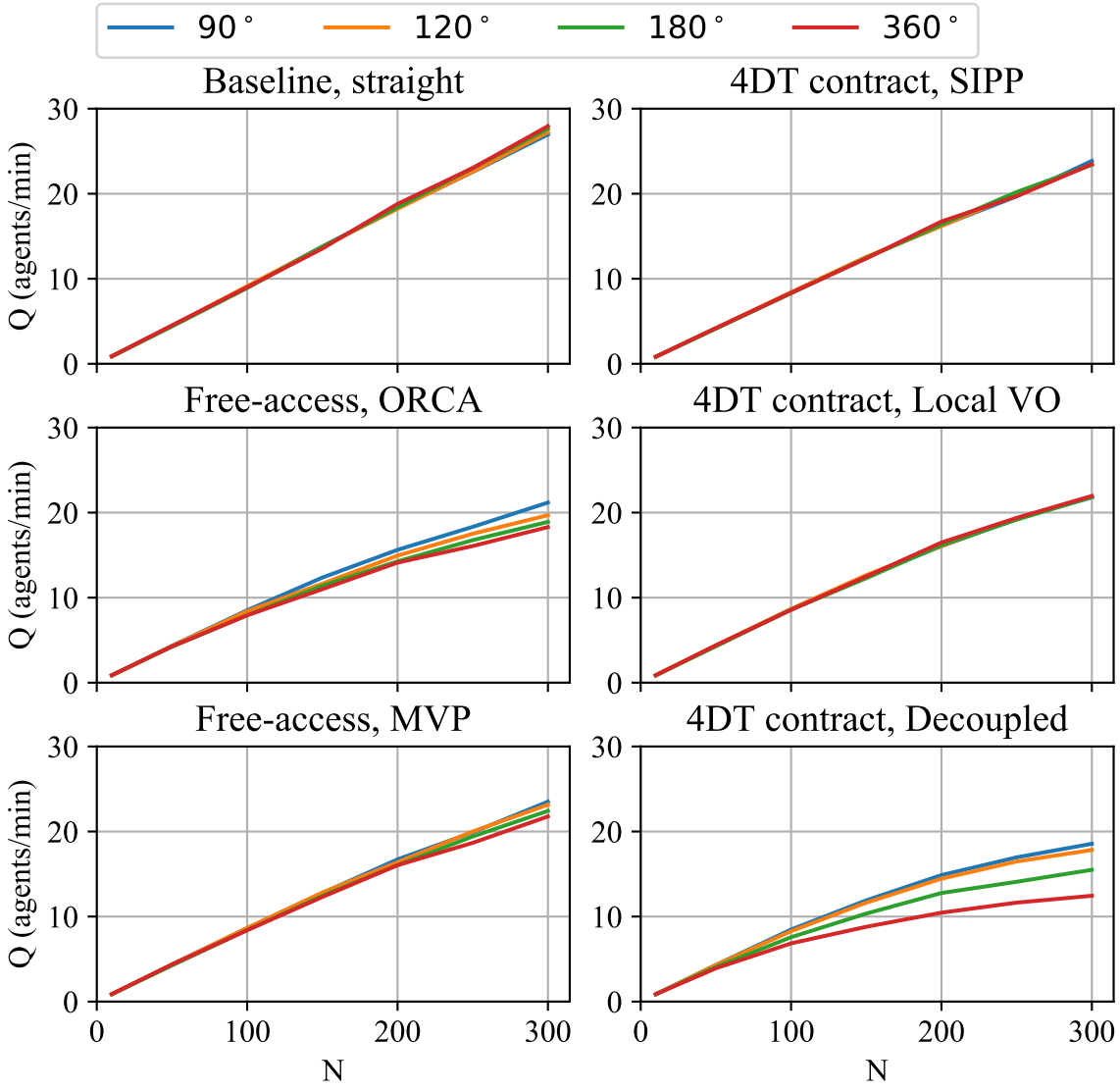


Figure 5.10: Evolution of throughput Q in function of agents density N for different deconfliction strategies and layer ranges.

Analysis of the impact of segregating traffic by heading range for different deconfliction strategies shows that the impact is very different depending on the strategy. As can be seen on Figure 5.10, segregating the traffic by layer has very little impact on the baseline throughput, i.e. the average distance for origin/destination pairs remains roughly constant

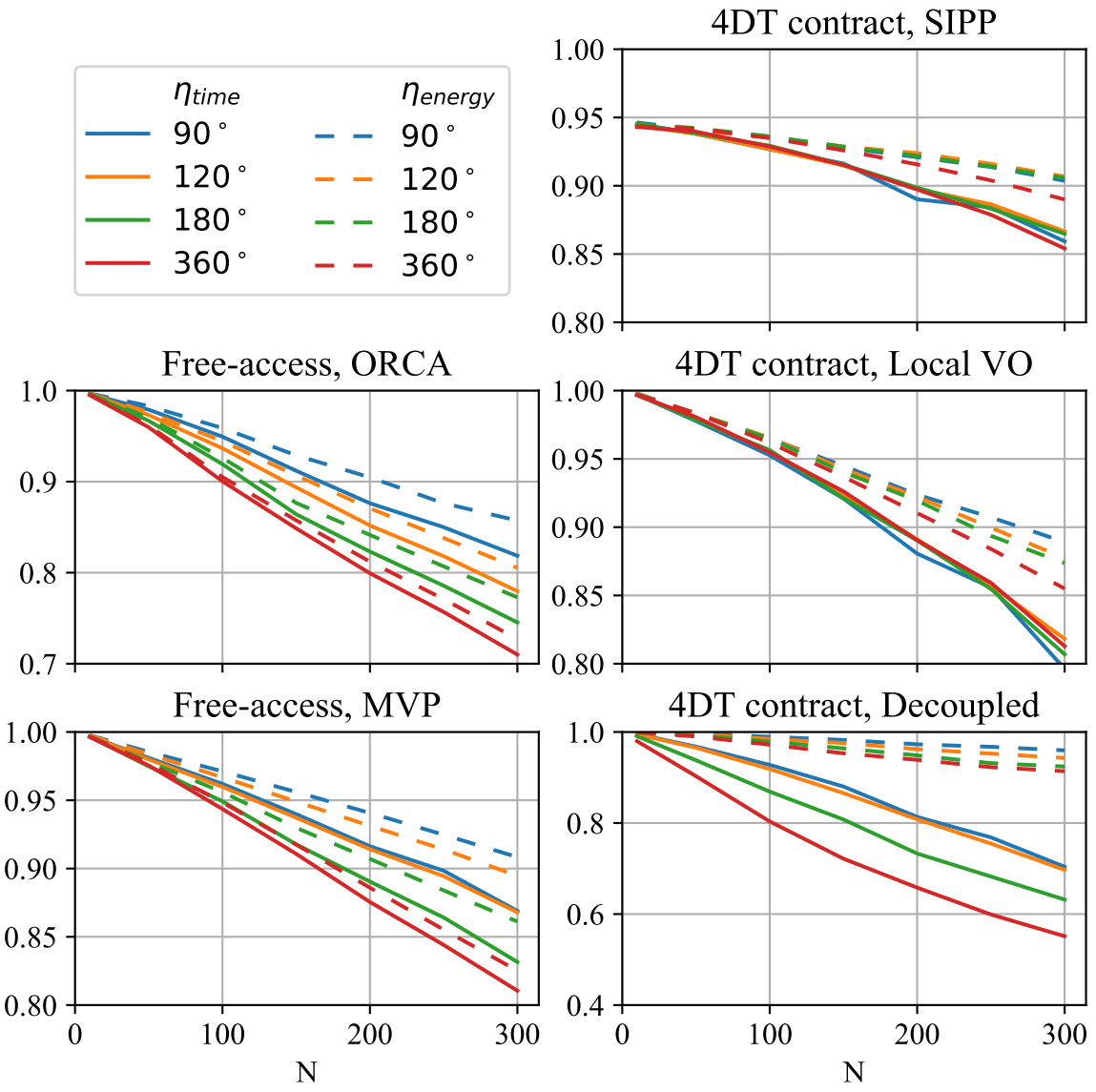


Figure 5.11: Evolution of energy and time efficiency in function of agents density N for different deconfliction strategies and layer ranges. Note that the y-scale is different between plots.

no matter the heading restriction. Among 4DT-contract architectures, only the decoupled approach shows that segregating the traffic by heading improves throughput performance, the local VO and SIPP algorithms have relatively constant throughput in function of heading range. For the decoupled method, the improvement added by each new layer diminishes, and although the throughput at 300 agents almost doubles when adding a $90°$ heading range restriction compared to the $360°$ baseline it is not enough to make the decoupled

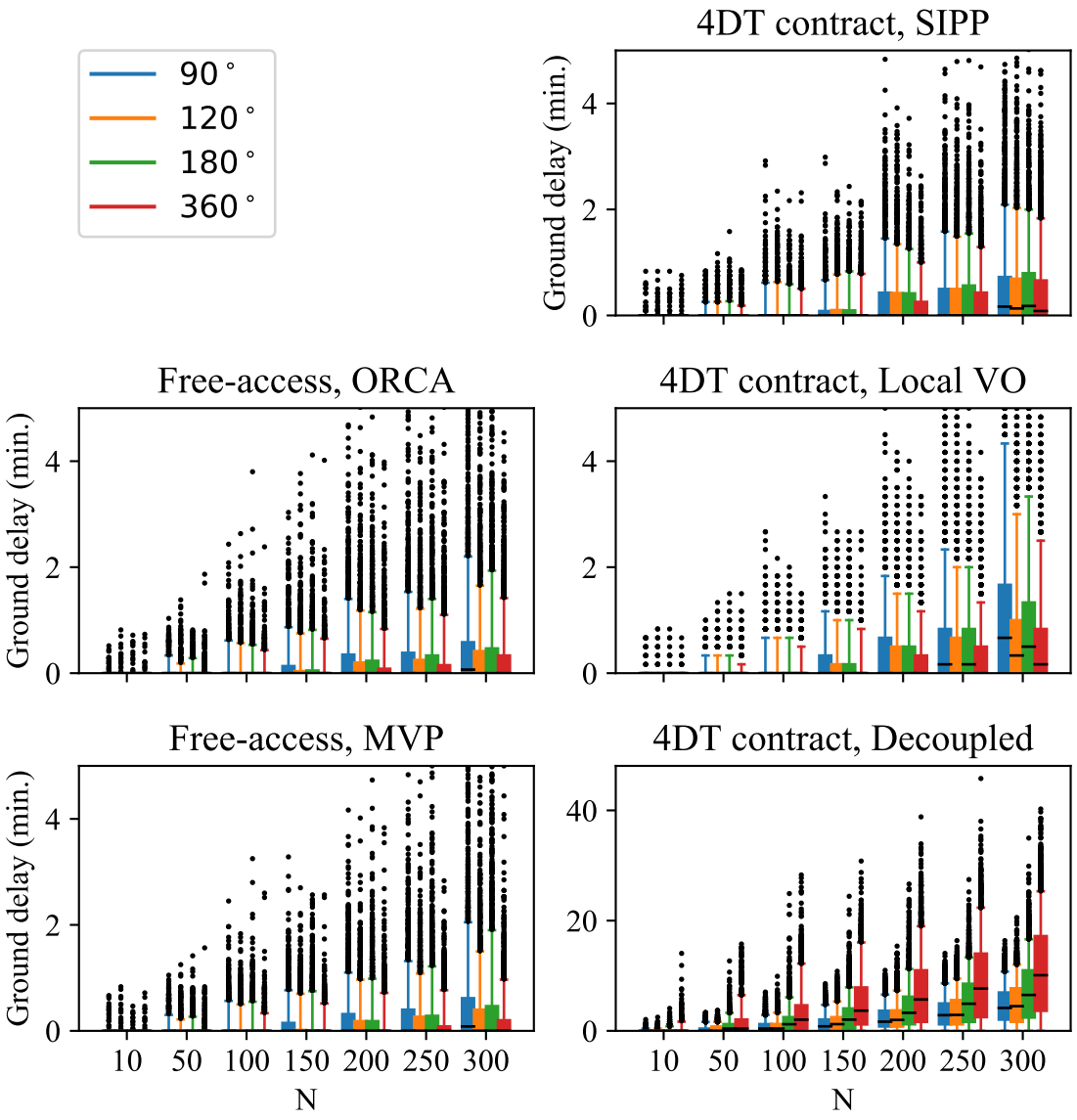


Figure 5.12: Distribution of ground delays in function of agents density N for different deconfliction strategies and layer ranges. Note that the plots for some strategies were truncated at 5 minutes ground delay to improve legibility, but at high densities there are some outliers that can go as high as 20 minutes ground delay for these methods.

method competitive compared to other 4DT-contract architectures at high density. The two free-access architectures show an improvement as the heading range is reduced but less than the 4DT-contract Decoupled architecture.

Looking at free-access architectures on Figure 5.11 shows that layers have a more distinct impact on energy efficiency than on time efficiency. The only difference between the

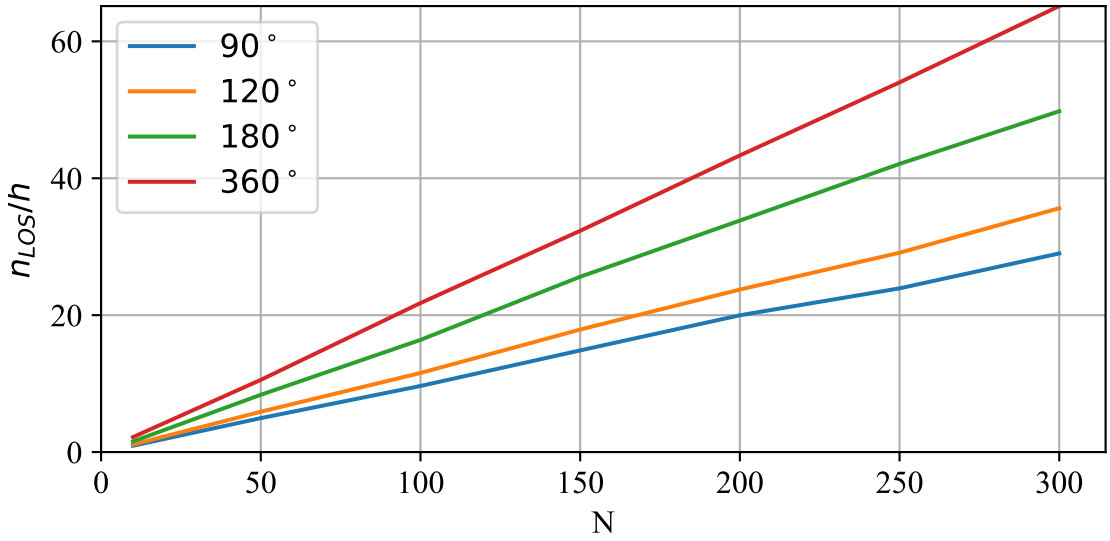


Figure 5.13: Evolution of the average loss of separation per flight hour in function of agents density N for different layer ranges for the baseline straight strategy.

two efficiencies for free-access architectures is the time spent waiting on the ground for the airspace above the agent's origin to be clear from intruders. When the range of heading is restricted, the localization of the agent's origin also becomes constrained. If the direction between the destination and the origin must be between 0 and $90°$, then the origin cannot be on the southern limit of the simulation area. This causes the density at the edge of the simulation area to increase since the same number of agents must be added and leads to higher wait times on the ground as the range of heading is reduced. This effect can be observed more directly on Figure 5.12. All methods except the Decoupled strategy show that at high densities, the position of the 75 and 95 percentiles of the distribution of ground delays tends to increase as the layer range is reduced. This effect is linked to the simulation. Looking at energy efficiency shows that adding layers improves the average efficiency as could have been expected. For 4DT-contract architectures, similarly to what was observed for throughput, the impact of layers is particularly clear for the decoupled strategy, but less so for the SIPP and local VO methods.

Looking at Figure 5.12, a few remarks can be made. The LocalVO method exhibits a clearly discretized distribution of ground delays. This is due to the fact that if a solution

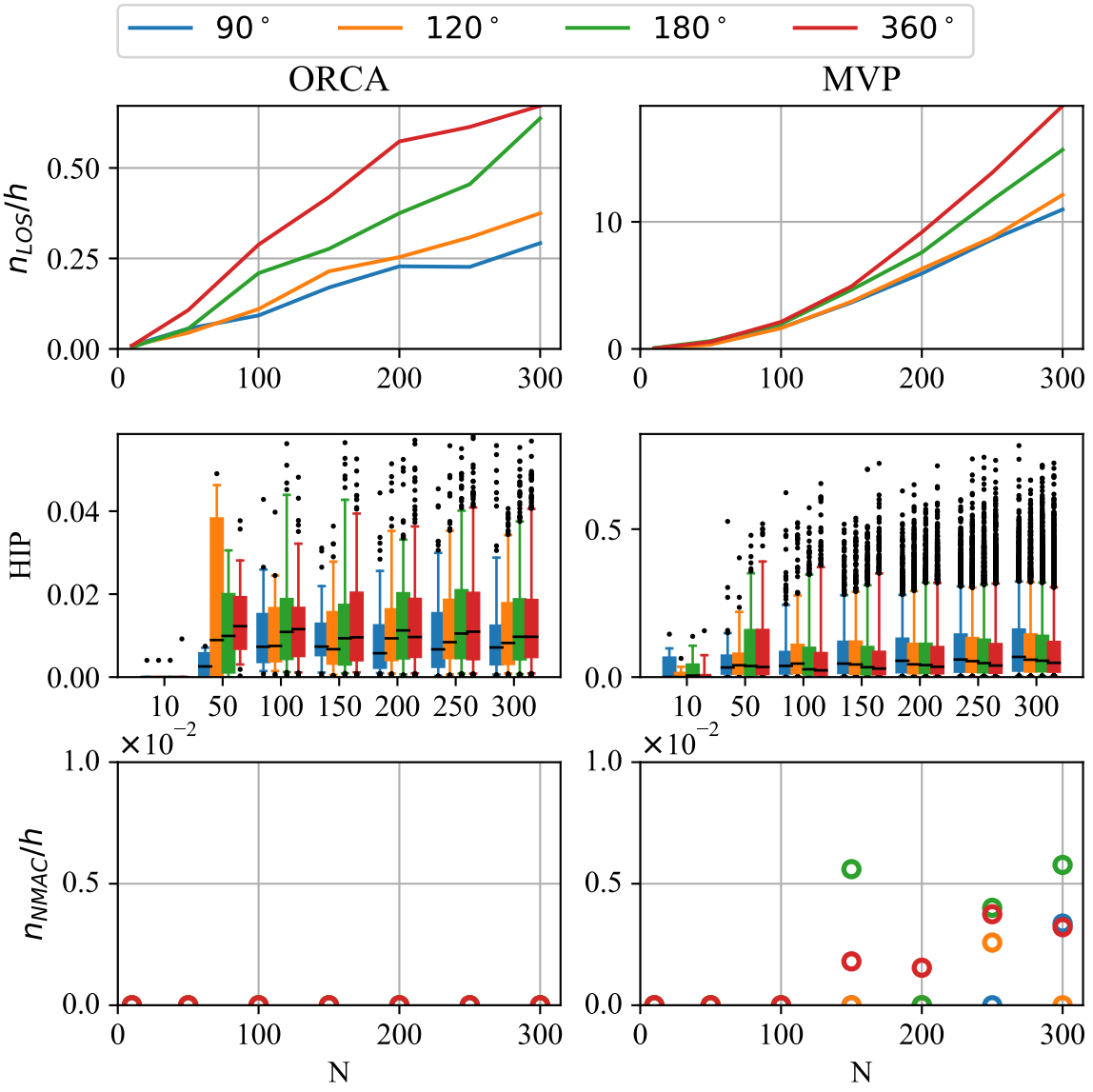


Figure 5.14: Evolution of the average loss of separation per flight hour (top), distribution of Horizontal Intrusion Parameter (middle), and average near mid-air collision per flight hour (bottom) in function of agents density N for different layer ranges for the ORCA (left) and MVP (right) algorithms.

cannot be found when departing at the desired time, the algorithm iterated by increasing the departure time by a fixed amount. Hence, ground delays are always a multiple of this amount. The decoupled method benefits the most from restricting the range of heading per altitude. It largely offsets the artifact of the simulation that creates higher density along some edges. The effect of restricting the heading range is clear even at medium densities

(100 agents), but the scale of improvements is reduced with the number of layers (i.e. there is a much sharper improvement when going from $360°$ to $180°$ than $120°$ to $90°$).

As can be seen on Figure 5.13, reducing the heading range decreases the number of losses of separation per flight hour for the baseline case. In the baseline case agents do not avoid each other and simply fly straight from their origin to their destination. This can give an indication of the number of conflicts that agents can expect to have to solve and the overall complexity of the airspace. This is imperfect since the algorithms used will change the behavior of agents and impact the number of conflicts, but still shows that as layers are added and the heading range decreases agents can expect to have to solve fewer conflicts. The more conflicts must be solved, the more agents will have to maneuver and the more the energy efficiency will be reduced. This is coherent with what was observed in Figure 5.11.

Figure 5.14 shows that in a free-access architecture controlled by the collision avoidance ORCA or MVP, the restriction of heading range reduces the number of losses of separation per flight hour but does not really have an impact on the number of NMAC. The benefit of adding one more layer decreases with the the number of layers. The distribution of HIP for ORCA is similar to what was observed in experiment 1.1, none of the conflicts are severe. For MVP, the distribution of severity is similar for all layers.

Simple Model Using the Number of Conflicts in the Baseline

The number of conflicts in the baseline where agents fly straight to their destination has been used as a proxy for airspace complexity in the literature. Here, we developed a simple modeling approach that relies on conflict count to estimate the performance of different alternatives with varying airspace structure. An overview of the method is given below. Simplified results from this analysis are then compared to those achieved from the more accurate 2D simulation discussed above.

STEP 1: Evaluate the alternatives with a free airspace structure using an agent-based sim-

ulation.

STEP 2: Evaluate the number of conflicts per flight hour in a baseline case where agents fly straight and at constant speed with and without airspace structure.

STEP 3: Fit simple models $f_{metric}^{alternative}(n_{LoS}^{baseline}/h)$ for each alternative and metric of interest using the results from step 1 and the results without a structure from step 2.

STEP 4: Predict the performance of one alternative based on the number of conflicts per hour

Step 1 uses the data that was generated during experiment 1.2. It gives the average value of the metrics of interest for each alternative without a structured airspace for values of N between 10 and 300.

Note that here step 2 is evaluated using the agent-based simulation but that, since agents just fly straight, a quicker event-based evaluation might be used instead and yield the same results. Data for the baseline when there is no airspace structure had been generated during experiment 1.1, data for the baseline with different layer range is generated as part of experiment 1.2.

For step 3 some thought is required when fitting a model. For the efficiencies and the number of LoS and NMAC, there is a normalization by the number of agents since the values are given per flight hour or per agent. Throughput on the other hand is not normalized and as such must be modeled differently. Here we fitted all metrics except throughput by using an ordinary least square to find the optimal parameters of a quadratic function $f_{metric}^{alternative}(n_{conflict}) = a \times n_{conflict}^2 + b \times n_{conflict} + c$. The fit was verified by looking at the coefficient of determination R^2. For both efficiencies across all 5 alternatives the R^2 value is greater than 0.995. For 4DT contract strategies, for n_{LoS}/h and n_{NMAC}/h the value is 0 so R^2 is not defined (the variance in the data is 0) but the error in modeling is 0. For the free-access ORCA alternative, similarly n_{NMAC}/h is equal to 0 and the error in modeling is 0. For the Free-access MVP architecture, there is more noise in the n_{NMAC}/h

value due to these events being rare and the R^2 value is only 0.855. Note that since layers decrease the number of conflicts per flight hour the fitted functions are always evaluated within the bounds used for training, there is no extrapolation. For throughput, a model that only takes into account the number of conflict will not work well. Let's consider N agents in the simulation area at the beginning of their trajectory, we can estimate the throughput if we know their average actual travel time including ground delays. If all N agents exit the simulation in $t_{avgactual}$, then the throughput will be $q = N/t_{avgactual}$. The average travel time is not modeled, however the average ideal travel time is known as it only depends on the pair of start and goals. The time efficiency can be used to estimate the average travel time. Indeed, remember that $\eta_{time} = \Delta t_{ideal}/\Delta t_{actual}$. Hence, we can estimate $t_{avgactual} = t_{avgideal}/\eta_{time}$, and in turn estimate the throughput $q = N \times \eta_{time}/t_{avgideal}$. Note that this is just a reasonable estimate, if the time efficiency is not uniformly distributed with respect with agents ideal travel time the estimation will not be correct. To estimate $t_{avgideal}$, the throughput of the baseline is used. A simple linear regression with an intercept set to 0 is fitted to the throughput $q(N) = a \times N$, where a is the inverse of the ideal average travel time. The fit obtained is good ($R^2 = 0.999$) and shows an average ideal travel time of 10.8 minutes.

