# Multi Agent Path Planning for Medical Urban Air Mobility Services in 3D Environments

Master of Science Thesis

Vincent van Gorp

TU Delft

# Multi Agent Path Planning for Medical Urban Air Mobility Services in 3D Environments

## Master of Science Thesis

by

Vincent van Gorp

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 05-06-2024.


| | | |
|---|---|---|
| Student number: | 4777697 | |
| Project duration: | 3 April, 2023 – 5 June, 2024 | |
| Thesis committee: | Dr.Ir. E. van Kampen | TU Delft, Chair |
| | Dr. M.J. Ribeiro | TU Delft, Daily Supervisor |
| | Dr. O.A Sharpanskykh | TU Delft, Daily Supervisor |
| | Dr. M. Lourenço Baptista | TU Delft, Examiner |


TU Delft

# Acknowledgements

The submission of this Master's thesis marks the culmination of my academic journey at the Faculty of Aerospace Engineering at the Technical University of Delft. I have always had a passion for the aviation industry, and upon finishing my degree this passion has only grown. Over the past six years, I have been able to develop myself both academically and personally and I have had the ability to explore and learn about the aviation industry. Overall, I have enjoyed my time in Delft and I cherish the people that I have met along the way, such as my study friends and "old" roommates. The last year of writing my master's thesis has not been easy, with many setbacks and moments that I wanted to quit. However, upon almost finishing my master's degree, I am very proud of this work and the persistence I have shown along the way.

I would like to express my gratitude to both my supervisors Alexei Sharpanskykh and Marta Ribeiro for their support during the entire thesis project. They have always been available to help me with all kinds of challenges and questions and I would like to thank them for their valuable insights and support.

Furthermore, I want to express my gratitude to my friends and family for their support throughout the past six years, which made my time in Delft more enjoyable. While writing my master's thesis, many of my fellow student friends supported me for which I am grateful. I want to give special thanks to three of them in no particular order: Loek, Sam, and Victor. I appreciate your availability for help and the feedback you provided when I was stuck, both technically and mentally. Your input significantly enhanced the quality of my thesis and the time I spent working on it. Thank you all.

To conclude, I have mixed feelings about the end of my student time. I feel disappointed that my study time has gone by so quickly, but simultaneously I am looking forward to what the future in the aviation industry will bring me.

Vincent van Gorp
Delft, May 2024

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| 3E | Enhanced triple pruning |
| ANSP | Air Navigation Service Provider |
| ASP | Answer Set Programming |
| ATC | Air Traffic Control |
| BCBS | Bounded Conflict-Based Search |
| BP | Bypassing Conflicts |
| BVLOS | Beyond Visual Line-Of-Sight |
| CA* | Cooperative A* |
| CBS | Conflict-Based Search |
| CBSwP | Conflict-Based Search with Priorities |
| CDR | Conflict Detection and Resolution |
| CNPC | Control and NonPayload Communication |
| CO-WHCA* | Conflict-Oriented Windowed Hierarchical Cooperative A* |
| CONOPS | Concept of Operations |
| CT | Constraint Tree |
| CTR | Control Zone |
| DAO | Dragon Age: Origins |
| DNS | Dynamic Potential Search |
| EASA | European union Aviation Safety Agency |
| ECBS | Enhanced Conflict-Based Search |
| EES | Explicit Estimation Search |
| eICTS | extended Increasing Cost Tree Search |
| EPEA* | Enhanced Partial Expension A* |
| eVTOL | electric Vertical Take-off and Landing |
| FAA | Federal Aviation Authority |
| FIMS | Flight Information Management System |
| GCBS | Greedy Conflict-Based Search |
| GIS | Geographical Information System |
| HCA* | Hierarchical Cooperative A* |
| HEMS | Helicopter Emergency Services |

| | |
|---|---|
| I&W | Ministery of Infrastructure and Water Management |
| ICTS | Increasing Cost Tree Search |
| ID | Independence Detection |
| ILP | Integer Linear Programming |
| ILT | Inspectie Leefomgeving en Transport (The human environment and transport inspectorate) |
| LA-MAPF | Multi-Agent Path Finding with Large Agents |
| LoS | Line-of-Sight |
| MA-CBS | Meta-Agent Conflict-Based Search |
| MAPF | Multi-agent Path Finding |
| MASS-GT | Multi-agent Simulation System for Goods Transport |
| MDD | Multi-value Decision Diagram |
| NAS | National Airspace System |
| OD | Operator Decomposition |
| OSM | OpenStreetMap |
| PBS | Priority-Based Search |
| PC | Payload Communication |
| PC | Prefering Cardinal conflicts |
| RF | Radio Frequency |
| rM* | Recursive M* |
| SAT | Boolean satisfiability |
| SDSP | Supplemental Data Service Providers |
| SIPP | Safe-Interval Path Planning |
| SoSID | Systems of Systems Inverse Design |
| STN | Simple Temporal Network |
| TASS | Tree-based Agent Swapping Strategy |
| TMX | Traffic Manager |
| TPG | Temporal Plan Graph |
| UAM | Urban Air Mobility |
| UAS | Unmanned Aircraft System |
| UAV | Unmanned Aerial Vehicles |
| USS | Unmanned Aircraft System Service Suppliers |
| UTM | Unmanned Aircraft System Traffic Management |
| V2V | Vehicle-to-Vehicle |
| VLOS | Visual Line-Of-Sight |
| WA* | Weighted A* |
| WHCA* | Windowed Hierarchical Cooperative A* |

# I

Scientific Paper

# Multi Agent Path Planning for Medical Urban Air Mobility Services in 3D Environments

Vincent van Gorp,*

Delft University of Technology, Delft, The Netherlands

**Abstract**

Urban Air Mobility (UAM) has seen remarkable growth over the past decade, especially considering the delivery of goods. Notably, the delivery of medical supplies via Unmanned Aerial Vehicles (UAVs) has emerged as a promising application, which is driven by its societal benefits and has been demonstrated successfully in rural environments. However, expanding these operations to urban areas presents a unique set of challenges, such as high traffic density and the need for online path planning of UAVs. Effective path planning and coordination are of utmost importance to allow for safe and efficient delivery operations in urban environments. Currently, path planning of UAVs is relatively unexplored, and for drone delivery models the trajectories of UAVs are often oversimplified. This research aims to fill the research gap by developing and evaluating an online path planning and coordination mechanism in 3D for a cooperative fleet of autonomous UAVs delivering medical supplies. The Rotterdam Area is selected as a use case, but the model can be adapted to other cities. Utilising an agent-based model formulation, this paper formulates a novel approach for multi-agent pathfinding in 3D and real-world environments, where the travelled paths of UAVs in drone delivery models are computed by taking into account the urban environment and kinematics of UAVs. Windowed Cooperative Safe Interval Path Planning (WC-SIPP) is used as a path planning and coordination mechanism, where kinematic constraints of UAVs and safety separation distances between UAVs are incorporated. Experimental studies show the capability of the algorithm to plan UAVs in different multi-layer urban environments. We establish that including more than one vertical layer is beneficial for UAVs to avoid conflicts by allowing them to change altitudes during flight. Additionally, we show that an increase in the number of layers does not lead to an increase in the delivery time of packages. However, an increase in available layers does come with a decrease in computational performance. Besides, the proposed method is able to plan the trajectories of UAVs and ensure safety separation between UAVs within a dynamic environment, where up to 20 unexpected obstacles are introduced into the environment. Furthermore, we demonstrated that for our operation model, assigning prioritisation based on either mission type or package request time did not influence the delivery time of packages.

## 1 Introduction

Over the past decade, Urban Air Mobility (UAM) services have been emerging all over the globe, offering both transportation of passengers and packages. Transportation of packages using unmanned aerial vehicles (UAVs) is already in operation in both rural and urban environments. In 2023 the total number of drone deliveries exceeded 1 million in total according to estimations of McKinsey [1], and it is expected that the number of deliveries will only grow in the coming years. Companies such as Zipline [2] and Wing [3] are operating fleets of UAVs to deliver a variety of goods, such as food, medicines and e-commerce. At this point, the delivery of medical items is most mature, with Zipline demonstrating the success of this use case in Rwanda [4]. Zipline is transporting medical supplies from distribution centres to various medical facilities all over the country. The company aims to deliver up to 2 million packages within the country by 2029. Looking at the public benefit of delivering medical items via UAVs, it can be deemed that the public benefit is larger compared to food and e-commerce. According to EASA [5], the social acceptance of drone delivery is higher in case the delivery has benefit for the community and does not serve individual needs. Therefore out of these applications, the delivery of medical supplies is the most promising use case. UAVs have only limited payload capacity, making them well-suited to deliver medical supplies, such as medicine, emergency supplies and vaccines. Even though Zipline has demonstrated the success of this use case in rural environment, integrating a similar system within an urban environment introduces new challenges and constraints. Therefore, effective path planning and coordination is of utmost importance to allow for efficient and safe operations in urban environment.

*Msc Student, Sustainable Air Transport, Faculty of Aerospace Engineering, Delft University of Technology

To analyse the complex dynamics and interactions among different actors for UAM services, our approach involves utilising agent-based modelling (ABM). ABM offers us the benefit of modelling individual behaviour and interactions of different agents, incorporating a dynamic realistic urban environment, and allowing for good model scalability and flexibility. The FAA and NASA [6] propose a theoretical framework outlining the key actors, components, and their interactions on a high level for the Unmanned Aircraft System Traffic Management (UTM) airspace. This framework acts as the basis of our Multi-Agent System (MAS). In order to accurately model the paths of UAVs, this work will focus on the modelling of 3D UAV operations by use of multi-agent pathfinding (MAPF). MAPF algorithms allow for conflict-free planning and coordination between various agents within a shared environment. Path planning is a critical aspect to allow for UAM services, and by following an agent-based modelling approach, this work can contribute by implementing novel techniques to improve multi-agent path planning of autonomous UAVs. Currently, path planning for UAV operations in 3D and real-world urban environments is relatively unexplored. Besides, there has been a focus in literature for medical supply delivery on the development of vehicle routing models, also known as drone delivery models [7, 8, 9]. These models often oversimplify the UAV's trajectory, depicting it as a direct straight line between the departure location and arrival destination, disregarding the operational environment. This results in models where the trajectory of the UAV is not accurately represented. Therefore, this research covers the following research gaps, being that **multi-agent pathfinding in 3D and real-world environments is relatively unexplored**, and **the travelled path of UAVs for drone delivery models is not represented accurately**. Following from these research gaps, the primary focus of this work is to develop a medical drone delivery model in a 3D urban environment representing the travelled paths of UAVs. To analyse path planning in urban environments, we selected the Rotterdam Area as the use case for the distribution of healthcare supplies. However, it must be noted that the model can also be adapted to other urban environments. Rotterdam is selected due to two main reasons. First, the inner city of Rotterdam contains many high rises in combination with organic, non-grid-like, city structures, making path planning less trivial compared to grid-like cities. Second, due to the presence of many healthcare facilities, such as hospitals, pharmacies, clinics, and general practitioners, a large market exists for the delivery of medical supplies. Ensuring timely delivery of medical supplies from warehouses to medical facilities is important for patient care, as delays could result in severe consequences, particularly during shortages [10]. Therefore, the delivery system should allow for medical emergency missions, which would need prioritisation over standard medical delivery. Due to the unexpected nature of emergency missions, the delivery system should allow online path planning.

Hence, in light of the aforementioned considerations the main research objective is specified as follows: **To develop and evaluate an online path planning and coordination mechanism in 3D for a cooperative fleet of autonomous UAVs delivering medical supplies in the Rotterdam Area.**

In this research, we propose a multi-agent system for the delivery of both standard and urgent medical supplies in the Rotterdam Area. Multi-agent pathfinding in a 3D environment is used to model the paths of UAVs, where conflict-free planning and coordination are key. We selected Windowed Cooperative Safe Interval Path Planning as the path planning and coordination mechanism. Kinematic constraints of UAVs are incorporated during planning, as well as separation distance constraints between UAVs. This results in a novel approach for multi-agent pathfinding in 3D and real-world environments, where the trajectories of UAVs are represented by considering the urban environment and the kinematics of UAVs.

This paper is structured as follows. First, a comprehensive overview of the current state-of-the-art research will be given in section 2. Afterwards, section 3 will elaborate on the case study. In section 4 the methodology will be presented. In this section an overview of the multi-agent system, specifications regarding its environment and agents, and the path planning and coordination method is given. Afterwards, the experimental setup is shown in section 5. The results of the experiments are presented and discussed in section 6. Lastly, section 7 provides our conclusions and recommendations for future work.

# 2 Literature Review

This section first presents state-of-the-art information regarding UAV delivery. Afterwards, an overview is presented of the current state-of-the-art research regarding multi-agent path finding (MAPF) in section 2.2. In section 2.3 research regarding the modelling of UAM operations in urban environments will be explored, focusing on applications of MAPF models for UAM services.

## 2.1 UAV Delivery

Urban Air Mobility is seen as one of the new revolutionary approaches to improve the transportation industry, which includes the transportation of packages using UAVs. Companies, such as Zipline [2] and Wing [3], are already operating fleets of UAVs delivering packages in rural and urban environments. For example, Zipline [2] is operating a fleet of autonomous UAVs delivering medical supplies all over Rwanda by making use of centralised distribution centres. They are able to deliver packages to healthcare facilities within an hour of ordering. Similar to Zipline, Wing [3] is also operating fleets of UAVs, where a centralised unmanned traffic management software is used to plan the routes of individual UAVs to deliver various packages. The delivery items can be subdivided into three categories, being retail & e-commerce, food, and medical items. The delivery of medical items is seen as the most promising use case due to its maturity, and the public benefit it offers to society. Besides, the social acceptance for delivering medical supplies is higher compared to the other categories according to EASA [5]. Food delivery is offered by Wing [3] in Australia, but occurs on a smaller scale compared to delivery of medical supplies. Furthermore, the delivery of e-commerce has difficulty with starting up, with Amazon shutting down its e-commerce drone delivery services [11]. Thus, in our research, the delivery of medical supplies in urban environment will be considered with a centralised path planning and coordination approach, similar to the operating models of Zipline and Wing.

## 2.2 Multi-Agent Pathfinding and Motion Planning

Multi-agent pathfinding is the problem of finding paths for multiple agents within a shared environment, where agents have to reach their goal without collisions between other agents and the environment [12]. MAPF problems can be categorised into two distinct groups, centralised and distributed planning methods. We have selected a centralised path planning method, as this fits best within the UTM framework proposed by the FAA and NASA [6]. According to this framework, it can be expected that path planning and coordination per Unmanned Aircraft System (UAS) Operator will be centralised, which is also similar to the operating models of Zipline ][2] and Wing [3]. In this work, a distinction is made between the individual path-finding of agents, low-level planning, and the collaborative path-finding of multiple agents, high-level planning. For the individual path planning of agents, Safe Interval Path Planning (SIPP) [13] has been selected with the objective of minimising travel time. SIPP is well suited for the planning of agents in a dynamic environment as it is able to use a smaller search space in comparison with A*. The A* algorithm is a path-finding algorithm that uses a heuristic search algorithm to find the path with the smallest cost given a start and goal node. Enhancements to the standard A* algorithm are Independence Detection (ID) [14] and Operator Decomposition (OD) [15]. Another promising A*-based algorithm is M* [16]. Even though these algorithms improve the algorithm by addressing the drawbacks of A*, being that the state-space and branching factor (number of possible positions an agent can occupy in the next time step) is exponential in the number of agents, these algorithms are not developed for agents moving in a dynamic environment. SIPP is developed for agents moving in a dynamic environment and offers us the benefit of not using a state for every configuration and time step pair, but uses a grouping of collision-free time steps. An extension of SIPP is SIPP with Kinodynamic Constraints [17]. Standard SIPP relies on the assumption that agents can stop instantaneously, an assumption which does not hold when considering UAV kinematics. This extension allows us to incorporate acceleration and deceleration during path planning. SIPP itself can be integrated with other MAPF solvers, where SIPP acts as the low-level planner. By use of SIPP, we extend our problem from path planning to motion planning, as we incorporate the trajectory of agents, considering kinematic features and dynamic obstacles when agents move towards their goal [18].

The high-level planner should be well suited for online planning and re-planning of operations and allow for mission prioritisation due to the nature of medical supply delivery. One of the most widely used multi-agent planners is Conflict Based Search (CBS) [19], which uses a two-level search in order to find an optimal solution for all agents. The goal of the high-level search is to find conflicts and add constraints, while the low-level search focuses on updating the paths of agents, consistent with the new constraints. CBS does not allow direct integration of prioritisation of agents. Therefore, an extension of CBS is Priority Based Search (PBS) [20] is proposed, which increases the computational performance of CBS and allows for prioritised planning. PBS does offer the benefit of prioritisation of agents, but does not directly offer online path planning. Online path planning can be incorporated into this algorithm, but implementation is not trivial. An algorithm that allows for direct implementation of online planning and mission prioritisation is Windowed Cooperative (WC) Planning [21]. WC has the benefit of reserving paths for a limited time window, and allows for changing prioritisation of agents during the simulation. WC is an online MAPF solver, and even though optimality and completeness are sacrificed, it allows for real-time path planning. Therefore, WC is selected as the coordination and conflict resolution approach, where the standard implemented A* algorithm is replaced with SIPP in order to plan the trajectories of UAV agents.

### 2.3 Modelling UAM Operations in Urban Environment

With the continuous advancement of UAV technology, recent research has been anticipating towards an increasing demand of UAM services. The development of UTM systems is imperative to ensure safe integration of UAV operations in low-altitude airspace. An important component of safe operations is the establishment of Conflict Detection and Resolution (CDR) methods. A distinction can be made between strategic and tactical CDR. Strategic CDR aims to find and prevent conflicts before they occur by adjusting flight paths or re-scheduling missions. This method is also referred to as pre-flight CDR. Tactical CDR aims to respond to conflicts as they arise during mission execution, where often only a small part of the flight plan is adjusted to keep the deviation as small as possible. Ho et al. [22] proposed a pre-flight CDR method that incorporates both take-off scheduling and speed adjustment. Take-off scheduling allows for operations to be postponed, while speed adjustment allow for decreased speed on given flight segments. Their method also incorporates the shape of UAVs, by representing an agent as a sphere. The representation of UAVs is based on previous work from Ho et al. [23]. Postponing flights, speed adjustment, and the UAV's shape will also be incorporated within this work, albeit with minor adjustments. The MAPF solver selected by Ho et al. is Enhanced Conflict Based Search (ECBS), which plans paths of UAV operations in batches. Previously planned batches are afterwards treated as flight path constraints for following batches, which is less suitable for online planning and incorporation of priority missions. Besides, Hoekstra et al. [24] performed research towards centralised separation management for UAM services, exploring the modelling of urban environment, separation management concepts, and different mission types. For the centralised concept within this work, flights are strategically deconflicted prior to take-off by assigning flights to different flight layers and afterwards delaying departure slots to resolve conflicts. However, this does not take into account rerouting of flights.

Besides the research from Ho et al. [22, 23], limited research is available on the modelling of 3D UAV operations in urban environments, with a focus on multi-agent path planning. Models that do consider the delivery of medical supplies [7, 8, 9] frequently oversimplify the trajectory of UAVs, portraying it as a direct line from departure to destination point. Therefore, this research will contribute to the existing body of knowledge research by exploring multi-agent path-finding in 3D and real-world environments. The case study presented in section 3 will serve as the basis to evaluate our multi-agent pathfinding method in a practical scenario.

## 3 Case Description

This section provides further details regarding the use case employed in this study. This is done by specifying the objective of the operation, and describing the concept of operations. The concept of operations acts as the foundation of the agent-based model.

In the context of transporting medical supplies, guaranteeing timely delivery from warehouses to medical facilities is paramount for patient well-being. This study focuses solely on delivery between inventory locations, commonly referred to as warehouses, and medical facilities. It is expected that drone delivery will be mainly utilised for last-mile delivery, as is similar to the current operation models of Zipline [2] and Wing [3]. Due to the limited payload of UAVs, the scope of this research is limited to operations of UAVs flying towards and from the medical facility, and delivering one package only. UAVs are able to deliver a package by lowering the package using a cable delivery system while hovering above the drop-off zone. These operations assume enough space for multiple drones to simultaneously deliver medical packages, without imposing a maximum number of hovering drones. To achieve effective operations, the objective of the operation is to minimise the travel time of UAVs, while ensuring no collisions during flight. Reducing travel time is desired as this directly affects the speed of delivering medical packages.