Finally, the results obtained by the simple models can be compared to results obtained using the 2D agent-based simulation. Table 5.6 shows the R^2 value of the models applied to alternatives using a $[0°, 180°]$ layer airspace structure. As can be seen, the quality of fit varies a lot depending on the metric and alternatives. Although it can be quite good for some alternatives and metrics, the R^2 is below 0.9 on a number of alternatives and metrics. The performance on the number of NMAC per flight hour is quite poor which is not surprising given that the fit was not very good in the first place and there is a lot of variance in the data. As illustrated on some sample plots in Figure 5.15 showing the comparison, the models tend to overestimate the performance of the alternatives with a layered airspace and the error increases with N.

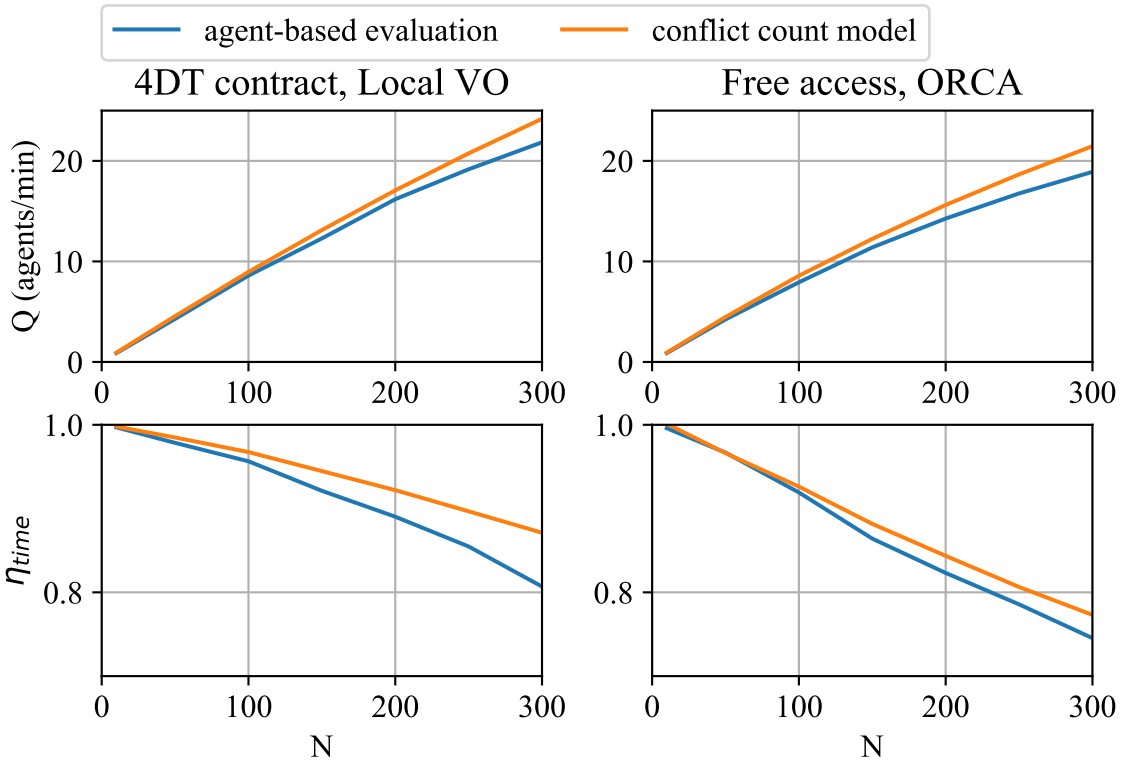


Figure 5.15: Comparison of the throughput (top) and time efficiency (bottom) for the layered (180°) 4DT Local VO alternative (left) and layered (180°) free-access ORCA alternative (right) obtained using the agent-based evaluation and using the simple conflict-based model in function of agent density N.

Table 5.6: Goodness of fit (R^2) of models based on conflict count in the baseline for a layered airspace with a heading range between $[0°, 180°]$ for 5 UTM alternatives

Metric	4DT contract			Free-access	
	Decoupled	Local VO	SIPP	MVP	ORCA
Q	0.972	0.973	0.994	0.989	0.95
η_{time}	0.974	**0.732**	**0.798**	0.93	0.962
η_{energy}	0.97	**0.897**	0.967	0.985	0.985
n_{LoS}/h	N/A	N/A	N/A	**0.839**	0.934
n_{NMAC}/h	N/A	N/A	N/A	**0.16**	N/A

Simple Model Using the Impact on one Alternative

In the literature, there have been studies where the airspace structure was evaluated while the autonomy algorithm was fixed. Here, we evaluate another simple model that assumes that the impact of airspace structure is uniform across alternatives.

STEP 1: Evaluate all the alternatives with a free airspace structure using an agent-based simulation.

STEP 2: Evaluate the free-access MVP alternative with varying airspace structure definition in the agent-based simulation.

STEP 3: Evaluate the percentage change in performance for each metric m, $\rho_m^{layer}(n) = m^{layer}(n)/m^{free}(n)$ for the MVP alternatives using the results from step 1 and step 2.

STEP 4: Predict the performance of the other alternatives using the ratio measured in step 3 and the performance of each alternative in an unstructured or free airspace that were generated as part of step 1: $m^{layer}(n) = \rho_m^{layer}(n) \times m^{free}(n)$

Step 1 uses data that was generated as part of experiment 1.1. Step 2 uses the evaluation of the free-access MVP alternative with a layered airspace structure that was generated in experiment 1.2. Contrary to the previous model that could use an event-based simulation to analyze baseline flights with layers, this model requires an agent-based simulation capable of modeling layers. Step 3 stores the ratios obtained by comparing MVP alternatives with and without airspace structure.

Finally, the performance of the model can be compared to the agent-based results and the goodness of fit can be evaluated with the coefficient of determination R^2. Results obtained for a comparison performed for a layered airspace structure with a range between $[0°, 180°]$ are shown in Table 5.7. Obviously the R^2 value is 1.0 for the MVP alternative by construction. The goodness of fit varies across alternatives but is particularly poor for the 4DT contract decoupled alternative, which is not a surprise when looking at Figure 5.10,

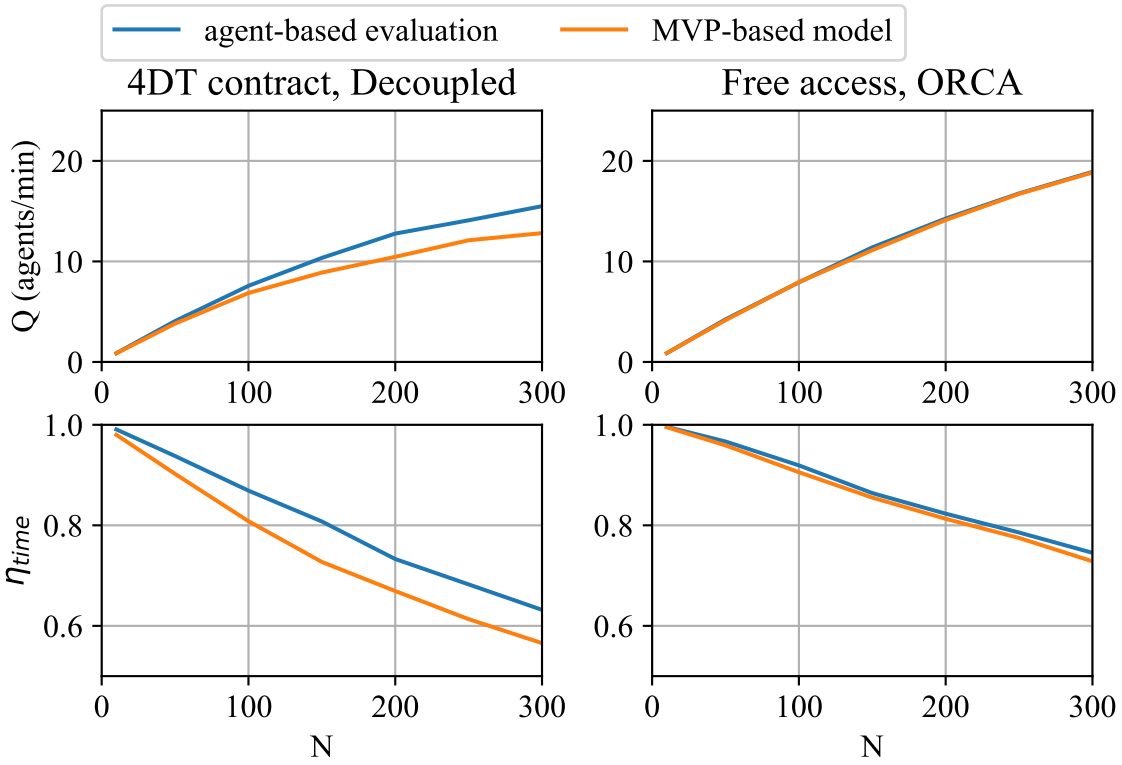


Figure 5.16: Comparison of the throughput (top) and time efficiency (bottom) for the layered (180°) 4DT Decoupled alternative (left) and layered (180°) free-access ORCA alternative (right) obtained using the agent-based evaluation and using the simple MVP-based model in function of agent density N.

the change in performance for the decoupled alternative due to layers is much higher than for the free-access MVP alternative. This model is better at predicting the performance of the other free-access alternative in terms of throughput and efficiencies, however it does not capture the number of LoS adequately. Some sample plots are provided in Figure 5.16 to show the comparison between the model and the agent-based simulation. This model underestimates the improvement obtained by the 4DT contract Decoupled alternative when airspace structure is added.

Table 5.7: Goodness of fit (R^2) of models based on MVP change in performance for a layered airspace with a heading range between $[0°, 180°]$ for 5 UTM alternatives

Metric	4DT contract			Free-access	
	Decoupled	Local VO	SIPP	MVP	ORCA
Q	**0.891**	0.995	0.998	1.0	1.0
η_{time}	**0.767**	0.938	**0.874**	1.0	0.984
η_{energy}	**0.704**	0.92	**-0.015**	1.0	0.983
n_{LoS}/h	N/A	N/A	N/A	1.0	**0.854**
n_{NMAC}/h	N/A	N/A	N/A	1.0	N/A

A negative value for R^2 means that the model is worse than a model that would just predict the average value. Since the variance of the η_{energy} metric is low for the 4DT contract SIPP alternative, a small error in the model can quickly results in the sum square of residuals being larger than the total sum of squares.

5.3.3 Conclusion

The layered airspace structure was shown to have a positive impact on the performance of UTM architectures. However the sensitivity of different architectures to the structure was quite different. A model that assumed a uniform sensitivity across architecture was shown to be a bad fit overall.

To illustrate further why airspace structures should be considered, the algorithm detailed in Appendix E is used to check whether there exists a set of weights such that the rankings of architecture is different between the decision tables Table 5.8 and Table 5.9. There are several possibilities and the following weight array is selected to illustrate the change in rankings: $W_A = [0.497, 0., 0.503, 0, 0]$.

Table 5.8: Decision Matrix for architectures with a free-airspace $N = 300$, ranking performed using the SAW methodology and the set of weight W_A

Attributes	Alternatives					Weights	Objective
	4DT contract			Free-access			
	Decoupled	LocalVO	SIPP	MVP	ORCA		
Q	12	22	23	22	18	0.497	Maximize
η_{time}	0.55	0.81	0.85	0.81	0.71	0	Maximize
η_{energy}	0.91	0.86	0.89	0.83	0.73	0.503	Maximize
n_{los}/h	0	0	0	19	0.67	0	Minimize
n_{nmac}/h	0	0	0	0.0032	0	0	Minimize
Scores	0.766	0.936	0.987	0.915	0.788		
Rank	5	2	1	3	4		

Table 5.9: Decision Matrix for architectures with a layer going from $0°$ to $180°$ and $N = 300$, ranking performed using the SAW methodology and the set of weight W_A

Attributes	Alternatives					Weights	Objective
	4DT contract			Free-access			
	Decoupled	LocalVO	SIPP	MVP	ORCA		
Q	15	22	23	22	19	0.497	Maximize
η_{time}	0.63	0.81	0.86	0.83	0.75	0	Maximize
η_{energy}	0.92	0.87	0.91	0.86	0.77	0.503	Maximize
n_{los}/h	0	0	0	16	0.64	0	Minimize
n_{nmac}/h	0	0	0	0.0058	0	0	Minimize
Scores	0.832	0.939	0.99	0.945	0.822		
Rank	4	3	1	2	5		

With this set of weight the ranking obtained in the two cases are different although the best alternative stays the same. With a k-factor the normalized matrix would be identical to

what is obtained with Table 5.8 so the results would be the same. The normalized matrix is not represented here.

Now we perform the same analysis but this time to evaluate whether the error introduced by the conflict count model is significant. The set of weight chosen is $W_B = [0, 0.174, 0.826, 0, 0]$. The decision matrices Table 5.10 and Table 5.11 show the performance, scores and rank of each architecture obtained using the SAW methodology.

Table 5.10: Decision Matrix for architectures with a layer going from 0° to 180° and $N = 300$ modeled using conflict count, ranking performed using the SAW methodology and the set of weight W_B

Attributes	Alternatives					Weights	Objective
	4DT contract			Free-access			
	Decoupled	LocalVO	SIPP	MVP	ORCA		
Q	17	24	25	24	21	0	Maximize
η_{time}	0.62	0.87	0.89	0.86	0.77	0.174	Maximize
η_{energy}	0.93	0.9	0.91	0.87	0.79	0.826	Maximize
n_{los}/h	0	0	0	12	0.59	0	Minimize
n_{nmac}/h	0	0	0	0.0025	0	0	Minimize
Scores	0.948	0.966	0.981	0.939	0.85		
Rank	3	2	1	4	5		

Table 5.11: Decision Matrix for architectures with a layer going from $0°$ to $180°$ and $N = 300$ evaluated in the agent-based simulation, ranking performed using the SAW methodology and the set of weight W_B

Attributes	Alternatives					Weights	Objective
	4DT contract			Free-access			
	Decoupled	LocalVO	SIPP	MVP	ORCA		
Q	15	22	23	22	19	0	Maximize
η_{time}	0.63	0.81	0.86	0.83	0.75	0.174	Maximize
η_{energy}	0.92	0.87	0.91	0.86	0.77	0.826	Maximize
n_{los}/h	0	0	0	16	0.64	0	Minimize
n_{nmac}/h	0	0	0	0.0058	0	0	Minimize
Scores	0.953	0.943	0.984	0.937	0.841		
Rank	2	3	1	4	5		

The second and third architecture are different between the two rankings. This validates the hypothesis that airspace structures should be evaluated concurrently with the other subsystems and proves that the model evaluated here is not suitable. However, there might be other models that could be carefully designed to capture more of the nuance of the interactions between the agents. For instance, instead of using simply the number of conflict, one could augment the model by considering conflicts involving more than two aircraft as another parameter.

Conclusion 1.3

The impact of airspace structure must be studied in conjunction with the other subsystems in an agent-based simulation, as not modeling it or modeling it through a simplified model might lead to different rankings and a different choice of architecture.

5.4 Conclusion

These three experiments validate the hypotheses that were formulated to answer the sub-research questions. New options for the preflight planning subsystem have been evaluated and shown to perform well. Experiments show that the four subsystems that were identified during the literature review have interactions that strongly impact the performance of the overall UTM system. Neglecting a subsystem or analyzing it separately would affect the outcome of the conceptual design stage. We can conclude that these four subsystems should be considered at an early stage of the design. The findings of the chapter are summarized on Figure 5.17

Note that there might be other subsystems that should be included in the analysis and that the list of subsystems evaluated in this section does not claim to be exhaustive.

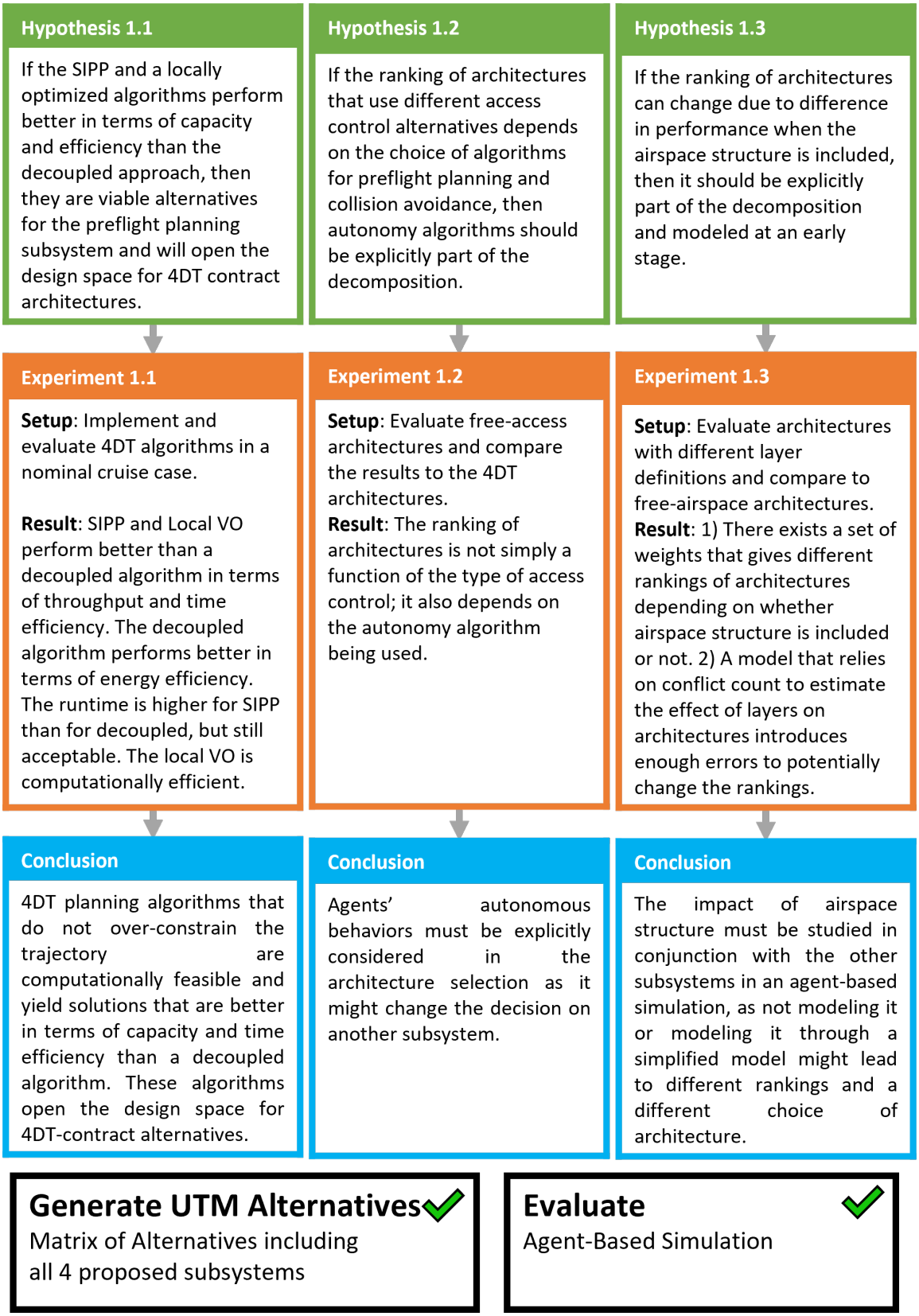


Figure 5.17: Summary of chapter 5

CHAPTER 6

SENSITIVITIES OF UTM ARCHITECTURES TO EXTERNAL FACTORS

Similarly to what was done in Chapter 4, in this chapter the results from Experiments 2.1 and 2.2 and 2.3 are presented in three separate sections. As explained in Chapter 3, these experiments are required to validate or disprove the hypotheses formulated as part of the research framing. Each experiment focuses on the impact of different external factors on the overall performance of different alternatives. Each section starts by presenting a reminder of what the experiment is trying to achieve and gives a quick overview of the experimental setup. Then, the results from the experiment are presented in details and general remarks and observations are made. Finally, these results are used to validate or disprove the hypothesis and answer the research question conclusively.

These experiments aim at evaluating the sensitivities of airspace architectures to factors that are not directly under the designer control.

6.1 Sensitivity to Traffic Demand

6.1.1 Goal and Setup

In the previous chapter, the demand was assumed to be uniform. Traffic was generated on the edges of the simulation area in a way that the distribution of heading was uniform. However, in reality urban air mobility traffic will not be uniform but rather will exhibit strong patterns similarly to what is observed in urban mobility studies. Previous studies have shown that non-uniform demand increases local density of traffic and overall decreases performance.

Research Question 2.1

Should demand be included at the conceptual design stage?

In [31], different traffic patterns based on a city map are used but the results are averaged out over the different runs, so the impact on each metric cannot be measured. In [97], the impact of different traffic patterns on conflict count is evaluated but potential interactions with autonomy algorithms are not evaluated. In [46], a population density map was used to generate origin/destination pairs but the impact of modeling the demand and how it might change based on the rules of access and autonomy algorithms were not evaluated. In [54], the number of conflicts is evaluated with changes in the demand by studying the effect of changing the number of access points, however the impact of conflict avoidance or preflight planning is not evaluated.

If the impact of demand is similar across alternatives or can be easily predicted by looking at the airspace complexity expressed as the number of conflicts then there would be no need to use an expensive agent-based simulation to quantify its effect. However, if there is a significant interaction between demand and the other subsystems that are being considered then an agent-based simulation should be used rather than a simplified approach.

Hypothesis 2.1

If the ranking of architectures can change due to difference in performance when demand is included, then it should be explicitly modeled at an early stage.

Because previous studies have not looked at autonomy algorithms in conjunction with traffic demand, there is no data available to evaluate the existence of interactions between them.

Here two different traffic patterns are examined. The first one is a hub and spoke pattern, which corresponds to a package delivery use case. The second traffic pattern is based on population density in Atlanta and assumes that the demand for travel to and from an area is proportional to its population density. This corresponds to an Air Taxi use case. Details on the data used and how it was interacted in the simulation can be found in Chapter 4. Contrary to the previous experiments agents take-off and land from within the simulation area rather than on the edges. The uniform density and heading results obtained from

experiment 1.1 are used to compare some of the results.

Similarly to the previous experiments, each run is repeated 10 times for different values of N, the number of agents in the simulation. The metrics are measured once the simulation has reached a steady state. When doing the analysis the airspace structure subsystem is fixed to a free-access with a single layer structure. The analysis is only done on 5 alternatives to reduce the number of cases to run and because it is sufficient to show that there is an interaction between the autonomy algorithms chosen for preflight planning or collision avoidance and the demand.