In addition, we select the Rotterdam Area as the use case for the distribution of healthcare supplies. This environment is selected due to the presence of many healthcare facilities, such as clinics, general practitioners, hospitals, and pharmacies. It is also selected due to its organic, non-grid-like city structure with a large variation in building types. Besides the environment, the actors present in this use case are derived from the theoretical UTM framework proposed by the FAA and NASA [6]. The actors present in this use case are the UAS Operator, UAVs, and medical facilities. How these actors are represented within our MAS, will be extensively discussed in section 4.2. The medical supplies will be delivered by UAVs, where we selected the DJI Matrice 600 [25]. This UAV has a payload capacity of up to 6 kilograms and a round trip range of 10 kilometres, which is sufficient for medical supply delivery in the selected areas of Rotterdam. For the delivery of medical supplies, a distinction can be made between two different mission types. The first is the delivery of standard medical supplies, and the second is the delivery of urgent medical supplies. Within this research, the exact content of each package is not specified. For standard mission types, one could envision the transportation of routine medication, diagnostic samples or non-emergency medical equipment. For urgent mission types, one could think of blood samples,

emergency medications and emergency medical equipment. In order to execute the different missions, one UAS Operator will be responsible for path planning and coordination of UAVs. It is assumed that no other UAS Operators and foreign UAVs are present during operations. The UAS Operator will have an infinite amount of UAVs at its disposal to plan missions. This decision has been taken, due to the research being centred on path planning rather than task allocation.

# 4 Methodology

In this section, we propose an agent-based modelling approach, with a focus on the modelling of 3D UAV operations for medical supply delivery in urban environment by use of multi-agent pathfinding. The implementation of the model is developed in Python. First, a specification on the environment will be presented in section 4.1. In section 4.2 the agents present within the model and their interactions will be described. Lastly, the proposed path planning and coordination mechanism will be presented in section 4.3.

## 4.1 Environment Specification

In this research, the Rotterdam Area has been selected as the use case for the distribution of healthcare supplies. Two different neighbourhoods in Rotterdam have been selected in order to assess our MAS. Both areas contain multiple healthcare facilities. A 500m by 500m neighbourhood between Erasmus Medical Centre and Leuvehaven has been selected as the small environment. For the large environment the neighbourhoods Spangen, Tussendijken, and a part of Bospolder has been selected, which spans an area of 750m by 1500m. The environment is represented by a graph $G = (V, E)$, consisting of vertices V and non-directional edges E. Each vertex $v_i$ has as an attribute the location coordinates $(x_{v_i}, y_{v_i}, z_{v_i})$. The 2D graph is obtained using data from OpenStreetMap (OSM) ][26], representing the street network in the selected neighbourhoods. The 3D graph is constructed by duplicating the 2D graph across multiple layers. Each vertical layer is separated by 10 meters, as the vertical safety separation distance between UAVs is set to be 7.62 meters (25 feet), similar to the values used for the Metropolis II project [24]. Each vertex in a layer is connected to its neighbour vertices within its own layer, as well as to those in the one layer above and below. No edge is present between a vertex and the vertices directly above and below it, as this would result in non-realistic UAV trajectories. A schematic overview of vertex connections in a 3D graph across multiple layers is shown in Figure 1.

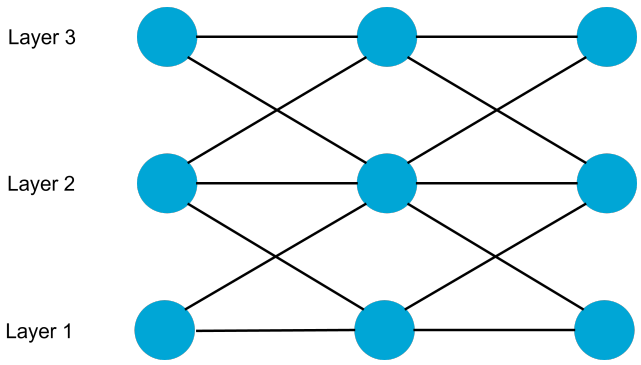


Figure 1: Schematic overview of vertex connections across multiple layers

A 2D graph representation of the small and large environment can be seen in Figure 2 and Figure 3, respectively. In this graph, warehouse locations are represented by blue vertices, and healthcare facilities by red vertices. These vertices are located in the bottom layer. For the small environment, 2 warehouse locations and 4 healthcare facilities are present. For the large environment, 3 warehouse locations and 5 healthcare facilities are present. Within these environments, the healthcare facilities represent real healthcare facilities, such as doctors, clinics and pharmacies. Due to the absence of existing warehouses, strategic locations were chosen to represent warehouses. The locations are predominantly located at the edges of the environment, as this represents the entry and exit points of the delivery area. It is anticipated that future warehouses will be situated on the periphery of the city.

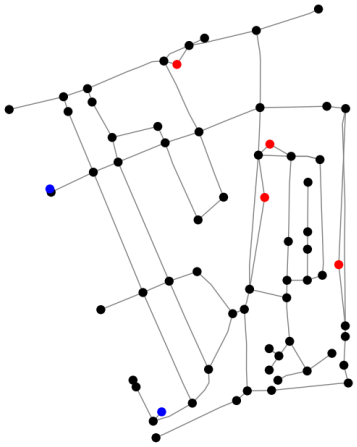

Figure 2: Graph representation for the small environment with blue nodes being warehouse locations, red nodes being healthcare facility locations

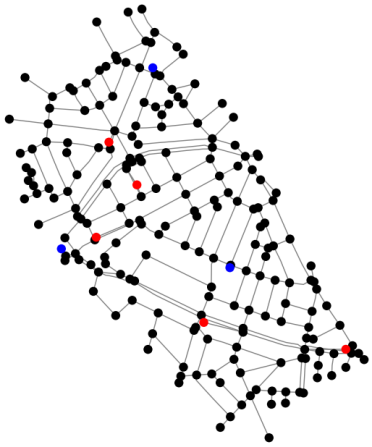

Figure 3: Graph representation for the large environment with blue nodes being warehouse locations, red nodes being healthcare facility locations

## 4.2 Agent Specification

This section presents the specifications related to the agents present in the MAS, which are the UAS Operator Agent, UAV Agent, and Healthcare Facility Agent. An overview of the MAS can be seen in Figure 4.

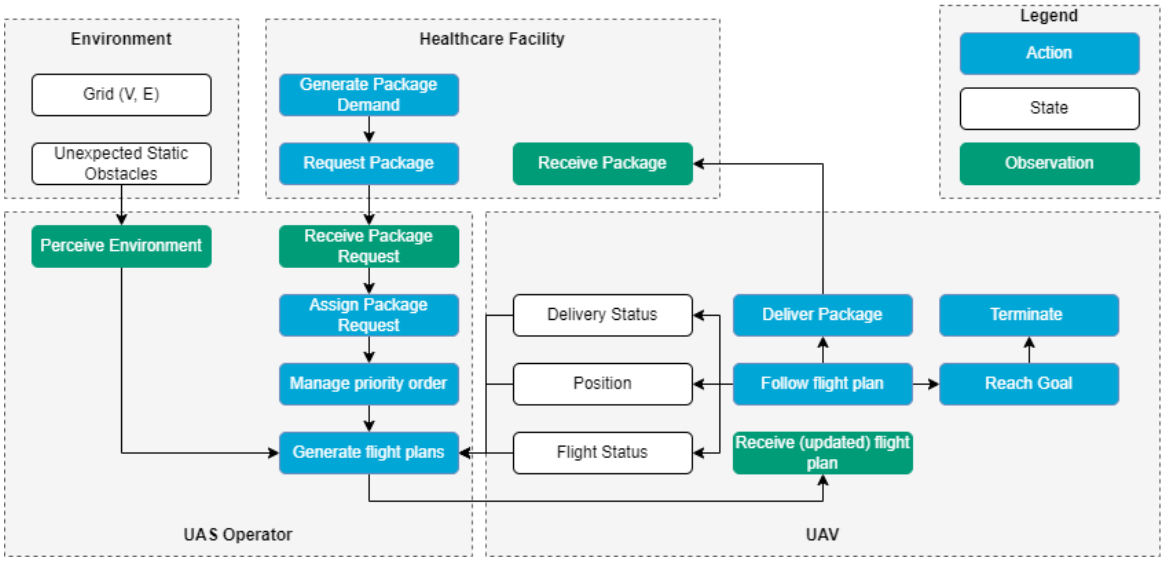


Figure 4: Overview of the multi-agent system

### 4.2.1 UAS Operator Agent

The UAS Operator agent is responsible for the management of the entire UAV fleet, including the planning and coordination of UAVs, in order to ensure fast medical package delivery. The operator has the goal to minimize the delivery time of packages, while ensuring no collisions between UAVs. Within the scope of this model, only 1 centralised UAS operator agent will be present, with no physical presence in the simulation. The agent is responsible for the centralised path planning and task allocation, with an unlimited amount of UAV agents at its disposal. This indicates that the UAS Operator will be the single central computing power, responsible for finding a global solution for all agents. The agent has the following properties:

- **Allocation property**: This property involves interaction between the UAS Operator and UAV agent. A requested package by a healthcare facility is allocated to a UAV agent at the request time, giving the UAV agent a start, package drop-off and goal location.

- **Priority management property**: This property involves interactions between the UAS Operator Agent and all active UAV agents. The priority planning schedule is based is based on 3 factors, being the delivery status (delivery or fly back), package type (urgent or standard), and package request time of a UAV agent. Priority is first given to UAVs delivering packages over flying back. Second UAVs are given priority if their package type is urgent over standard. Last, priority is given for the earliest request time. The combination of these factors will determine the priority order.
- **Path planning property**: This property involves interaction between the environment, the UAS Operator Agent, and a UAV agent. The operator is able to plan the path of a UAV agent, after initialisation, given the representation of the environment, previously computed paths, and collision-free periods (safe intervals). The UAS operator agent makes use of WC-SIPP during path planning, which will be elaborated upon in section 4.3. During path planning, kinematic constraints from the UAV agent are taken into account.
- **Path update property**: This property involves interaction between the UAS Operator Agent, and a UAV agent. For each planning window, following the WC-SIPP method, the UAS operator shares the updated path with the UAV Agent.
- **Safety Separation Property:** This property includes interaction between the UAS operator agent and UAV agents. During path planning, the operator ensures sufficient separation between UAV agents. The horizontal separation distance is 32 meters and the vertical separation is 7.62 meters (25 feet). These values are derived from the Metropolis II project [24].
- **Check for environment availability property**: This property involves interaction between the environment and the UAS Operator Agent. The agent is able to perceive the environment, including the availability of all vertices during each path planning window. A vertex can be unavailable if it becomes blocked by an unexpected static obstacle. This could for example occur when temporary airspace restrictions are imposed.

#### 4.2.2 UAV Agent

The UAV agent is responsible for the task execution of the imposed tasks by the UAS Operator agent. This includes initialisation at the correct start position, following the provided path to the healthcare facility, delivering the package to the customer agent, and flying back to the warehouse location. During operations, it is assumed that the agent has sufficient battery life to complete a mission without recharging. With a round trip range of 10 kilometres and a maximum environment size of 750m by 1500m, this is deemed a valid assumption. The agent has a physical presence within the simulations, and its flying properties originate from the DJI Matrice 600 [25]. Overall, an unlimited number of UAVs is available for the UAS operator. The UAV agent has the following properties:

- **Initialisation property**: This property includes interaction between the UAV agent, UAS Operator, and the environment. A UAV agent will be able to initialise at its starting position once the spawn time of the agent is equal to the simulation time step, which is provided by the UAS Operator agent. The UAS Operator agent will send information regarding the start location, the package type, and the goal to the UAV agent.
- **Receive Trajectory Property**: This property includes interaction between the UAV agent and the UAS operator agent. For each planning window, following the WC-SIPP method, the UAV agent receives the updated path from the UAV agent, including the imposed speeds along this path.
- **Motion Properties**: This property includes interaction between the UAV agents, the UAS Operator, and the environment. At each time point, the UAV agent follows the heading, acceleration, and speed given along its route by the UAS Operator Agent. The agent has the capability to accelerate/decelerate with a constant acceleration of 2 $[m/s^2]$ and has a maximum vertical speed of 5 $[m/s]$ and a maximum horizontal speed of 18 $[m/s]$ [25]. It is assumed that the agent is only able to move with a speed in horizontal direction, that is a multiple of the UAV's acceleration. When considering movement between two layers, movement in vertical direction is necessary. For movement in vertical direction, it is assumed that the initial and final vertical speed of a motion is 0 $[m/s]$, as within each layer the agent is restricted to only move in horizontal direction. If the agent needs to change altitude, it will instantly start this movement in the quickest way possible. The movement consists of an acceleration, constant speed, and deceleration phase. Regarding horizontal movement, UAVs are only allowed to either accelerate/decelerate to a predetermined speed on an edge. At each node, a command can be given to increase/decrease its speed using a constant acceleration/deceleration. A speed change will occur at the beginning of the edge. The distance and time to accelerate/decelerate is given by Equation 1 and Equation 2. The time flying at a constant speed after acceleration/deceleration is given by Equation 3. The total time travelling the edge is the addition of $t_{\mathbf{acc}}$ and $t_{v_{constant}}$.

$$d_{acc} = v_0 \cdot t_{acc} \pm \frac{1}{2} \cdot a \cdot t_{acc}^2 \quad (1) \qquad t_{acc} = \frac{|v_1 - v_0|}{a} \quad (2) \qquad t_{v_{constant}} = \frac{d_{edge} - d_{acc}}{v_1} \quad (3)$$

Speed changes can be imposed to either avoid conflicts with other agents or to allow for an upcoming turn. In order to allow for realistic operations, turn speed restrictions are imposed, based on the heading change required for the turn. The turn speeds are taken from USEPE, a research project exploring separation methods for drone operations in urban environment [27]. Besides, as both the warehouse location and healthcare facility are located in the bottom layer, the UAV will operate predominantly in the bottom layer. In case the lowest layer is occupied, the other layers are used as an alternative. The UAV will come back to the lowest layer as possible afterwards.

- **Package delivery property**: This property includes interaction between the UAV Agent, Healthcare Facility Agent, and the UAS Operator Agent. Once a UAV reaches the healthcare facility location, it will able to hover above the target location and deliver the package to the healthcare facility agent. It is assumed that for delivery of a package by use of a cable system, the delivery takes 20 to 30 seconds, using a uniform distribution $U(20, 30)$. No specifics regarding the exact delivery time are currently available from either Zipline [2] or Wing [3], but a time range was deduced through the analysis of video material. Upon delivery, the UAS Operator Agent will receive an update regarding the UAV's delivery status.
- **Termination Property**: This property includes interaction between the UAV Agent, the UAS Operator Agent, and the environment. The UAV agent will terminate itself from the environment once it has reached its goal. Upon termination, the UAS Operator Agent will receive an update regarding the UAV's flight status.

#### 4.2.3 Healthcare Facility Agent

Lastly, the Healthcare Facility Agent is able to request medical packages from the UAS Operator. It can either request an urgent or standard package. Every medical facility is represented by one independent agent. The healthcare facility is represented by a vertex in the simulation environment. This location can only be used by UAV agents to deliver packages. Within the urban environment, real-world locations have been selected as delivery sites. The drop-off locations are situated at the front of these locations. The delivery location is assumed to be a safe area, where multiple UAVs are able to deliver requests to a healthcare facility agent. This agent has the following properties:

- **Package Request Property**: This property includes interaction between the Healthcare Facility Agent and the UAS Operator Agent. The Healthcare Facility agent is able to generate package requests at a pre-defined demand rate. In order to generate packages a Poisson distribution is used, with $\lambda$ being the package rate per hour per facility.
- **Package Receiving Property**: This property includes interaction between the healthcare facility agent, and a UAV agent. Once the UAV agent arrives at the delivery location, the healthcare facility agent is able to receive the package.

### 4.3 Path Planning and Coordination Mechanism

In the previous section, the agents present in the multi-agent system are introduced. This section will elaborate on the path planning and coordination mechanism utilised by the UAS Operator Agent. We modelled the path planning of UAV agents as a Multi-Agent Path Finding (MAPF) problem. The path planning and coordination mechanism consists of two parts, which are the single-agent planner and the multi-agent planner. For the multi-agent path planning, Windowed Cooperative (WC) planning has been used, which is discussed in section 4.3.1. For the single-agent planner, Safe Interval Path Planning (SIPP) has been used, which is elaborated upon in section 4.3.2.

#### 4.3.1 Windowed Cooperative Planning

For the multi-agent path planning and coordination algorithm, we have selected Windowed Cooperative (WC) Planning. WC allows us to plan paths online, and incorporate and change mission prioritisation during planning. The algorithm can be found in Algorithm 1. The multi-agent cooperative path planning occurs in windows, where the window size $w$ is fixed to a predefined size. A window is a time interval for which the constraints of the previously planned agent are considered for the planning of an agent. After this window, the remaining paths of the previously planned agents are not reserved. Therefore, conflicts that would occur after this window are not resolved in the planning window. Only conflicts occurring during the window are resolved. The agent will be following its path until replanning occurs, which is halfway through the window, with $k = \frac{w}{2}$. (Re-)planning occurs in line 7 for Algorithm 1. If (re-)planning is true, the path planning order is assigned based on the priority of the agents. The paths of agents are planned sequentially using the individual agent planner, which

is in our research SIPP. The priority order of agents is based on the criteria discussed in section 4.2. After assigning the priority order of the agents, the lists of both static and dynamic obstacles are updated. Static obstacles are unexpected obstacles that will be introduced in the current path planning window and dynamic obstacles are the paths of previously planned agents. Before the complete replanning cycle, constraints from the previous planning window are partially taken into account in the current planning window. In our work, if a UAV agent is travelling on an edge during replanning, the agent will follow the previously computed plan until the next vertex, and replanning for this agent will occur from this vertex onwards. This indicates that the path from the preceding window is partially reserved for the current window. UAV agents are only allowed to travel to the neighbouring vertex when they enter an edge with the imposed speed by the operator agent, and therefore no additional speed changes along this edge are allowed. After assigning the priority order and updating the static and dynamic obstacles, the replanning of all agents can commence. For the individual planning of agents, SIPP is used, with the following input information: starting position ($n_{start}$), starting time ($t_{start}$), starting velocity ($V_{start}$), and starting heading ($\psi_{start}$). This is needed to compute a feasible trajectory for the UAV agent, considering the kinematic constraints. After (re-)planning, the path of the agent will be updated. This means that the initial path is disregarded and the new path is assigned to the agent. To ensure that agents head in the correct direction the single agent path planning is executed from the UAV's current starting position to its goal position, with only taking into account the constraints of the previously planned agent for the current window.

Besides, a drawback of windowed planning is the potential for bottleneck situations, as agents' paths are planned sequentially and constraints from previously planned agents could result in not being able to find a feasible path. This issue can be resolved by using deterministic re-scheduling [28]. If the planner is not able to find a path for an agent due to higher-priority reservations from other agents, that agent's priority is raised to the highest level during replanning. This is done to allow the agent to plan its path with a minimal number of constraints, increasing its success in finding a feasible path. After resorting the priority order of the agents, the previously planned paths are removed from both the set of paths, as well as from the dynamic obstacle list. Within one window multiple reschedules can take place, until a feasible set of paths for all agents is found. In case no successful path is found, the simulation is set to be unsuccessful.

**Algorithm 1** Windowed Cooperative Safe Interval Path Planning

**Input:**
  Set of agents $A$, Graph Urban Environment $G(V, E)$, Dynamic Obstacle List $DO$,
  Static Obstacle List $SO$
**Output:**
  Set of paths $P_a$ for each agent $a_i$, a $\in$ A
**Function:**
**if** Planning is True: **then**
  PlanningSchedule = sort(A, prioritisation criteria)
  SO.update()
  DO.update($P_a$) # Update DO using previously computed $P_a$
  $P_a = \{\}$
  **while** Not all paths found **do**
    **for** each $a_i$ *in* A **do**
      path = SIPP($n_{start}$, $t_{start}$, $V_{start}$, $\psi_{start}$, $G(V, E)$, DO, SO)
      **if** path found **then**
        $P_a$.append(path)
        DO.update($P_a$)
      **if** path not found **then**
        PlanningSchedule = Re-sort(A) # using Deterministic Rescheduling
        $P_a = \{\}$
        DO.remove($P_a$) # Remove new planned paths before starting replanning
        **if** no new PlanningSchedule is found **then**
          SimulationFailed is True

### 4.3.2 Safe Interval Path Planning

To plan the paths of individual agents, we utilise SIPP [13], a path planning method developed for path planning in dynamic environments. Our method is developed in Python and is based on the open-source code by Bose, Markelov and Noyes [29]. SIPP offers us the benefit of not using a state for every graph location and time step pair, but uses a grouping of collision-free time steps for every location. This grouping is called a safe interval,

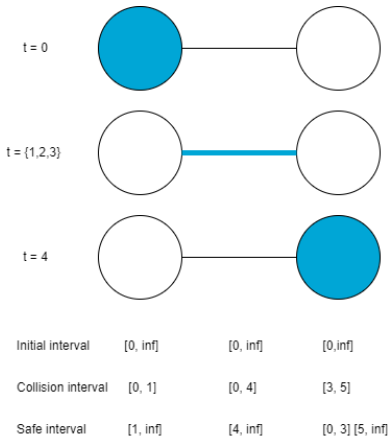


Figure 5: Example path of an agent on a graph $G(V, E)$ and the resulting collision and safe intervals

and represents the time interval during which the agent can occupy the location. A visual representation of how safe intervals are updated is given Figure 5. The number of safe time steps for a location can be unbounded, but the number of safe intervals per location is finite and generally small, which results in a smaller search space compared to A*. In order to track the movement of dynamic obstacles, the paths of agents are stored in a dynamic obstacle list. In our work, dynamic obstacles are agents that have a higher priority than the planned agent. Standard SIPP relies on the assumption that agents are able to stop instantaneously. This assumption does not hold when considering kinematic constraints from UAV agents are considered and must be taken into account accordingly. An extension of SIPP, SIPP with Kinodynamic Constraints [17], relaxes the assumption of agents being able to stop instantaneously. In our work, we will also relax this assumption. Besides, SIPP also relies on the fact that an agent is able to wait at locations. In our work, UAV agents are prohibited to wait during flights, as this is deemed to be inefficient in terms of energy usage and flight time.

Planning with safe intervals consists of two phases, which are the graph construction and the graph search. For the graph construction, a timeline for each configuration is created using the paths of previously planned agents. As we incorporate variable speeds of agents, the configuration consists of the graph location and horizontal speed of the vertex or edge. Additionally, we also incorporated safe intervals on edges, to allow for agents following each other on the same edge when travelling in the same direction. If they would be travelling in opposite directions, constraints would be imposed. After the graph construction, the graph search is performed in order to find a path for a UAV agent. For the graph search, an overview of the algorithm is illustrated in Figure 6 using a flow diagram. On the left-hand side the function findPath is illustrated. For this algorithm, the heuristic function is adapted to incorporate both time and distance in equal weights, using an Euclidean distance heuristic. The distance between two vertices is calculated using the Haversine formula, which also incorporates altitude. The findpath function is used to find a path from a starting position to a goal position, given the starting position, time, speed and heading of an agent. The function itself is not altered from the original function and more detailed information regarding this function can be found in Part III, Supporting Work, Chapter 1. In order to find a path successors are created from a state, beginning with the starting state. A successor is a state that can be reached from the current state by taking a valid action. In our work, the function to create successor states is adapted for our use case, where the getSuccessor function is illustrated in flow diagram Figure 6 on the right-hand side. During the creation of successors, it is ensured that the to-be-performed motion is feasible considering the kinematic constraints of the agent. This is necessary, as the agent is not able to instantaneously change its speed. Besides, healthcare facility locations are seen as safe positions, meaning that no collisions occur at these vertices. In our work, multiple UAVs are allowed to deliver packages at the same time. For all other locations, if a collision occurs at timestep $t$ or the separation distance is violated during the motion, the successor is disregarded.

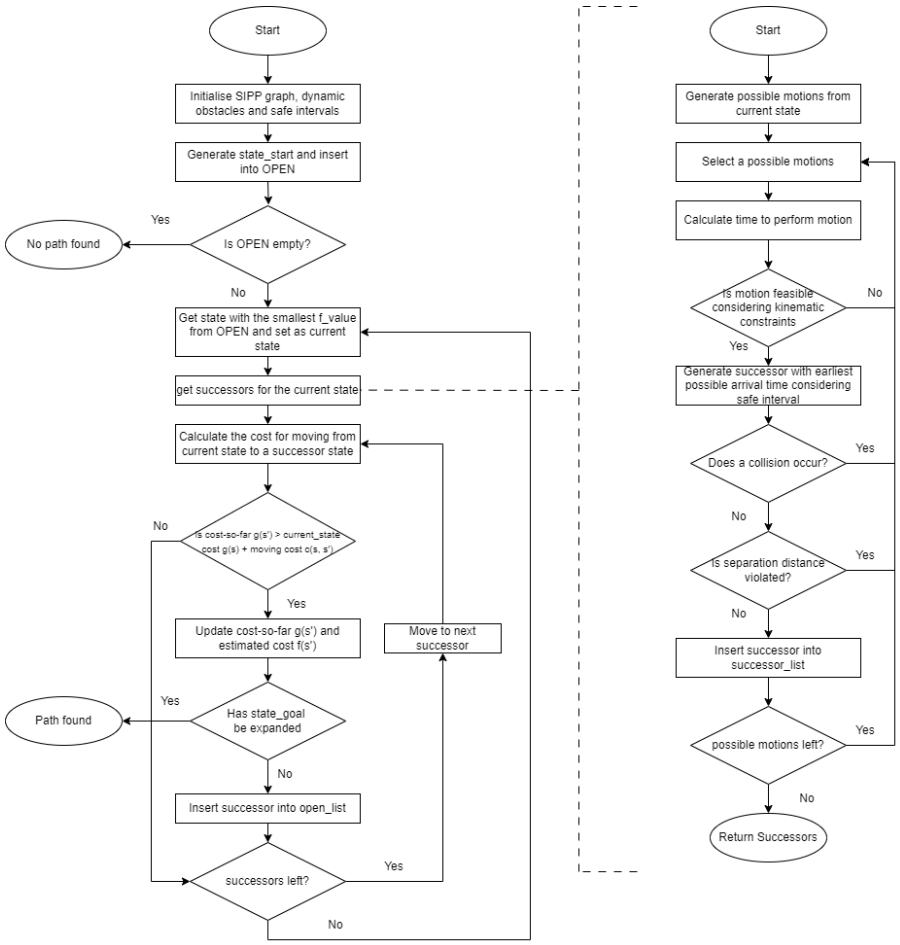


Figure 6: Flow diagram of the findPath function on the left hand side, including the getSuccessors function on the right hand side

# 5 Experimental Setup

In this section, we present the experimental setup to simulate the transportation of medical supplies by use of UAVs in urban environments. The operations are based on the case description provided in section 3, where the operating time window for each experiment is set to 1 hour. To analyse the performance of the multi-agent model, several experiments are executed. Each experiment is performed using the urban environment representations, as described in the environment specification in section 4.1. The agents present in the multi-agent system, as described in section 4.2, are the same for all experiments. Every experiment consists of a set of scenarios, and every scenario is a set of different independent variable settings. An overview of all scenarios can be found in Part III, Supporting Work, Chapter 2. In order to have a constant coefficient of variation each scenario is simulated 100-200 times, which ensures reliable results. More details regarding the variability of simulation results can be found in Part III, Supporting Work, Chapter 3. This section will first elaborate on the independent variables and key performance indicators, and afterwards discuss the four different experiments.

## 5.1 Independent Variables

For each experiment, a subset of independent variables is varied in order to analyse the multi-agent system, which are as follows:

- **Window Size:** is the window size selected for the WC-Planning Algorithm. Three different window sizes are used, being 20, 30 and 40 $[s]$ in size.
- **Urban Environment Type:** Two different urban environments are used, referred to as a small and a large environment, as described in section 4.1.
- **Demand Rate:** is the number of total packages requested per medical facility per hour. Three different demand rates are used, which are 2, 4 and 6 packages/facility/hour.
- **Number of Available Layers**: is the number of available layers for operations in the urban environment. Three different values are used, which are 1, 3 and 5 available layers. If the number of layers is not varied,

the standard number of available layers is set to 3. As described in section 4.2, the UAVs will be delivering packages in the lowest layer, where both the warehouse and healthcare facility locations are situated. Other layers are used as alternatives when the lower layer is occupied.
- **Urgency Distribution for Packages:** is the percentage of total packages across all facilities that have the package type urgent. Three different distributions are used, being 10%, 25% and 50%. If the urgency distribution is not altered, a distribution of 25% is selected.
- **Priority Planning Order:** is the priority planning ordering method used by the UAS Operator. Three distinct planning orders exist which are Mission Type and Request Time (MTRT) ordering, Request Time (RT) ordering, and Random (R) ordering. If the priority planning order is kept constant, the standard priority planning order is MTRT.
- **Number of unexpected obstacles** is the number of unexpected static obstacles introduced during path planning. An unexpected obstacle will be represented by unavailable vertices. 5 different values are used, being 0, 5, 10, 15, and 20. For experiments A, B, and D, the number of unexpected obstacles is set to 0.

## 5.2 Key Performance Indicators

To analyse the performance and results of the different experiments, the following Key Performance Indicators are measured for the subsequent experiments:

- **Computational Time:** is the computational time $[s]$ for planning the total path of each UAV agent.
- **Nodes Explored:** is the number of nodes explored during the path planning search of a UAV agent
- **Success rate:** is the number of successful simulations, over the total number of simulations per scenario. A simulation is deemed successful if all delivery missions within that simulation are completed without collisions.
- **Travel Distance:** is the travelled distance $[m]$ of each UAV agent.
- **Travel Time:** is the total travel time $[s]$, including delivery, per UAV agent.
- **Average Speed:** is the average speed $[m/s]$ of each UAV agent.
- **Delivery Time Package:** is the delivery time $[s]$ measured from the time of request until the time of delivery.

## 5.3 Experiment A: Window Size

The first experiment focuses on investigating the impact of the window size used for WC-SIPP. The selected window size during path planning is a critical parameter that can significantly influence both the success rate and computational performance of the algorithm. A larger window size allows the operator to anticipate and plan for a larger part of upcoming dynamic obstacles, which can lead to fewer bottleneck situations and a higher success rate. However, this advantage comes with a possible increased computational cost, as planning over a larger time horizon with a larger set of constraints can be more computationally heavy. However, with a larger window size, replanning needs to occur less often, which could have a positive influence on the computational time. It is expected that if a path of an agent cannot be found easily and more nodes need to be explored to plan around imposed constraints, reducing the window size will have a positive impact on the computational time. However, if a path can be found relatively easily for agents, computing the motions during path finding will have the largest influence on the computational time, and therefore increasing the window size will lead to a lower computational time. Within the first experiment, we aim to select the most suitable window size for both the small and large urban environments. This is achieved by varying the window size (20, 30, 40) and the demand rate per medical facility (2, 4, 6). A window size of up to a maximum of 40 seconds is selected, as we would like to allow for the incorporation of urgent missions, we need to allow for timely change of mission prioritisation. With a maximum window size of 40, the agent is capable of travelling a maximum distance of 360 meters before replanning occurs. The other independent variables are kept constant, with 3 available layers, 25% urgency distribution for packages, mission type and request time priority planning order, and no introduced unexpected obstacles. The results of this experiment are presented in section 6.1. The hypotheses are described below and are tested with a significance level of $\alpha = 0.05$. Each hypothesis is applicable to all demand scenarios.

- **Hypothesis** $H_{A1}$: A window size of 40 seconds will lead to a higher success rate compared to window sizes of 20 and 30 seconds for WC-SIPP.
- **Hypothesis** $H_{A2}$: A window size of 40 seconds will lead to lower computational times compared to window sizes of 20 and 30 seconds for WC-SIPP.
- **Hypothesis** $H_{A3}$: A window size of 40 seconds will lead to a smaller number of explored nodes compared to window sizes of 20 and 30 seconds for WC-SIPP.

## 5.4 Experiment B: Multi-layer path planning

In the second experiment, we aim to investigate the scalability and performance of multi-layered path planning in 3D by varying the number of available layers for both the small and large environments. The scalability of the algorithm in multi-layered space is of interest, as it reflects the ability to handle larger and more intricate urban environments. By adjusting the number of layers (1, 3, 5) for various demand rates (2, 4, 6), we seek to analyse how the performance of the algorithm scales and assess the impact on the KPIs of the drone delivery system. The highest-performing window size, from experiment 1, will be selected for this and the following experiments. The other independent variables are kept constant. The results of this experiment are presented in section 6.2. The hypotheses are described below and are tested with a significance level of $\alpha = 0.05$. Each hypothesis is applicable to all demand scenarios.

- **Hypothesis** $H_{B1}$: Increasing the number of layers will lead to higher computational times.
- **Hypothesis** $H_{B2}$: Increasing the number of layers will lead to faster delivery times.
- **Hypothesis** $H_{B3}$: Path planning using multiple layers will lead to a higher success rate compared to path planning in a single layer.

## 5.5 Experiment C: Path planning in a dynamic environment

In the third experiment, we aim to evaluate the online path planning and coordination mechanism to effectively plan paths in dynamic environments. To achieve this, we introduce static obstacles into the environment, varying in number of obstacles introduced. The objective is to assess how well the algorithm is able to adapt to unforeseen obstacles, while maintaining efficient path planning for drone delivery missions. By simulating scenarios with strategically placed static obstacles, we aim to simulate real-world conditions, where the environment can change rapidly. Unexpected obstacles are placed at vertices that are frequently visited by UAVs. Unexpected obstacles are introduced at random time steps with $U\ (0, 3600)$, and are placed at strategically placed locations, to increase their impact on the operation. Obstacles placed at less frequently visited vertices have a lower impact. When an obstacle is scheduled to appear within the current planning window, the UAS Operator is made aware. This approach allows the operator to adjust flight plans accordingly. For this experiment, the number of unexpected obstacles (0, 5, 10, 15, 20) is varied. The obstacle size is set to be 1 unavailable vertex for a duration of 3 windows. The other independent variables are kept constant. It must be noted that for this experiment, we conduct a total of 200 simulations per scenario to obtain a consistent coefficient of variance. The results of this experiment are presented in section 6.3. The hypotheses are described below and are tested with a significance level of $\alpha = 0.05$. Each hypothesis is applicable to all demand scenarios.

- **Hypothesis** $H_{C1}$: Increasing the number of unexpected obstacles up to 20 leads to higher computational times.
- **Hypothesis** $H_{C2}$: Increasing the number of unexpected obstacles up to 20 leads to an increase in delivery time.

## 5.6 Experiment D: Priority Planning Order

In the fourth experiment, we aim to investigate the effect of different priority planning orders on the performance of the path planning algorithm for the drone delivery system. Specifically, we evaluate three different methods for prioritising UAVs and analyse the impact on the operations. In the previous experiments, the priority planning order was determined by considering the mission type of UAVs (urgent, standard) and their package request time. However, in this experiment, we will explore two different methods for prioritising UAVs. The first method involves prioritising UAVs solely based on their request time, allowing us to investigate the influence of prioritising mission types. The second method entails assigning UAVs a random priority order, providing insight into the effect of prioritising request time. Additionally, we will vary the distribution between urgent and standard packages to asses how different ratios of urgent to standard packages influence the performance of the priority planning ordering method. Three different urgency distributions are varied 10%, 25% and 50% for different demand rates and priority planning orders. The other independent constants are kept constant. The results of this experiment are presented in section 6.4. The hypotheses are described below and are tested with a significance level of $\alpha = 0.05$. Each hypothesis is applicable to all demand scenarios.

- **Hypothesis** $H_{D1}$: The distribution ratio between urgent and standard packages influences the delivery time of urgent missions compared to standard missions, when prioritising UAVs based on mission type and request time.
- **Hypothesis** $H_{D2}$: Prioritising UAVs based on mission type and request time improves the delivery time of UAVs with urgent missions compared to prioritising based solely on request time or random prioritisation.
- **Hypothesis** $H_{D3}$: Prioritising UAVs based on request time improves the delivery time of UAVs compared to random prioritisation

# 6 Results and Discussion

In this section, we present the findings from the experiments to evaluate the performance of our path planning and coordination mechanism for UAVs delivering medical supplies under varying conditions. The results are structured into four subsections, each dedicated to a specific experiment.

## 6.1 Experiment A:Window Size

The first experiment focuses on investigating the impact of the window size used for WC-SIPP on the algorithm's performance. In this experiment, we aim to select the most suitable window size for both small and large environments. In Figure 7 and Figure 8, the success rate of the algorithm is shown for both the small and large environments, respectively. The success rate is deemed to be the most critical KPI, as it demonstrates the ability of the MAPF algorithm to consistently find viable and collision-free paths for all UAVs within a simulation.

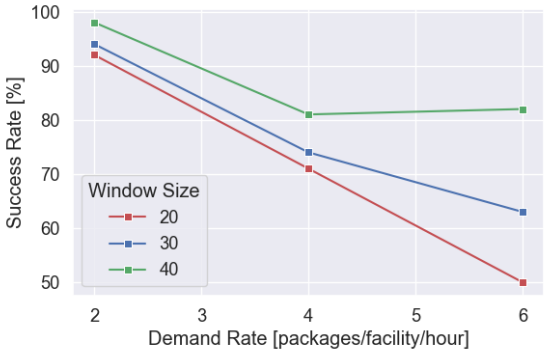


Figure 7: Simulation success rate for the small environment comparing different window sizes (20,30,40) for various demand rates

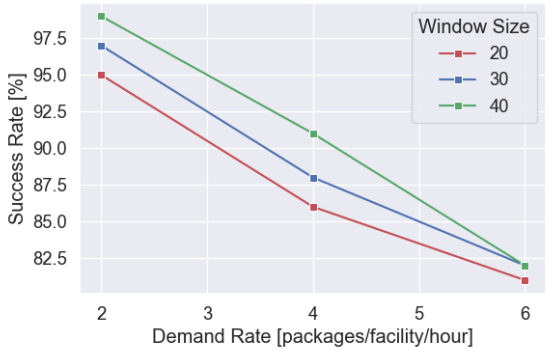


Figure 8: Simulation success rate for the large environment comparing different window sizes (20,30,40) for various demand rates

It can be seen that for both the small and large urban environment, a window size of 40 seconds outperforms the other window sizes in terms of success rate for all demand scenarios, which supports hypothesis $H_{A1}$. However, for the large environment with a demand rate of 6, the same success rate is obtained when comparing window sizes 30 and 40. Therefore, a closer look is taken into the computational performance and explored nodes during path planning for the large environment. The computational time and explored nodes for various window sizes and demand rates are visualised in Figure 9 and Figure 10. Here it is illustrated that for every demand scenario, a window size of 40 offers better computational performance compared to the other window sizes. A better computational performance is directly related to the fact that for a window size of 40, replanning occurs less frequently, and therefore the total number of explored nodes is less considering the complete path search. Even though more constraints are taken into account during path planning with a larger window size, the effect in increased explored nodes is marginal when compared to an increased replanning frequency. This indicates that a path can be found close to the optimal path, as the number of explored nodes does not increase significantly, when increasing the demand rate. The computational time and explored nodes results for various window sizes and demand rates for the small environment can be found in Part III, Supporting Work, Chapter 4.

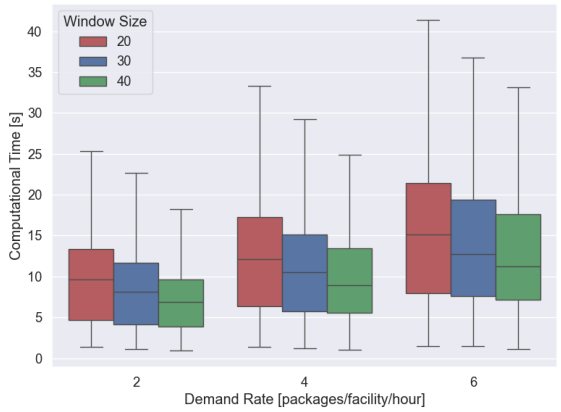


Figure 9: Computational time with varying window sizes and demand rates for the large environment

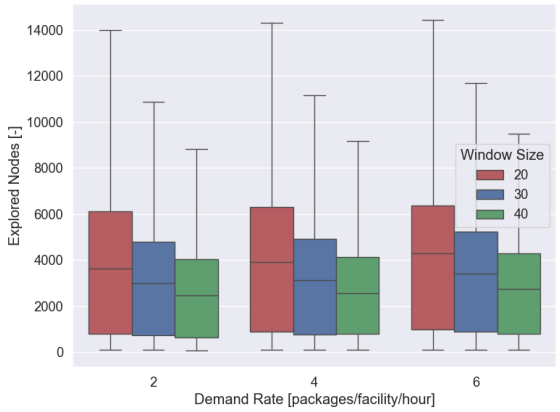


Figure 10: Explored nodes during path search with varying window sizes and demand rates for the large environment

In order to test the statistical significance of the experiment results, the Kruskal Wallis statistical test [30] is used. This test is utilised as the data from 3 data groups is numerical, not normally distributed and unpaired. The p-values for both environments and all demand scenarios are given in Table 1. Green values indicate that there is a significant difference is obtained, with p-value $< 0.05$, while red values indicate there is no significant difference obtained, p-value $> 0.05$. In Table 1 it can be seen that for both environments and all demand scenarios, statistical significance is obtained, except for one scenario. This is for the small environment and highest demand rate when considering the computational time KPI. This could partially be attributed to the fact that for the small environment with the highest demand rate, the number of UAVs operating at the same time in close proximity is larger compared to other scenarios. The UAS Operator ensures safety separation during path planning, which involves checking for violations in separation distances for each possible motion. Checking for violations in safety separation occurs during the generation of successors, as described in Figure 6. This process relies on calculating the distance between the exact location of active UAVs. The locations of active UAVs are stored in a separation list, which needs to be searched for every time step per possible motion. This process is computationally inefficient, and could potentially lead to prolonged computational times, as larger window sizes require the storage of more constraints. However, further research is necessary to confirm this explanation and determine the effect of this function.