6.1.2 Results

Overview of the impact of different traffic demands on different alternatives

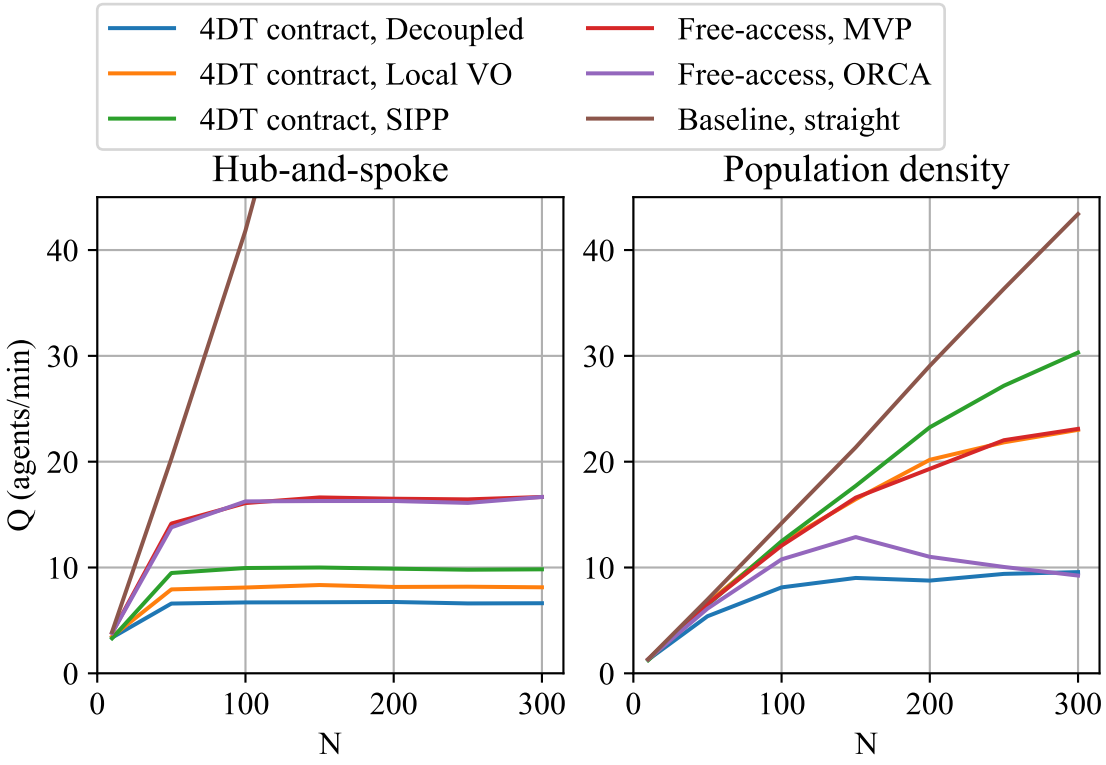


Figure 6.1: Evolution of throughput Q in function of agents density N for different architectures and traffic demand patterns

Looking at the throughput for the straight baseline shows a large difference between

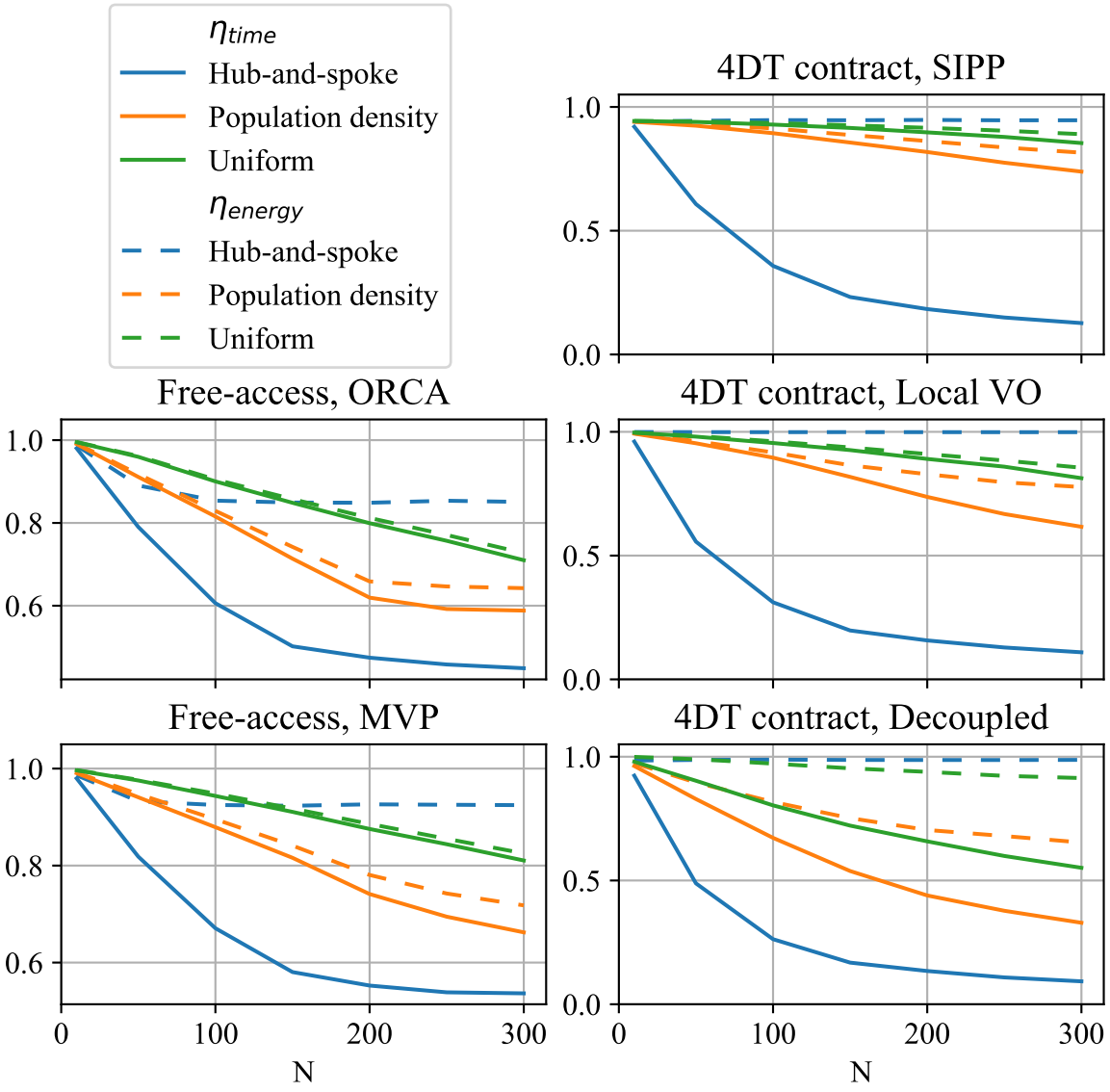


Figure 6.2: Evolution of time and energy efficiency in function of agents density N for different architectures and traffic demand patterns

the hub and spoke and the population density traffic demand patterns. As can be seen on Figure 6.1, for the population density and the hub and spoke pattern, the throughput is linear with respect to the number of agents, but for hub and spoke at 300 agents the throughput is about 140 agents/min, while for population density the throughput for that same number of agents is only about 45 agents/min. This is because the average trip length is different between the two demand patterns. In the hub and spoke model, agents' origins and destinations are always placed in the same quarter of the simulation area, hence the trips

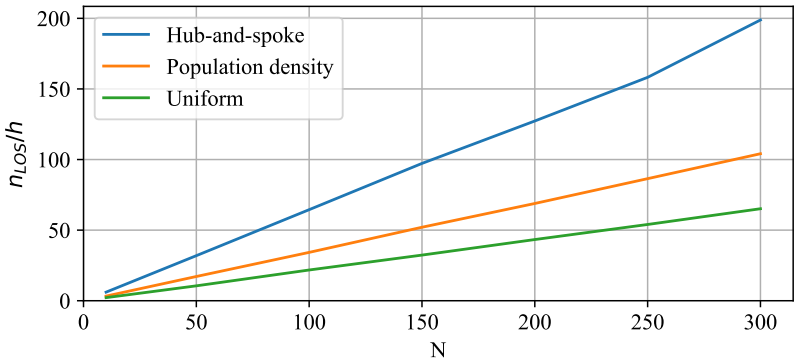


Figure 6.3: Evolution of the average number of LoS per flight hour in function of the the number of agents N in the baseline straight case for three traffic demand patterns

made by one agent are on average shorter than those generated using the population density over the whole simulation area. While the throughput is an intuitive metric to characterize the capacity of the airspace it cannot always be used to compare different scenarios to one another. This is why here the straight baseline is used for comparison and not the results from experiment 1.

For the hub and spoke pattern, 4DT-contract architectures can be seen to reach a maximum throughput value at a relatively low density (50 agents in the simulation). In this traffic demand pattern, the throughput is heavily constrained by the hubs. Indeed let's consider a hub that creates one agent that immediately moves away from the hub at constant speed, the hub will be able to spawn a new agent safely only once that agent is at the minimum separation distance. Assuming the agent speed is 20m/s and the minimum separation distance is 500m, this means the hub can only create one agent every 25 seconds at the most. The maximum inflow of agents leaving the hub is then 2.4 agents/min for safe strategies. Given that there are 4 hubs in the hub and spoke method, that agents on the return leg are counted independently, and ignoring safety constraints when landing at the hub, the maximum safe throughput would be $2.4 \times 4 \times 2 = 19.2$ agents/min. This is close to the maximum observed for the Free-access architectures, roughly 16.5 agents/min. The reason why safety constraint can be ignored when considering the return leg is because of

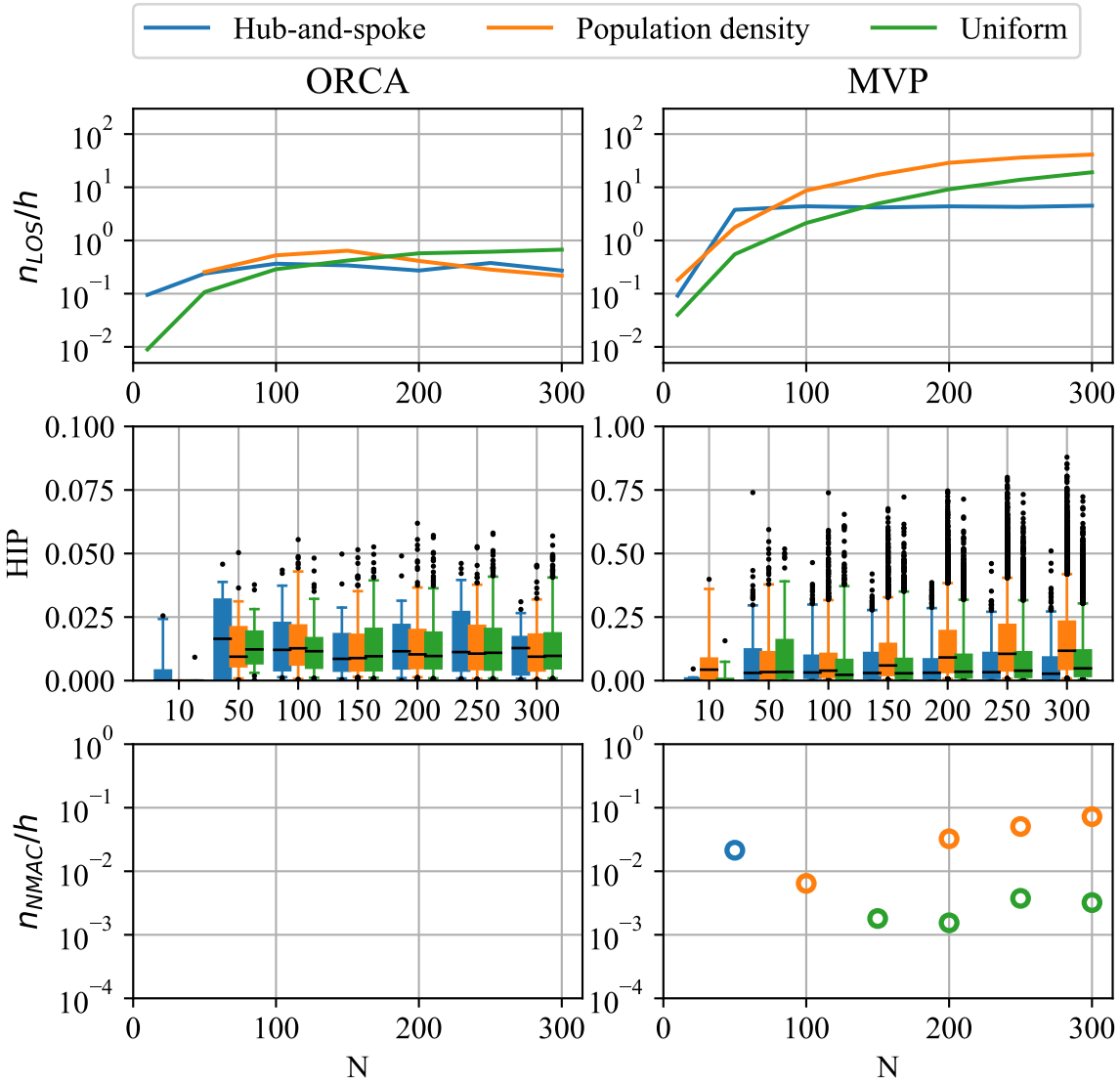


Figure 6.4: Evolution of average number of losses of separation per agent flight hour (top, shown on a log scale), HIP (middle), and number of severe losses of separation (bottom, shown on a log scale) in function of agents density N for different Free-access architectures

the tolerance, agents within 1 km of their goal are considered in the ”approach” phase and are removed from the simulation.

For the population density pattern, the SIPP method outperforms all the other methods at high densities in terms of throughput, while the LocalVO and the MVP methods perform similarly. In the ORCA approach, an inflection in the throughput can be seen as density increases. This is similar to what can be seen in car traffic simulations. This inflection

indicates that there is a ”traffic jam”, agents slow down due to traffic, which result in longer travel time the more agents are added to the simulation. ORCA is the only algorithm among the 6 that are studied that slows down the vehicles in dense conditions. The decoupled approach, which performs poorly compared to all other methods, ends up being equivalent to ORCA at high densities.

Figure 6.2 shows the energy and time efficiencies of the five methods (the baseline straight strategy has an efficiency of one by construction and is not shown) for the two different demand patterns and compare them to the uniform case that was obtained in experiment 1. As a reminder, the energy efficiency, shown by dashed lines, is higher than the time efficiency, shown by solid lines, by definition, as the time efficiency takes into account ground delays. For the three 4DT-contract architectures, the energy efficiency in the hub and spoke cases are constant with the number of agents and close to 1 (below 1 for SIPP due to the grid constraint in the solver). However, the time efficiency drops quickly. This indicates that 4DT-contrat architectures in the hub and spoke cases are constrained by the hub capacity, the path to and from the customers are close to a straight line but large ground delays are required to be able to take-off. For the population density cases both the time and energy efficiency are reduced when compared to the uniform baseline. For the two Free-access architectures, the hub and spoke cases also create larger ground delays than in the uniform and population density cases. However, the effect is much smaller than on the 4DT-contract architectures.

Figure 6.3 shows the average number of LoS per flight hour for the baseline straight method. This can give an idea of how challenging a demand pattern is. The uniform traffic pattern results as could have been expected in the lowest number of LoS. The hub and spoke pattern which concentrates traffic around 4 hubs is more challenging.

Figure 6.4 gives information on the types of losses of separations encountered by the Free-access architectures. The 4DT-contract architectures are not shown here as they guarantee safety and have zero LoS. Similarly to what was observed in previous experiments,

the ORCA method is more successful than MVP at avoiding LoS. In the hub and spoke case the number of LoS is roughly constant once the maximum throughput is reached. In the population density case for ORCA the number of LoS peaks at the same density as the throughput ($N = 150$ agents). In the hub and spoke case, the MVP alternative manages to avoid near mid air collisions except for one outlier when $N = 50$.

Simple Model Using the Impact on one Alternative

From the graphs, especially looking at the throughput for the ORCA alternative with a population density pattern, it is pretty clear that not all alternatives react in a similar manner to the addition of traffic patterns.

If, similarly to what was done in experiment 1.3, we implement a model that assumes the impact of a factor is equal over all alternatives, then the resulting fit is very poor. Here, results are presented for a population-density based pattern when the impact of changing the demand is assumed equal to what can be observed with the MVP alternative

Table 6.1: Goodness of fit (R^2) of models based on MVP change in performance with a population density based demand for 5 UTM alternatives

Metric	4DT contract			Free-access	
	Decoupled	Local VO	SIPP	MVP	ORCA
Q	**-0.057**	0.996	0.921	1.0	**-1.477**
η_{time}	**0.82**	0.965	**0.707**	1.0	0.954
η_{energy}	**0.064**	0.933	**0.277**	1.0	0.944
n_{LoS}/h	N/A	N/A	N/A	1.0	**-21.99**
n_{NMAC}/h	N/A	N/A	N/A	0.992	N/A

The R^2 value is not 1 for the n_{NMAC}/h for the free-access MVP alternative, because in the uniform case there are no near-mid-air collisions for $N = 100$. Since the model is a k-factor applied to the baseline it cannot capture the apparition of NMACs with the change

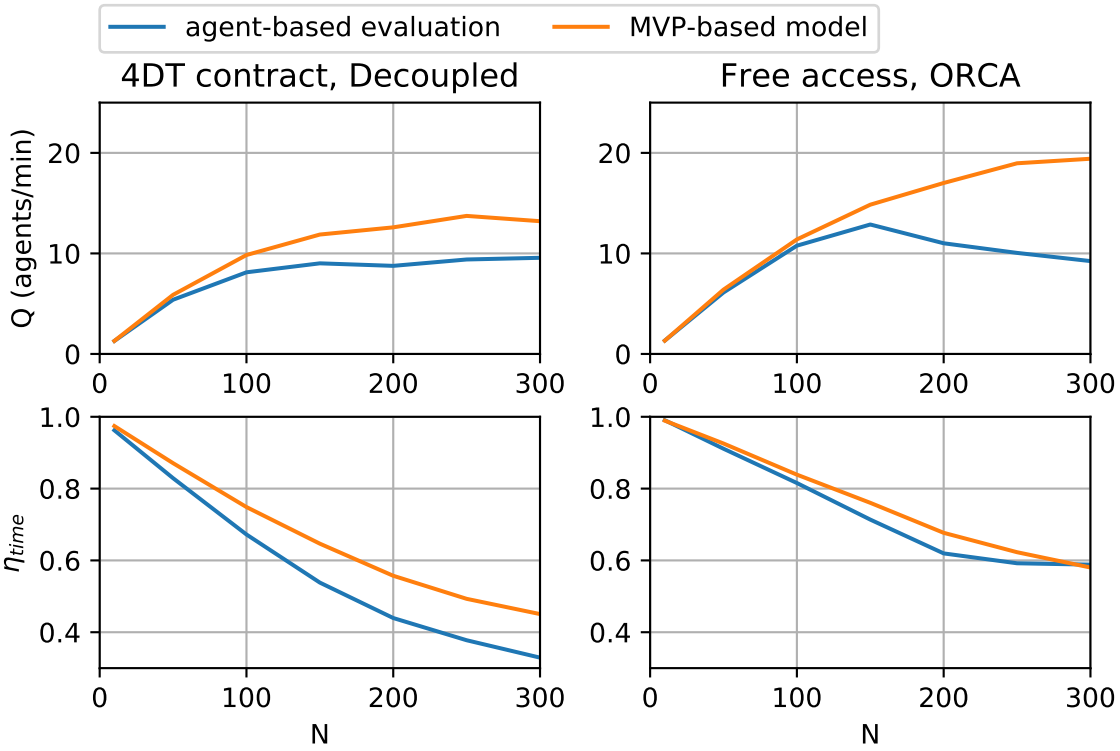


Figure 6.5: Comparison of the throughput (top) and time efficiency (bottom) 4DT Decoupled alternative (left) and free-access ORCA alternative (right) with population density based obtained using the agent-based evaluation and using the simple MVP-based model in function of agent density N.

in demand.

Note that there is no clear division between free-access alternatives and 4DT contract alternatives in terms of goodness of fit. Even though the baseline for the model is a free-access alternative, it does a better job modeling the 4DT contract Local VO alternative than the other free-access alternative that uses ORCA for collision avoidance. The model underestimates the performance of some alternatives (4DT contract SIPP, energy efficiency of the 4DT contract Local VO) and overestimates other (4DT contract Decoupled, free access ORCA).

Figure 6.5 shows the prediction of the MVP-based model on two alternatives and two metrics compared to what is obtained with the agent-based simulation.

6.1.3 Conclusion

This experiment demonstrates that traffic demand patterns have a large impact on performance and that this impact is not uniform across all strategies as a k-factor model is a bad fit. To validate that demand is indeed important to consider at the conceptual design stage, we show that the ranking of architectures may depend on the inclusion of demand in the evaluation for certain sets of weights. Using the algorithm detailed in Appendix E, the following weights are identified: $W_C = [0.784, 0, 0, 0, -0.216]$. This results in the rankings shown in the decision tables Table 6.2 and Table 6.3.

Table 6.2: Decision Matrix at $N = 300$ with uniform demand, ranking performed using the SAW methodology and the set of weights W_C

Attributes	Alternatives					Weights	Objective
	4DT contract			Free-access			
	Decoupled	LocalVO	SIPP	MVP	ORCA		
Q	12	22	23	22	18	0.784	Maximize
η_{time}	0.55	0.81	0.85	0.81	0.71	0	Maximize
η_{energy}	0.91	0.86	0.89	0.83	0.73	0	Maximize
n_{los}/h	0	0	0	19	0.67	0	Minimize
n_{nmac}/h	0	0	0	0.0032	0	-0.216	Minimize
Scores	0.416	0.733	0.784	0.511	0.611		
Rank	5	2	1	4	3		

Table 6.3: Decision Matrix at $N = 300$ with a population-density based demand, ranking performed using the SAW methodology and the set of weights W_C

Attributes	Alternatives					Weights	Objective
	4DT contract			Free-access			
	Decoupled	LocalVO	SIPP	MVP	ORCA		
Q	9.6	23	30	23	9.2	0.784	Maximize
η_{time}	0.33	0.62	0.74	0.66	0.59	0	Maximize
η_{energy}	0.65	0.78	0.82	0.72	0.64	0	Maximize
n_{los}/h	0	0	0	41	0.22	0	Minimize
n_{nmac}/h	0	0	0	0.072	0	-0.216	Minimize
Scores	0.247	0.595	0.784	0.381	0.239		
Rank	4	2	1	3	5		

The architectures ranked third, fourth and fifth are different depending on whether demand is assumed to be uniform or based on the population density. If the designer only cared about the energy efficiency, the 4DT-contract decoupled architecture would be selected when the evaluation is conducted with uniform demand while the 4DT-contract SIPP architecture would be selected when the evaluation is conducted with population-density based demand. This validates the hypothesis that demand is important to model early in the design process, and that neglecting it could lead to a sub-optimal selection of architectures.

Conclusion 2.1

Demand must be explicitly included as part of the evaluation of architectures since not including it might lead to choosing an architecture with worse performance than the one that would be chosen when demand is considered.

6.2 Sensitivity to Static Obstacles

6.2.1 Goal and Setup

Large portions of low-altitude urban airspace are not freely accessible. Large protected areas surround class B airports, and it is common to find smaller business airports around urban center which also have restricted airspace around them. At very low altitude, tall buildings also represent obstacles. These obstacles constrain the location of landing zones (LZs), impacting the demand. They also force agents to make a detour as the path between their origin and destination might be blocked by a static obstacle. This motivated the following research question:

Research Question 2.2

Should static obstacles be included at the conceptual design stage beyond their effect on demand?

The following hypothesis was developed to answer the research question:

Hypothesis 2.2

If the ranking of architectures can change due to difference in performance when static obstacles are included compared to when only the impact on demand is modeled, then static obstacles should be explicitly modeled at an early stage.

Here two altitudes are considered, and the city of Atlanta is used as the inspiration for the airspace and building constraints. At 200 feet there are clusters of buildings in the center of the simulation area due to tall buildings in the business districts of the city. At the edge of the simulation area some areas are constrained but they remained limited. At 800 feet there are very few obstacles caused by buildings, but the areas to avoid are much larger and strongly constrain the number of available LZs. Five scenarios were considered in this experiment:

- Uniform: LZs are uniformly distributed over the whole simulation area and there are

no obstacles

- Altitude 200 feet with obstacles: static obstacles that are present at 200 feet are used to generate LZs and should be avoided by all agents.
- Altitude 200 feet with no obstacle: LZs are distributed according to obstacles that are present at 200 feet but there are no static obstacle to avoid in the air
- Altitude 800 feet with obstacles: static obstacles that are present at 800 feet are used to generate LZs and should be avoided by all agents.
- Altitude 800 feet with no obstacle: LZs are distributed according to obstacles that are present at 800 feet but there are no static obstacle to avoid in the air

Comparing performance in scenarios where obstacles are not present but the LZs are still distributed as if they were to scenarios where obstacles are always active will allow to evaluate whether static obstacles should be modeled in details or not early in the design process.

6.2.2 Results

Looking at the straight baseline, it can be seen that the throughput for the three demand scenarios are fairly different, with the 800 feet altitude having the highest throughput. This makes sense as throughput is sensitive to the average travel distance. Indeed, since at 800 feet the edges of the simulation area are largely covered by static obstacles due to airport airspace constraints, most flights originate or end in the middle of the simulation area, resulting in a shorter average distance. At 200 feet that effect is reduced and the throughput is similar to what can be observed for the uniform demand case. The three demand scenarios hence cannot really be compared to one another on the throughput metric due to the difference in average travel time, similarly to what was explained in the previous section. In the straight baseline, the throughput with or without obstacles are identical

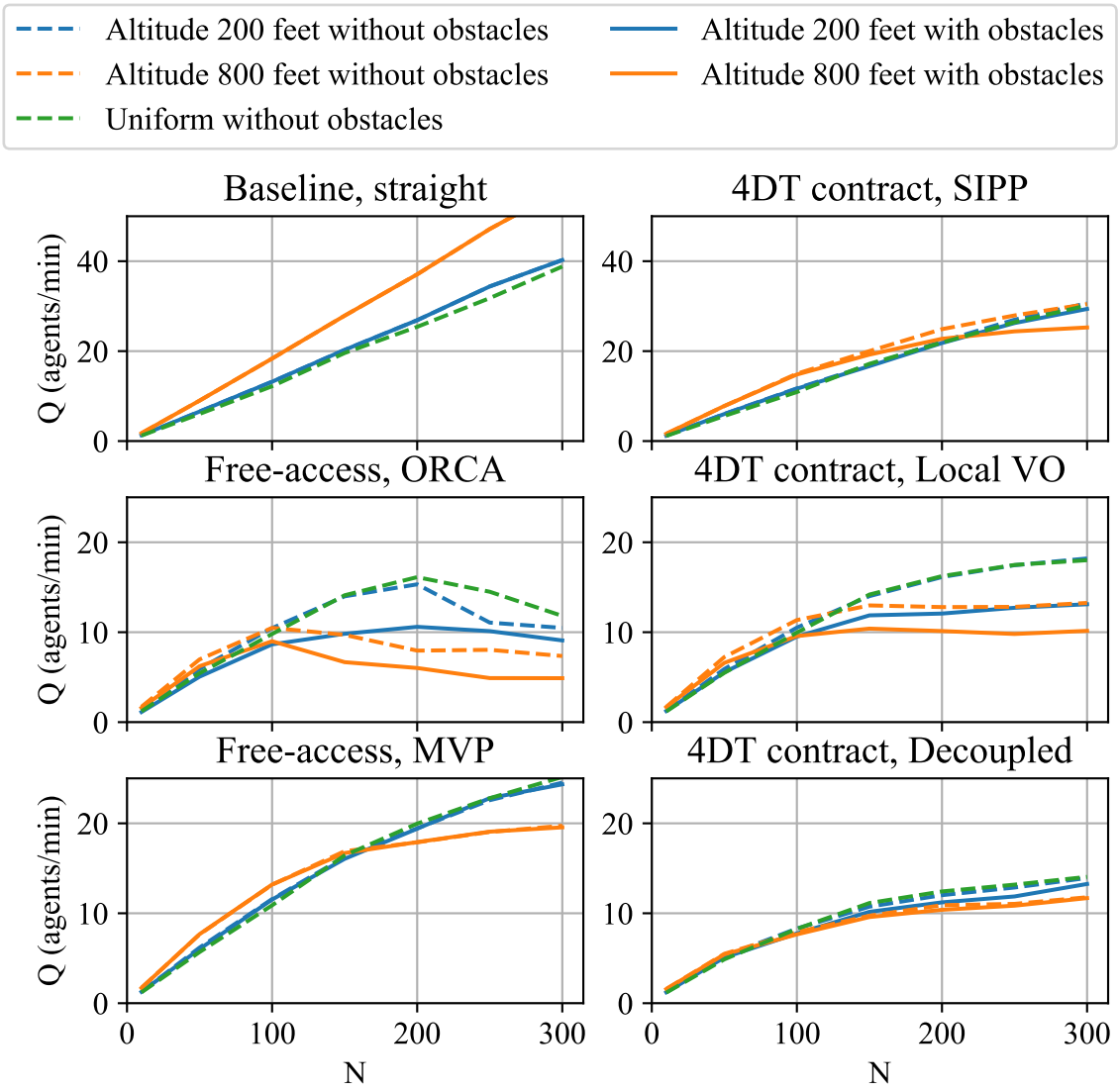


Figure 6.6: Evolution of throughput Q in function of agents density N for different architectures and traffic demand patterns

since no maneuver is performed to avoid the static obstacles. The throughput obtained at the altitude of 200 feet is similar to the throughput obtained in the uniform case as the distribution of LZ is similar between the two. This similarity can also be seen when comparing the throughput of the altitude 200 feet without obstacles to the uniform case when other algorithms are used. Only ORCA shows a difference between the two at high densities.

Looking at the efficiencies shows that, as could have been expected, obstacles decrease

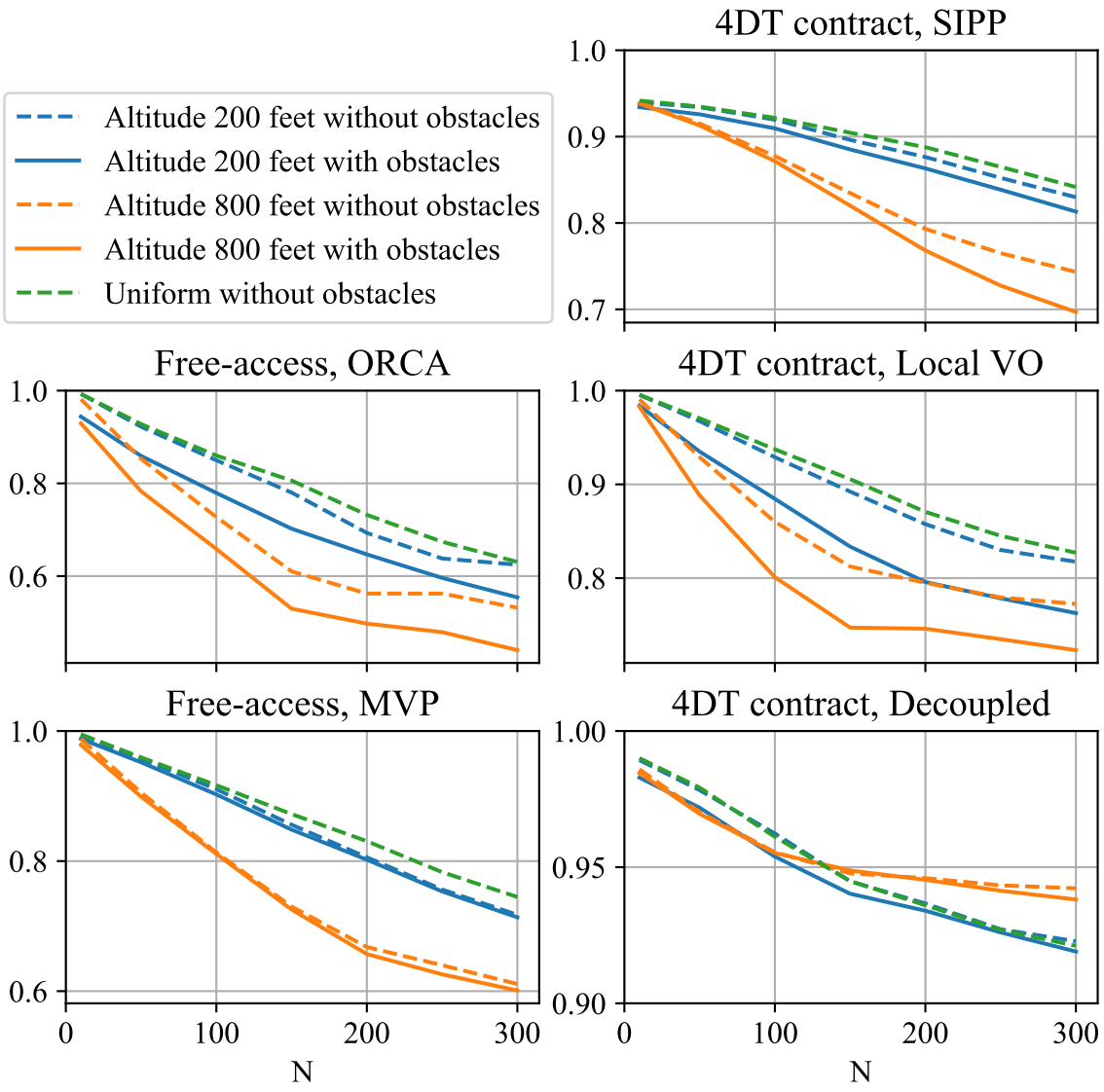


Figure 6.7: Evolution of the energy efficiency in function of agents density N for different architectures and traffic demand patterns

the efficiency. The efficiency metrics appear to be more sensitive than the throughput metric to the static obstacles. Static obstacles force agents to go around and take a longer path which decreases the efficiency. Figure 6.7 shows that for all methods the dashed line, representing the simulation without static obstacles, are above the solid lines, representing the simulations with static obstacles. However, once again, the magnitude of the change due to static obstacles is quite different depending on the method. The decoupled and MVP approach show a very small change while the other alternatives have a larger drop in

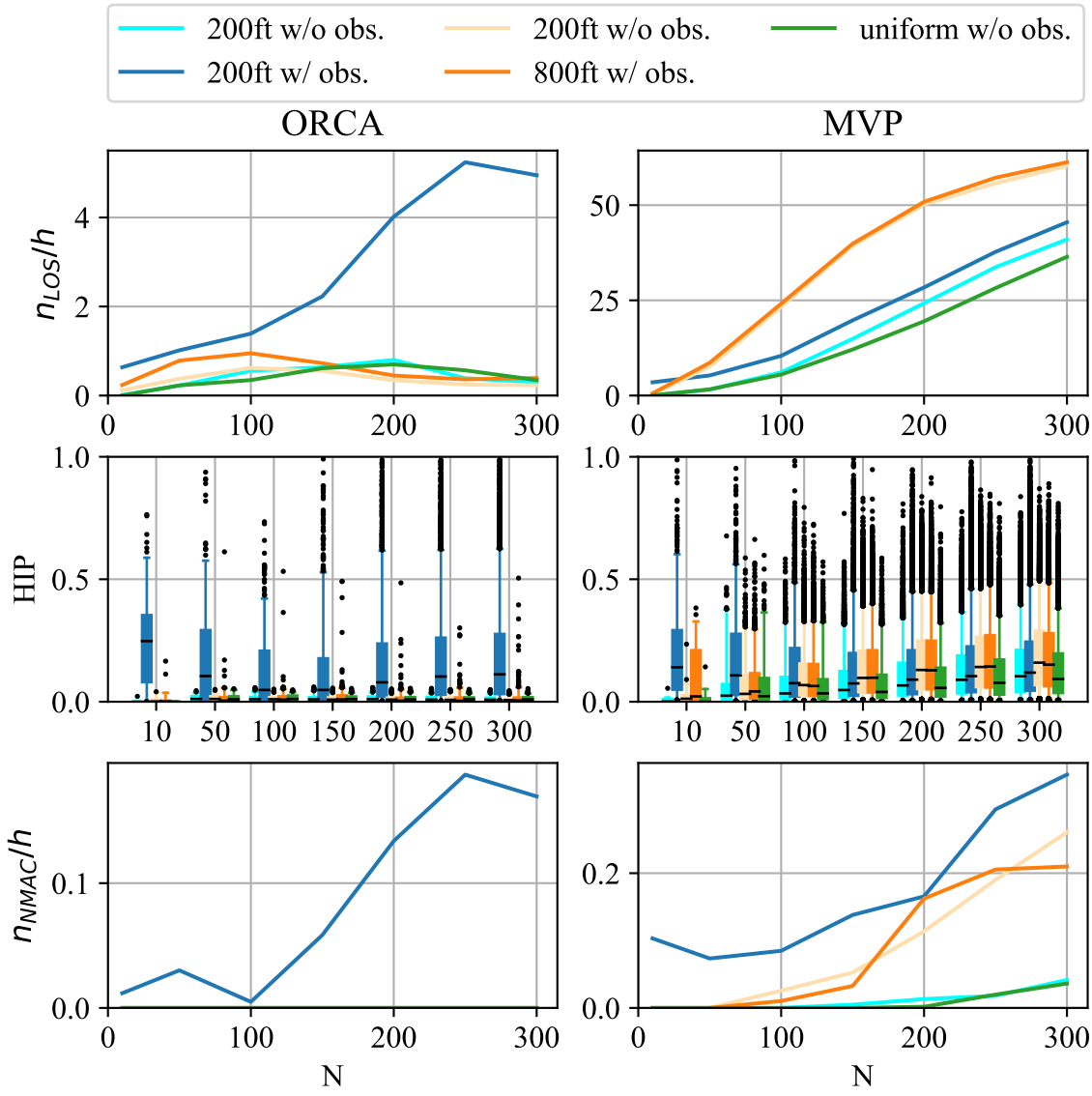


Figure 6.8: Evolution of average number of losses of separation per agent flight hour (top, shown on a log scale), HIP (middle), and number of severe losses of separation (bottom, shown on a log scale) in function of agents density N for different Free-access architectures

efficiency due to the static obstacles.

As before, the 4DT-contract architectures guarantee the absence of any LoS. The NMAC distance had to be redefined to work with static obstacles since they can have varying sizes and some obstacles' radii are smaller than the previously defined NMAC distance of 500 feet (152.4m). Since NMAC is used here to discuss the severity of a LoS, a conflict involving a static obstacle was considered a NMAC if the closest distance between the center of

the obstacle and the agent was smaller than a percentage of the minimum desired separation distance. The percentage was selected to match what was used for NMAC between two agents. Since the minimum desired separation distance between two agents is 500m, the percentage was set at $30.48\% = 152.4/500$. With this new definition, ORCA starts showing some NMAC when there are many small static obstacles (at 200 feet with obstacles). The number of LoS per flight hour for ORCA for the 800 feet altitude layer, exhibits a pattern similar to what was observed for the throughput. It peaks at 100 agents. For the MVP method, looking at the n_{LoS}/h shows that the decrease in the number of LZs is more challenging that the addition of static obstacles at 200 feet.

Simple Model using the impact on one Alternative

Similarly to what has been done in the other experiments, a simple model to evaluate the impact of static obstacles on the performance of each alternative was developed. As explained above static obstacles severely constrain the distribution of LZs which impacts the demand patterns. A conclusion of the previous experiment was that changes in demand had to be modeled in the agent-based simulation directly as there was an interaction between demand and autonomy algorithms that was not well-captured by a model that assumed a uniform impact between alternatives. Taking this into account, the simple model proposed here evaluates the change in performance between the scenario at a given altitude with obstacles against the scenario with the same distribution of LZ for take-off and landing but no obstacles. The change in performance is evaluated for the MVP algorithm, and this model is evaluated across all 5 alternatives. As shown in Table 6.4, and as could have been expected from the figures above, the resulting fit is poor. The MVP alternative is relatively insensitive in terms of throughput and efficiencies contrary to other alternatives like 4DT contract Local VO and free access ORCA.

Table 6.4: Goodness of fit (R^2) of models based on MVP change in performance with obstacles for 5 UTM alternatives at an altitude of 200 feet

Metric	4DT contract			Free-access	
	Decoupled	Local VO	SIPP	MVP	ORCA
Q	0.976	**0.417**	0.997	1.0	**0.38**
η_{time}	0.997	**0.794**	0.975	1.0	**0.849**
η_{energy}	0.993	**0.695**	0.971	1.0	**0.818**
n_{LoS}/h	N/A	N/A	N/A	1.0	**-1.523**
n_{NMAC}/h	N/A	N/A	N/A	**0.662**	**-1.43**

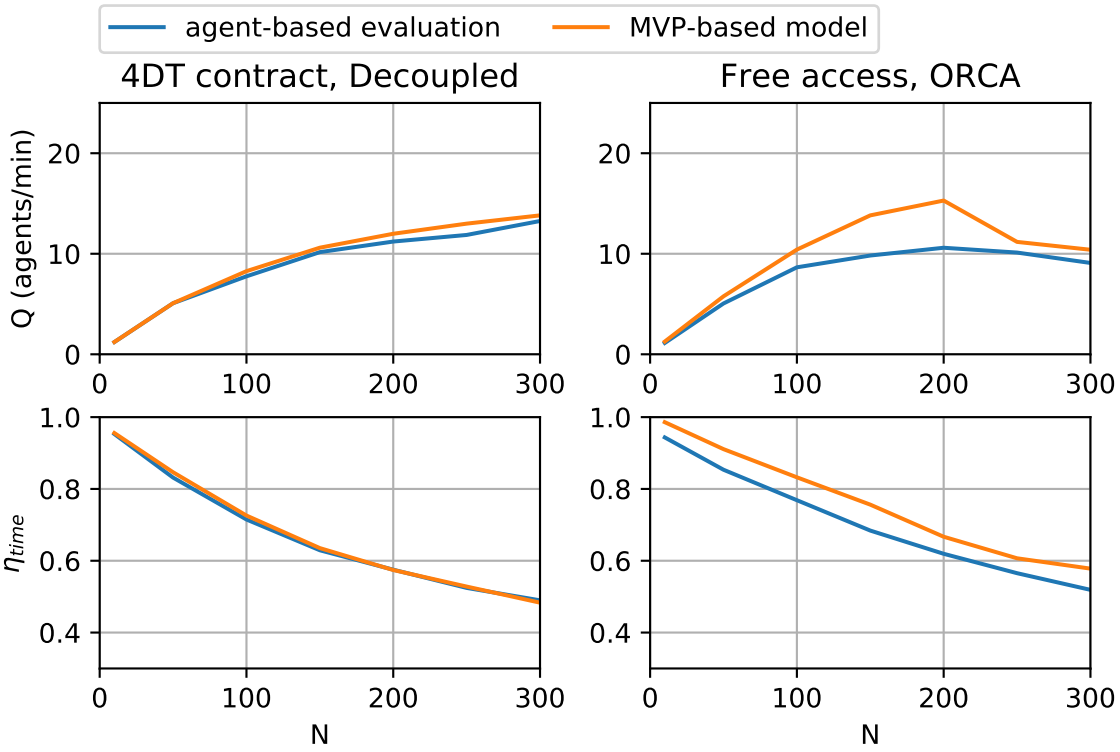


Figure 6.9: Comparison of the throughput (top) and time efficiency (bottom) 4DT Decoupled alternative (left) and free-access ORCA alternative (right) with static obstacles using the agent-based evaluation and using the simple MVP-based model in function of agent density N.

6.2.3 Conclusion

This experiment shows that static obstacles have an impact on performance of the strategy beyond the impact on demand. Moreover, this experiment has shown that depending on the strategy the impact is different. This has also highlighted that for some strategies, static obstacles present an additional challenge due to the presence of local minima. Although

these strategies can be improved through various heuristics, they are less robust and require additional work to obtain the desired behaviors.

To validate that static obstacles should be considered at the conceptual design stage, we show that for certain sets of weights the ranking of architectures obtained when static obstacles are only considered when defining demand is different than the ranking obtained when static obstacles are modeled in the analysis. Using the algorithm detailed in Appendix E, the following weights are identified: $W_D = [0.593, 0, 0, 0, -0.407]$. This results in the rankings shown in the decision tables Table 6.5 and Table 6.6.

Table 6.5: Decision Matrix at $N = 300$ and an altitude of 200 feet, static obstacles only impacting demand, ranking performed using the SAW methodology and the set of weight W_D

Attributes	Alternatives					Weights	Objective
	4DT contract			Free-access			
	Decoupled	LocalVO	SIPP	MVP	ORCA		
Q	14	18	31	25	10	0.593	Maximize
η_{time}	0.49	0.53	0.78	0.68	0.58	0	Maximize
η_{energy}	0.92	0.82	0.83	0.72	0.62	0	Maximize
n_{los}/h	0	0	0	41	0.3	0	Minimize
n_{nmac}/h	0	0	0	0.042	0	-0.407	Minimize
Scores	0.269	0.352	0.593	0.0673	0.203		
Rank	3	2	1	5	4		

Table 6.6: Decision Matrix at $N = 300$ and an altitude of 200 feet, static obstacles constraint active, ranking performed using the SAW methodology and the set of weight W_D

Attributes	Alternatives					Weights	Objective
	4DT contract			Free-access			
	Decoupled	LocalVO	SIPP	MVP	ORCA		
Q	13	13	29	24	9.1	0.593	Maximize
η_{time}	0.49	0.4	0.76	0.68	0.52	0	Maximize
η_{energy}	0.92	0.76	0.81	0.71	0.55	0	Maximize
n_{los}/h	0	0	0	46	4.9	0	Minimize
n_{nmac}/h	0	0	0	0.35	0.17	-0.407	Minimize
Scores	0.268	0.265	0.593	0.0842	-0.0159		
Rank	2	3	1	4	5		

Although the best architecture does not change, all other architecture rankings are affected. This goes to show that static obstacles must be considered when evaluating a strategy, which proves the hypothesis and answers the research question.

We can also remark that many small static obstacles are more challenging to handle for many of these algorithms than a large exclusion zone. Indeed, many small obstacles create the potential for local minima in which the agents can get trapped. Moreover by moving into relatively tight spaces, agents reduce the space available to maintain a safe distance to other agents. Performance of the free-access alternatives could probably be improved by adding a static preflight planning step that avoids challenging areas.

Conclusion 2.2

Static obstacles must be explicitly included as part of the evaluation of architectures since not including it might lead to choosing an architecture with worse performance than the one that would be chosen when only the non-uniform demand is considered.

6.3 Sensitivity to Priority Traffic

6.3.1 Goal and Setup

A good traffic management system should allow the system to be preempted to allow high priority flights, such as helicopter air ambulances, to proceed quickly.

Research Question 2.3

Should priority traffic be included at the conceptual design stage?

Hypothesis 2.3

If the ranking of architectures can change due to the change in performance when priority traffic is included, then priority traffic should be included at an early stage.

Priority traffic does not change significantly the operation of free-access alternatives since agents must already avoid other agents and static obstacles. However, 4DT contract alternatives must handle the disruption to their original flight plan. As detailed in Chapter 4, 4DT contract agents that are still on the ground will replan a full valid 4D trajectory while agents that were in the air will follow their flight plan until they arrive at a certain distance of the projected conflict, at which point they will switch to a reactive collision avoidance method. Since the ORCA collision avoidance main advantage is that it relies on collaborative avoidance but that in the case of a perturbed 4DT contract most agents will still be following their flight plans and not collaborate reactively, the MVP collision avoidance algorithm was selected for this experiment.

Priority agents were generated with start and end point outside of the simulation area to limit conflicts at take-off. These agents fly straight and at constant speed from their origin to their destination. The number of priority agents was set at 1 and 5. The simulation was controlled so that the number of priority agents in the simulation was constant. Each run was repeated 10 times similarly to what was done in other experiments. Regular agents were generated on the edge of the simulation area and the heading distribution was uniform. The baseline used for comparison of the results comes from experiment 1.1.

6.3.2 Results

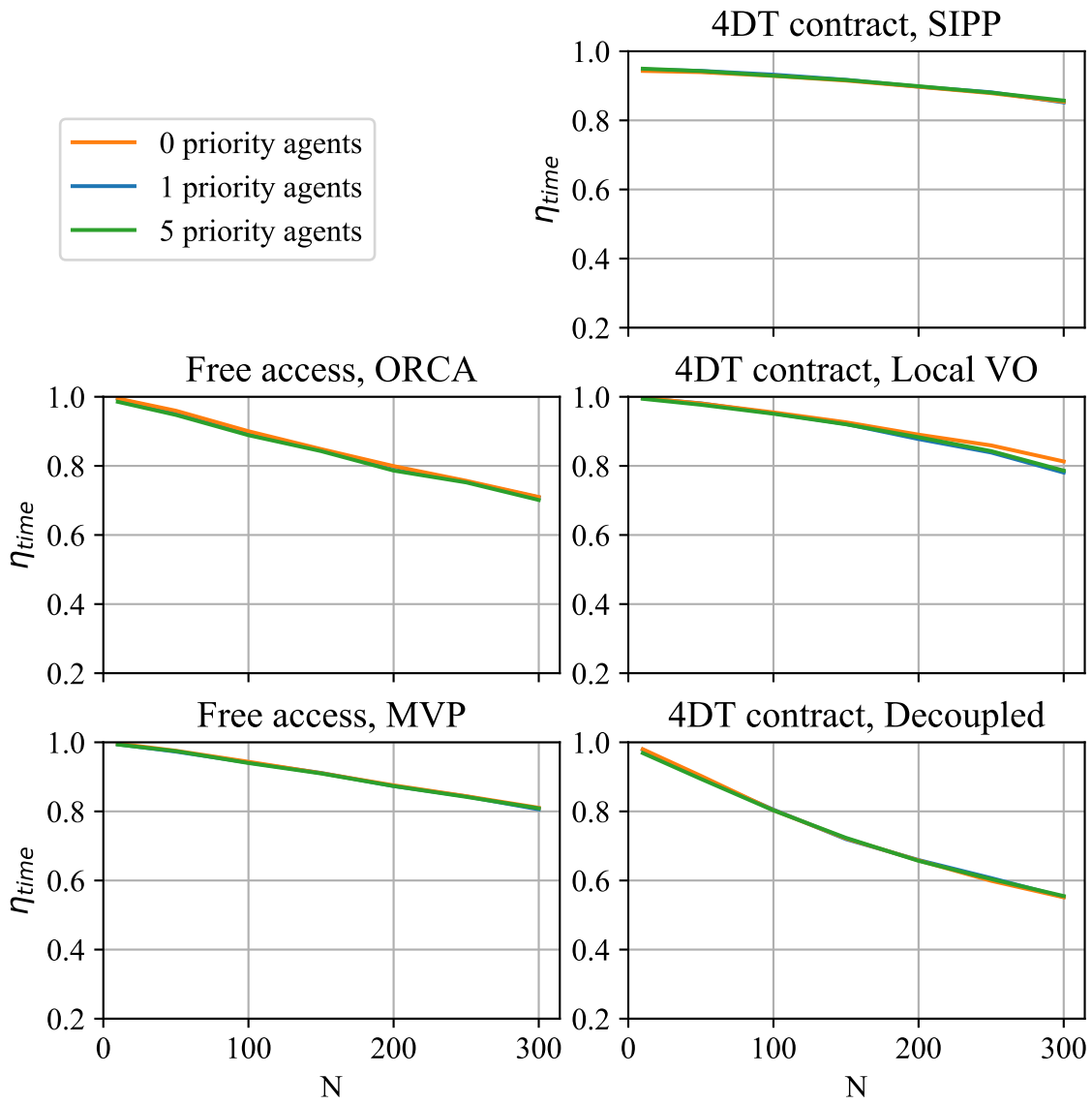


Figure 6.10: Evolution of average time efficiency in function of agents density N for different UTM alternatives with a varying number of priority traffic

As could have been expected, the addition of a few priority agents does not have a strong impact on the throughput or efficiency of any of the alternatives. This is illustrated in Figure 6.10, and the results are similar for throughput and energy efficiency.