Besides the Kruskal-Wallis statistical test [30], the Vargha-Delaney [31] effect size test is also conducted to assess the magnitude of differences between the different results. Using the Vargha-Delaney test a small effect size is observed when considering the computational time and explored nodes for a window size of 20 and 40. This supports the observed results in Figure 9 and Figure 10. Therefore, we accept both $H_{A2}$ and $H_{A3}$.

Table 1: Statistical analysis comparing the performance of the algorithm for window sizes 20, 30, and 40, varying the environment type and demand rate. Values given in the table are p-values obtained from the Kruskal Wallis statistical test

| **Environment** | Small | | | Large | | |
|---|---|---|---|---|---|---|
| **Demand rate** | 2 | 4 | 6 | 2 | 4 | 6 |
| **KPI** | | | | | | |
| Computational Time | 3.4E-06 | 6.1E-03 | 0.62 | 3.7E-08 | 1.9E-12 | 3.7E-14 |
| Explored Nodes | 1.0E-04 | 4.3E-08 | 6.7E-11 | 1.4E-07 | 5.0E-15 | 1.4E-25 |

## 6.2 Experiment B: Multi-layer path planning

For the second experiment, the focus is on investigating the scalability and performance of multi-layer path planning. This is tested by varying the number of available layers for various demand rates for both environments. In this section, we primarily focus on the results from the large environment, as this environment better displays the scalability and performance of multi-layer path planning. The results for the small environment can be found in Part III, Supporting Work, Chapter 5. Figure 11 and Figure 12 illustrate the effect on the computational time and explored nodes for the large environment. It is evident from these results that as the number of layers increases, the computational performance does not scale well.

A statistical analysis is conducted using the Kruskal-Wallis statistical test [30] for various demand rates and environments, comparing path planning for different numbers of available layers, which can be seen in Table 2. Table 2 supports our finding, showing that as the number of layers increases, computational times significantly increases across all demand rates. Therefore, we accept $H_{B1}$. Besides the Kruskal-Wallis statistical test [30], the Vargha-Delaney [31] effect size test is also conducted to assess the magnitude of differences between the different results. Using this test, it can be concluded for computational time KPI the effect size is large for all scenarios when increasing the number of layers. For the explored nodes KPI, the effect size is normal to large. An overview of the A-values can be found in Part III, Supporting Work, Chapter 5.

The increase in computational time for varying the number of layers can be explained by considering the number of explored nodes, as shown in Figure 12. As the number of layers increases, the number of explored nodes increases rapidly, resulting in poor computational performance. This indicates that the current heuristic does not search the 3D urban environment graph in an efficient manner.

Moreover, we can see in Table 2 that for the other KPIs only a significant difference is observed for the average speed for both the small and large environment, except for the lowest demand rate. However, when using the Varhga-Delaney effect size test, the effect size of increasing the number of layers is negligible on the average speed. Furthermore, it can be noticed that the delivery time KPI is not significantly improved, which leads to rejecting $H_{B2}$.

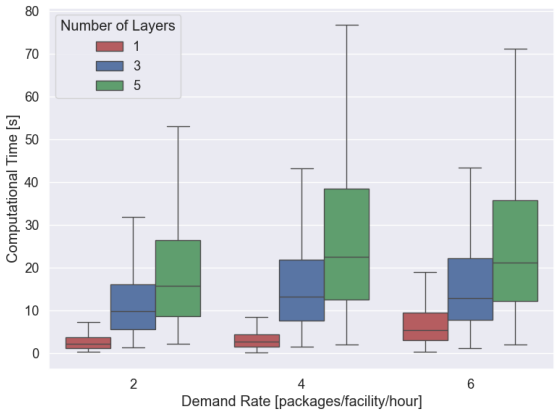


Figure 11: Computational time with varying number of layers and demand rates for the large environment

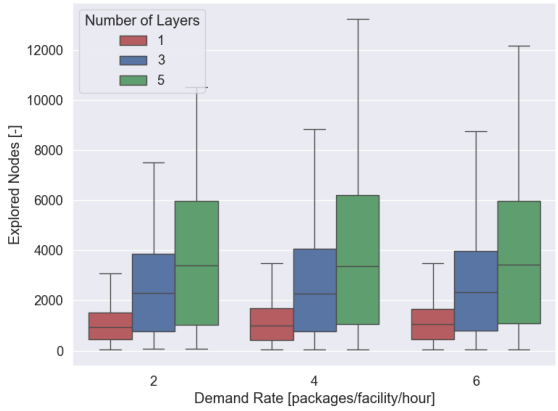


Figure 12: Explored nodes during path search with varying number of layers and demand rates for the large environment

Table 2: Statistical analysis comparing the performance of the algorithm for different numbers of available layers, varying the demand rates. Values given in the table are p-values obtained from the Kruskal Wallis statistical test

| **Environment** | Small | | | Large | | |
|---|---|---|---|---|---|---|
| **Demand Rate** | 2 | 4 | 6 | 2 | 4 | 6 |
| **KPI** | | | | | | |
| Computational Time | 3.4E-81 | 2.1E-185 | 0.0E+00 | 6.8E-155 | 0.0E+00 | 4.6E-297 |
| Explored Nodes | 1.8E-27 | 0.0E+00 | 0.0E+00 | 4.2E-41 | 8.8E-80 | 8.6E-127 |
| Travel Distance | 0.85 | 0.97 | 0.71 | 0.96 | 0.91 | 0.93 |
| Travel Time | 0.94 | 0.74 | 0.86 | 0.98 | 0.92 | 0.86 |
| Average Speed | 0.01 | 0.02 | 1.1E-05 | 0.17 | 1.9E-06 | 1.9E-11 |
| Delivery Time | 0.86 | 0.96 | 0.95 | 0.98 | 0.96 | 0.96 |

Besides the scalability and performance of the path planning and coordination algorithm, it is also interesting to analyse the behaviour of UAV agents in multi-layer path planning. An overview of the travel time spent in each layer for various number of available layers is given in Table 3. In this overview, it can be seen that when increasing the number of available layers to operate in, the UAV agent prefers to still fly predominantly in the lowest altitude layer. This behaviour is expected as both the warehouse and medical facility locations are located in this layer. In Table 4 the number of average layer changes per simulation is given for both the

small and large environments. The number of layer changes is counted per change of layer, indicating that a change from layer 1 to layer 2 and back to layer 1 is counted as 2 layer changes. In Table 4, it can be observed that the average number of layer changes is small. At first glance, it may seem unnecessary to incorporate a multi-layer environment, as the UAVs predominantly fly in the bottom layer, and the number of layer changes is small. However, upon examining the success rate for various number of available layers for both environments, it becomes apparent that the inclusion of more than one layer is warranted. The success rate for both the small and large environment is shown in Figure 13 and Figure 14, respectively. It can be observed that the success rate is lower for the 1 layer environment compared to the multiple vertical layer environments, which supports $H_{B3}$. Therefore, it can be concluded that the upper layers are mainly used to avoid conflicts and are only visited shortly. After avoiding a conflict, the UAVs are returning to the bottom layer.

Table 3: Overview of travel time spent in each layer for a varying number of available layers (1,3,5) for both the small and large environment.

| | | **Time spent [%]** | | | | |
|---|---|---|---|---|---|---|
| **Environment** | **Layers** | **Layer 1** | **Layer 2** | **Layer 3** | **Layer 4** | **Layer 5** |
| Small | 1 | 100 | - | - | - | - |
| Small | 3 | 96.0 | 3.5 | 0.5 | - | - |
| Small | 5 | 94.3 | 3.7 | 1.4 | 0.6 | 0 |
| Large | 1 | 100 | - | - | - | - |
| Large | 3 | 98.2 | 1.4 | 0.4 | - | - |
| Large | 5 | 98.0 | 1.6 | 0.3 | 0.09 | 0.0009 |

Table 4: Overview of the average number of layer changes per simulation for varying number of available layers for both the small and large environment

| | | **Changes between layers** | | | |
|---|---|---|---|---|---|
| **Environment** | **Layers** | **Layers 1-2** | **Layers 2-3** | **Layers 3-4** | **Layers 4-5** |
| Small | 3 | 8.9 | 1.9 | - | - |
| Small | 5 | 8.7 | 4.0 | 2.1 | 0 |
| Large | 3 | 6.5 | 2.7 | - | - |
| Large | 5 | 7.7 | 3.5 | 1.5 | 0.01 |

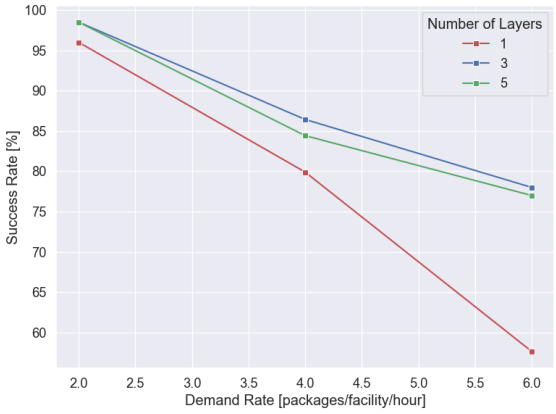


Figure 13: Simulation success rate for the small environment comparing different number of available layers (1,3,5) for various demand rates

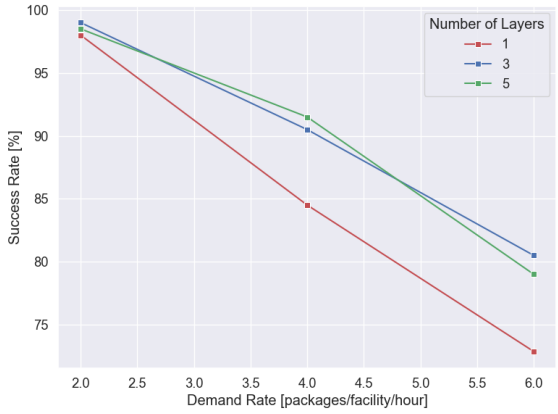


Figure 14: Simulation success rate for the large environment comparing different number of available layers (1,3,5) for various demand rates

Lastly, in order to analyse the flown routes and average speed for UAV agents, heat maps are constructed. In this section, only the results from the large environment containing 1 layer are shown to demonstrate the behaviour of UAV agents. The results for the small and large environments with various number of available layers can be found in Part III, Supporting Work, Chapter 5. The first heat map, presented in Figure 15, shows the average number of times nodes and edges are visited across all demand scenarios. The second heat map, presented in Figure 16, shows the average speed across all demand scenarios. These figures illustrate that UAV

agents prefer to fly direct routes from warehouse locations to medical facilities and vice versa. Parts of the environment that contain many changes of directions and that are not directly on the route of agents are visited less frequently. This can be explained as the flying speed of UAVs is more restricted in these parts, which would result in longer travel times. This behaviour is expected as the objective of the UAS Operator agent is to plan paths of UAVs with minimal travel time. It is also evident that certain parts of the environment are never visited, as they are not directly on the route of agents. In our MAS, locations of warehouses have been selected more arbitrarily, due to the absence of existing warehouses. It is anticipated that future warehouses will be situated on the periphery of the city. Analysis of Figure 15 suggests that the placement of warehouses impacts the flown routes by UAVs. Therefore, it is imperative to consider these routes when determining warehouse locations.

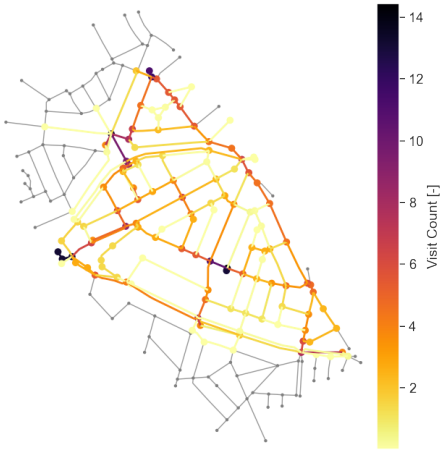


Figure 15: Heat map of the flown paths for the large environment with 1 available layer, for all demand scenarios, showing the number of times a node and edge is visited.

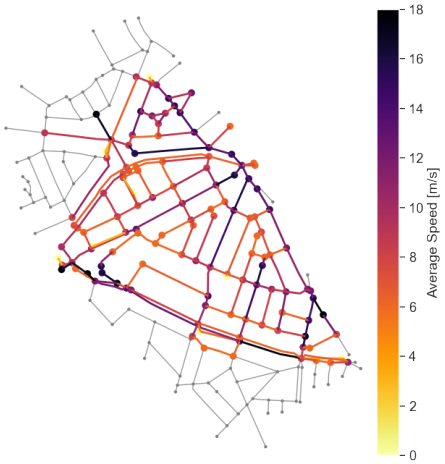


Figure 16: Heat map of the flown paths for the large environment with 1 available layer, for all demand scenarios, showing the average speed per node and edge.

## 6.3 Experiment C: Path planning in a dynamic environment

For the third experiment, we aim to assess the effectiveness of online path planning and coordination in dynamically changing environments. To accomplish this, we introduce static obstacles varying in number. Static obstacles are strategically placed and have a size of 1 vertex and a duration of 3 windows (120 seconds). For the locations of unexpected obstacles, the reader is referred to Part III, Supporting Work, Chapter 6. The number of unexpected obstacles introduced ranges from 0 to 20 obstacles, with incremental steps of 5. The goal is to evaluate the algorithm's ability to adjust to unexpected obstacles while ensuring efficient path planning for drone delivery. Figure 17 and Figure 18 show the success rate of the simulation model for various demand rates with an increasing number of unexpected obstacles introduced into the environment. As can be seen from both figures the success rate does not decrease significantly with an increasing number of unexpected obstacles for a constant demand rate.

Next, we analyse the performance KPIs of the algorithm when exposed to an increasing number of unexpected obstacles. A statistical analysis is conducted using the Kruskal-Wallis statistical test [30] for various demand rates, for which the results can be seen in Table 5. Here it is demonstrated that for a varying number of unexpected obstacles, only a significant difference can be observed for the computational time KPI for both environments and all demand rates. For the average speed KPI, only for the small environment, a significant difference can be observed. However, when using the Varhga-Delaney effect size test [31], the observed effect size for the average speed KPI, is negligible. For the computational time, a negligible effect is observed, except for the large environment with a demand rate of 2. Here a small effect is observed. An overview of the results of the Vargha-Delaney test is shown in Table 6. Combining these insights, we reject both hypotheses $H_{C2}$ and $H_{C3}$.

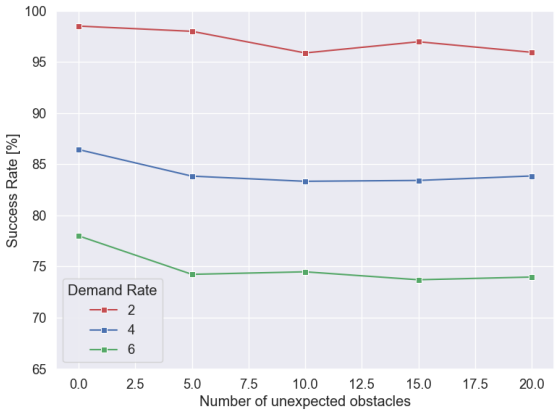


Figure 17: Simulation success rate for the small environment for various demand rates and number of unexpected obstacles

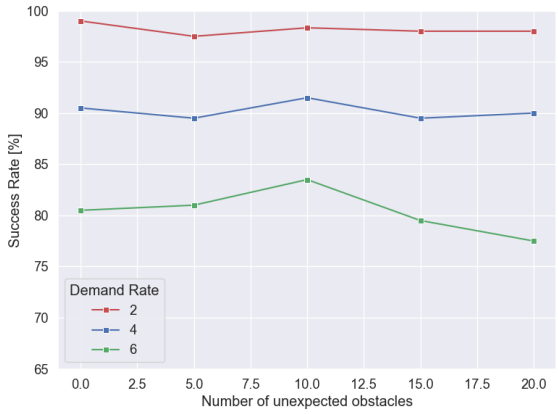


Figure 18: Simulation success rate for the large environment for various demand rates and number of unexpected obstacles

Table 5: Statistical analysis comparing the performance of the algorithm for different numbers of introduced unexpected obstacles, varying the demand rates. Values given in the table are p-values obtained from the Kruskal Wallis statistical test

| **Environment** | Small | | | Large | | |
|---|---|---|---|---|---|---|
| **Demand Rate** | 2 | 4 | 6 | 2 | 4 | 6 |
| **KPI** | | | | | | |
| Computational Time | 1.6E-05 | 1.5E-16 | 3.0E-04 | 8.4E-29 | 6.6E-34 | 3.4E-34 |
| Explored Nodes | 0.94 | 0.88 | 0.93 | 1.00 | 1.00 | 1.00 |
| Travel Distance | 0.98 | 0.97 | 0.95 | 1.00 | 1.00 | 1.00 |
| Travel Time | 0.86 | 0.64 | 0.62 | 1.00 | 1.00 | 0.96 |
| Average Speed | 4.81E-02 | 2.93E-03 | 1.50E-05 | 0.95 | 0.96 | 0.53 |
| Delivery Time | 1.00 | 1.00 | 0.99 | 1.00 | 1.00 | 0.92 |

Table 6: Statistical analysis comparing the performance of the algorithm for 0 and 20 introduced unexpected obstacles, varying the demand rates. Values given in the table are A-values obtained from the Vargha Delaney effect size test

| **Environment** | Small | | | Large | | |
|---|---|---|---|---|---|---|
| **Demand Rate** | 2 | 4 | 6 | 2 | 4 | 6 |
| **KPI** | | | | | | |
| Computational Time | 0.55 N | 0.5 N | 0.45 N | 0.59 S | 0.54 N | 0.52 N |
| Average Speed | 0.55 N | 0.55 N | 0.55 N | 0.5 N | 0.5 N | 0.51 N |

Next, the behaviour of UAV agents is analysed when introducing up to 20 unexpected obstacles into the environment. In Table 7 an overview is given of the travel time spent in each layer for both the small and large environment when introducing 0 and 20 unexpected obstacles. It can be observed that in case 20 obstacles are introduced into both environments, the travel time spent in the bottom layer is reduced, while the travel time spent in the second layer is increased. UAV agents briefly move to the second layer in order to avoid the unexpected obstacles, that are primarily introduced into the bottom layer. This effect can also be observed when measuring the number of layer changes, which is given in Table 8. The number of layer changes from the bottom layer to the second layer is increased on average with 42.3% and 38.3% for the small and large environments, respectively. The increased time spent in the second layer also becomes evident when comparing the heat maps of the flown paths for the small environment. The heat maps of the second flight layer for the small environment with 0 and 20 introduced unexpected obstacles are shown in Figure 19 and Figure 20. It can be observed that the nodes and edges in this layer are visited more frequently when introducing 20 obstacles compared to 0 obstacles.

Table 7: Overview of travel time spent in each layer for both the small and large environment with 0 and 20 unexpected obstacles.

| | Time spent [%] | | | | | |
|---|---|---|---|---|---|---|
| | 0 Obstacles | | | 20 Obstacles | | |
| **Environment** | **Layer 1** | **Layer 2** | **Layer 3** | **Layer 1** | **Layer 2** | **Layer 3** |
| Small | 96.0 | 3.5 | 0.5 | 94.6 (-1.4) | 4.8 (+1.3) | 0.6 (+0.1) |
| Large | 98.2 | 1.4 | 0.4 | 97.7 (-0.5) | 1.9 (+0.5) | 0.4 |

Table 8: Overview of the number of layer changes per simulation for both the small and large environment with 0 and 20 unexpected obstacles.

| | Changes between layers | | | |
|---|---|---|---|---|
| | 0 Obstacles | | 20 Obstacles | |
| **Environment** | **10-20** | **20-30** | **10-20** | **20-30** |
| Small | 8.8 | 1.9 | 12.6 (+3.8) | 2.1 (+0.2) |
| Large | 6.5 | 2.7 | 8.9 (+2.4) | 2.8 (+0.1) |

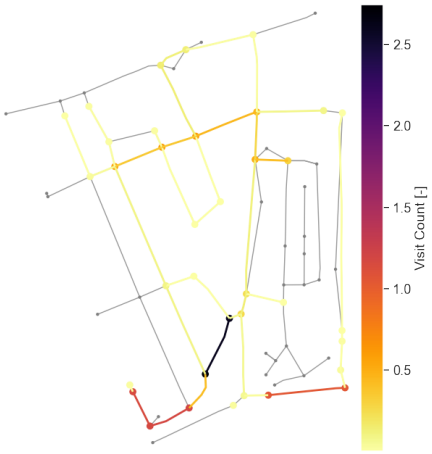


Figure 19: Heat map of the flown paths for the large environment with 3 available layers, for all demand scenarios, showing the number of times a node and edge is visited for layer altitude 20 with 0 unexpected obstacles

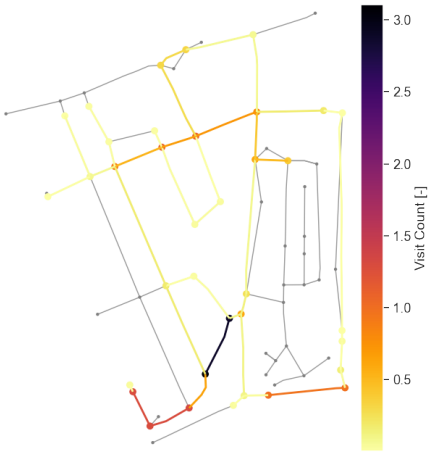


Figure 20: Heat map of the flown paths for the large environment with 3 available layers, for all demand scenarios, showing the number of times a node and edge is visited for layer altitude 20 with 20 unexpected obstacles

## 6.4 Experiment D: Priority Planning Order

For the fourth and last experiment, we focus on investigating the effect of different priority planning in order to assess the performance of the path planning algorithm for the drone delivery system. The experiments are only performed considering the large environment. Three different methods for prioritising UAVs are analysed, which are mission type and request time (MTRT) prioritisation, request time (RT) prioritisation, and random (R) prioritisation. Additionally, we will vary the distribution between urgent and standard packages (10%, 25%, 50%). In this experiment, we first investigate if prioritising urgent missions over standard missions leads to a significant difference in UAV performance. A statistical analysis is conducted using the Mann-Whitney U test [32] for various demand rates and urgency distributions, for which an overview of the calculated p-values is given in Table 9. Here it can be seen that for various demand rates and urgency distribution, no significant difference in UAV performance can be observed. Therefore, we reject $H_{D1}$. This indicates that assigning priority based on mission type does not result in a significant difference in UAV performance in our MAS when comparing travel distance, travel time, average speed, and delivery time. This could be attributed to the fact that UAVs

have the advantage of avoiding conflicts by changing layers for a short period of time. Changing altitude during operation only leads to a small increase in travel distance, travel time and delivery time, which is deemed to be insignificant in our operations. For a visual comparison measuring the difference in performance of UAVs with urgent and standard missions, the reader is referred to Part III, Supporting Work, Chapter 7.

Table 9: Statistical analysis comparing UAV performance for urgent and standard missions for different package urgency distribution and demand rates. The priority planning order method used is mission type and request time prioritisation. Values given in the table are p-values obtained from the Mann-Whitney U statistical test

| **Demand Rate** | | 2 | | | 4 | | | 6 | |
|---|---|---|---|---|---|---|---|---|---|
| **Urgency Distribution** | 10 | 25 | 50 | 10 | 25 | 50 | 10 | 25 | 50 |
| **KPI** | | | | | | | | | |
| Travel Distance | 0.78 | 0.52 | 0.69 | 0.62 | 0.23 | 0.37 | 0.7 | 0.98 | 0.41 |
| Travel Time | 0.72 | 0.44 | 0.98 | 0.78 | 0.32 | 0.39 | 0.71 | 0.89 | 0.38 |
| Average Speed | 0.97 | 0.9 | 0.3 | 0.45 | 0.84 | 0.29 | 0.87 | 0.74 | 0.93 |
| Delivery Time | 0.47 | 0.25 | 0.14 | 0.93 | 0.27 | 0.63 | 0.88 | 0.099 | 0.79 |

Furthermore, we investigate if prioritising UAVs based on mission type and request time significantly improves the performance of UAVs with urgent missions compared to prioritising based solely on request time or random prioritisation. A statistical analysis is conducted using the Kruskal Wallis test [30] for various demand rates and urgency distributions, for which an overview is given in Table 10. Here it can be seen that for various demand rates and urgency distribution, no significant difference is observed for prioritising urgent mission types when compared to assigning prioritisation based solely on request time and compared to assigning random prioritisation. This leads to rejecting $H_{D2}$. Similar to the previous investigation, this result could be attributed to the fact that UAVs have the advantage of quickly changing layers to avoid conflicts, with no significant increases in travel distance, travel time and delivery time.

Table 10: Statistical analysis comparing UAV urgent mission performance for different priority planning ordering methods (MTRT, RT, R) for different package urgency distribution and demand rates. Values given in the table are p-values obtained from the Kruskal Wallis statistical test

| **Demand Rate** | | 2 | | | 4 | | | 6 | |
|---|---|---|---|---|---|---|---|---|---|
| **Urgency Distribution** | 10 | 25 | 50 | 10 | 25 | 50 | 10 | 25 | 50 |
| **KPI** | | | | | | | | | |
| Travel Distance | 0.97 | 0.85 | 0.96 | 0.79 | 0.93 | 0.57 | 0.78 | 0.65 | 0.83 |
| Travel Time | 0.91 | 0.83 | 0.94 | 0.62 | 0.88 | 0.25 | 0.60 | 0.32 | 0.7 |
| Average Speed | 0.97 | 0.94 | 0.76 | 0.94 | 0.36 | 0.83 | 0.62 | 0.37 | 0.74 |
| Delivery Time | 0.95 | 0.99 | 0.82 | 0.98 | 0.96 | 0.99 | 0.98 | 0.99 | 0.98 |

Last, we investigate if prioritising UAVs based on request time significantly improves the performance of UAVs compared to assigning prioritisation randomly. A statistical analysis is conducted using the Mann-Whitney U statistical test [32], where we compare the UAV performance for request time prioritisation to random prioritisation for various demand rates. An overview of the p-values can be seen in Table 11. In this overview, it can be seen that there is a significant difference in travel time and average speed when considering a demand rate of 4. When considering a demand rate of 6, the travel distance, travel time, and average speed are significantly different. However, when using the Vargha-Delaney A-test [31], a negligible effect size is measured. This indicates that even though there is a significant difference for the specific cases, the increase in performance is negligible. Therefore, we reject $H_{D3}$. Combining the findings from this experiment, it can be concluded that in our model of operations, assigning priority based on either mission type or request time does not lead to improved performance.

Table 11: Statistical analysis comparing UAV performance for different priority planning ordering methods (RT, R) for different demand rates. Values given in the table are p-values obtained from the Mann-Whitney U statistical test and A-values obtained from the Vargha-Delaney effect size test

| | p-value | | | A-value | | |
|---|---|---|---|---|---|---|
| **Demand Rate** | 2 | 4 | 6 | 2 | 4 | 6 |
| **KPI** | | | | | | |
| Travel Distance | 0.43 | 0.13 | 0.02 | 0.51 | 0.51 | 0.51 |
| Travel Time | 0.24 | 0.01 | 4.5E-04 | 0.51 | 0.52 | 0.52 |
| Average Speed | 0.30 | 1.9E-03 | 7.7E-07 | 0.51 | 0.52 | 0.53 |
| Delivery Time | 0.88 | 0.73 | 0.68 | 0.50 | 0.50 | 0.50 |

## 6.5 Discussion

This section discusses the obtained results from each experiment. We aim to gain valuable insights and understand the implications of our findings within the large scope of this context. In total four different experiments were conducted to test different aspects of the multi-agent system. For the first experiment, we aimed to investigate the impact of selecting different window sizes for WC-SIPP. For this experiment, it was demonstrated that an increasing window size leads to a higher success rate. The operator is able to plan UAVs by taking more constraints into account, leading to fewer bottleneck situations. From this experiment, it became evident that a decrease in replanning frequency led to a lower computational time. This is noteworthy, as Silver [21] demonstrates that for an increase in window size, a higher computational time is measured. In our work, calculating the trajectory is computationally more heavy compared to taking into account constraints from other agents, leading to this result. For further research, it needs to be explored at which point taking into account constraints from other agents would have computational drawbacks by increasing the window size and demand rate.

Furthermore, we investigated the scalability and performance of path planning in multiple layers. It became evident that an increase in the number of available layers leads to an increase in computational time for both the small and large environments. When comparing the results between the small and large environments, an increase in computational time can be observed. This indicates that as the search space grows larger, the computational time increases rapidly. The main reason for an increase in computational time when varying the number of layers is the rapid increase in expanded nodes. Even though the number of available layers has increased, the delivery time of UAVs has not improved. Combining these insights, improvements are needed for the heuristic function, in order to allow for a more efficient search in the search space. The current Euclidean heuristic incorporates both time and distance in equal weights. Using a more advanced heuristic could therefore reduce the number of explored nodes and decrease the computational time when searching in 3D space. A possibility would be to adjust the algorithm to primarily search in the bottom layer and only search in upper layers when conflicts arise, as the majority of operations occur in the bottom layer. The upper layer however is required in our environment, as planning in a single vertical layer environment leads to a decrease in success rate.

From this experiment it also became evident that in operations where UAVs are predominantly flying in the bottom layer and using the upper layers to avoid conflicts, a horizontal spread is more beneficial in terms of success rate of operations compared to a vertical spread. This can be observed when comparing the success rate from the small environment with the large environment. It must be noted that the small environment contains 4 medical facilities in an area of 500 by 500 meters, while the large environment only contains 5 medical facilities in an area of 750 by 1500 meters. When comparing the success rate for both environments, it can be observed that the success rate is better for the large environment, even though more operations are considered. This can be attributed to the fact that UAVs are spread more across the environment during operations and can therefore more easily avoid conflicts and keep horizontal distance. When comparing an increase in the number of available layers, no increase in success rate can be observed when comparing the 3 vertical layer environment to the 5 vertical layer environment, demonstrating that a vertical spread in our operations is not as important as having a horizontal spread.

Moreover, when considering UAV operations for medical last-mile delivery from warehouse locations to healthcare facilities in urban environments, it can be expected that larger urban environments will need to be considered for operations. This aligns with the work from Ho et al.[23], which examined a 14.35 by 17.1-kilometer area with two planning layers and 30x30 meter voxels. Our model offers improved accuracy in representing paths near delivery locations. However, as the environment size expands, careful consideration of the environment representation as well as the number of available vertical layers is essential to maintain computational feasibility.

Besides, we demonstrated the capability of the online path planning and coordination mechanism to effectively plan paths in dynamic environments. By introducing up to 20 static obstacles into the environment, we demonstrated that the success rate does not decrease significantly. It must be noted for this experiment only one type of static obstacle is introduced to demonstrate the capability of the algorithm to plan UAVs in a dynamic environment. However, integrating more realistic uncertainty factors like weather conditions and changing regulatory restrictions, such as imposed no-fly zones, is needed to assess better the capability of the algorithm to plan in dynamic and more realistic environments. Our model also assumes that the UAV is able to perfectly follow the given trajectory, which is also an assumption that needs to be relaxed for real-world implementation.

For the final experiment, we investigated the influence of prioritising UAVs based on mission type and request time. It became evident that for our model, prioritisation of agents based on either request time or mission type did not influence the performance of the delivery model. Therefore, if an operator is considering the delivery of short-distance packages, mission or request time prioritisation does not offer a decrease in delivery time. However, it must be noted that the operations in this model are simplified, as only a time window of 1 hour is considered with each UAV only performing one mission. Further analysis would be needed to assess if UAVs performing multiple missions during a larger operation time window, would still lead to insignificant differences in the performance of UAVs when prioritising them based on mission type and request time.

To summarize, the proposed path planning and coordination mechanism, WC-SIPP, is able to plan paths and coordinate UAV agents in multi-layer 3D urban environments considering the delivery of medical supplies. Planning in a multi-layer environment is beneficial for UAVs to avoid conflicts, while not increasing the delivery time of packages. However, the total number of available layers should be considered carefully, as it leads to a rapid increase in computational time. The proposed method is able to plan paths in a dynamic environment including replanning during flight due to changes in mission priority and introduction of unexpected obstacles. This makes it suitable for pre-flight conflict detection and resolution by incorporating in-flight replanning. In our model prioritising based on either mission type or request time did not improve the delivery time of UAVs for simplified missions. However, when considering more realistic and more complex operations, this result may not remain valid.

# 7 Conclusions & Recommendations

This paper presents and evaluates an online path planning coordination mechanism in 3D for a cooperative fleet of autonomous UAVs delivering medical supplies in the Rotterdam Area. The Rotterdam area is selected as a use case, but the model can be adapted to other cities. We propose a multi-agent system for the delivery of both standard and urgent medical supplies. Multi-agent pathfinding in a 3D environment is used to model the paths of UAVs for a medical drone delivery model. Our approach utilises Windowed Cooperative Safe Interval Path Planning, where kinematic constraints of UAVs are incorporated during path planning, as well as separation distances between UAVs. Using two different neighbourhoods in Rotterdam as case studies, we first demonstrated the importance of selecting an appropriate window size for operations, as it is key for reducing the occurrence of bottlenecks situations during simulations.

Second, we demonstrated the ability of the algorithm to plan paths in multiple flight layers, showing that an increase in the number of layers does not influence the performance of operating UAVs. UAVs operate predominantly in the bottom layer and use other layers to avoid conflicts. A multi-layered environment therefore also outperforms a single-layer environment in terms of success rate. However, an increase in the number of available layers comes with a significant increase in computational time.

Third, we showed the capability of the algorithm to plan and coordinate paths of UAVs within a dynamic environment by introducing up to 20 unexpected obstacles. The obstacles are primarily introduced at frequently visited vertices and by making use of layer changes the path planning and coordination mechanism is able to successfully plan around these obstacles.

Last, we demonstrated that for our operation model, assigning prioritisation based on either mission type or request time did not significantly influence the performance of UAVs. To conclude, our analysis has shown the promising potential of using WC-SIPP for online multi-agent pathfinding in a 3D urban environment. The model is adaptable to other urban environments and can be utilised to model the path planning and coordination of a cooperative fleet of autonomous UAVs considering multiple warehouse and healthcare locations.

While our research demonstrated the implementation of an online path planning and coordination mechanism for medical urban air mobility services in a realistic 3D environment, further research is necessary. Currently,

the urban environment is represented by the street network in the selected environments. Building representations in the form of keep-out polygons should be included in the environment, to allow for a more realistic urban representation. Besides, scaling to a larger urban environment would allow for a more realistic operational environment. The operations in our model are simplified and the incorporation of better task allocation methods and more complex medical missions could provide further insight into the developed framework. To allow for more complex operations, the computational performance of path planning in multi-layered environments needs to be addressed. Using a more advanced heuristic could reduce the number of explored nodes and decrease the computational time when searching in 3D space.

Besides, to allow for a more accurate representation of the travelled paths of UAVs, the kinematics of UAVs can be improved. Improvements can be made by considering turn radius limitations, reduced restrictions for edge motions, and rotational movements.

Moreover, investigating diverse drone types and including battery performance characteristics is important for analysing UAV delivery services, especially when considering an increase in urban environment size. Integrating this would allow for more accurately represented operations. The operation model could also be improved by incorporating a refined demand model based on the healthcare infrastructure, which goes beyond the current Poisson distribution. Lastly, integrating more realistic uncertainty factors like weather conditions and regulatory restrictions is essential to operate in dynamic environments effectively.

# References


[1] A. Cornell, S. Mahan, and R. Riedel, "Commercial drone deliveries are demonstrating continued momentum in 2023," 10 2023, accessed on 26-03-2024. [Online]. Available: https://www.mckinsey.com/industries/aerospace-and-defense/our-insights/future-air-mobility-blog/commercial-drone-deliveries-are-demonstrating-continued-momentum-in-2023

[2] Zipline, "Welcome to the best delivery experience not on earth." 2023, accessed on 26-03-2024. [Online]. Available: https://www.flyzipline.com

[3] Wing Aviation LLC, "Better delivery." 2023, accessed on 26-03-2024. [Online]. Available: https://wing.com

[4] K. Korosec, "Zipline is now the national drone service provider for rwanda," December 2022, accessed on 26-03-2024. [Online]. Available: https://techcrunch.com/2022/12/15/zipline-is-now-the-national-drone-service-provider-for-rwanda/

[5] European Union Aviation Safety Agency (EASA), "Study on the societal acceptance of urban air mobility in europe," EASA, Tech. Rep., 03 2021.

[6] Federal Aviation Administration, "Unmanned aircraft system traffic management: Concept of operations v2.0," FAA, Tech. Rep., 03 2020.

[7] B. Rabta, C. Wankmüller, and G. Reiner, "A drone fleet model for last-mile distribution in disaster relief operations," *International Journal of Disaster Risk Reduction*, vol. 28, pp. 107–112, 2018.

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

[9] P. Kitjacharoenchai, B. Min, and S. Lee, "Two echelon vehicle routing problem with drones in last mile delivery," *International Journal of Production Economics*, vol. 225, p. 107598, 2020.

[10] X. Shang, G. Zhang, B. Jia, and M. Almanaseer, "The healthcare supply location-inventory-routing problem: A robust approach," *Transportation Research Part E: Logistics and Transportation Review*, vol. 158, p. 102588, 2022.

[11] U. Shakir, "Amazon is shutting down its drone delivery service in california as it looks to other markets," 2024, accessed on 07-05-2024. [Online]. Available: https://www.theverge.com/2024/4/22/24137383/amazon-prime-air-drone-delivery-closing-lockeford-california-phoenix-arizona

[12] R. Stern, *Multi-Agent Path Finding – An Overview*. Cham: Springer International Publishing, 2019, pp. 96–115.

[13] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 5628–5635.

[14] T. Standley and R. Korf, "Complete algorithms for cooperative pathfinding problems," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, ser. IJCAI'11. AAAI Press, 2011, p. 668673.

[15] T. Standley, *Finding Optimal Solutions to Cooperative Pathfinding Problems*, ser. AAAI'10. AAAI Press, 2010, p. 173178.

[16] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *Proceedings of (IROS) IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2011, pp. 3260 – 3267.

[17] Z. Ali and K. Yakovlev, "Safe interval path planning with kinodynamic constraints," *arXiv e-prints*, pp. arXiv–2302, 2023.

[18] C. Zhou, B. Huang, and P. Fränti, "A review of motion planning algorithms for intelligent robots," *Journal of Intelligent Manufacturing*, vol. 33, no. 2, pp. 387–424, 2022.

[19] G. Sharon, R. Stern, A. Felner, and N. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[20] H. Ma, D. Harabor, P. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 7643–7650, Jul. 2019.

[21] D. Silver, "Cooperative pathfinding." *Proceedings of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference, AIIDE 2005*, pp. 117–122, 01 2005.

[22] F. Ho, A. Gonçalves, B. Rigault, R. Geraldes, A. Chicharo, M. Cavazza, and H. Prendinger, "Multi-agent path finding in unmanned aircraft system traffic management with scheduling and speed variation," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 5, pp. 8–21, 2021.

[23] F. Ho, A. Goncalves, A. Salta, M. Cavazza, R. Geraldes, and H. Prendinger, "Multi-agent path finding for uav traffic management: Robotics track," *Autonomous Agents and Multiagent Systems 2019*, 2019.

[24] A. Morfin Veytia, C. Badea, J. Ellerbroek, J. Hoekstra, N. Patrinopoulou, I. Daramouskas, V. Lappas, V. Kostopoulos, A. Vidosavljevic, J. van Ham, E. Sunil, P. Alonso, J. Terrazas, D. Bereziat, A. Vidosavljevic, and L. Sedov, "Metropolis ii: Benefits of centralised separation management in high-density urban airspace," *SESAR Innovation Days*, 12 2022.

[25] DJI, "Matrice 600 specs," 2024, accessed on 28-03-2024. [Online]. Available: https://www.dji.com/nl/matrice600

[26] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[27] M. Nikolaeva, O. Aranda, O. Dahle, and M. Baena, "USEPE consolidated concept of operations," ISDEFE, Tech. Rep., Dec. 2022.

[28] A. Andreychuk and K. Yakovlev, "Two techniques that enhance the performance of multi-robot prioritized path planning," *arXiv preprint arXiv:1805.01270*, 2018.

[29] A. Bose, I. Markelov, and S. Noyes, "Multi-agent path planning in python," https://github.com/atb033/mu1ti-agent_path_planning, 2021.

[30] E. Ostertagova, O. Ostertag, and J. Kováč, "Methodology and application of the kruskal-wallis test," *Applied mechanics and materials*, vol. 611, pp. 115–120, 2014.

[31] M. Hess and J. Kromrey, "Robust confidence intervals for effect sizes: A comparative study of cohensd and cliffs delta under non-normality and heterogeneous variances," in *annual meeting of the American Educational Research Association*, vol. 1. Citeseer, 2004.

[32] T. MacFarland and J. Yates, "Mann–whitney u test," *Introduction to nonparametric statistics for the biological sciences using R*, pp. 103–132, 2016.