Free-access alternatives have similar performance in terms of safety with or without priority agents. However, as shown in Figure 6.11, 4DT-contract alternatives are impacted

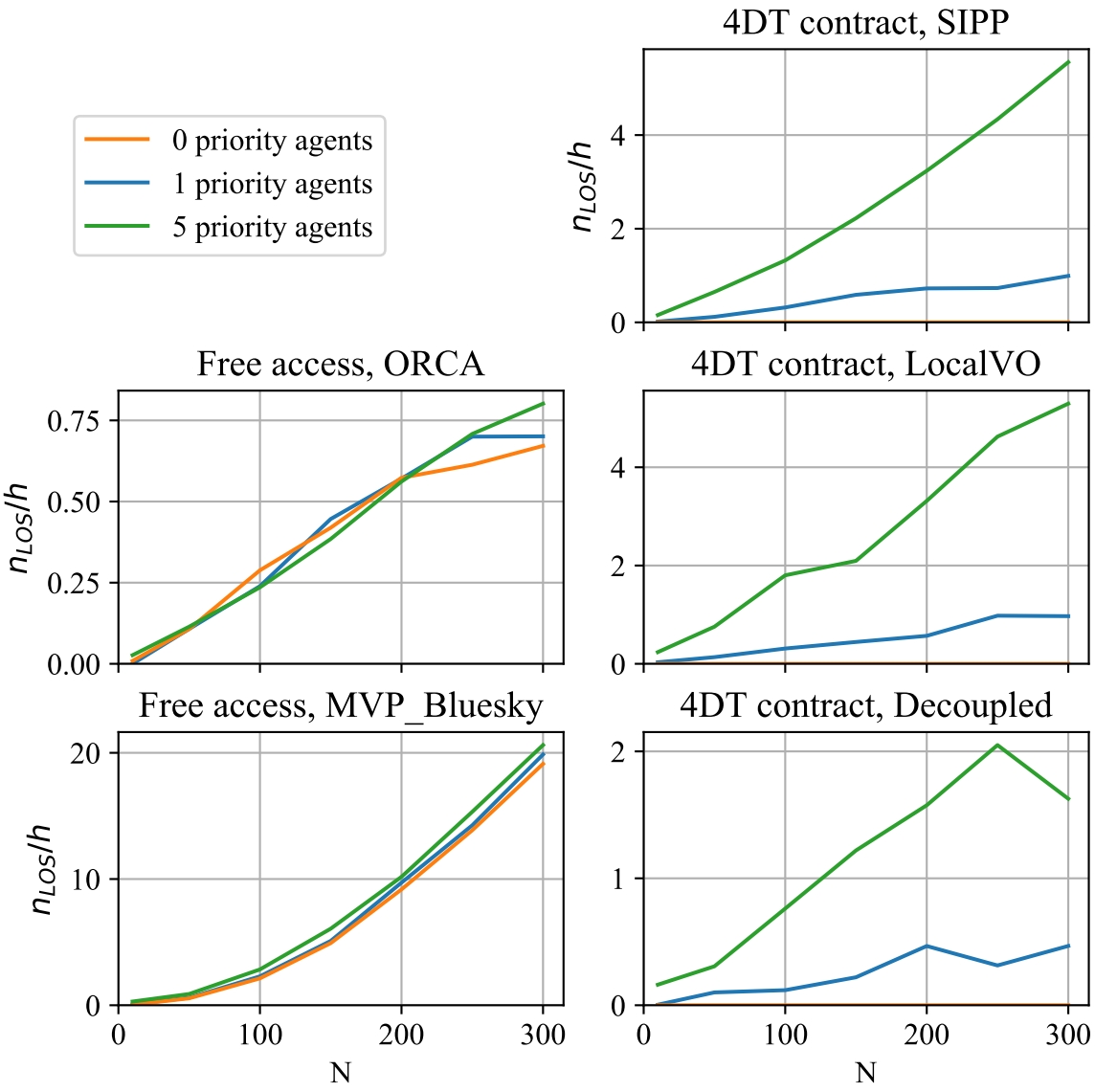


Figure 6.11: Evolution of average number of losses of separation per agent flight hour in function of agents density N for different UTM alternatives with a varying number of priority agents.

in terms of safety because they switch to reactive behaviors when a priority agent perturbs their plan while they are in the air. The number of losses of separation per flight hour increases with the number of priority agents included in the simulation. The Decoupled method exhibits the fewest LoS of all the 4DT-contract architectures methods. This is because, as had been observed in previous experiments, the decoupled method strongly limits the throughput and the total number of agents in the air. As a consequence, the

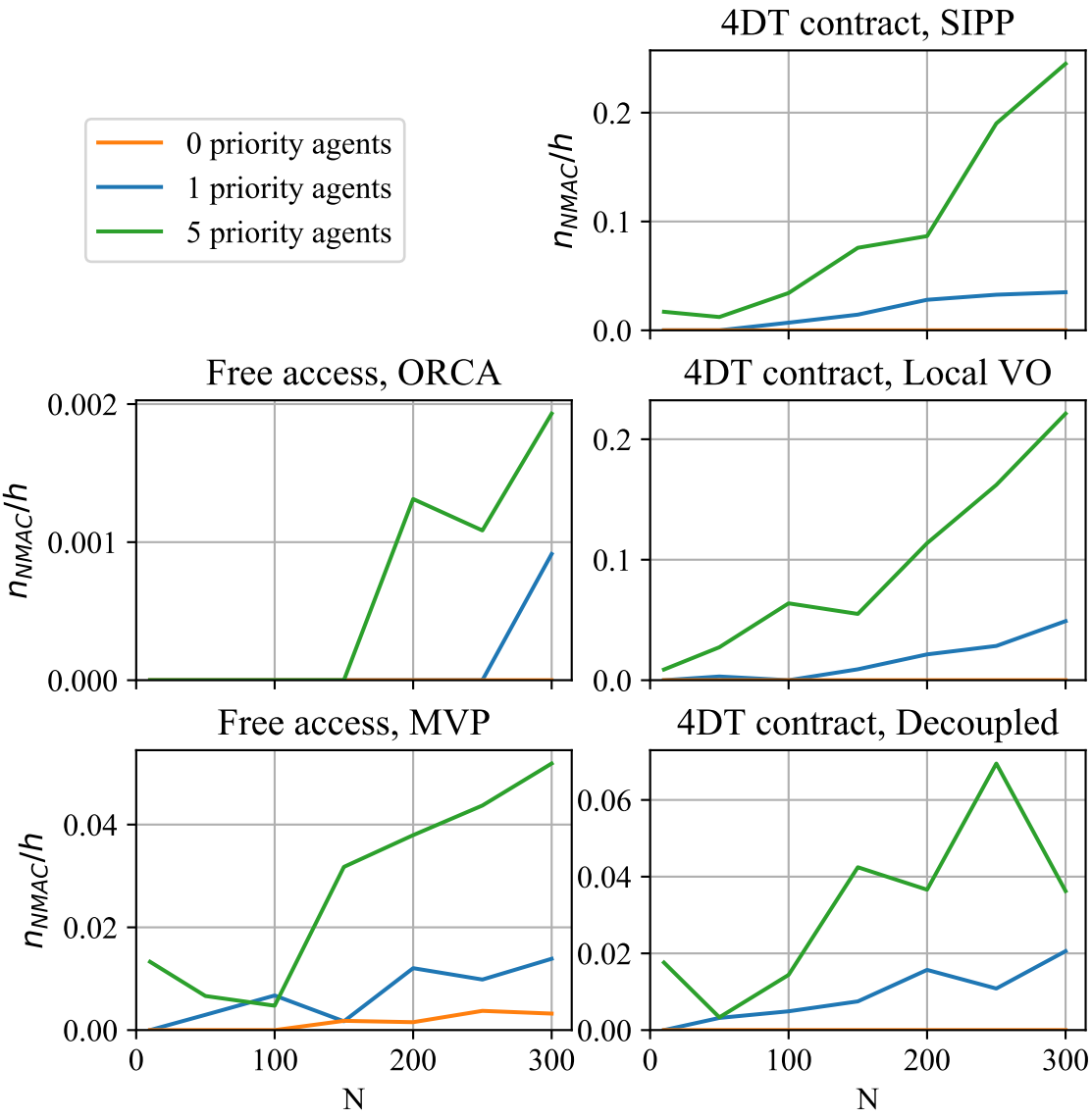


Figure 6.12: Evolution of average number of Near Mid Air Collisions per agent flight hour in function of agents density N for different UTM alternatives with a varying number of priority agents.

actual density of agents encountered by an agent in the decoupled alternative is lower than for other 4DT-contract architectures. As observed in Free-access architectures the number of LoS per flight hour increases with the number of agents.

As can be seen on Figure 6.12, the addition of priority traffic increases the likelihood of a severe loss of separation on all alternatives. Although the addition of priority traffic had very little impact on the number of LoS for both free access alternatives, the number of

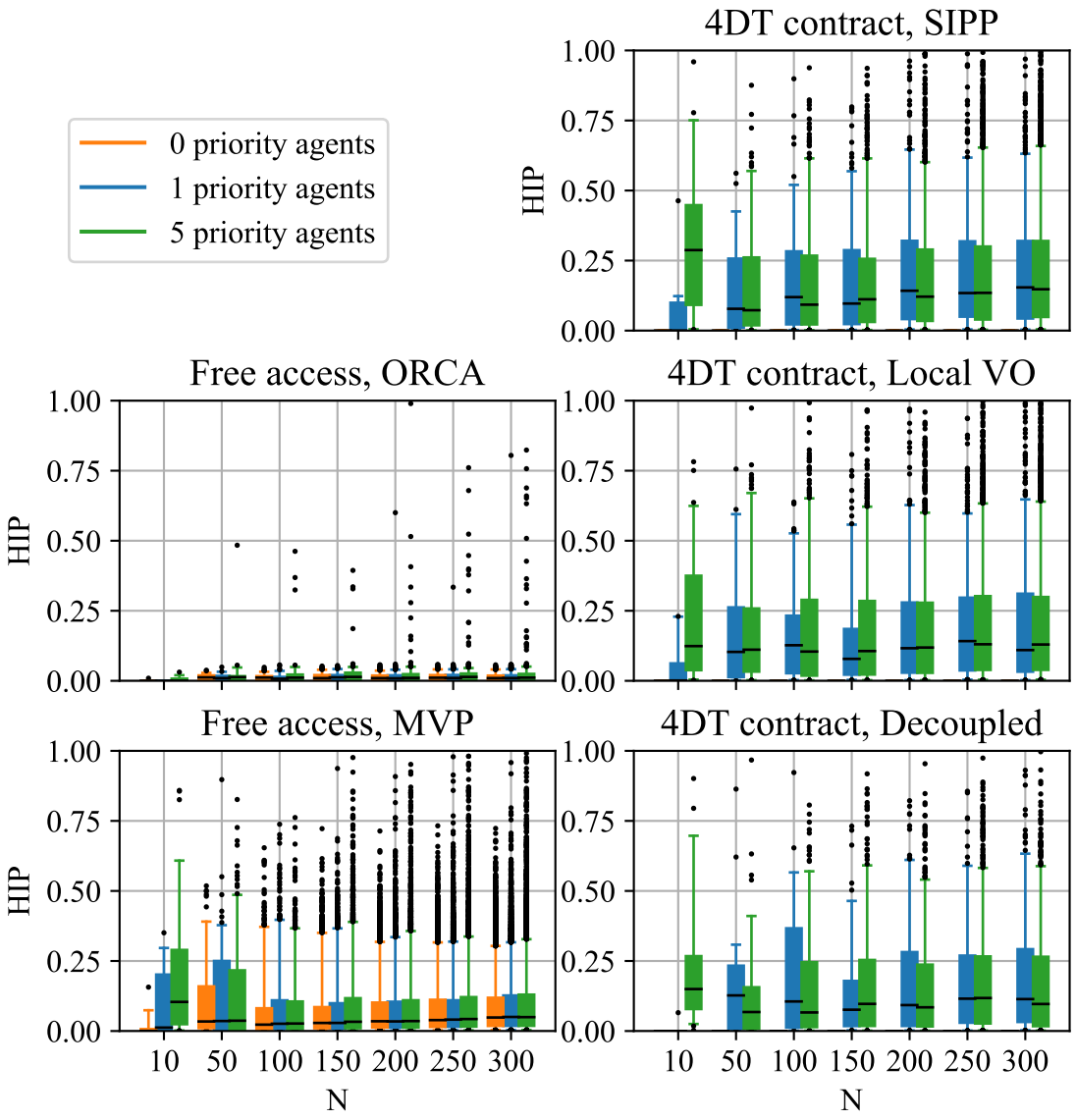


Figure 6.13: Distribution of LoS severity measured by the Horizontal Intrusion Parameter in function of agents density N for different UTM alternatives with a varying number of priority agents.

NMACs per flight hour increases with the number of priority agents. For the ORCA based free-access alternative, NMACs start to appear at medium-high density. At low density, the collision avoidance method successfully avoids severe losses of separation. The MVP-based alternative is less successful and the rate of severe LoS is roughly 10 times higher than when there is no priority traffic. Other alternatives perform worse than free-access MVP in terms of n_{NMAC}/h. The rate of NMACs per flight hour is higher by an order of

magnitude for 4DT contract alternatives using SIPP or LocalVO for preflight planning than the free-access MVP alternative. The difference in performance between the alternatives even though they all three rely on MVP for collision avoidance can be explained by the fact that in 4DT contract alternatives agents that have not been perturbed by a priority agent continue to follow their flight plan. The burden of avoiding collisions rests solely on the few perturbed agents. Even though MVP does not explicitly rely on the collaboration of other agents, it works poorly when other agents do not actively try to avoid the ownship. Hence, the fact that all the agents actively try to avoid each other (except for the priority agents) in free access provide an increased robustness when faced with perturbations compared to the rigid organization of 4DT contract alternatives. Once again, the difference between the 4DT Decoupled alternative and the two other 4DT contract alternatives in terms of safety can be explained by the fact that in the decoupled alternative the total number of agents in the air at a given time is limited, hence perturbed agents encounter a lower density, resulting in fewer severe losses of separation.

Figure 6.13 allows to visualize the distribution of LoS severity for the different alternatives. The number of priority agents, type of preflight planning algorithm, and number of agents do not impact very strongly the distribution. For all 4DT contract alternatives 75% of LoS have a HIP inferior to 0.25. The number of outliers that have a high HIP value increases with the number of agents and the number of priority agents. This is consistent with the increased n_{NMAC}/h observed in Figure 6.12. The MVP free-access alternative manages to keep most conflicts at a lower severity. This is due, as explained above, to the fact that in the free-access alternative all agents take part in the collision avoidance effort.

A k-factor model was not fitted on the results since it would obviously be a bad fit. Indeed, the number of losses of separation and near mid air collisions is 0 when there are no priority agent for 4DT-contract architectures. As a result a k-factor model would always predict 0 for these architectures.

6.3.3 Conclusion

The safety performance of 4DT contract alternatives degrades with the addition of unplanned priority agents, as could have been expected. The effect of priority agents is not uniform across alternatives and might be hard to predict before hand. For the free-access alternatives, although the number of loss of separation per flight hour remains relatively unchanged, the number of severe losses of separation as measured by n_{NMAC}/h increases significantly with the number of priority agents in the simulation. Due to the difference in logic, the performance of purely reactive alternatives (Free-access) cannot be extrapolated to predict the performance of perturbed 4DT-contract alternatives.

The decision table obtained when priority traffic is not considered is presented in Table 6.7, while the decision table obtained while adding 5 priority agents to the simulation is presented in Table 6.8. Due to the large change in safety performance that occur between the two analyses, there are many weights that result in a change of rankings. The decision tables presented here use $W_E = [0.5, 0, 0, 0, -0.5]$.

Table 6.7: Decision Matrix at $N = 300$ without priority traffic, ranking performed using the SAW methodology and the set of weight W_E

Attributes	Alternatives					Weights	Objective
	4DT contract			Free-access			
	Decoupled	LocalVO	SIPP	MVP	ORCA		
Q	12	22	23	22	18	0.5	Maximize
η_{time}	0.55	0.81	0.85	0.81	0.71	0	Maximize
η_{energy}	0.91	0.86	0.89	0.83	0.73	0	Maximize
n_{los}/h	0	0	0	19	0.67	0	Minimize
n_{nmac}/h	0	0	0	0.0032	0	-0.5	Minimize
Scores	0.265	0.468	0.5	-0.0364	0.39		
Rank	4	2	1	5	3		

Table 6.8: Decision Matrix at $N = 300$ with 5 priority agents, ranking performed using the SAW methodology and the set of weight W_E

Attributes	Alternatives					Weights	Objective
	4DT contract			Free-access			
	Decoupled	LocalVO	SIPP	MVP	ORCA		
Q	12	22	24	22	18	0.5	Maximize
η_{time}	0.55	0.79	0.86	0.81	0.7	0	Maximize
η_{energy}	0.91	0.87	0.9	0.82	0.72	0	Maximize
n_{los}/h	1.6	5.3	5.6	21	0.8	0	Minimize
n_{nmac}/h	0.036	0.22	0.24	0.052	0.0019	-0.5	Minimize
Scores	0.181	0.0109	0	0.35	0.374		
Rank	3	4	5	2	1		

The two analyses result in completely different rankings. This shows that the effect of perturbations due to priority agents is significant and requires to be modeled explicitly as part of the agent-based simulation.

Note that although the 4DT-contract strategies implemented here have been shown to perform poorly when priority traffic is added, this does not mean that all 4DT-contract architectures are inherently unable to handle perturbations. If an algorithm more robust than MVP was used when the agent needs to switch to a reactive collision avoidance method, performance would be greatly improved.

Conclusion 2.3

Priority traffic must be explicitly included as part of the evaluation of architectures since not including it might lead to choosing an architecture with worse performance than the one that would be chosen when it is considered.

6.4 Conclusion

These three experiments have shown that demand, static obstacles, and priority traffic should be explicitly modeled as part of the agent-based simulation early in the design process. The experiments have shown that there is an interaction between the alternatives and the external factors, and that the effect of one of these external factors cannot be accurately modeled without taking into account the collision avoidance or preflight planning method implemented in the alternative. Neglecting these external factors might lead to the selection of an architecture that would be sub-optimal when these factors are considered. A summary of the chapter is provided in Figure 6.14.

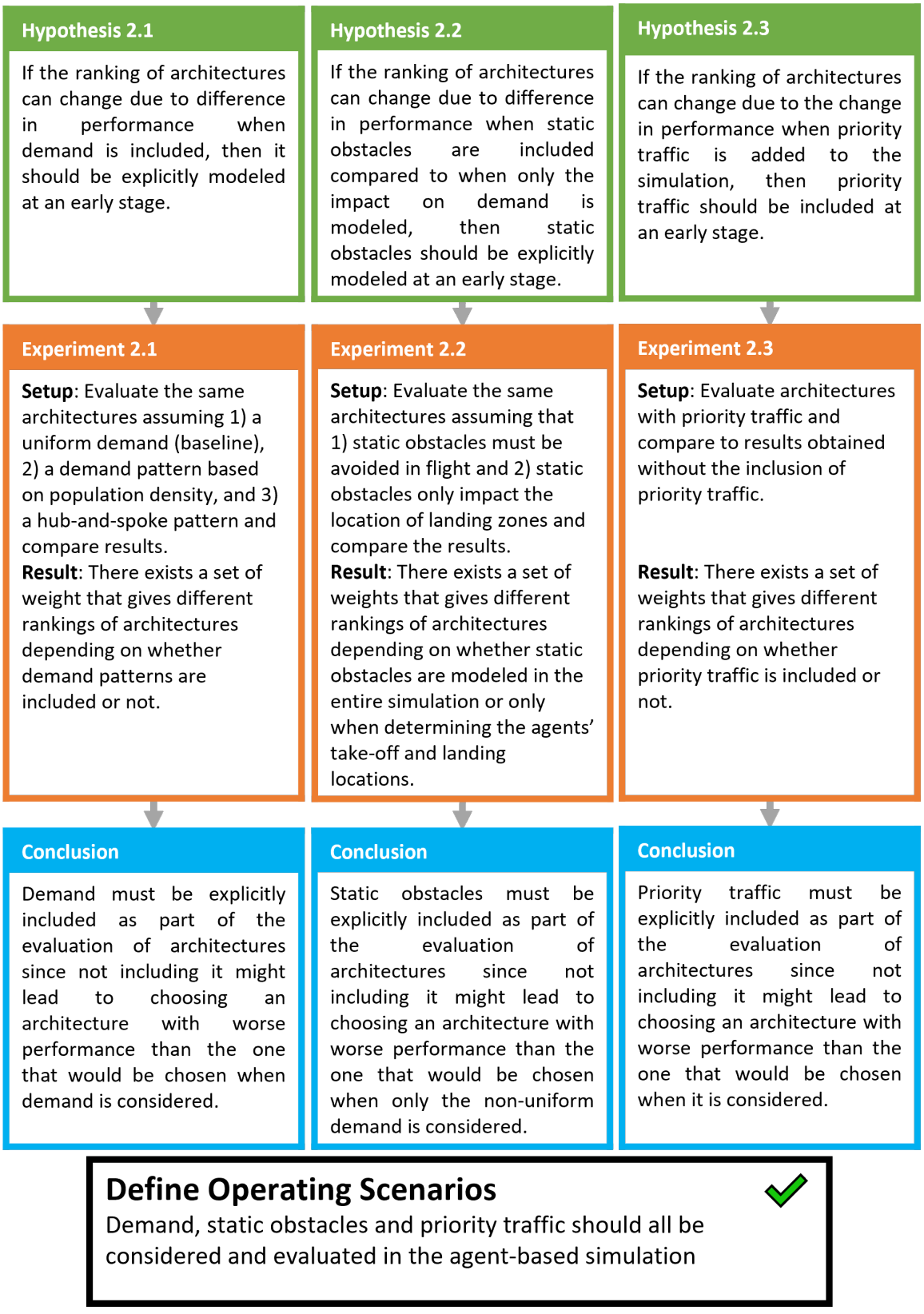


Figure 6.14: Summary of chapter 6

CHAPTER 7
VERIFICATION OF THE PROPOSED APPROACH

In the previous chapters, the research questions that were raised as part of the literature have been answered. In chapter 5, the importance of considering autonomy algorithms, whether for preflight planning or for collision avoidance, airspace structures, and access control rules when evaluating ConOps was shown. For airspace structures it was further shown that a simple conflict based model to estimate the impact of structures on the overall airspace did not capture all interactions properly and could lead to a different ranking of architectures, which might lead the designer to select a suboptimal architecture at the conceptual design stage. In chapter 6, the impact of external factors on different architectures was evaluated. Architectures were shown to have different sensitivities to external factors, and neglecting them could lead to a different ranking of architectures, illustrating why they must be included at an early stage of the design evaluation.

In this chapter, the proposed approach is compared to the baseline approach. This baseline approach represents the method by which a UTM architecture would have been conducted prior to the development of the proposed approach. Similarly to the other result chapters, we first recall the experimental setup, then provide an analysis of the results and finally draw conclusions.

7.1 Goal and Setup

The goal of this experiment is to verify the hypothesis that was formulated to answer the guiding research question of the thesis:

Overarching Research Question

How should UTM architectures be systematically and quantitatively assessed and compared at a conceptual design stage?