# II

Literature Study
previously graded under AE4020

# 1

# Introduction

Urban Air Mobility (UAM) is an emerging industry focusing on the transportation of passengers and delivery of parcels within urban landscape. It can be expected that UAM services will take off in the coming decade [39]. Regarding the transportation of passengers, companies such as Blade [8] and Uber Elevate [45] are already offering air taxi services in a few cities, albeit on a small scale. The delivery of parcels via Unmanned Aerial Vehicles (UAVs) has already been in operations since 2016 [123] and the number of parcels being delivered yearly has only increased ever since. In response to the increasing demand of UAM services, path planning and coordination methods are required to allow for efficient, sustainable and safe operations in urban environment.

In order to model the complex dynamics and interaction between various stakeholders in the urban environment, this thesis will focus on modelling 3D UAV operations by use of multi-agent path finding (MAPF). MAPF algorithms allow for conflict-free path planning and coordination of multiple agents within a shared environment. By focusing on path planning for UAM services, this work can contribute to development of real-world implementation of UAV transport system, which is a relatively unexplored field. Path planning is a critical aspect to allow for UAM services, and by following an agent-based approach novel techniques and/or model extensions can be proposed to improve path planning of autonomous UAVs and address specific challenges to UAV operations. It can be expected that with an evolving demand of UAV operations, regulations of UAV operations will change as well, which includes adaptations in flight restrictions. Also safe integration with current airspace needs to be ensured, while also autonomously avoiding collisions with other UAVs. Besides avoiding collisions between UAVs, it will be equally important to develop effective systems to detect and avoid static obstacles, such as high rises and restricted airspace. Modelling of 3D UAV operations is needed to capture the complex dynamics and interactions between agents and their shared environment. Combining the modelling of 3D UAV operations with dynamic airspace constraints will impose novel challenges for the development and evaluation of an agent based model.

This literature study is divided into 5 main chapters. The context in which path planning and coordination of UAVs will occur, will be provided in chapter 2. This include an overview of urban air mobility services, business models, companies, and vehicle types. Furthermore, the operational framework, which is called Unmanned Aircraft System Traffic Management, will be discussed. Combining the aforementioned information, a use case will be selected, which is deemed most suitable for path planning and coordination of UAVs in urban environment. Next, chapter 3 will elaborate on several factors that will influence the modelling of the urban environment. This includes a selection of a city and its key characteristics. Also, various software tools and modelling methods will be discussed, which could be suitable for the modelling of the environment and its operations. The information provided in the previous chapters will be combined to create a concept of operations for UAV operations in urban environment, which will be presented in chapter 4. This includes an analysis of the existing system, justification of an extension, and the new proposed system. For the proposed system, system objectives, requirements and assumptions are presented. Following from the concept of operations, multi-agent path finding (MAPF) algorithms are presented in chapter 5. MAPF algorithms allow us to find an efficient and safe method for planning and coordination of multiple UAVs in a shared environment. After presenting several MAPF algorithms a trade-off is performed in order to select

the most suitable method for the problem at hand. In chapter 6 a research proposal is presented based on the findings of the aforementioned chapters. This includes an overarching main research objective, which is further divided into research questions and sub-questions. These questions enable us to meet the research objective in a structured manner.

# 2

# Urban Air Mobility

Urban Air Mobility (UAM) is a rapidly emerging industry focusing on transportation of both passengers and cargo within the urban environment. This chapter will focus on Urban Air Mobility and the different use cases, vehicles, and traffic management systems in order to define the selected problem for this thesis. As the concept of delivering items via UAVs in urban environment is relatively new, a concept of operations (CONOPS) is needed in order to outline the operational framework, the objectives, and requirements of the system. This chapter will give an overview of the different stakeholders involved for UAM service, different suitable UAM vehicle types, and an overview of the proposed airspace and architecture needed for UTM. The selected use case will act as the foundation of the CONOPS. Afterwards, chapter 3 will provide reasoning for selecting the Rotterdam Area as urban landscape, including several characteristics that need to be taken into account for modelling the urban environment. The information presented in this chapter and the following chapter will be combined in order to provide a CONOPS, which will be presented in chapter 4.

section 2.1 will present different UAM services and provide reasoning for the most suitable service. section 2.2 will provide an overview of different vehicle types used for UAM. section 2.3 will discuss different urban airspace concepts, as well as the proposed architecture for operations within urban airspace. Lastly, in section 2.4 the selected use case will be discussed, presenting the agents within the multi-agent system.

## 2.1. Urban Air Mobility Services

Urban Air Mobility (UAM) is seen as one of the new revolutionary approaches to improve the transportation and logistic industry. The services offered for UAM can be divided into two different operation types, which are the transportation of passengers or packages, both using unmanned aerial vehicles (UAV). The benefit of passenger transport using UAVs, also referred to as air taxis, is that shorter distances can be flown using a higher speed compared to car travel. This can lead to a significant reduction in travel time and possibly even help to decongest traffic in metropolitan areas. Companies that already offer passenger services are e.g. Uber Elevate [45] and Blade [8]. Both companies offer taxi services using helicopters, but offer this only in limited cities, albeit at a small scale. In the future it is expected that these services will be taken over by electric Vertical Take-off and Landing (eVTOL) aircraft, where promising eVTOL vehicles [39] are being developed by Lilium [58], Volocopter [108] and Joby [48]. Even though developments for passenger transport in UAM seem promising, currently there are no companies operating eVTOLs for passenger transport. Transportation of packages using drones is however already in operation. In early 2022, over 2000 commercial drone deliveries were occurring every day worldwide and the number of deliveries is only growing [16]. This does include parcel delivery in both rural and urban environments. Even though this amount is relatively small compared to the total number of parcel deliveries worldwide, it does show the maturity of UAV delivery compared to passenger transport. Companies that are already operating fleets of drones to deliver a variety of goods, such as food, medicines, and e-commerce, are e.g. Zipline [123], UPS [105], and Wing [116]. A future possibility could also be to use passenger eVTOL vehicles for larger parcel deliveries, but this is deemed outside of the scope of this thesis.

Both passenger transport and parcel delivery are interesting use cases in order to investigate planning and

coordination of UAVs in urban environment. However, due to three main reasons the delivery of parcels will be chosen as use case, which are the following:

- First, parcel delivery using UAVs is currently more mature than passenger transport using UAVs. Most eVTOLs are currently either in design or test phases. Up to now, no fleets of eVTOLs transporting passengers exist. Parcel delivery using drones both in rural and urban environment is already in a further development stage and therefore more promising.

- Second, during operations it can be expected that the number of drones to deliver packages will be larger than the number of eVTOLs transporting passengers in the coming decade, making the planning and coordination of drones in urban environment more challenging.

- Third, autonomous flight operations for drones do already exist and are more likely to be included on a large scale for parcel delivery compared to passenger transport. One of the main reasons is that even though technology-wise it will be possible to fly eVTOLs autonomously, there are big public hurdles to be taken, such as accountability in case anything goes wrong. For drones, this is less problematic, as no passengers are onboard of UAVs, reducing the risk of human injury caused by an UAV. Therefore, the development of planning and coordination mechanisms for drones will be more beneficial for autonomous operations.

To conclude, both passenger and parcel delivery are expected to take off in the coming decade, with parcel delivery being the most suitable use case. The next section will elaborate on the different UAV delivery services available.

### 2.1.1. Urban Air Vehicle delivery services

Drone-delivery services are on the rise and are expected to become more widespread in the coming decade. In general, there are two promising business models that could be interesting use cases. The first one being the delivery of packages for the last-mile delivery of large retailers and delivery companies, such as Amazon and DHL. This would be the delivery of packages from a warehouse to customer [16]. The other business model is hyper-local delivery of packages. This would be the delivery of packages from local stores to customers using a third party delivery company in a small area in a very short time frame [73]. Both will be explained in greater detail below.

#### Last-mile delivery

Last-mile delivery is currently the most expensive and time-consuming part within the shipping process, but it is key to satisfaction of customers. According to Capgemini [47] 40 percent of customers use delivery services at least once per week. The cost involved with last-mile delivery accounts to 41 percent of the total supply chain costs [47]. With big retail companies, such as Amazon and DHL, offering free delivery services as well as same or next day delivery, the cost of the last-mile will only further increase. Due to the short time frame of delivery as well as the use of heavy transportation vans, the optimisation of routes becomes difficult, especially when considering energy emissions and transportation costs. A transportation vehicle is expected to drive to multiple locations before dropping of a package to a customer. The package is often a 1000 times less the weight of the vehicle, resulting in high energy and transportation costs. Matching the size of the delivery vehicle with the package size could allow for lower energy costs. An example of this is the use of delivery of food using bikes, such as UberEats and Just Eat Takeaway. This is considered to be hyper-local delivery and will be discussed later in this section. The downside of bike delivery is the high labour costs involved. A solution to both key problems within in the last-mile delivery could be potentially solved using drone deliveries, as energy emissions could be reduced by better matching the size of the package to the delivery vehicle as well as using autonomous flight to reduce the labour costs. According to McKinsey [16] if a drone would deliver one package to a customer, with a drone operator overseeing a total of 20 drones, the overall cost of delivery would become cost competitive with other transport modes, such as delivery by electric car or electric/ICE van. Last-mile delivery could have two integration possibilities for large retailers. The first one being that drones will fly from warehouse locations towards customers and afterwards fly back to pick up a new package or to recharge. Another possibility would be to integrate drone and truck delivery. This would mean that a truck would still be needed to carry parcels to a certain area as well as the drones. The drones would then be used for the final delivery of goods [26].

#### Hyper-local delivery

Where last-mile delivery focuses on the delivery of retail items of large corporations, hyper local delivery focuses on both the delivery of food and retail items of local stores only. This often achieved by using third party delivery companies. A good example of this is the delivery of fresh food by UberEats and Just Eat Takeaway. Due to the nature of the product, fast delivery is required in order to keep the food fresh and warm. The delivery distance is often within a radius of 5-15 kilometers depending on the delivery services and the time to delivery can range from 15 minutes to 2 hours taken from the time of order. Similar to last-mile delivery, the transportation vehicle does often not match the size of the delivery items in case cars or vans are used. When bike carriers are used the size of the delivery vehicle already matches the packages size better, but here high labour costs are involved for third party companies. One of the challenges with hyper-local delivery is the management of its fleet (delivery personnel). Due to the unexpected demand of items to be delivered, the size of the fleet does often not match the demand. The usage of autonomous delivery drones could therefore be a solution, as little to no labour costs are involved when a drone is not operating. The hyper-local delivery model does also suit the delivery of medicine. The delivery of medicine using drones in mainly rural areas is currently carried out by Zipline [123].

Both business models can be used for the delivery of parcels using UAVs. The remainder of this section will focus on already operating delivery companies using UAVs and their delivery services.

### 2.1.2. Delivery Companies

This section will shortly discuss the operation models of two market-leaders within the drone-delivery sector, which are Zipline [123] and Wing [116]. Zipline has reached the milestone of delivering 500,000 packages worldwide and Wing has completed over 300,000 drone deliveries. As both companies are beyond the proof-of-concept phase, key insights can be obtained from the operation model for the drone delivery model problem.

#### Zipline

First, Zipline [123] is the market-leader when it comes to the delivery of parcels using drones. The company was founded in 2016 and is currently operating in 7 different countries ranging from the United States to Rwanda and Japan. The company's primary business model is the operation of warehouses containing medical supplies. By using in-house designed drones medical supplies, such as blood and vaccines, are transported to hospitals and health facilities. The drones used for medical supplies, which are called the Platform 1 Zip, have a service radius of 100 km and are able to fly at an altitude of 90 meters. The packages are delivered using a specially designed package and make use of parachute at a dropping altitude of 30 meters. The package will then land in the designated landing area, which has a diameter of 5 meters. The drones are able to fly autonomously, but drone operators are present in the warehouses to monitor the flights. The company started operations in Rwanda and has a currently a partnership with the Rwandan government to aim for 2 million deliveries and 200 million kilometers of autonomous flight by 2029 [55]. This concept is successful due to the fact that transportation of medical supplies using traditional transportation methods is very time consuming, due to poor infrastructure available in Rwanda. By using centralised distribution centres of medical supplies and autonomous drones, it possible to deliver packages 24 hours per day to hospitals within an hour of ordering.

Besides the delivery of medical equipment using the Platform 1 Zip drone, the company is also working on a Platform 2 Zip drone, which is being build for precise delivery in urban environment. The drone can be used to deliver packages containing retail & e-commerce items, food, or healthcare items. The drone is expected to fly from a warehouse or restaurant location to the customers location and would hover tens of meters above the drop-off location. Upon arrival a build-in delivery drone would be dropped down using a cable which is able to quietly drop-off the package. The benefit of this system is that the drone will stay up in the air, reducing noise pollution as well as the chance of injury caused by the drone to bystanders. Zipline [123] states that this delivery service is 7 times faster compared to traditional automobile delivery, and shall result in 97% fewer emissions compared to ground delivery using petrol vehicles. It must be noted that the Platform 2 Zip drone is not yet in service, but does seem promising based on the successful operations of the Platform 1 Zip drone. Also the statements regarding the transportation speed and the emission saving potential cannot be checked due to limited information available.

### Wing

Next, Wing [116], which is a subsidiary of Alphabet (also the parent company of Google), is the number two drone delivery company in the world, with over 300,000 deliveries made. The company is currently operating in Australia, the United States, and Finland. Wing focuses on hyper-local delivery using their in-house developed drones. The packages that can be delivered by this drone are similar to the packages that can be delivered by the Platform 2 Zip drone. Packages are picked up using a cable and auto loader. The auto loader is a machine which allows drones to pick-up a package autonomously with the use of a grappling hook. The drone climbs to an altitude of 45 metres and flies towards the customers location. Afterwards the drone will descend to an altitude of 7 meters and lowers the package towards the desired location. By hoovering and lowering the package, the delivery method reduces noise, chance of human injury and energy consumption. After package delivery, the drone will return to the Wing site to recharge. The goal of Wing is to build a network that does not consist of point-to-point routes. The network intents to follow complex and changing routes when needs are evolving. This has the benefit that packages can be picked-up and dropped off in a more efficient manner, while also allowing the drone to recharge at various hubs when necessary [54]. The Wing drone is able to fly autonomously. A centralised unmanned traffic management software is used to plan the routes of an individual drone. After planning, the drone is able to take-off and fly towards the customers location. If problem are encountered during flight, contingency actions are taken automatically by the drone.

It can be concluded that both hyper-local delivery and last-mile delivery can be successfully offered by delivery companies, such as Zipline and Wing. Both companies make use of centralised hubs to assist in drone operations, such as providing charging or battery swap services. For last-mile delivery, warehouses are used, resulting in a hub-and-spoke network. For hyper-local delivery a point-to-point network is more common. Besides the specific drones developed by Zipline and Wing, several other Unmanned Aerial Vehicles are available for UAM services. The available vehicle types will be discussed in the next section.

## 2.2. Urban Air Mobility Vehicle Types

Unmanned Aerial Vehicles (UAVs) have risen in popularity and are being used on a daily basis for a variety of applications. The UAV design is often tailored to the specific tasks needed to be performed, such as transportation and surveillance. This section will only focus on UAVs tailored to delivery of packages, often referred to as delivery drones. An overview will be given of drones that are currently able to deliver packages including technical specification regarding the Wing Type, Weight of the drone, Maximum Payload, Maximum Range, Operating Altitude, Maximum and/or Cruise Speed, and Delivery Mechanism. The maximum range is equal to the total round-trip range. Based on the capabilities of the delivery drones, assumptions can be made for delivery modelling. For the wing type, four different classifications exist, which are fixed wing, multirotor, hybrid (combination of fixed wing and multirotor), and single rotor [67]. Fixed wings drones have the benefit that they are able to fly large distances at high speeds, but do lack maneuverability, as they are not able to hover, rotate and move backwards. Multirotor drones on the other hand do offer high level of maneuverability and agility, due to their symmetrical design using multiple rotors ranging often between 4 and 8. The main limitations of the multirotor design are their short endurance and low payload capabilities due to their small size. The hybrid design offers the best of both the fixed wing and multirotor design, allowing the drone to vertically take-off and land. After take-off the configuration of the rotors can be changed to the fixed wing configuration to allow for better range and endurance. The downside of this design is the higher cost and complexity involved. Lastly, single-rotor UAVs make use of only 1 rotor for vertical lift and take-off (VTOL) and use a small rotor to stabilise, which is similar to a helicopter. They are often used for military purposes and do not seem suitable to operations in urban environment.

An overview of promising and operational UAVs is given in Table 2.1. All drones except the Zipline P2 have conducted test flights. From Table 2.1 several key insights can be obtained regarding the operations of delivery drones suitable for urban environment. First, the most promising wing type is the hybrid configuration, as it offers both the larger range and endurance, as well as good levels of maneuverability. Payload capacity ranging from 1.2 - 6 kilogram is able to suffice the delivery of food, medicine, and e-commerce. The Wingcopter 198 [118] is capable of delivering three different parcels up to a total payload weight of 5 kilograms using a cable delivery mechanism. With a range of 20 kilometers or more, most European cities can be largely covered, which 5 out of 6 drones are capable of. According to the European Union Aviation Safety Agency (EASA) [23] the maximum flight altitude of drones is 120 meters. Allowing the drones to fly at high altitudes

has three main benefits, which are reduction in noise pollution, lower chance of human injury, and more obstacle free paths. Lastly, none of the delivery drones opt to have a delivery system, which require the drone to land at a customers location to drop off a package. Using different cable designs, such as a grappling hook for Wing [116] and a delivery droid for Zipline [123], packages can be delivered at higher altitudes. Cable delivery mechanisms allow for precision delivery, which is preferred in urban environment. To our knowledge, no research has been done up to the safety of using cable mechanisms for delivery. To ensure safe operations, this should be studied in more detail to assess human injuries caused by different malfunctions during delivery. For rural environment, parachute delivery mechanisms would suffice. The benefit of the latter mechanism is the increased turn around time for the drone during delivery. As one of the benefits of UAV delivery is vehicle design optimisation for a specific delivery type, there will not be one drone that will be best all over the board. Therefore, a combination of drones could also be feasible within one fleet. To ensure safe operations a traffic management system is needed, which does not only encompasses the UAV itself. This will be discussed in more detail in the next section.

Table 2.1: Overview of delivery UAVs

| Drone | Wing Type | Weight [kg] | Payload [kg] | Range [km] | Altitude [m] | Speed [km/h] | Delivery Mechanism |
|---|---|---|---|---|---|---|---|
| **Zipline P1** [123] | Hybrid | 21 | 1.75 | 160 | 80-120 (cruise)<br>30 (delivery) | 101 (cruise)<br>128 (max) | Parachute |
| **Zipline P2** [123] | Hybrid | Unkown | 3.6 | 38 | 90 (cruise) | 112 (max) | Delivery Droid |
| **Wing** [116] | Hybrid | 5.2 | 1.2 | 20 | 45 (cruise)<br>7 (delivery) | 104.4 (cruise) | Cable |
| **Dji Mavic 2 Enterprise** [20] | Multirotor | 0.9 | 0.2 | 8 | 6000 (max) | 72 (max) | Cable |
| **Wingcopter 198** [118] | Hybrid | 20 | 6 | 65-85 | 5000 (max) | 100 (Cruise)<br>144 (Max) | Cable |
| **Wingcopter 178** [117] | Hybrid | 12 | 6 | 35-75 | 5000 (max) | 86 (Cruise)<br>151 (Max) | Cable |

## 2.3. Unmanned Aircraft System Traffic Management

Due to expected growth of unmanned aircraft systems (UAS) within the (urban) airspace, the need for effective management of UAS traffic rises. Effective management is needed to be able to ensure safe operations for both ground traffic and conventional air traffic. The integration of UAS traffic within the airspace falls under the system called Unmanned Air Systems (UAS) Traffic Management (UTM). This section will give an overview of different concept for UTM Airspace, different stakeholders involved within falling under UTM Architecture, and lastly the expected UAV communication and coordination.

### 2.3.1. UTM Airspace

According to both the FAA [27] as well as EASA [23], the airspace in which drones are allowed to fly is 120 meters in Europe and 400 feet (122 meters) in the United States. NASA and the FAA propose a concept of operations for complex UTM operations, which occur below 400 feet above ground level [27]. This includes both UTM operations that are uncontrolled (part of class G airspace) and controlled (part of class A,B,C,D, or E). The class G airspace is thus the portion of the national airspace system (NAS) that is not part of the controlled airspace. This means that ATC has no responsibility to provide separation services. The management of the airspace will be cooperative between UAS and the interaction between UTM will be minimal. The UTM architecture will be further explained in subsection 2.3.2.

Due to the fact that a large part of the UAV operations will occur in uncontrolled airspace below 400 feet above ground level with minimal interaction of ATC, operators will be given extensive freedom in their operation management. The Metropolis project [85] investigated new airspace design concepts for the urban environment. Different traffic scenarios were modelled to provide a better understanding of air traffic within the selected airspace. This included extreme traffic scenarios as well. For the airspace design four different airspace structures were taken into account, which are Full Mix, Layers, Zones and Tubes. An example of the design structures is shown in Figure 2.1.

- **Full Mix:** The Full Mix design is the least restricted airspace design in which vehicles are allowed to fly at

every altitude and direction within the allowed airspace, without being constrained to a structural layout. Airborne separation is expected to be ensured using coordination and communication between UAVs.

- **Layers:** The airspace is split up into multiple horizontal layers, in which a layer will have an operating altitude bandwidth. Every layer will correspond to a certain heading angle, which will be repeated for the different layers.
- **Zones:** The Zones design is based on the conventional airspace structure. Different zones are allocated to different vehicle types, speed ranges and global directions. This is done to aid UAV separation. Different compared to the Full Mix and the Layers design, a possibility would be to dynamically adjust the zones to allow for better traffic handling.
- **Tubes:** The last design option considered is Tubes, which has the opposite design philosophy compared to the Full Mix concept. The Tubes provide a dense route structure that is fixed within the airspace. Similar to the Zones concept, different tubes will be allocated to different vehicle types, directions and speeds. Also a possibility exist to dynamically adjust the tubes, based on the traffic.

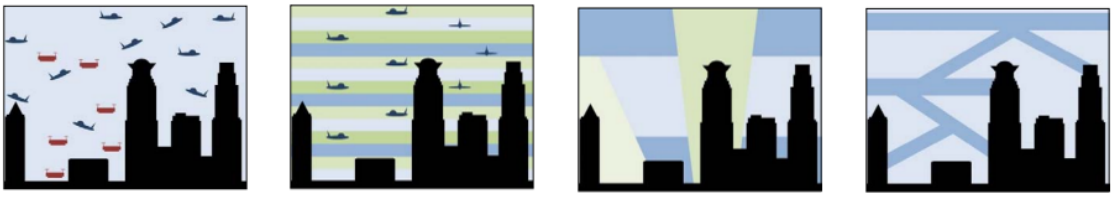

Figure 2.1: Airspace design concepts. From left to right these are Full Mix, Layers, Zones, and Tubes. Figures taken from Metropolis [85]

In order to select the best urban airspace design, Sunil et al. [98] compared four different airspace concepts in terms of capacity, safety and efficiency. It was demonstrated the tubes concept has a lower airspace capacity compared to the other 3 concepts. For this concept more airspace is required for separation in the pre-planning phase, reducing the overall capacity. However, this does not indicate that higher traffic volumes are safely facilitated in the other concepts. Safety can be assessed by the mean number of conflicts alerts occurring per flight. For the Zones and Tubes concept the number of conflicts is higher compared to the Layers and Full Mix concept. The reason for this is that due to the structured airspace, higher traffic concentrations exist, resulting in more conflict alerts at high traffic volumes. For the other concepts, the traffic is more distributed within the airspace, resulting in less conflict alerts. The efficiency of each concept is assessed by the route efficiency, which is defined as the shortest distance between start and goal location divided by the actual travelled distance. Here both the Layers and Full Mix concept outperform the Zones and Tubes concept. Due to less restrictions within the airspace, a more direct path can be flown, resulting in more efficient routes.

Therefore, it can be concluded that both the Full Mix and Layers concept outperform the Zones and Tubes concept in terms of capacity, safety and efficiency. The Full Mix and Layers concept perform rather equally, but the Layers concept slightly outperforms the Full Mix concept on one aspect. The Layers concepts outperforms Full Mix when comparing the results for number of conflict alerts and number of intrusions. Due to the fact this difference is minimal, but the implementation of a layered airspace will be more complex, the choice has been made to choose the Full Mix concept, where separation will be provided by using communication and coordination of UAVs. Above a flight altitude of 400 feet (122 meters), a layered concept could be the better option.

### 2.3.2. UTM Architecture

Besides the fact that the UTM airspace will be different compared to conventional airspace, the key actors involved in UTM will also be dissimilar. The FAA and NASA [27] propose a theoretical architecture that identifies the key actors and components, their relationships, functions and information flows on a high-level for the UTM airspace. The architecture can be seen in Figure 2.2. This architecture will act as the foundation of the different roles and interactions between agents. In order to accurately represent the entities in the UTM framework as agents in a multi-agent system, it is key to understand the different entities and their com-

plex interactions. Afterwards, the proposed UTM architecture can be simplified to allow for modelling of a multi-agent system.

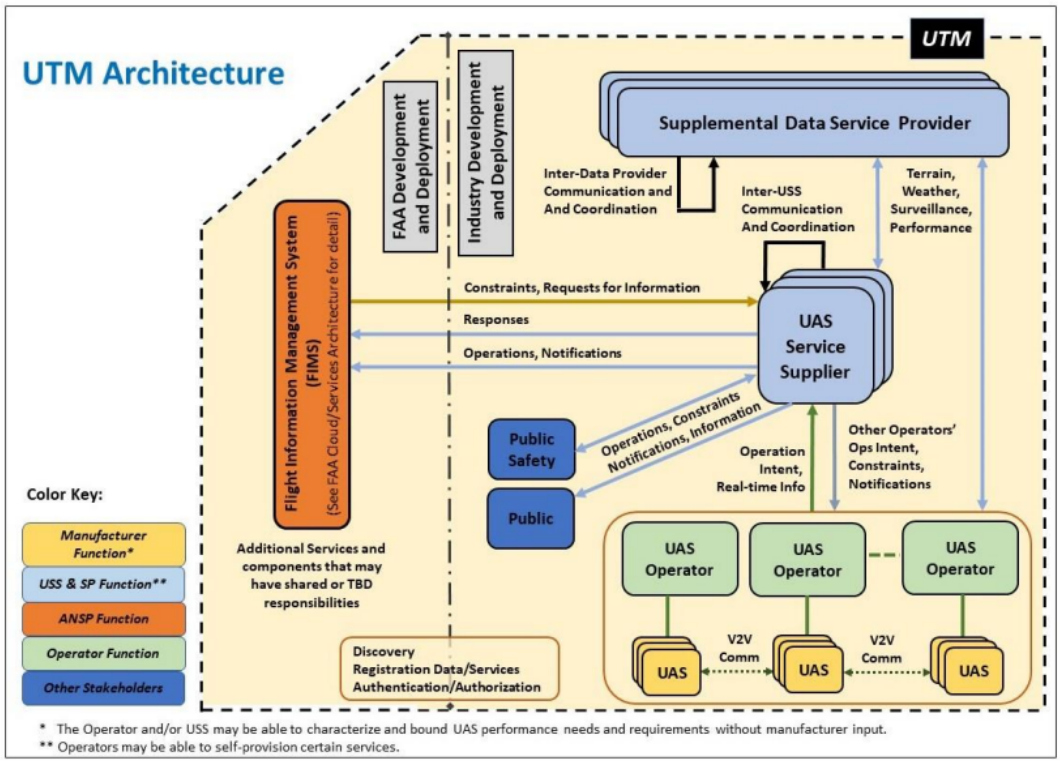


Figure 2.2: Theoretical UTM Architecture proposed by FAA and NASA. Figure taken from NASA [27]

Information and data will be shared between various stakeholders, including operator to operator, vehicle to vehicle, and operator to FAA communication. By sharing flight information and coordinating flights to de-conflict flight paths, safe operations can be ensured . The communication and coordination between all stakeholders will be via a distributed information network. This will be different compared to conventional traffic management, which is centralised and partially done by communicating by voice between air traffic control (ATC) and pilots. UAV communication will be further explained in subsection 2.3.3.

Within the UTM Architecture [27], three key stakeholders will be discussed in more detail, which are the FAA, Operators, and UAS Service Suppliers (USS). First, the FAA has the federal authority over aircraft operations in all airspace classes, from class A to G, as well as the regulatory and oversight authority of civil operations in the NAS. The European counterpart is EASA. The primary role of the FAA within UTM is to provide airspace constraint data to airspace users as well as to provide the operational and regulatory framework. In order to provide regulatory and operational oversight the FAA interacts with the Flight Information Management System (FIMS), which has the function of air navigation service provider (ANSP). Second, the operator is the responsible party for the management of operations. The operators will plan flights and share this information all while meeting regulatory responsibilities. The term operator can be used for both manned and unmanned operators. For unmanned operators a distinction can be made between operators that use visual line of sight (VLOS) or beyond visual line of sight (BVLOS) for their operations. For the delivery of packages using UAVs it is expected that operations will be done by BVLOS operators, highlighting the need for clear communication and coordination between UAS. Lastly, support of operations for UAS operators can be provided by third party USS. The services provided by USS are threefold. The first service is supporting communication between Operators and federated UTM actors (such as the FAA) to meet regulatory and operational requirements. The second service is to provide operators with operational information to ensure safe and efficient missions. The third service is to archive operational data in historical databases.

To conclude, different entities will provide different functions within the UTM Architecture. The entities

within this architecture will provide the basis for the agents within the delivery modelling problem. Different compared to conventional ATC, which uses a centralised planning and coordination approach, UTM will make use of a distributed planning and coordination approach. Operators will use a centralised approach for managing their fleet.

### 2.3.3. UAV Communication

Based on the proposed UTM Architecture described in subsection 2.3.2, Chakrabarty et al [14] developed an operational architecture using vehicle to vehicle (V2V) communication. The manner in which V2V communication will occur is of importance for the UAV delivery problem as it will act as the basis for how different agents will be able to communicate and share information with each other.

The proposed communication architecture is visualised in Figure 2.2 and includes the following actors: UASs, UAS Operators, UAS Service Suppliers (USS), Supplemental Data Service Providers (SDSP), and Flight Information Management Systems (FIMS). The communication and coordination will be through a distributed and highly automated network in which the following information will be shared. The FAA will provide real-time airspace constraints to the UAS operators via the FIMS and to the USS. The UAS operator is expected to operate in accordance to the imposed rules. Besides the information shared by the FIMS, other relevant information, such as terrain & obstacle data, specialised weather data, and constraint information is needed for safe operations. This will be provided by the SDSP. Currently, it is not clear if this information will be directly shared to the UAS or will be indirectly shared via the UAS operator and/or USS. The benefit of sharing this information directly to UAS is that this allows the aircraft to directly re-plan its path if necessary. The communication between a UAS Operator and UASs will be bidirectional, allowing communication of initial plans and possible plan adjustments to be shared between the two stakeholders. Also sharing the location information of all UASs to the UAS operators allows the operators to ensure that all constraints provided by the FIMS and SDSP are adhered to. Lastly, V2V communication will play an important role in ensuring safe operations and separation between UASs operating in close proximity. What is deemed to be close proximity needs to be defined. This means that the operator will be aware of each UASs location at any time, but not all UAS will be aware of the location of every UAS. [14]

The means of communication between UAS operators and UAS, as well as between UAS and UAS still needs to be defined in the future. The communication itself can be split up into Control and NonPayload communication (CNPC) and Payload Communication (PC) [122]. PC communication is for instance the sharing of aerial images or videos to the operator. This is deemed outside of the scope of this thesis. The CNPC shares information related to flight operations. Four distinct communication technologies can be used for both air-to-air and air-to-ground communication, which are Direct Link, Satellite, Ad Hoc Network and Cellular network.

The direct link network is mostly used in the past for air-ground communication using unlicensed radio frequency (RF) band. The benefit of this network is that it is simple to implement and has low costs. The downsides of this network are limited to Line-of-Sight (LoS) operations, insecure connections, and difficulties in scalability. [122]
Next, a satellite network would allow for UAV BLoS communication, also allowing UAVs to fly in remote areas, without other means of coverage, such as WiFi or 5G. The main disadvantages of this network are the high cost involved during operations and the high latency. [122]
Furthermore, the mobile ad hoc network has the benefit over the satellite network that no infrastructure is required, and the network itself is self-organised. Communication occurs between multiple devices. The disadvantage of the network is the high cost and complexity involved, as well as limited scalability. [122]
Lastly, using a cellular network for the communication between UAVs has the benefit of supporting large-scale communication networks in a cost-effective manner. This is not deemed possible for the previously mentioned technologies. The only downside of using cellular networks is the limited coverage in remote areas. However, as this thesis is only focused on UAV delivery in urban environment, access to a cellular network can be expected. [122]

Several cellular technologies exist, such as WiFi, 4G+ and 5G [104]. Yang et al [120] researched the capabilities offered by 4G+ networks as well as it limitations. Using a 2.6GHz carrier frequency with 20MHz carrier bandwidth on a LTE-Advance network, it was demonstrated that most latency data samples were concentrated between 200 and 300 milliseconds at heights of both 50 meters and 100 meters, which would encompass the

operating altitudes for UAVs. Without further enhancements to the current network, these latencies could be too high for V2V communication. The benefit of a 4G+ network is the range in which a signal can be received, which is 30 kilometers. Therefore, it could potentially be sufficient for operator to drone communications, while limiting the amount of information that would need to be shared. A 5G network could offer a lower latency in combination with the ability to share information at higher bit rates. One of the limitations of the 5G network is limited range, which could be 50-500 meters. This range would be sufficient for V2V communication. Verizon [107], AT&T [6], and T-Mobile [30] have invested in the development of 5G-networks and have already tested and demonstrated the capabilities a 5G network could offer for BVLOS flight operations using UAVs.

## 2.4. Urban Air Mobility Selected Use Case

After exploring different UAM Services and UAM Vehicle Types as well as the proposed UAS Traffic Management structure, the problem formulation will be discussed, which will form the basis of this thesis. This section will discuss the selected UAM Service, fleet and airspace design. Combining this information the agents present within the multi-agent system will be proposed. The main focus of the multi-agent system will be the path planning and coordination of UAVs, for which the multi-agent pathfinding framework is most suitable.

### 2.4.1. UAV Medical Supply Delivery

Two distinct types of UAM services will be offered in the future, which are passenger transport and parcel delivery, as discussed in section 2.1. The latter will be selected as this thesis UAM service, due to three reasons. First, parcel delivery using UAVs is more mature compared to passenger transport. Second, the planning and coordination of UAVs for parcel delivery will be more challenging due to the high number of UAVs involved. Third, a higher level of autonomy will be involved, emphasising the need for planning and coordination methods for UAV delivery fleets. Within the delivery sector, UAVs can be utilised for various purposes, such as the delivery of food, medical supplies and e-commerce. Table 2.2 presents a trade-off of the benefits of selecting each delivery item. The delivery of medical items using UAVs is most mature, with Zipline [55] demonstrating the success of this use case in Rwanda. Food delivery is also taking place in Australia offered by Wing [116], but is taking place on a smaller scale. Currently, the delivery of retail & e-commerce items is not mature, with Amazon Air [74] not being able to come off the ground. In general the market size of people ordering retail & e-commerce and food items is significantly larger compared to medical items. The public benefit of delivering medical items via drones in a fast manner is deemed higher compared to food and retail & commerce. Lastly the social acceptance of drone delivery is higher according to EASA [22]. This is if the use case has benefit for the community and does not serve individual needs, being the case for the delivery of medical items.

Table 2.2: Trade-off table representing benefits of selecting each delivery item

| Delivery Item | Maturity | Market Size | Public Benefit | Social Acceptance |
|---|---|---|---|---|
| *Retail & E-Commerce* | - - | ++ | - - | - - |
| *Food* | +- | ++ | +- | - - |
| *Medical Items* | ++ | +- | ++ | ++ |

Therefore out of these applications, the delivery of medical supplies offers the most benefit to the public, as it significantly improves access to healthcare services. Both Zipline [123] and Wing [116] offer services to deliver medical services to both healthcare facilities as well as to the general public. This technology is already a game-changer in Rwanda [55], but can also become of importance within the urban environment. Due to the limit capacity of delivery drones, as discussed in section 2.2, the delivery of medical items will be limited to up to 5 kg. This would allow for the delivery of e.g. medicines, vaccines, emergency (blood) supplies and diagnostic samples [24]. The delivery of medical supplies using traditional supply methods is already well established with companies as PostNL [77] using transport vans and Velomedi [106] using bike carriers. Challenges associated with traditional modes of transport can be overcome using UAVs, resulting in an efficient and reliable way to deliver medical supplies. The medical supplies will originate from several warehouse locations and will afterwards be delivered to customers, similar to the business model of Zipline [123]. Zipline only delivers items to medical facilities, which will also be the customers in this work. A possibility could also be to allow drones to pick-up diagnostic samples from customers and deliver those to warehouses.

The UAV Medical Supply Delivery problem is both interesting from an application and academic perspective. From an application perspective, medical drone delivery can assist medical facilities with managing inventory, especially for short-term perishable items. It can also increase on time delivery of medical supplies, as delays caused by traffic congestion can have severe consequences for patients. Due to improvements in the logistics of healthcare items as well as the public benefit UAV medical delivery can offer, this use case will be one of the first deployed UAM services in urban environment. Therefore, by focusing on the path planning and coordination of autonomous medical drones, this work can contribute to the development of this real-world implementation. From an academic point of view, by following an agent based approach the complex dynamics and interactions between agents can be captured in a 3D urban environment. As most multi-agent pathfinding methods are developed for 2D environments, this work can explore state-of-the-art algorithms and techniques, which need to be scaled for 3D environments, while being computational efficient and possibly allow for real-time planning. This work can also explore coordination & prioritisation strategies and communication protocols between agents, from which emergent behaviour of urban UAV operations can be analysed.

Next, for the delivery of medical supplies within the urban environment different UAVs would be suitable for operations. For this thesis two different UAVs will be considered, which are the Wingcopter 178 and 198, as described in section 2.2. The Wingcopter 178 will be the main delivery drone. A possible extension could be to include the Wingcopter 198 into operations. The main reason for choosing these two different drone types, is the amount of technical information available, which includes range estimations for different payloads, flight and climb speeds in different mission modes and maximum wind resistance. The Wingcopter 178 will be able to deliver 1 parcel up to 6 kilogram with a range of 35 kilometers, while the Wingcopter 198 will be able to deliver 3 parcels up to a total of 5 kilograms with a range of 65 kilometer. The parcels will be delivered using a cable mechanism.

Regarding UTM, the UTM Airspace design structure that will be selected is Full Mix, as described in Figure 2.1. The different entities that will form the basis of the agents within the multi-agent system are the FIMS, USS, SDSP, UAS Operators and UAS, as described in subsection 2.3.2. This will be explained in the following subsection.

### 2.4.2. Agents in Multi-Agent System

Focusing on the UAV Medical Supply Delivery problem, as described in the previous subsection, the following agents will be proposed, where the UTM architecture acts as the basis, as can be seen in Figure 2.2.

- **Authority:** This agent consist of two different entities within the UTM architecture framework. On the hand side this agent represents EASA/FAA via the flight information management system (FIMS), which will provide regulations and constraints to the urban airspace, such as the maximum flying altitude, restricted airspace areas and the allowed operations window. On the other side it will represent the Supplemental Data Service Provider (SDSP), which will provide information regarding the terrain, obstacle and wind directions to the Operator. As both the FIMS and SDSP will fulfill a data provision role to the UAS Operator, the choice has been made to merge both entities into one agent. This agent will be only providing information to the Operator agent, but does not have a role in the planning and coordination of UAVs. Therefore the authority agent will have no physical presence in the simulation, as its only goal is to provide information regarding its environment to other agents using one-way communication.

- **Operator**: Similar to the aforementioned agent, the operator agent will be the combination of two entities, which are the USS and the UAS Operator. The operator will be a commercial entity that will be responsible for the management of the UAV fleet, which includes the global planning and coordination of UAVs, as well as ensuring the fulfillment of the demand. The operator will also be responsible for adhering to regulations and restrictions imposed by the Authority agent. This agent will be a self-interested agent, with the goal of maximising its profit. The operator itself will use centralised planning method for the management of its fleet. It is expected when multiple operators operate in the same airspace, the coordination between the operators will be distributed, as no ATC entity will be present. Therefore, the operator agent will also have no physical presence in the simulation. The agent will have advanced cognitive properties as it is able to plan conflict-free paths of all UAV agents. Enabling

UAV agents to reach their goal in a fast and conflict-free manner is the goal of this agent. The operator agent will be able to communicate to both the UAV agent and the customer agent using bidirectional communication.

- **UAV**: The UAV agent is responsible for the task execution. This includes picking up the correct parcel, following the correct path to the customer, and delivering the parcel to the customer. The path to the customer is provided by the operator. The UAV properties will originate from the technical specifications of the Wingcopter 178 initially. This agent will have limited cognitive capabilities compared to the Operator agent. It will be able to follow the path imposed by the operator, but only knows its own path. When in close proximity with other UAV agents, the agent will be able to use bidirectional communication to other UAV agents to share their plans to ensure no conflicts will occur. If needed, the agent will be able to re-plan its path. This agent will have a physical presence within the environment, will be reactive to its environment, and will be mobile.

- **Customer**: The customer agent will be able to request specific items within a certain timeframe at preferred locations. This will probably be certain pick-up locations within dedicated zones. The customer agent will be able to go to the pick-up location and receive the item. A possibility would be that certain medical supplies will be requested that have a higher priority compared to other delivery items, such as medicine. Another possibility could be that customers would be able to deliver diagnostic samples to pick-up zones. This agent will have limited cognitive properties as it will only be able to order medical supplies via the operator and afterwards receive the package, delivered by the UAV agent. The goal of this agent is to order and receive their supplies. The agent will be physically present in the simulation, represented by goal nodes of UAV agents. The agent will be able to communicate to the operator to share order and delivery confirmation information, as well as share delivery confirmation information to the UAV agent.