Overarching Hypothesis

If both the proposed and baseline approaches are used to select UTM architectures under the same assumptions and result in different rankings, then the oversights and simplifications introduced by the baseline approach would effectively lead to the selection of a worse architecture and the proposed approach should be used instead.

The hypothesis is tested under two different scenarios that are representative of UAM applications. The first scenario corresponds to an Air Taxi scenario, while the second is a package delivery scenario. The external factor selected for the two scenarios are presented in Table 7.1.

Table 7.1: External factors for the two scenarios considered

Scenario	Demand	Static Obstacles	Priority Traffic
Air Taxi	Population-density based	800 feet map	Yes (2)
Package Delivery	Hub-and-spoke	200 feet map	No

Helicopters usually operate above 500 feet altitude unless a lower altitude is required by the mission (aerial inspection for instance). Current regulations in the US for sUAS limit the altitude of the vehicles to 400 feet. This is why an altitude of 800 feet was selected for Air Taxi while an altitude of 200 feet was selected for the package delivery scenario. For Air Taxi, landing pads are assumed to be placed over the city in a square grid every 1 km and the demand for origin or destination is assumed to be proportional to the population density. For Package Delivery, package drop-offs zones are assumed to be placed over the city in a square grid every 1 km. The demand for packages is assumed to be uniform. There are four warehouses and customers are served by the nearest warehouse

exclusively. Drones are assumed to land at the package drop-off zone and request a new flight plan to go back to the warehouse (4DT contract) or wait until the airspace above them is clear to take-off (free access). Each leg of the flight is counted separately when measuring throughput. At an altitude of 800 feet, Air Taxis will not be the only vehicles to operate. As outlined in the NASA UTM ConOps, some traffic, such as Helicopter Air Ambulance (HAA) should be given priority over regular operations. At lower altitude, a ConOps that mandate participation in the UTM system can be imagined.

This thesis does not claim that the design of these scenarios and the results of the analysis are accurate, but they are useful in demonstrating why the proposed approach should be used when designing UTM architectures at the conceptual design stage.

For the baseline approach simplifications of the different UTM systems are made and external factors are not considered. Four different architectures are evaluated.

With the proposed approach, ten architectures are evaluated on these scenarios for $N = 300$ and each run is repeated ten times, resulting in 200 simulations. The simulations are 2.5D, and two layers are simulated. For free airspace architecture assignment to one layer is done randomly, while for layered airspace architectures assignment is done based on rules (layer A if the heading is between -180 and $0°$, layer B if between 0 and $180°$). The following section presents the performance of four architectures evaluated using the baseline analysis, followed by the performance of ten architectures evaluated using the proposed approach on both scenarios.

7.2 Results

7.2.1 Baseline

The baseline approach corresponds to an evaluation of UTM architectures that could have been conducted prior to the work developed in this thesis. The baseline fixes the algorithm used for preflight planning and collision avoidance, respectively a decoupled algorithm and MVP, and evaluates the impact of layers using a conflict count model. The baseline

approach neglects the effect of demand, static obstacles and priority traffic.

The decoupled 4DT-contract results from experiment 1.1, the MVP free-access results from experiment 1.2, and the conflict count model that was developed as part of experiment 1.3 are used to generate the baseline results. Since the scenarios assume 300 agents split on two layers, the results that were obtained for $N = 150$ are used and the throughput value is doubled. This yields the performance shown in Table 7.2.

Table 7.2: Performance of 4 UTM architectures evaluated using the baseline approach

Airspace Structure	Access Control	Preflight Planning	Collision Avoidance	Q	η_t	η_e	n_{LoS}/h	n_{NMAC}/h
Free	Free	None	MVP	25	0.91	0.92	4.9	0.0018
Layers	Free	None	MVP	26	0.93	0.94	3.3	0.00089
Free	4DT contract	Decoupled	None	18	0.72	0.95	0	0
Layers	4DT contract	Decoupled	None	21	0.77	0.96	0	0

7.2.2 Air Taxi

Table 7.3 presents the performance of the ten alternatives on the five metrics of interest in the Air Taxi scenario.

Table 7.3: Performance of ten UTM architectures in an Air Taxi scenario

Airspace Structure	Access Control	Preflight Planning	Collision Avoidance	Q	η_t	η_e	n_{LoS}/h	n_{NMAC}/h
Layers	4DT contract	Decoupled	MVP	25	0.54	0.96	0.034	0
Layers	4DT contract	Local VO	MVP	25	0.49	0.82	1.1	0
Layers	4DT contract	SIPP	MVP	44	0.69	0.83	3.2	0.014
Layers	Free	None	MVP	36	0.67	0.77	35	0.17
Layers	Free	None	ORCA	11	0.59	0.64	0.47	0.017
Free	4DT contract	Decoupled	MVP	20	0.48	0.96	0.049	0.0031
Free	4DT contract	Local VO	MVP	22	0.41	0.77	1.7	0.002
Free	4DT contract	SIPP	MVP	42	0.66	0.79	3.2	0.016
Free	Free	None	MVP	33	0.62	0.7	46	0.27
Free	Free	None	ORCA	8.8	0.57	0.62	0.42	0.0098

Restricting access to a layer based on heading improves performance of all architec-

tures.

7.2.3 Package Delivery

Table 7.4 presents the performance of the ten architectures on the five metrics of interest in the Package Delivery scenario.

Table 7.4: Performance of ten UTM architectures in a Drone Delivery scenario

Airspace Structure	Access Control	Preflight Planning	Collision Avoidance	Q	η_t	η_e	n_{LoS}/h	n_{NMAC}/h
Layers	4DT contract	Decoupled	None	26	0.63	1	0	0
Layers	4DT contract	Local VO	None	30	0.63	0.96	0	0
Layers	4DT contract	SIPP	None	37	0.67	0.95	0	0
Layers	Free	None	MVP	32	0.67	0.96	2.2	0.22
Layers	Free	None	ORCA	33	0.6	0.89	0.34	0.0056
Free	4DT contract	Decoupled	None	28	0.48	1	0	0
Free	4DT contract	Local VO	None	32	0.43	0.82	0	0
Free	4DT contract	SIPP	None	42	0.59	0.93	0	0
Free	Free	None	MVP	34	0.57	0.91	6.5	0.21
Free	Free	None	ORCA	31	0.47	0.79	0.36	0.038

In the package delivery case, layers negatively impact the throughput of most architectures even though they improve their average time and energy efficiencies. Although this appears to be counter-intuitive it can be explained by the fact that the distribution of efficiencies between free-airspace and layered alternatives are very different and affect agents differently in functio of their ideal travel time. A small change in the total time of travel will have a much larger effect on the efficiency of agents that have a small ideal travel time than on the efficiency of agents that have a large ideal travel time. Throughput however is not normalized and will behave the same either way.

7.3 Conclusion

If the SAW methodology is used to rank alternatives in the air taxi scenario with a set of preferences $W_E = [0.36, 0.36, 0.09, -0.09, -0.09]$ that puts a strong weight on throughput

and time efficiency and an equal weight on the other metrics, then the top five architectures and their scores are:

1. Layered 4DT-contract SIPP (0.80)
2. Free-airspace 4DT-contract SIPP (0.76)
3. Layered free-access MVP (0.60)
4. Layered 4DT-contract Decoupled (0.58)
5. Layered 4DT-contract Local VO (0.53)

If these same weights are used to rank the architectures evaluated using the baseline, the following ranking is obtained:

1. Layered Free-access (0.71)
2. Layered 4DT-contract (0.70)
3. Free-airspace 4DT-contract (0.62)
4. Free-airspace Free-access (0.61)

If only one architecture can be selected at the end of the conceptual design phase, the baseline approach would lead to the selection of a layered free-access architecture, when the detailed analysis conducted as part of the proposed approach shows that a layered 4DT-contract architecture would better satisfy the designer's preferences. According to the analysis conducted as part of the proposed approach, the Layered Free-Access alternative would result in a score that is 25% lower than what is achieved using a Layered 4DT-contract architecture.

If that same methodology is used to rank alternatives in the package delivery scenario with a set of weight that puts more emphasis on throughput and energy efficiency $W_F = [0.33, 0.11, 0.33, -0.11, -0.11]$ then the top five architectures identified by the proposed approach are:

1. Free-airspace 4DT-contract SIPP (0.74)
2. Layered 4DT-contract SIPP (0.72)
3. Layered 4DT-contract Local VO (0.66)
4. Layered free-access ORCA (0.65)
5. Layered 4DT-contract Decoupled (0.64)

With these same weights the following ranking is obtained with the baseline approach:

1. Layered 4DT-contract (0.70)
2. Free-airspace 4DT-contract (0.64)
3. Layered Free-access (0.64)
4. Free-airspace Free-access (0.52)

The baseline approach would lead to the selection of a layered approach, whereas the proposed approach shows that it might not be necessary to impose a structure for this scenario given the designer's preferences.

Obviously the weights that were chosen for these two scenarios are weights that highlight the difference in choices that result from the different approaches. If a set of weight that prioritized safety was selected in the air taxi scenario, then a 4DT contract architecture with layers would be selected, similar to what would be chosen when the full framework proposed here is used. However, the baseline would significantly underestimate the throughput and time efficiency achievable by these architectures, which might lead to incorrectly conclude that a UTM architecture cannot be both economically viable and safe.

Conclusion

The elements identified throughout the thesis play an important role in the selection of UTM architectures. Neglecting them leads to the selection of a different and worse-performing architecture. The proposed approach should therefore be followed to explicitly include in the decomposition and model them during evaluation.

A summary of the chapter is provided in Figure 7.1

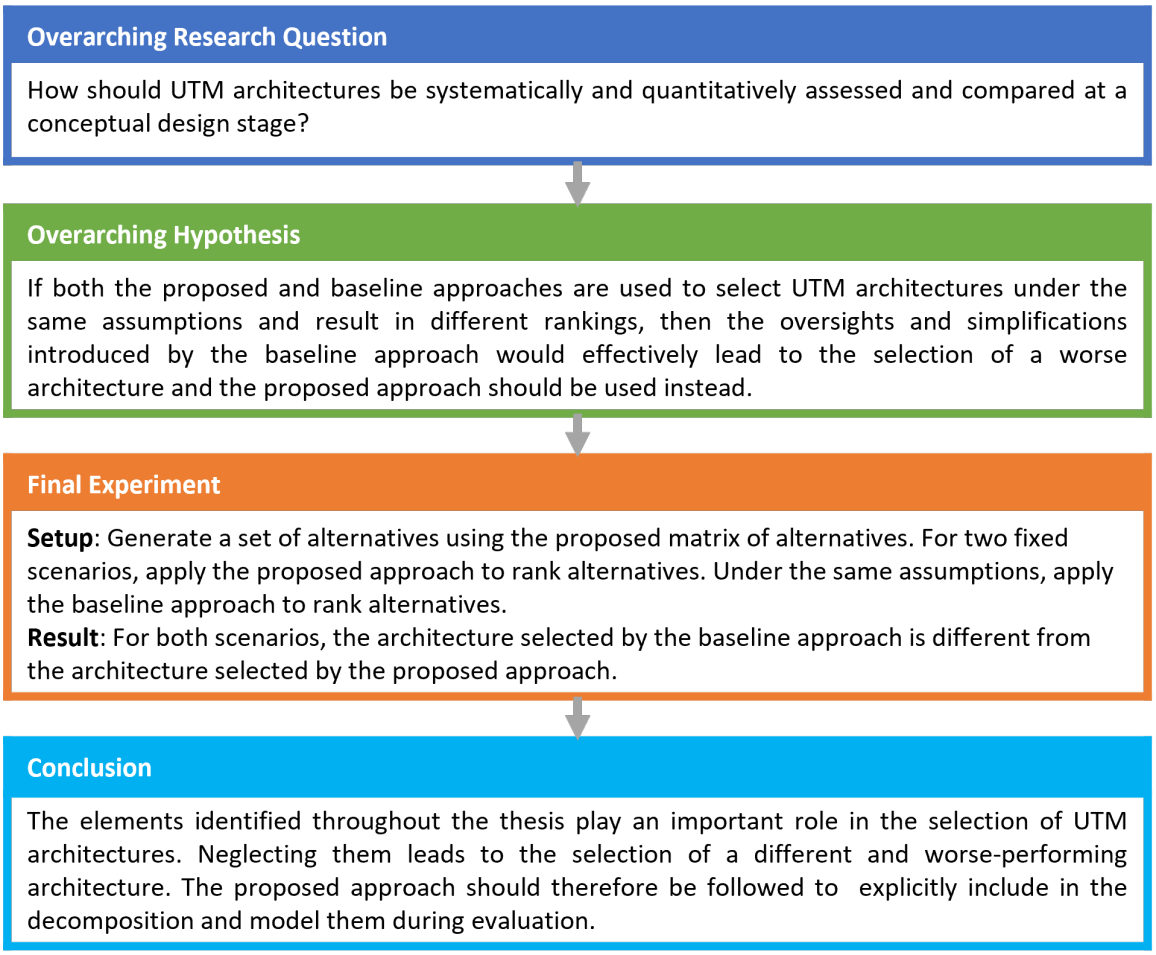


Figure 7.1: Summary of chapter 7

CHAPTER 8
CONCLUSION

This chapter first presents a rapid summary of the work that was conducted in this thesis. It then lists the main contributions. Finally, it proposes a few options to continue the work that has just been presented.

8.1 Summary

An analysis of the literature on UTM showed that there were many proposals and concepts of operations but that there was a lack of a common framework to compare and contrast the alternatives. Moreover, it was found that certain elements of the UTM were not evaluated in many of the studies that were found. Studies tended to focus either exclusively on the autonomous aspects, comparing algorithms for collision avoidance to each others, or on the structure or demands of the airspace. This thesis proposed a framework to bridge this gap by providing a tool in which different elements could be combined and evaluated. This work has looked at different subsystems and external factors and their effects on different metrics. It has shown that the choice of algorithm could not be decoupled from the airspace structure or external factors, as performance varied widely depending on the alternative selected. Moreover, it has shown that neglecting or making simplifications could lead to significantly change the architecture selection.

The work presented in this thesis was divided into two main research areas.

The first research area focused on evaluating the impact of different UTM subsystems on the overall performance of the airspace. However, before a comparison of free-access and 4DT-contract architectures could be developed, the need for better 4DT planning algorithms than the ones usually presented in the UTM literature was highlighted. A new algorithm based on velocity obstacles was introduced and compared to a variation on the

A* algorithm called SIPP, and to a decoupled approach similar to what can be found in the literature. This new algorithm was shown to perform well in terms of throughput and time efficiency when compared to the decoupled algorithm. With a large number of agents, this algorithm under performs SIPP slightly, but it is much more computationally efficient. The ranking of architectures using different access control methods was shown to be dependent on the algorithm used for collision avoidance and preflight planning. Making assumptions on the agent's autonomous behavior can have a large impact on the outcome of the decision-making process. Segregating agents to certain altitudes based on their heading was shown to improve performance of all studied UTM architectures, however the impact was shown to be very different depending on the architecture. This difference in sensitivities can cause a change in the rankings if the airspace structure is not modeled. Moreover, a model based on conflict count, which had been used as a measure of airspace complexity in the literature, was shown to not capture all the interactions and potentially also result in a different choice of architectures. As a conclusion of the work conducted in the first research area, it was determined that all four proposed subsystems (Airspace Structure, Access Control, Collision Avoidance, Preflight Planning) should be included when evaluating UTM architectures at the conceptual design stage.

The second research area evaluated the impact of the inclusion of several external factors on the performance and rankings of different UTM architectures. Architectures were shown to have different sensitivities to external factors, causing the inclusion of an external factor in the evaluation to potentially change the ranking obtained using the SAW method. The first external factor that was considered is demand using two different demand patterns: a population-density pattern and a hub-and-spoke pattern. Then, static obstacles at two different altitudes were considered and their effect beyond their impact on the demand were evaluated. Finally, priority emergency traffic was added to the simulation. A mitigation method for 4DT-contract architectures was implemented to allow agents to re-plan or switch to a reactive collision avoidance behavior when perturbed by an agent with a higher

priority. As a conclusion of the work conducted in the second research area it was determined that these three external factors should be considered when evaluating architectures at the conceptual design stage.

Finally, the proposed approach was validated against a baseline approach on two UAM scenarios. The baseline approach neglects the impact of external factors, models airspace structures using a conflict count model, and fixes the algorithm used for preflight planning and collision avoidance. In both scenarios using the baseline approach to select an architectures would lead to the selection of an architecture that does not perform as well as when the selection is performed using the detailed proposed approach.

8.2 Contributions

The main contribution of this work is the development of an approach to systematically generate and evaluate UTM architectures under flexible assumptions. The impact of different elements on the overall architecture performance and decision-making outcomes were systematically evaluated. The UTM system of systems was shown to exhibit strong coupling between the subsystems and external factors. The importance of considering these elements early in the design was highlighted in multiple experiments. This work is useful to UTM designers to understand which subsystems and external factors must be included at a conceptual design stage. If a simulation or tool neglects the subsystems and factors presented in this thesis, the designer will be aware that the ranking of alternatives might be impacted. Moreover, to evaluate whether a different subsystem or external factor should be included in the evaluation, the UTM designer can follow the same approach that was followed in the experiment to evaluate the sensitivity of the metrics and architecture and the impact on the decision-making outcome. This is assuming the UTM designer has access to a flexible simulator that can model the new external factor or subsystem.

Additional contributions were made in the 4D trajectory planning domain. The Velocity Obstacle problem was formulated as a MIQCP problem and used as a component of a local

4DT planning method. This local approach has a low complexity and yields good performance when compared to global methods or decoupled approaches. The Safe Interval Path Planning was adapted from the robotics literature to a UTM context. These two methods can be used for 4DT planning in the UTM context.

An agent-based simulation was developed to enable the experiments that had been designed. This tool and the implementation of all the algorithms developed for the thesis are made freely available under a MIT license [1].

Finally, while validating the hypotheses that were formulated in the research framing section, a lot of insights were gained on the behaviors of UTM systems. Some interesting results are listed here. A local planner performs similarly to a global planner for 4DT-contract architectures at medium-high density of agents. Segregating the airspace by layer based on agents' heading reduces the number of conflicts, but in a controlled system the impact is limited. Some metrics are more sensitive than others, for instance throughput was less sensitive to the structure changes than the efficiency metrics. The average HIP value is relatively insensitive to the density and is generally not very helpful in gaining insights on the system. The peak in throughput for a given density was only observed using the ORCA collision avoidance method.

The work conducted as part of the research area 1 was published at the Scitech conference in January 2021 [98].

8.3 Future Work

There is a lot more research to be done in the UTM domain, and the field is quickly evolving. A few potential avenues to expand the work that was presented in this thesis are presented here. Note that due to the fast evolution of the domain, the work presented here represents the state of the UTM domain as of August 2021, assumptions made in the experiments and the tools might become obsolete in the next few years.

[1] https://github.com/colineRamee/UTM_simulator.git

Although this research has shown conclusively the need for including airspace structure, access control rules, preflight planning, and collision avoidance at an early stage in the design of a UTM system, it does not mean that other subsystems should not be included as well. Similarly, demand, static obstacles, and priority traffic were studied and shown to impact the architecture selection, but there might be other external factors that should be included. A few elements that would be interesting to evaluate following the same approach that was used throughout the experiments are listed here.

- Weather: different types of weather factors will impact performance. Convective weather could be represented as non-participating obstacles that must be avoided. Wind will impact the preflight planning as agents should try to find the best path to their destination given a wind field. Wind will also introduce uncertainty in all agents velocities, assumptions or models of the control laws' performance will be required.

- Vehicle: vehicle kinematics and autonomy performance would be interesting to model. The simulator assumed that vehicles' could change velocity instantaneously, a more realistic model with bounded acceleration and turn rates would introduce complexity for 4DT-contract planners but would be closer to actual vehicle limitations. Coupled with perturbations from the weather, this introduces noise in the vehicles' position and velocity. Vehicles' autonomy performance might be limited by sensor/communication performance. Sensors are noisy and depending on how well the autonomy account for these uncertainties the performance will vary. Uncertainties might be handled by padding the minimum separation distance but this will reduce the capacity of the airspace. Vehicles were considered to be homogeneous in the simulation, they all had the same performance. Looking at how heterogeneous agents behave would be an interesting research direction. Moreover, all vehicles were considered to be hover capable. Adding conventional take-off and landing (CTOLs) vehicles and short take-off and landing (STOLs) to the simulation would be interesting.

- Altitude changes: most of the experiments were conducted in 2D. The validation experiment was conducted in 2.5D but interactions between altitude levels during climb and descents were ignored. Measuring the impact of this simplification would be interesting. Extending the simulator to handle altitude changes would also allow to test out other preflight planning and collision avoidance methods. The number of altitude levels that could be used by UAM traffic is not yet clearly defined so some assumptions would be required.

- Noise: an important metric that was not considered in this work is noise. Minimizing noise for the population living in an area where UAM vehicles operates will be critical for the system's acceptability. This will be addressed in part by the vehicles' design but might also impact preflight planning. Indeed, operators might try to avoid certain densely populated areas or limit the number of flights per hour over a given area to limit disturbances.

- Integration in the NAS: in the study presented here the only non-UAM operations that were considered were emergency flights flying straight through the area. In reality UAM might have to share the airspace with other NAS users. At low altitude, operations in LAANC airspace was considered to be always allowable, and at higher altitude, operations inside controlled airspace was considered to be always forbidden. In reality access to these different airspace will be granted based on usage from other NAS users, and based on controller workload.

This thesis made a number of approximations and simplifications. With access to a more advanced simulator it would be interesting to quantify the errors induced by these approximations and whether they would be significant at the preliminary design stage. This would allow to validate the simulator that was developed for this thesis. As it stands, the simulation that was developed here is useful to gain an understanding of the system but the numerical results obtained are very preliminary and are associated with a high uncertainty.

As information about the future vehicles and regulations for UAM becomes available, this work could be refined to provide more directly applicable results.

In this research, a single strategy for 4DT contract in the presence of priority traffic was implemented, and it was shown to fail to avoid near mid air collisions at high density. Different strategies could be tested to make a 4DT contract more robust to uncertainty. For instance, an agent could plan to fly only at 80% of its maximum speed and when perturbed it would have some method to minimize its deviation to the flight plan. Another approach would be to increase the size of the reserved volume around the agent, if the agent is lightly perturbed it would still remain in its protected area. Both these methods would come at a cost in efficiency and throughput, which makes it an interesting trade-off for the designer.

Similarly to what has been done in the literature, different collision avoidance methods beyond the two that were implemented in the thesis could be tested. Decentralized strategies were chosen, but it would be interesting to investigate collaborative algorithms. The tool developed here should allow such comparative studies easily.

In a first stage of the development of UAM it is likely that there will only be few LZs per urban areas. As was shown in the hub-and-spoke experiment, the bottleneck to performance is then likely to come from the approach and departure of those sites. Further studies are required on how to organize traffic at the LZ scale for both small package delivery UAVs and larger passenger-carrying eVTOLs.

Appendices

APPENDIX A

TIME OF CLOSEST APPROACH AND TIME TO LEAVE

For several algorithms as well as for collision checking it is necessary to find the time between two agents assuming a constant velocity. The formulas are derived here. For the safe interval path planning algorithm it is moreover necessary to find the smallest delay, if it exists, that allows the ownship to move from its current position to the next point on the grid without losing separation with surrounding intruders.

A.1 Time of Closest Approach

Let's consider two agents A and B, whose positions are respectively $\mathbf{P_A}(t_0)$ and $\mathbf{P_B}(t_0)$ at time t_0, and velocities are the constant $\mathbf{V_A}$ and $\mathbf{V_B}$. We want to find the time t^* at which the distance between the two agents is minimum.

We define $\mathbf{R}_{AB}(t)$ the relative position of agent B with respect to A at time t, and write the distance between the two agents at time t as $r_{AB}(t)$. By definition:

$$\begin{aligned}
\mathbf{R}_{AB}(t) &= \mathbf{P_B}(t_0) + (t - t_0) * \mathbf{V_B} - \mathbf{P_A}(t_0) - (t - t_0) * \mathbf{V_A} \\
r_{AB}(t) &= ||\mathbf{R}_{AB}(t)|| \\
r_{AB}(t) &= \sqrt{\mathbf{R}_{AB}^T.\mathbf{R}_{AB}}
\end{aligned}$$

Solving for $\min_t(r_{AB})$ is equivalent to solving for $2\frac{d\vec{R}_{AB}}{dt}^T.\vec{R}_{AB} = 0$ and hence:

$$t^* = t_0 - \frac{(\mathbf{V_B} - \mathbf{V_A})^T.(\mathbf{P_B}(t_0) - \mathbf{P_A}(t_0))}{\|\mathbf{V_B} - \mathbf{V_A}\|^2} \tag{A.1}$$

If $\mathbf{V_A} = \mathbf{V_B}$ the distance between the agents is constant and t^* is set equal to t_0.

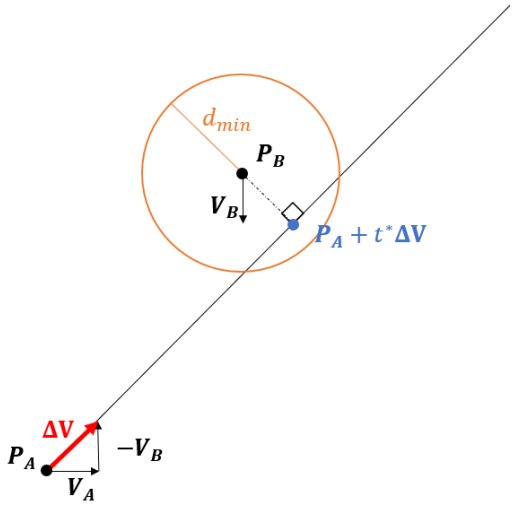


Figure A.1: Finding the distance of closest approach by reasoning geometrically in the intruder reference frame

As illustrated on Figure Figure A.1 the distance of closest approach can be found graphically and equation Equation A.1 interpreted as the projection of the relative position on the relative velocity vector. To interpret this figure, similarly to what is done when building a Velocity Obstacle set, we reason in the frame of the intruder, i.e. the velocity of the intruder is subtracted to the ownship and the intruder's position is considered constant.

The equations presented above assume that the agents' velocity are constant. In most of the applications considered in this thesis this is not the case. The agents are added and removed from the simulation at different times, and they do not always travel in a straight line at constant speed. Their velocity however is considered constant on a given trajectory segment. Hence, the analyses for closest time of approach must be conducted piecewise. Considering a time segment $[t_1, t_2]$ such that the agents' velocity is constant during that interval, clamping t^* will yield the time on the interval when the two agents are closest to each other. Indeed, $r_{AB}(t)$ is a quadratic equation of t, with the factor of the squared term being positive (i.e. the distance increases as time goes toward $+/-\infty$). As a result either t^* is on the interval, t^* occurs before the interval starts and the agents are now moving away from each other, or t^* occurs after the interval and the agents are getting closer to

each other.

$$t^*_{([t_1,t_2]} = \max(t_1, \min(t_2, t^*))$$

A.2 Time to Leave

Assuming agent A and agent B will lose separation if agent A leaves its current position at a velocity $\mathbf{V_A} \neq 0$, we want to find a delay t' if it exists for which the minimum distance between the two agents is exactly the minimum separation distance.

$$r_{AB}(t^*) = d_{min}$$

The position of agent A at time t can be expressed as:

$$\mathbf{P_A}(t) = \begin{cases} \mathbf{P_A}(t_0) + (t - t')\mathbf{V_A}, & \text{if } t > t' \\ \mathbf{P_A}(t_0), & \text{otherwise} \end{cases}$$

Let's define $\Delta\mathbf{P} = \mathbf{P_B}(t_0) - \mathbf{P_A}(t_0)$ and $\Delta\mathbf{V} = \mathbf{V_B} - \mathbf{V_A}$
Assuming $r_{AB}(t^*(t' = 0)) < d_{min}$ and $t' < t^*$, the formula for t^* as a function of t' can be derived similarly to what was done in the previous section.

$$r_{AB}(t, t') = \|\Delta\mathbf{P} + \Delta\mathbf{V} \cdot t + \mathbf{V_A} \cdot t'\| \tag{A.2}$$

$$t^*(t') = t_0 - \frac{\Delta\mathbf{V} \bullet (\Delta\mathbf{P} + \mathbf{V_A} \cdot t')}{\|\Delta\mathbf{V}\|^2} \tag{A.3}$$

We can plug back equation Equation A.3 into equation Equation A.2 to find the minimum distance between the agents as a function of the delay.

$$
\begin{aligned}
&r_{AB,min}(t') = r_{AB}(t^*(t')) \\
&r_{AB,min}(t') = \left\| \Delta\mathbf{P} + \Delta\mathbf{V} \cdot (t_0 - \frac{\Delta\mathbf{V} \bullet (\Delta\mathbf{P} + \mathbf{V_A} \cdot t')}{\|\Delta\mathbf{V}\|^2}) + \mathbf{V_A} \cdot t' \right\|
\end{aligned} \tag{A.4}
$$

We want to find t' such that $r_{AB,min}(t') = d_{min}$, since $d_{min} > 0$ and $r_{AB,min} \geq 0$ this is equivalent to solving the quadratic equation in t': $r_{AB,min}(t')^2 = d_{min}^2$.
To simplify the expression we introduce the following terms:

$$
\begin{aligned}
\mathbf{\Gamma} &= \Delta\mathbf{P} + \Delta\mathbf{V} \cdot (t_0 - \frac{\Delta\mathbf{V} \bullet \Delta\mathbf{P}}{\|\Delta\mathbf{V}\|^2}) \\
\mathbf{\Lambda} &= \Delta\mathbf{V} \cdot (-\frac{\Delta\mathbf{V} \bullet \mathbf{V_A}}{\|\Delta\mathbf{V}\|^2}) + \mathbf{V_A}
\end{aligned}
$$

Hence equation Equation A.4 becomes:

$$
r_{AB,min}(t') = \|\mathbf{\Gamma} + \mathbf{\Lambda} \cdot t'\|
$$

And so the quadratic equation to be solved is:

$$
d_{min}^2 = \|\mathbf{\Gamma}\|^2 + 2\mathbf{\Lambda} \bullet \mathbf{\Gamma} \cdot t' + \|\mathbf{\Lambda}\|^2 t'^2
$$

The determinant of the quadratic equation is:

$$
\Delta = (\mathbf{\Lambda} \bullet \mathbf{\Gamma})^2 - (\|\mathbf{\Gamma}\|^2 - d_{min}^2) \cdot \|\mathbf{\Lambda}\|^2
$$

If $\Delta \geq 0$ the possible delays are:

$$t'_a = \frac{-\mathbf{\Lambda} \bullet \mathbf{\Gamma} - \sqrt{\Delta}}{\mathbf{\Lambda}}$$
$$t'_b = \frac{-\mathbf{\Lambda} \bullet \mathbf{\Gamma} + \sqrt{\Delta}}{\mathbf{\Lambda}}$$

This can be interpreted geometrically similarly to what was done in the previous section. This helps understand the different domains of the solution. The minimum distance between the agents can either occur while agent A is moving or while agent A is waiting. Figure Figure A.2 shows how the position at which the minimum distance occurs changes with the delay. Since the figure is shown in agent B reference frame, when agent A is waiting it moves along $-\mathbf{V_B}$, when agent A's delay has passed it moves along $\Delta\mathbf{V}$. So to find the time to leave on the figure simply find the point where $\Delta\mathbf{V}$ is tangent to the minimum separation circle, the intersection between the tangent and the waiting line $-\mathbf{V_B}$ gives the position where agent A has to leave $\mathbf{P_A} - \mathbf{V_B}t'$, where t' is the time to leave. The blue circle shows the minimum distance if the ownship waits indefinitely $d_{min,wait}$. The two agents are at this distance at time $t_{min,wait}$. As can be seen on the figure, if agent A leaves at that time or shortly after, the minimum distance does not occur at $t_{min,wait}$ but on the segment where agent A is moving. The purple dot and its associated delay t'_2 shows when the minimum distance when not moving is equal to the minimum distance while moving with a delay. For any delay $t <= t'_2$ the minimum distance occurs when the agent is moving, for any delay $t > t'_2$ the minimum distance occurs at $t_{min,wait}$ when the agent is waiting.

As shown on Figure Figure A.3 if $d_{min,wait}$ is smaller than the desired separation distance d_{min} then there is no solution for the time to leave because $t' > t'_2$. With a delay of t' the minimum distance is $d_{min,wait}$.

Since we perform this analysis when there is a loss of separation at the time of closest approach without delay and no current loss of separation, we can see graphically that t'_a is

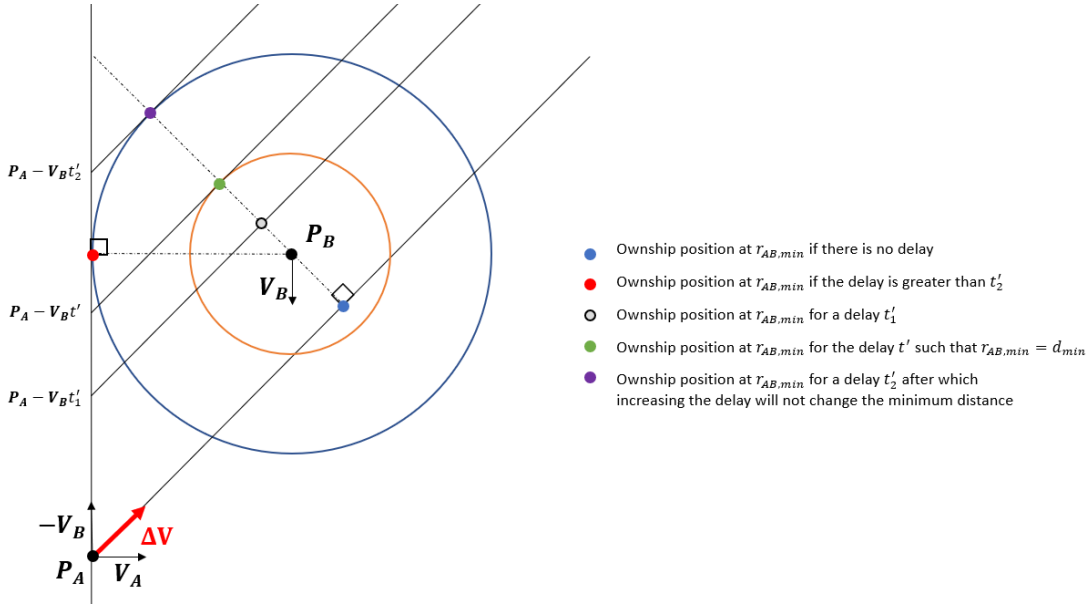


Figure A.2: Finding the distance of closest approach for different delays by reasoning geometrically in the intruder reference frame

negative. Since a negative delay is not a valid solution in our case it is ignored. Moreover, it can be seen that if $d_{min,wait} < d_{min}$ there can be no solution as $t_2' < t'$

Obviously there is no solution if the current distance between the two agents is less than d_{min}. The solution is degenerate if $\mathbf{V_B} = -\mathbf{V_A}$. Indeed, in that case $\Delta\mathbf{V}$ is collinear with $-\mathbf{V_B}$, hence the minimum distance does not change with delays. The minimum distance is also fixed if $\mathbf{V_B}$ is null.

If agent A is on the ground or at a different altitude while it waits, the minimum separation while waiting does not matter. The solution is given by the latest intersection between the $\mathbf{P_A} - \mathbf{V_B}t$ line and the orange circle.

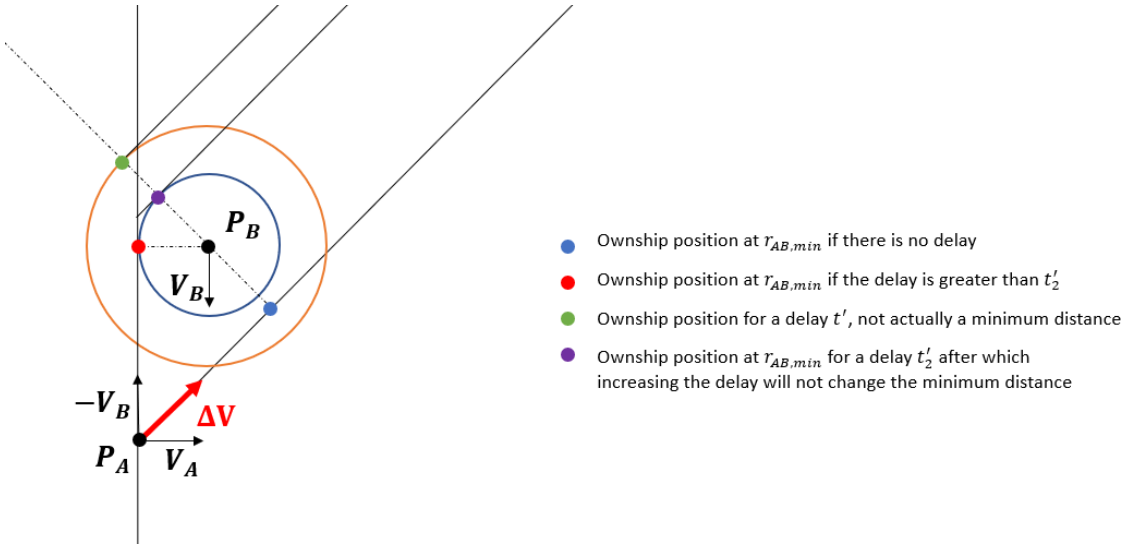


Figure A.3: Example of a situation where there is no valid time to leave

APPENDIX B

INTERVALS

In the Safe Interval Path Planning algorithm it is necessary to keep track of free intervals at all points in the grid. The following algorithm shown in pseudo code was used to update free intervals.

Algorithm 5 Free Intervals Update

```
1:  function UPDATEINTERVAL(freeIntervals, occupied)
2:      keepIterating = True
3:      i=0
4:      n = length(freeIntervals)
5:      while keepIterating do
6:          free = freeIntervals[i]
7:          if free[0] < occupied[1] and occupied[0] < free[1] then
8:              if free[0] < occupied[0] and occupied[1] < free[1] then
9:                  freeIntervals.pop(i)
10:                 freeIntervals.insert(i,[free[0],occupied[0]])
11:                 freeIntervals.insert(i+1,[occupied[1],free[1]])
12:                 keepIterating=False
13:             else if occupied[0] ≤ free[0] and free[1] ≤ occupied[1] then
14:                 freeIntervals.pop(i)
15:                 n -= 1
16:             else if occupied[0] ≤ free[0] then
17:                 free[0]=occupied[1]
18:                 i+=1
19:             else if free[1] leq occupied[1] then
20:                 free[1]=occupied[0]
21:                 i +=1
22:         else
23:             i+=1
24:         if i ≥ n or free[1] ≥ occupied[1] then
25:             keepIterating = False
26:     return freeIntervals
```

In this pseudo code, freeIntervals is a list of intervals, where an interval is a list of two elements $[a, b]$ such that $a \leq b$, and occupied is a single interval. If the two intervals overlap

(checked line 7), there are four possibilities. First, the occupied interval is included in a free interval (line 8). Second, the occupied interval entirely covers the free interval (line 13). Third, the occupied interval begins before the free interval (line 16). And fourth, the occupied interval ends after the end of the free interval. For the SIPP algorithm the free intervals list is initialized as $[[0, \infty]]$. When static obstacles are present and cover the center of the grid cell, the free intervals list is initialized to $[[]]$

A similar algorithm was used when identifying available directions in one of the heuristic for the LocalVO method. The main difference is that the interval represents angles and wraps around at -180° and 180°. This is handled by dividing an occupied interval $[a, b]$ that wraps around (i.e. $b < a$) into two occupied interval $[a, 180^\circ]$ and $[-180^\circ, b]$.

APPENDIX C

THEORETICAL CAPACITY

The baseline throughput that is found by running the simulation with agents that fly straight and at constant velocity was used in multiple experiments. However this baseline does not consider safety. The following analysis considers the maximum number of agents that can coexist safely in the airspace at the same time, however it does not consider agents start and end position.

As explained in [58] and [99] a theoretical maximum safe density of agents can be computed by looking at uniform circle packing. The densest plane lattice packing is a hexagonal lattice, which would only allow one direction of travel per layer. A square packing would allow a bidirectional flow of traffic but lower density. The efficiency of a packing is defined as the ratio of area covered by a circle over the total area $\eta_p = \frac{A_{circles}}{A_{ref}}$. For a hexagonal packing $\eta_p = \frac{\pi}{2\sqrt{3}} \approx 0.91$, for a square packing $\eta_p = \frac{\pi}{4} \approx 0.79$. Since $A_{circle} = N\pi r^2$, the maximum number of agents that can safely be in the simulation area can be found. Assuming a simulation area of 20 by 20 km and a minimum separation distance of 500 m (which means $r = 250m$), the maximum number of agents for the hexagonal packing is 1847 agents, and 1600 agents for the square packing.

The throughput of these configuration can also be computed. In the hexagonal configuration considering agents travel in the direction shown by the arrow on Figure Figure C.1, then there is a vehicle every $2r$ and the number of vehicles per line is $n = \frac{length}{2r}$. The next line of vehicles follow at $\delta x = \frac{3}{2}R = \sqrt{3}r$, which means that a new line of vehicles exit the simulation every $\delta t = \frac{\delta x}{v}$. Hence the throughput is $q = \frac{n}{\delta t} \propto \frac{1}{r^2}$. With the previously stated assumptions this yields a throughput of 110 agents per minute. When considering a direction of travel perpendicular to the arrow this yields the same throughput. In the square configuration, there is also a vehicle every $2r$ and the next line of vehicles is $2r$ behind.

This yields a throughput of 96 vehicles per minute.

In reality the airspace utilization and throughput is much smaller, but these values can be useful to normalize the results.

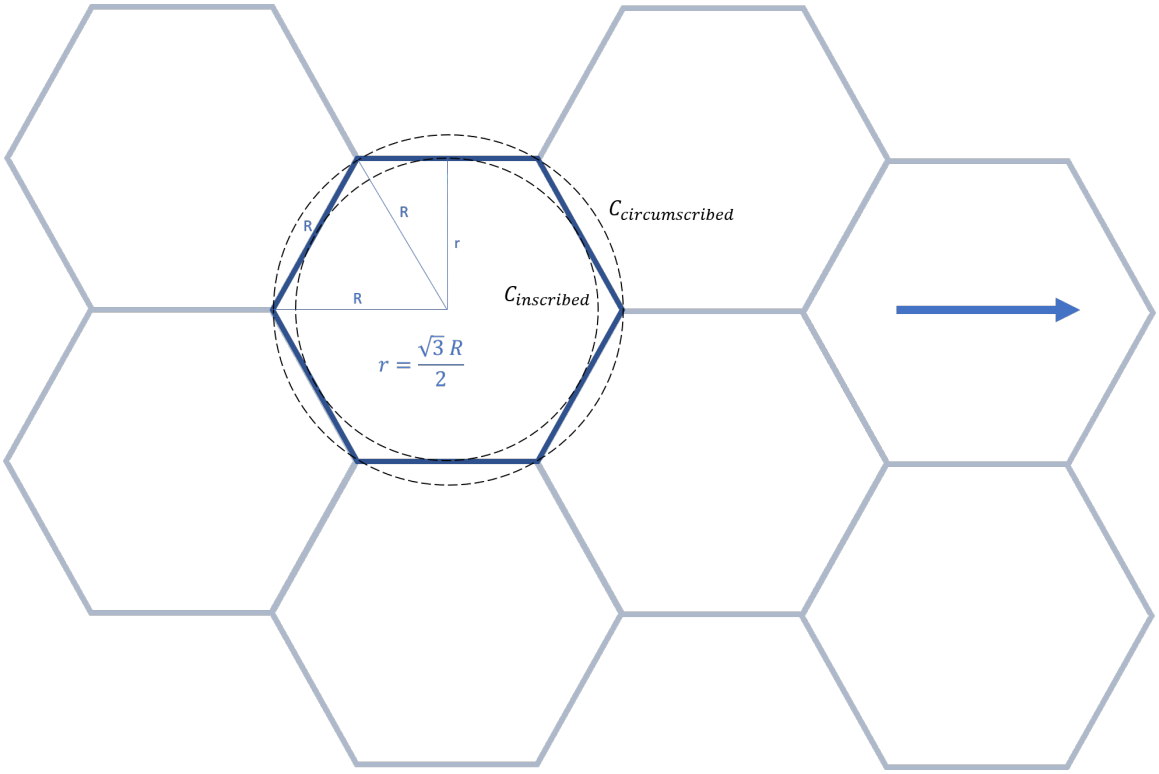


Figure C.1: Illustration of a hexagonal configuration.

APPENDIX D

POPULATION DENSITY CODE

```
import geopandas as gpd
import pandas as pd
from math import cos, pi
import numpy as np
import random
import json

# Equirectangular projection (not very accurate)
earth_radius = 6.3781*10**6
class Projection:
  def __init__(self, center):
    self.lat0 = pi*center[1]/180.0
    self.lon0 = pi*center[0]/180.0

  def lonlat2cartesian(self, lonlat):
    """lat, lon are in degrees, x,y are in m, equirectangular projection
    """
    x=earth_radius*(lonlat[0]*pi/180.0-self.lon0)*cos(self.lat0)
    y=earth_radius*(lonlat[1]*pi/180.0-self.lat0)
    return np.asarray([x,y])

  def cartesian2lonlat(self, xy):
    """ lat, lon are in degrees, x,y are in m, equirectangular
   projection """
    lat=(xy[1]/earth_radius+self.lat0)*180.0/pi
    lon=(xy[0]/(earth_radius*cos(self.lat0))+self.lon0)*180.0/pi
    return np.asarray([lon, lat])
```

```
26
27 # Process the population data
28 point_0 = [-84.396622, 33.772571]  # ASDL coordinates
29 pop_density = gpd.read_file('Population__by_Census_Tract__2018-shp/
       Demographic_Population_Tract_ACS2018.shp')
30 # Create bounding box to only select the area required
31 atl_proj = Projection(point_0)
32 pt1 = atl_proj.cartesian2lonlat([-10000, 10000])
33 pt2 = atl_proj.cartesian2lonlat([10000, 10000])
34 pt3 = atl_proj.cartesian2lonlat([10000, -10000])
35 pt4 = atl_proj.cartesian2lonlat([-10000, -10000])
36 min_lat = min(pt1[1], pt2[1], pt3[1], pt4[1])
37 max_lat = max(pt1[1], pt2[1], pt3[1], pt4[1])
38 min_lon = min(pt1[0], pt2[0], pt3[0], pt4[0])
39 max_lon = max(pt1[0], pt2[0], pt3[0], pt4[0])
40 bounded_pop_density = pop_density.cx[min_lon:max_lon, min_lat:max_lat]
41
42 # Initialize the potential take-off and landing points
43 n = 20
44 spawn_points = {'Latitude':[], 'Longitude':[], 'coordinates_xy':[]}
45 for i in range(0, n+1):
46   for j in range(0, n+1):
47     pt = atl_proj.cartesian2lonlat([-10000 + i * 1000, -10000 + j *
       1000])
48     spawn_points['coordinates_xy'].append([-10000 + i * 1000, -10000 + j
        * 1000])
49     spawn_points['Latitude'].append(pt[1])
50     spawn_points['Longitude'].append(pt[0])
51 df=pd.DataFrame(spawn_points)
52 gdf = gpd.GeoDataFrame(df,
53                geometry=gpd.points_from_xy(df.Longitude, df.Latitude), crs
       ={'init': 'epsg:4326'})
54 # Join LZ points with the density data to get density at each point
```

```
spawn_points_with_density = gpd.sjoin(gdf,
                    bounded_pop_density,
                    how="left",
                    op='within')

# Process the result so it can be used as a random distribution
tot_density = spawn_points_with_density['rPopDensit'].sum()
spawn_points_with_density['density_share']=spawn_points_with_density['
    rPopDensit']/tot_density
coordinates = []
cumulative_density = []
densities = []
sum = 0
for index, row in spawn_points_with_density.iterrows():
  sum+=row['density_share']
  cumulative_density.append(sum)
  # Offset coordinates so that the top left point has cordinates (0,0)
  coordinates.append(np.array(row['coordinates_xy'])+np.array([10000,
    10000]))
  densities.append(row['rPopDensit'])

# Save file in a format that can be easily loaded by the main code
LZ_distribution = {'cumulative_distribution': cumulative_density, '
    coordinates_xy': coordinates.tolist()}
with open('../data/atlanta_lz_pop_density.json', 'w') as f:
json.dump(LZ_distribution, f)

# Routine to draw at random from the distribution and port it to the
    main code
val = random.random()
index = np.searchsorted(cumulative_density, val)
selected_point = coordinates[index]
```

APPENDIX E

CHANGE IN RANKINGS

A multi-attribute decision making (MADM) problem starts with a normalized decision table or decision matrix that presents the value of the attributes for each alternative and a set of weights indicating the relative importance that the designer assigns to each attribute.

The simple additive weighting (SAW) method is a simple and commonly used MADM method [47]. In this method the score of each alternative is expressed as:

$$P_j = \sum_{i=1}^{M} w_i m_{ij}$$

Where w_i is the weight given to attribute i, M is the number of attributes, m_{ij} is the value of attributes i for alternative j, and P_j is the performance score of alternative j.

The alternatives are ranked based on their performance score, the alternative with the highest score being the most desirable.

In this thesis, we are looking at whether to include certain phenomenon in our models and investigating whether errors in modeling could lead to significant changes in the decision taken at the conceptual design stage. Does a set of weights exist such that the ranking obtained with one model is different than the ranking obtained with another model?

Or expressed in a more formal way: given two decision matrices M^1 and M^2 obtained by different models and two alternatives Alt_a and Alt_b, is there a set of weights w_i such that:

$$P_a^1 > P_b^1$$

$$P_a^2 < P_b^2$$

To evaluate in a systematic way the existence of a set of weight such that the preference between two alternatives can change, we formulate the question as a linear programming (LP) problem.

There are two linear inequality constraints:

$$\sum_{i=1}^{M} w_i(m_{i,b}^1 - m_{i,a}^1) \leq 0$$
$$\sum_{i=1}^{M} w_i(m_{i,a}^2 - m_{i,b}^2) \leq 0$$

Where $m_{i,a}^1$ is the value of attribute i for alternative a obtained with model 1. And there is one equality constraint due to the normalization of the weights:

$$\sum_{i=1}^{M} |w_i| = 1$$

Which can be expressed linearly by splitting the weights into two sets: w^+ for positive weights associated with desirable attributes and w^- for negative weights associated with undesirable attributes.

$$\sum_{i \in w^+} w_i - \sum_{i \in w^-} w_i = 1$$

The variables are bounded:

$$\begin{cases} 0 \leq w_i \leq 1, i \in w^+ \\ -1 \leq w_i \leq 0, i \in w^- \end{cases}$$

The cost or value function does not really matter as the primary focus of the problem is to identify whether a feasible solution exist. A potential value function to maximize can be

the sum of the difference in score between the alternatives for both models:

$$\max_{w_i}(P_a^1 - P_b^1) + (P_b^2 - P_a^2)$$
$$\max_{w_i} \sum_{i=1}^{M} w_i(m_{i,a}^1 - m_{i,b}^1 + m_{i,b}^2 - m_{i,a}^2)$$

Note that this is not exactly the canonical form of the problem but it is a form that is accepted by solvers and is intuitive to understand.

This can be used to explore all the combinations of alternatives and identify whether for any of them there exists a set of weights such that preferences are changed between the two models, as illustrated in Algorithm Algorithm 6. In this pseudo code, pref is a vector of size m, with pref[i]=1 if w_i should be maximized, and pref[i]=-1 if w_i should be minimized. The solveLP function solves the following linear problem:

$$\min_x c^T x$$
$$\text{Subject to: } A_u x \leq b_u$$
$$A_e x = b_e$$
$$l \leq x \leq u$$

In the actual implementation, the linprog function of the optimize package of the scipy library was used to solve the linear programming problem.

When considering a model parameter that can be modeled as a k-factor across alternatives, i.e. there exists k_i such that $m_{ij}^2 = k_i m_{ij}$ for all alternatives j, and if the normalization of the decision matrix is done by using the largest element of the matrix and the k_i are strictly superior to zero, then the normalized matrices M^1 and M^2 are equal.

If architectures dominate each other in a Pareto sense then there would be no set of weights that would result in a change in preference between architectures.

Algorithm 6 Can Rankings Change?

```
1:  function CANRANKINGSCHANGE(M1, M2, pref)
2:      m, n = size(M1)
3:      bu = zeros(2,1)
4:      Ae = pref
5:      be = 1
6:      l = minimum(zeros(m,1), pref)
7:      u = maximum(zeros(m,1), pref)
8:      for i=1 to n do
9:          Alternative1A = M1[:,i]
10:         Alternative2A = M2[:,i]
11:         for j=1 to n do
12:             if i≠j then
13:                 Alternative1B = M1[:,j]
14:                 Alternative2B = M2[:,j]
15:                 Delta1 = Alternative1B - Alternative1A
16:                 Delta2 = Alternative2A - Alternative2B
17:                 Au = [Delta1; Delta2]
18:                 c = Delta1 + Delta2
19:                 hasSolution = solveLP(c, Ae, be, Au, bu, l, u)
20:                 if hasSolution then
21:                     return True
22:     return False
```

REFERENCES

[1] J. Holden and N. Goel, "Fast-Forwarding to a Future of On-Demand Urban Air Transportation," Uber Elevate, San Francisco, Tech. Rep., 2016, pp. 1–98.

[2] (2018). "Amazon Prime Air." Available at https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011, (visited on 11/02/2019).

[3] (2018). "Transforming the way goods are transported." Available at https://x.company/projects/wing/, (visited on 11/02/2019).

[4] Crown Consulting Inc, Ascension Global, Georgia tech Aerospace Systems Design Lab, and McKinsey and Company, "UAM Market Study Executive Summary," NASA, Tech. Rep., 2018.

[5] Booz Allen Hamilton, "Executive Briefing Urban Air Mobility Market Study," Tech. Rep., 2018.

[6] L. (Gipson. (2018). "NASA Embraces Urban Air Mobility, Calls for Market Study." Available at https://www.nasa.gov/aero/nasa-embraces-urban-air-mobility, (visited on 11/02/2019).

[7] AUVSI, *Commercial UAS Exemptions by the Numbers*, 2016.

[8] PricewaterhouseCoopers International Limited (PwCIL), "Clarity from above PwC global report on the commercial applications of drone technology," Tech. Rep. May, 2016, pp. 3/40.

[9] Office of inspector general United States Postal Service, "Public Perception of Drone Delivery in the United States," United States Postal Service, Tech. Rep., 2016.

[10] E. Ackerman and E. Strickland, "Medical delivery drones take flight in east africa," *IEEE Spectrum*, vol. 55, pp. 34–35, 2018.

[11] (2018). "General aviation and part 135 activity surveys - cy 2016." Available at https://www.faa.gov/data_research/aviation_data_statistics/general_aviation/CY2016/, (visited on 02/22/2019).

[12] P. D. Vascik, R. J. Hansman, and N. S. Dunn, "Analysis of Urban Air Mobility Operational Constraints," *Journal of Air Transportation*, vol. 26, no. 4, pp. 1–14, Oct. 2018. arXiv: 0307085 [cond-mat].

[13] P. D. Vascik and R. J. Hansman, "Scaling Constraints for Urban Air Mobility Operations: Air Traffic Control, Ground Infrastructure, and Noise," in *2018 Aviation*

Technology, Integration, and Operations Conference, 2018, pp. 1–25, ISBN: 978-1-62410-556-2.

[14] S. B. Cwerner, "Vertical flight and urban mobilities: The promise and reality of helicopter travel," *Mobilities*, vol. 1, no. 2, pp. 191–215, 2006.

[15] L. Gipson. (2019). "Urban air mobility grand challenge." Available at https://www.nasa.gov/uamgc, (visited on 02/23/2019).

[16] NASA Aeronautics Research Mission Directorate, "Urban Air Mobility (UAM) Grand Challenge Industry Day," NASA ARMD, Tech. Rep., 2018.

[17] P. D. Vascik, H. Balakrishnan, and R. J. Hansman, "Assessment of Air Traffic Control for Urban Air Mobility and Unmanned Systems," *International Conference on Research in Air Transportation*, no. June, 2018.

[18] E. R. Mueller, P. H. Kopardekar, and K. H. Goodrich, "Enabling Airspace Integration for High-Density On-Demand Mobility Operations," in *AIAA Aviation Technology, Integration and Operations Conference*, 2017, pp. 1–24.

[19] J. Harrington and R. Richardson. (2018). "Nasa flies large unmanned aircraft in public airspace without chase plane for first time." Available at https://www.nasa.gov/press-release/nasa-flies-large-unmanned-aircraft-in-public-airspace-without-chase-plane-for-first, (visited on 11/02/2019).

[20] J. Grogan, "107 BVLOS Waivers," 2019.

[21] FAA, "First Responder Tactical Beyond Visual Line of Sight (TBVLOS) 91.113 Waiver Guide," 2020.

[22] ——, "Air Traffic by the Numbers," FAA, Tech. Rep., 2018.

[23] Amazon.com Inc., "Revising the Airspace Model for the Safe Integration of Small Unmanned Aircraft Systems," in *NASA UTM 2015: The Next Era of Aviation*, 2015, pp. 2–5.

[24] A. Bauranov and J. Rakas, "Designing airspace for urban air mobility: A review of concepts and approaches," *Progress in Aerospace Sciences*, vol. 125, p. 100 726, 2021.

[25] (2018). "Uas data exchange (laanc)." Available at https://www.faa.gov/uas/programs_partnerships/data_exchange/, (visited on 11/02/2019).

[26] NASA, "Unmanned Aircraft System Traffic Management (UTM)," Tech. Rep., 2018.

[27] M. Johnson, J. Jung, J. Rios, J. Mercer, J. Homola, T. Prevot, D. Mulfinger, and P. Kopardekar, "Flight Test Evaluation ofan Unmanned Aircraft System Traffic Management (UTM) Concept for Multiple Beyond-Visual-Line-of-Sight Operations," in *Proceedings of the Twelfth USA/Europe Air Traffic Management Research and Development Seminar (ATM2017)*, 2017.

[28] G. Zhu and P. Wei, "Low-Altitude UAS Traffic Coordination with Dynamic Geofencing," *16th AIAA Aviation Technology, Integration, and Operations Conference*, no. June, 2016.

[29] SESAR, *U-space Blueprint*, 2017.

[30] C. Barrado, M. Boyero, L. Brucculeri, G. Ferrara, A. Hately, P. Hullah, D. Martin-Marrero, E. Pastor, A. P. Rushton, and A. Volkert, "U-space concept of operations: A key enabler for opening airspace to emerging low-altitude operations," *Aerospace*, vol. 7, no. 3, pp. 1–18, 2020.

[31] E. Sunil, J. Ellerbroek, J. Hoekstra, A. Vidosavljevic, M. Arntzen, F. Bussink, and D. Nieuwenhuisen, "Analysis of Airspace Structure and Capacity for Decentralized Separation Using Fast-Time Simulations," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 1, pp. 38–51, 2017.

[32] E. Sunil and J. Hoekstra, "The Influence of Traffic Structure on Airspace Capacity," *Proceedings of the 7th International Conference on Research in Air Transportation*, 2016.

[33] D.-S. Jang, C. A. Ippolito, S. Sankararaman, and V. Stepanyan, *Concepts of Airspace Structures and System Analysis for UAS Traffic flows for Urban Areas*, 2017.

[34] L. Sedov and V. Polishchuk, "Centralized and Distributed UTM in Layered Airspace," in *8th International Conference on Research in Air Transportation*, 2018, pp. 1–8.

[35] D. N. Mavris, D. A. DeLaurentis, O. Bandte, and M. A. Hale, "A stochastic approach to multi-disciplinary aircraft analysis and design," in *36th AIAA Aerospace Sciences Meeting and Exhibit*, 1998.

[36] D. N. Mavris and O. Pinon, "An Overview of Design Challenges and Methods in Aerospace Engineering," in *Complex Systems Design and Management*, O. Hammami, D. Krob, and J.-L. Voirin, Eds., Springer, 2011, ch. 1, ISBN: 978-3-642-25202-0.

[37] V. Bulusu, R. Sengupta, E. R. Mueller, and M. Xue, "A Throughput Based Capacity Metric for Low-Altitude Airspace," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, pp. 1–9, ISBN: 978-1-62410-556-2.

[38] S. P. Hoogendoorn and V. Knoop, "Traffic flow theory and modelling," in *The Transport System and Transport Policy: An Introduction*, B. van Wee, J. A. Annema, and D. Banister, Eds., Cheltenham, UK and Northampton, MA, USA: Edward Elgar Publishing, 2013, ch. 7, pp. 125–159, ISBN: 9780857936905.

[39] C. Munoz, A. Narkawicz, J. Chamberlain, M. C. Consiglio, and J. M. Upchurch, "A Family of Well-Clear Boundary Models for the Integration of UAS in the NAS," in *14th AIAA Aviation Technology, Integration, and Operations Conference*, 2014, ISBN: 978-1-62410-282-0.

[40] J. Hardy, D. P. Jack, and K. D. Hoffler, "Sensitivity Analysis of Detect and Avoid Well Clear Parameter Variations on UAS DAA Sensor Requirements," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, ISBN: 978-1-62410-556-2.

[41] S. P. Cook, D. Brooks, R. Cole, D. Hackenberg, and V. Raska, "Defining Well Clear for Unmanned Aircraft Systems," in *AIAA Infotech @ Aerospace*, 2015, pp. 1–20, ISBN: 978-1-62410-338-4.

[42] (2014). "Category: Loss of separation." Available at https://www.skybrary.aero/index.php/Category:Loss_of_Separation, (visited on 03/04/2019).

[43] E. Sunil, J. Ellerbroek, and J. M. Hoekstra, "CAMDA: Capacity Assessment Method for Decentralized Air Traffic Control," no. Cd, 2018.

[44] H. Lee and K. Bilimoria, "Properties of air traffic conflicts for free and structured routing," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2001, ISBN: 9781563479786.

[45] U.S. Department of Transportation, "Near Midair Collision Reporting," in *Aeronautical Information Manual*, Aviation Supplies and Academics, Inc., 2018, ch. 7-6-3, p. 997, ISBN: 978-1-61954-536-6.

[46] V. Bulusu, V. Polishchuk, R. Sengupta, and L. Sedov, "Capacity Estimation for Low Altitude Airspace," in *17th AIAA Aviation Technology, Integration, and Operations Conference*, 2017, pp. 1–15, ISBN: 978-1-62410-508-1.

[47] R. V. Rao, "Introduction to multiple attribute decision-making (MADM) methods," in *Decision Making in the Manufacturing Environment: Using Graph Theory and Fuzzy Multiple Attribute Decision Making Methods*, 2007, ch. 3, pp. 27–41.

[48] F. Roman, N. Rolander, M. G. Fernandez, B. Bras, J. Allen, F. Mistree, P. Chastang, P. Depince, and F. Bennis, "Selection without Reflection is a Risky Business...," in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics, Inc., 2004, p. 18.

[49] D. N. Mavris, O. J. Pinon, and D. Fullmer, "Systems design and modeling: A visual analytics approach," in *27th Congress of the International Council of the Aeronautical Sciences 2010, ICAS 2010*, vol. 5, 2010, pp. 3886–3913, ISBN: 9781617820496.

[50] R. M. Keller, "Ontologies for aviation data management," in *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, vol. 2016-Decem, 2016, ISBN: 9781509056002.

[51] N. Peinecke and A. Kuenz, "Deconflicting the urban drone airspace," in *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, vol. 2017-Septe, 2017, ISBN: 9781538603659.

[52] A. (Joulia, T. (Dubot, and J. (Bedouet, "Towards a 4D Traffic Management of Small UAS Operating at Very Low Level," in *Proceedings of the 30th Congress of the International Council of the Aeronautical Sciences (ICAS2016)*, 2016, 2016_0064, ISBN: 9783932182853.

[53] P. Sachs, C. Dienes, E. Dienes, and M. Egorov, "Effectiveness of Preflight Deconfliction in High- Density UAS Operations," Altiscope, Tech. Rep., 2018, p. 17.

[54] M. Faisal Bin Mohamed Salleh, D. Y. Tan, and C. H. Koh, "Preliminary Concept of Operations (ConOps) for Traffic Management of Unmanned Aircraft Systems in Urban Environment," in *AIAA Information Systems-AIAA Infotech @ Aerospace*, 2017, pp. 1–13.

[55] J. Mercer, K. Lee, P. Lee, R. Hoffman, B. Gore, and N. Smith, "Examining Airspace Structural Components and Configuration Practices for Dynamic Airspace Configuration," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2008, pp. 1–23.

[56] (2019). "Free route airspace (fra)." Available at https://www.eurocontrol.int/articles/free-route-airspace, (visited on 04/09/2019).

[57] J. M. Hoekstra, R. N. Van Gent, and R. C. Ruigrok, "Designing for safety: The 'free flight' air traffic management concept," *Reliability Engineering and System Safety*, vol. 75, no. 2, pp. 215–232, 2002.

[58] V. Bulusu, "Urban Air Mobility: Deconstructing the Next Revolution in Urban Transportation - Feasibility, Capacity and Productivity," Ph.D. dissertation, University of Califonia, Berkeley, 2019.

[59] J. Hoekstra, S. Kern, O. Schneider, F. Knabe, and B. Lamiscarre, "Metropolis – Simulation Results and Analysis Report," pp. 1–56, 2015.

[60] N. Durand and N. Barnier, "Does ATM Need Centralized Coordination? Autonomous Conflict Resolution Analysis in a Constrained Speed Environment," *Air Traffic Control Quarterly*, vol. 23, no. 4, pp. 325–346, 2017.

[61] J. Klooster, S. Torres, D. Earman, M. Castillo-Effen, R. Subbu, L. Kammer, D. Chan, and T. Tomlinson, "Trajectory synchronization and negotiation in Trajectory Based Operations," in *29th Digital Avionics Systems Conference*, IEEE, 2010, 1.A.3–1–1.A.3–11, ISBN: 9781424466184.

[62] H. A. P. Blom and G. J. Bakker, "Safety Evaluation of Advanced Self-Separation Under Very High En Route Traffic Demand," *Journal of Aerospace Information Systems*, vol. 12, no. 6, pp. 413–427, 2015.

[63] Federal Aviation Administration, "Introduction to TCAS II," FAA, Tech. Rep., 2011.

[64] SC-147 Traffic Alert & Collision Avoidance System (TCAS), "DO-386 Vol I Minimum Operational Performance Standards for Airborne Collision Avoidance System Xu (ACAS Xu) (Vol I), andDO-386 Vol II Minimum Operational Performance Standards for Airborne Collision Avoidance System Xu (ACAS Xu) (Vol II: Algorithm Design)," RTCA, Tech. Rep., 2020, pp. 1, 935.

[65] M. J. Kochenderfer, J. E. Holland, and J. P. Chryssanthacopoulos, "Next-Generation Airborne Collision Avoidance System," *Lincoln laboratory Journal*, vol. 19, no. 1, pp. 17–33, 2012.

[66] G. Manfredi and Y. Jestin, "An introduction to ACAS Xu and the challenges ahead," *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, vol. 2016-Decem, no. Acas X, pp. 1–9, 2016.

[67] K. D. Bilimoria, "A Geometric Optimization Approach to Aircraft Conflict Resolution," in *18th Applied Aerodynamics Conference*, 2000.

[68] M. S. Eby, "A Self-Organizational Approach for resolving Air Traffic Conflicts," *The Lincoln Laboratory Journal*, vol. 7, no. 2, pp. 239–253, 1994.

[69] J. Maas, E. Sunil, J. Ellerbroek, and J. M. Hoekstra, "The Effect of Swarming on a Voltage Potential-Based Conflict Resolution Algorithm," in *Proceedings of the 7th International Conference on Research in Air Transportation*, 2016.

[70] K. D. Bilimoria, B. Sridhar, S. R. Grabbe, G. B. Chatterji, and K. S. Sheth, "FACET: Future ATM Concepts Evaluation Tool," *Air Traffic Control Quarterly*, vol. 9, no. 1, pp. 1–20, 2001.

[71] J. M. Hoekstra and J. Ellerbroek, "BlueSky ATC Simulator Project: an Open Data and Open Source Approach," *7th International Conference on Research in Air Transportation*, pp. 1–8, 2016.

[72] P. Fiorini and Z. Shiller, "Motion Planning in Dynamic Environments Using Velocity Obstacles," *The international journal of robotics research*, vol. 17, pp. 760–772, 1998.

[73] S. Balasooriyan, "Multi-aircraft Conflict Resolution using Velocity Obstacles," Master thesis, TU Delft, 2017.

[74] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, "The Hybrid Reciprocal Velocity Obstacle," *IEEE Transactions on Robotics*, vol. 27599, no. October, pp. 1–11, 2009.

[75] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n -Body Collision Avoidance," in *Springer Tracts in Advanced Robotics*, 2011, pp. 3–19.

[76] N. Durand, "Constant speed optimal reciprocal collision avoidance," *Transportation Research Part C: Emerging Technologies*, vol. 96, pp. 366–379, 2018.

[77] C. Munoz, A. Narkawicz, G. Hagen, J. Upchurch, A. Dutle, M. Consigilo, and J. Chamberlain, "DAIDALUS : Detect and Avoid Alerting Logic for Unmanned Systems," in *IEEE/AIAA 34th Digital Avionics Systems Conference (DASC).*, 2015, pp. 1–12, ISBN: 9781479989409.

[78] X. Yang and P. Wei, "Autonomous Free Flight Operations in Urban Air Mobility using Monte Carlo Tree Search," in *International Conference for Research in Air Transportation*, 2018.

[79] S. J. Russel and P. Norvig, "A* search: Minimizing the total estimated solution cost," in *Artificial Intelligence: a Modern Approach*, Third Edit, Prentice Hall, 2009, ch. 3, pp. 93–99, ISBN: 0-13-604259-7.

[80] S. M. Lavalle, "Extensions of Basic Motion Planning," in *Planning algorithms*, Cambridge University Press, 2006, ch. 7, pp. 311–367.

[81] M. Phillips and M. Likhachev, "SIPP: Safe Interval Path Planning for Dynamic Environments," in *IEEE International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 4708–4715, ISBN: 9781467317375.

[82] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," *Proceedings of European Control Conference*, no. 3, pp. 2603–2608, 2001.

[83] M. Tra, E. Sunil, J. Ellerbroek, and J. Hoekstra, "Modeling the intrinsic safety of unstructured and layered airspace designs," in *12th USA/Europe Air Traffic Management R and D Seminar*, 2017.

[84] V. Bulusu, R. Sengupta, and Z. Liu, "Unmanned Aviation: To Be Free or Not To Be Free?" *7th International Conference on Research in Air Transportation*, 2016.

[85] P. D. Vascik, J. Cho, V. Bulusu, and V. Polishchuk, "Geometric Approach Towards Airspace Assessment for Emerging Operations," *Journal of Air Transportation*, pp. 1–10,

[86] B. B. Frank, J. Hoekstra, and B. Heesbeen, "Traffic manager: A flexible desktop simulation tool enabling future ATM research," *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, vol. 1, pp. 1–10, 2005.

[87] M. Xue, J. Rios, and J. Silva, "Fe 3 : An Evaluation Tool for Low-Altitude Air Traffic Operations," pp. 1–13, 2018.

[88] N. Wanngren, "Dynamove – Multi Agent Navigation using Velocity Obstacles," Master, KTH, 2011.

[89] L. Gurobi Optimization, *Gurobi optimizer reference manual*, 2019.

[90] S. H. Kim, "Conflict Risk Assessment of Structured and Unstructured Traffic of Small Unmanned Aircraft Systems," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, ISBN: 978-1-62410-556-2.

[91] A. Weinert, S. Campbell, A. Vela, D. Schuldt, and J. Kurucar, "Well-clear recommendation for small unmanned aircraft systems based on unmitigated collision risk," *Journal of Air Transportation*, vol. 26, no. 3, pp. 113–122, 2018.

[92] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[93] D. Silver, "Cooperative Pathfinding," in *AAIDE 1*, 2005, pp. 117–122.

[94] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-Angle Path Planning on Grids," *Journal of Artificial Intelligence Research*, vol. 39, pp. 533–579, 2010.

[95] J. van den Berg and M. Overmars, "Roadmap-based Motion Planing in Dynamic Environments," *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 21, no. 5, pp. 1598–1605, 2004.

[96] Research and Analytics Division of the Atlanta Regional Commission. (2020). "Population (by census tract) 2018." Available at https://opendata.atlantaregional.com/datasets/population-by-census-tract-2018, (visited on 09/29/2020).

[97] R. Golding, "Metrics to characterize dense airspace traffic," Altiscope, Tech. Rep. June, 2018.

[98] C. Ramee and D. N. Mavris, "Development of a framework to compare low-altitude unmanned air traffic management systems," in *AIAA Scitech 2021 Forum*, 2021, p. 0812.

[99] J. Krozel, M. Peters, and K. Bilimoria, "A Decentralized Control Strategy for Distributed Air/Ground Traffic Separation," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2000.

VITA

Coline Ramée obtained a dual degree in Aerospace Engineering from Supélec and the Georgia Institute of Technology in 2016. She has been working at the Aerospace Systems Design Laboratory since Fall 2014. Her interests include autonomy and air vehicles. She worked on several student robotic competitions while at Georgia Tech.