# 3

# Environment Modelling

To allow for the modelling of medical urban air mobility services, a model needs to be created representing an urban environment. This chapter will focus on several aspects that are key for modelling the urban landscape. First, the reason for selecting Rotterdam as use case will be explained in section 3.1. This will also include an overview of the medical facilities, possible warehouse placements, and restricted & limited access areas for UAV operations in Rotterdam. Second, geofencing will be discussed in section 3.2, to allow for the modelling of virtual boundaries around buildings and no fly zones, which can afterwards be incorporated within UAV path planning. In section 3.3, a simulation tool for urban freight transport will be discussed, facilitating demand modelling of medical facilities in Rotterdam.

## 3.1. Rotterdam Area Environment

Rotterdam has been selected as the use case for the distribution of healthcare supplies due to several reasons. First of all, the inner city of Rotterdam contains many high rises in combination with a non grid like, but rather organic, city structure, making path planning more interesting. Flying over residencies by UAVs will be possible in certain neighbourhoods, while in other parts of the city UAVs will fly mostly over road infrastructure. Second, situated on the north side of Rotterdam is Rotterdam the Hague Airport (RTHA), which will influence the UAV operations and the allowed airspace it is permitted to fly in. Third, restricted flying areas will also be located in large parts of the city itself due to location of the Port of Rotterdam, which is currently restricted airspace. Fourth, due to the presence of three large hospitals as well as many healthcare facilities distributed over the city, a large market exist for the delivery of medical supplies. Lastly, for the Rotterdam Area an open-source simulation tool for modelling the demand of urban freight transport is available, called MASS-GT, which will be discussed in section 3.3. Combining the aforementioned reasons, Rotterdam is an interesting use case for the modelling of UAV operations, which will need many considerations regarding allowed and non-allowed flying zones during operations for several mission types.

This section will go into detail regarding the Rotterdam Area environment. It is important to consider factors such as the placement of hospitals and pharmacies as well as restricted flying areas in order to accurately represent the Rotterdam Area and allow for correct modelling of UAV paths. This section will first elaborate on the customers, which are medical facilities. Afterwards focus is on suppliers, which will be warehouses and distribution centres. Lastly restricted and limited access areas will be discussed.

### 3.1.1. Medical Facilities

First, healthcare logistics can be innovated by using UAVs, which enables medical facilities to overcome logistical challenges. The effectiveness of using drones for delivering supplies to healthcare facilities has already been demonstrated in Rwanda by Zipline [55, 123]. In order to accurately model the delivery of various medical supplies, it is important to get an understanding of which facilities are present in Rotterdam and which medical supplies they will be needing. An overview of different healthcare facilities is presented in Table 3.1, including the medical supplies that can be delivered to the facility. The estimated number of medical facilities available for delivery in Rotterdam is also shown in Table 3.1. The three largest possible customers are Erasmus MC Hospital, Maaststad Hospital, and Ikazia Hospital, with a total of 1350, 600 and 350 hospital

beds respectively. The total number of possible delivery cites for medical supplies is 427. It must be noted that multiple medical facilities can be located within the same building, but operate under different names. However, with a maximum of 427 possible delivery cites, the market for delivery of medical supplies is sufficient. The medical supplies will originate from warehouses/distribution centres, which will be elaborated on in the following subsection.

Table 3.1: Overview of different types of healthcare facilities, the medical supplies they will need, and an estimated number of available facilities in Rotterdam according to Zorgkaart The Netherlands [124]

| **Healthcare Facility** | **Medical Supplies** | **# Facilities Rotterdam** |
|---|---|---|
| Hospital | Medicine, Vaccines, Emergency Supplies, Diagnostic Samples | 16 |
| General Practitioner Offices | Medicine, Vaccines, Emergency Supplies, Diagnostic Samples | 164 |
| Pharmacies | Medicine | 75 |
| Specialist Clinics | Medicine, Emergency Supplies, Diagnostic Samples | 30 |
| Mental Health Institutions | Medicine | 75 |
| Diagnostic Centres | Diagnostic Samples | 3 |
| Nursing Homes | Medicine, Emergency Supplies, Diagnostic Samples | 64 |

### 3.1.2. Warehouses

Similar to the business model of Zipline [123], several warehouse locations will be able to provide the demand for all healthcare facilities. Around distribution centres numerous flight maneuvers will take place, which can cause noise nuisance in the vicinity of warehouses. Therefore, it can be expected that warehouses will be located mostly at industrial areas, located outside of the city centre of Rotterdam. An example of possible warehouse locations in the Rotterdam area is visualised in Figure 3.1. The warehouse locations originate from the MASS-GT project [18], which are based on the real-world locations of warehouses. This project will be discussed in more detail in section 3.3. Figure 3.1 only shows warehouses with a floor space over 25,000 [$m^2$], representing large warehouses. A benefit of using large warehouse locations outside the city centre is the possible incorporation of a vertiport. A vertiport is a mobility hub and landing space for both passenger and cargo UAVs. For the problem at hand, only landing and recharing/battery swapping of cargo UAVs will be considered. Furthermore, the port of Rotterdam also opened the first vertiport location in the Netherlands, showing the willingness of Rotterdam to include UAV operations within the city [72]. For the modelling of the medical delivery problem, a selection of warehouses will be made, based on their respective location to customer sites.

Figure 3.1: Warehouse (>25,000 [$m^2$]) placement in Rotterdam according to the MASS-GT Model [18] created in QGIS [78]

### 3.1.3. Rotterdam Restricted/Limited Access

The placement of medical facilities and warehouse locations will have profound impact on path planning of UAVs. However, one of the biggest influences on the path planning of UAVs will be the placement of restricted and limited access areas within urban environment. Even though drones are physically capable of flying throughout the entire city, it is not preferred due to privacy, nuisance and environmental reasons. Furthermore, Rotterdam also has multiple no or limited fly zones due to the nearby airport and seaport. These

areas will have different impacts on the flight planning of drones and will be discussed in more detail below. The influence of the airport, seaport, and city centre, parks & recreational areas will be discussed below respectively.

#### Rotterdam The Hague Airport

Rotterdam The Hague Airport (RTHA) is located on the north side of Rotterdam and heavenly influences the allowed airspace to fly in. Around RTHA a control zone (CTR) is established, which encompasses the entire city of Rotterdam [82]. Within the CTR drone flight is limited, and is only allowed if the operator has a license provided by the The Human Environment and Transport Inspectorate (ILT) [82]. Besides the license, permission for operating the flight also needs to be granted by LVNL [82]. If both the license and access are granted, a maximum flying height of 120 meters is imposed, similar to norms provided by EASA [23] and the FAA [27]. Therefore, it is assumed that drone operations will be feasible in Rotterdam as long as permission is granted by LVNL. In order to cause no nuisance to air traffic, a possibility could be to restrict the flying height based on the distance from the airport.

#### Port of Rotterdam

Besides the airport limiting flight operations in Rotterdam, the port of Rotterdam is a no fly zone for drone operations. Part of the no fly zone of the port of Rotterdam, situated close to the city centre is shown in red in Figure 3.2. The Port of Rotterdam is currently working together with the Ministry of Infrastructure and Water Management (I&W), Ministry of Defence and ILT in order to create a prototype of the U-Space above the port of Rotterdam [71]. This is still in development phase and up to now, no areas have been marked which are allowed to fly through. However, a possibility could be to allow drones delivering emergency medical supplies to fly over the port airspace, even though this would be restricted for other drone operations. This is similar to the operation of helicopter emergency medical services (HEMS), which are also allowed to fly through certain restricted areas.

#### City Centre, Parks & Recreational Areas

The municipality of Rotterdam [33] states that it is currently working on policies regarding restricted/limited access urban airspace as well as possible landing sites for UAVs. However, up to our knowledge no policy has been currently adopted regarding limited access areas in Rotterdam. It could for instance be possible that it is not preferred that drones fly over crowded areas, parks and recreational areas in order to reduce nuisance. An example of restricted flying areas is visualised in Figure 3.2, which highlights the city centre of Rotterdam in yellow, and parks & recreational areas in green. However, in case urgency is required, such as for emergency supplies, exceptions can be made. According to a study performed by EASA [22] on the societal acceptance of UAM in Europe use cases that are in general public interest, especially for health and safety services, are deemed more acceptable. Even higher noise levels could be acceptable if the number of operations for emergency purposes will be limited. Therefore, different levels of geofencing can be used based on the mission type of the drone.

## 3.2. Geofencing

Geofencing is a key component for UAV operations, allowing for safe flight operations. Geofencing can be described as the use of virtual boundaries and geographic zones based on a geographical location [52]. The virtual boundaries can be used to restrict flights within no fly zones, but also to keep clearance from obstacles such as high-rises. Therefore, it is expected that for UTM, the envisioned geofencing system will have initially two kinds of airspace, which are fly-zones (keep-in geofence) and no-fly-zones (keep-out geofence). In general these volumes will be statical, for example for highrises. However, an addition could be to have dynamically adjusted virtual boundaries, to allow for either separation between UAVs or to enable more direct paths for emergency missions.

In order to ensure that UAVs will not fly in no-flying zones, geofencing needs to be taken into account during UAV path planning. Geofencing is used both for obstacle and restricted airspace avoidance as well as flight separation of UAVs. Using parameters such as vehicle speed, geofence boundary safety size and polygon boundaries parameters, a flight plan can be created that does not violate the constraints of the environment. A trajectory keep-in geofence can be used for the modelling of the planned flight plan. A path finding logic is used for path planning between a departure and destination location. The proposed algorithms by Kim and Atkins [52] use a set of 3D polygons to represent different geofence volumes. A visibility graph approach is


Figure 3.2: Map of Rotterdam city centre (yellow), parks & recreational areas (green), Port of Rotterdam (Red) created using Google Earth Pro [36]

taken for path planning. As the number of vertices needed to create geofence volumes influences the computational speed of the graph generation, map simplification algorithms are needed. A possible method to reduce the number of vertices is by clustering geofence volumes, which reduces the number of vertices. The downside of this method is that more airspace is restricted in the model as would be in a real-world scenario. Another possibility could be to reduce the complexity of the models shape, by using pre-defined geometric shapes, such as beams, cubes, and cylinders.

The aforementioned geofencing methods can both be applied to airspace as well as obstacles. In order to represent obstacles in the Rotterdam Area, map data of Rotterdam will need to be processed. A possible source in order to get geographical city data is OpenStreetMap (OSM) [37]. OSM is an open source provider of map data, including data of the Rotterdam area. In order to create a 3D keep-out geofence representation of the region, several steps need to be taken. From the OSM data input, a data formatting step is needed in order to reduce the complexity of the environment by representing buildings by polygon vertices, building heights and street level coordinates. The next step is to create larger polygons by merging multiple building polygons together. Then, 2D and even 3D keep-out geofences can be created. An example of post-processed map data for southern Manhattan, New York, is shown in Figure 3.3, which includes only buildings with a height over 20 meters. The coloured polygons represent single buildings, while the black polygons represent merged polygons. The same method could be applied in order to model part of the Rotterdam environment. In order to even further reduce the complexity of the model, buildings up to certain heights situated close to each other can be represented by one single polygon with a uniform height. For example, if three buildings next to each other have maximum heights of 8, 10 and 12 meters, a possibility could be to cluster these buildings and create a polygon with a height of 15 meter, which includes a safe separation distance in addition. A similar approach can be taken for other heights. Based on the selected area, different height categories will be determined. Geofencing could also be used to keep safe separation from other dynamic obstacles, such as birds. However, as the trajectory of birds is difficult to predict, this will be deemed outside of the scope of this thesis.

To conclude, geofencing methods will be key for the modelling of UAV operations in urban environment. It will be needed to represent virtual keep-out zones around building and restricted & limited access areas. Besides the modelling of the environment, the modelling of the demand will also be key for medical UAV operations, which will be discussed in the next section.

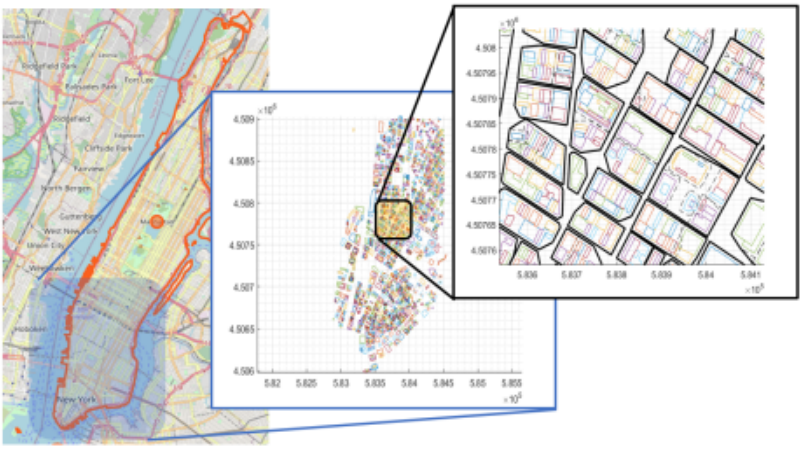

Figure 3.3: Post-processed map data for southern Manhattan, representing building with heights over 20 meter, where buildings are clustered together in geofence volumes. Figure taken from Kim and Atkins [52]

## 3.3. Demand Modelling

In order to accurately represent the demand of medical items, a demand model needs to be incorporated within the system. The Multi-agent Simulation System for Goods Transport (MASS-GT) [18] is a multi-agent simulation model designed for modelling of urban freight transport. The tool is open-source and originates from a project at the TU Delft. The model incorporates different stakeholders within the multi-agent approach, such as producers, customers and carriers. The stakeholders originate from four markets which are the commodity market, logistics services market, transport market, and infrastructure market. Data from the Central Bureau of Statistics Netherlands regarding transport shipments via road transport is used to calibrate the model. Besides the transport shipment data, also data regarding vehicle tours, trip patterns, firm population and commodity flows is incorporated. The model allows for the simulation of urban freight in the Rotterdam Area, including The Hague and Leiden. It also includes the locations of distribution centres, categorised based on floor space. Parcel depots of several large shipment companies have also been included, such as DHL, PostNL and UPS. The study area itself is divided into zones, for which each zone the estimated demand for certain types of commodities can be modelled.

The MASS-GT model [18] can potentially be used in multiple ways as part of the modelling of the UAV medical supply delivery problem. First, warehouse placements can be used to represent several source locations for medical supplies. Furthermore, the demand of commodities in combination with the number of medical facilities in a zone can possibly be used to estimate the expected demand of medical supplies.

# 4

# Concept of Operations

The use of unmanned aerial vehicles (UAVs) for the delivery of parcels is an emerging concept. One of the most prominent delivery items are healthcare supplies. As the concept of delivering medical items via UAVs is relatively new, a concept of operations (CONOPS) is needed in order to outline the operational framework, the objectives, and requirements of the system. chapter 2 gave an overview of the different stakeholders involved for UAM service, different suitable UAM vehicle types, and an overview of the proposed airspace and architecture needed for UTM. The chapter also included explanation for choosing UAV Medical Supply Delivery as the selected use case. chapter 3 provided reasoning for selecting the Rotterdam Area as urban landscape, including several characteristics that need to be taken into account for modelling the Rotterdam environment. This chapter will combine the information presented in the aforementioned chapters in order to provide a CONOPS. Even though Zipline [123] has already been operating in Rwanda since 2016, operations in urban environment have not yet been implemented. The CONOPS will serve as an blueprint outlining the operational framework, objectives, and system requirements & assumptions, which will be discussed in section 4.1, section 4.2, and section 4.3 respectively.

## 4.1. Operational Framework

The operational framework for UAV Medical Supply Delivery will act as the foundation from which the system objectives, requirements, and assumptions will be derived. This section will first provide a short overview of the existing system. Afterwards, a justification of why an extension is needed for the current system will be given. Lastly, the future system will be described.

### 4.1.1. Existing system

This subsection will give a short overview of the current system for healthcare and pharmaceutical logistics, mainly focusing on the delivery of healthcare supplies from warehouse locations to medical facilities.

The pharmaceutical supply chain has the following basic structure [38]. First drugs are manufactured at production sites of pharmaceutical manufacturers. Pharmaceutical manufacturers are the source for prescription drugs within the supply chain, and are considered the first element in the supply chain. Afterwards pharmaceuticals are transported to wholesale distributors. Wholesale distributors directly purchase pharmaceutical products from manufactures, store the products in warehouses and afterwards are responsible for the shipment of products to medical facilities, such as hospitals and pharmacies. The current method of transportation for healthcare items is via ground transportation within the same city. Upon receiving of the shipment, medical facilities store the products before dispensing the items to patients. It must be noted that this is the basic structure of the pharmaceutical supply chain and many variations exist due to constantly evolving players. For the supply of other medical items, such as vaccines [61] or blood supplies [1], the supply chain can be different. However, the same basic elements exist. For all items a source is the first element in the supply chain. The second element in the supply chain is a large inventory location. The third and last element is the delivery location, which are often medical facilities. This thesis will only focus on the delivery between inventory locations and medical facilities, as it is expected that drone delivery will be mainly be utilised for last-mile delivery.

There are multiple challenges involved for delivery and warehousing of healthcare items. Three main challenges are:

- **Managing inventory:** Medical facilities have to balance an adequate inventory in order to meet patients needs, while avoiding unnecessary inventory cost [56]. Especially with short-term perishable items, such as blood supplies, a careful balance is needed [95]. To ensure an adequate inventory, inventory management needs to be optimised, which can be challenging for medical facilities as accurate demand forecasts are needed.
- **Delivering in time:** Ensuring timely delivery of medical supplies from warehouses to medical facilities is important for patient care. Delays or inefficiencies, due to for example traffic congestion, can have severe consequences. For certain medical items, shortages are not allowed. Therefore, efficient routing and scheduling of transport is needed to guarantee medical supplies. [87]
- **Controlling temperature:** For many medical supplies, such as vaccines and blood supplies, specific temperature control is needed in order to maintain the integrity of the products. The delivery of these items is part of the cold supply chain and offers different challenges compared to standard supply delivery. The cold supply chain could be improved by providing faster and more reliable distribution. [61]

An extension to the current transportation and supply chain system, could be the use of drones for delivering medical supplies to facilities. The extension can be justified as it allows for easier inventory management, timely delivery of emergency items, and even allow for faster distribution within the (cold) supply chain. Centralised warehouses have the capability to ship products to multiple medical facilities within a city by use of drones. This in order to better help balancing the overall inventory, and allowing for medical requests within a short time frame.

### 4.1.2. Proposed System

subsection 4.1.1 focused on the simplified current medical supply chain and why an extension of using drones for delivery between distribution centres and medical facilities can be justified. This subsection will provide an overview of the proposed system, by combining information presented in chapter 2 and chapter 3 regarding UAM services and environment modelling respectively.

First, the FAA and NASA [27] propose a theoretical architecture identifying the key actors and components on a high-level for the UTM Airspace. The UTM architecture can be found in subsection 2.3.2. It is expected that for the delivery of medical items, the same entities will be involved, with one or multiple key UAS operators being in charge of UAV operations within an urban area, such as the city of Rotterdam. Other key actors involved in UTM are the Flight Information Management System (FIMS), Supplemental Data Service Provider (SDSP), and the UAVs. Besides the actors involved from an aeronautical perspective, distribution centres and medical facilities from the medical perspective will be taken into account. Especially large distribution centres placed outside the city will be of importance, as it allows for the incorporation of a vertiport. Regarding the medical facilities, several facilities will be taken into account, each with specific supply needs. A complete overview of healthcare facilities and needed supplies is given in Table 3.1.

Furthermore, the FIMS will provide airspace constraint data, such as no-fly zones. It will also provide the regulatory framework. The SDSP will provide terrain and weather data. In order to model medical UAV operations in the Rotterdam Area, data from both sources needs to be included. Terrain information can be obtained from a geographical city data source, called OpenStreetMap. For No-fly zones a geofencing system can be used, as described in section 3.2. Keep-out geofences can also be used for keeping safe separation from obstacles and other UAVs. In order to allow emergency missions to reach their destination as fast as possible, geofencing volumes need to be dynamically adjustable to allow these UAVs to pass through no-flying zones, similar to HEMS operations. Regarding the airspace itself, no further constraints are imposed, as the Full Mix concept is selected for class G operations.

In order to adress the UAV medical delivery problem, a comprehensive framework is needed to show the different stages involved. The framework includes the process involved, starting from the moment a medical package is order up to its delivery. A preliminary overview can be seen in Figure 4.1, showing the process from ordering to delivering parcels by UAVs. During both the initial path planning of the UAVs by the operator as well as the in-flight path planning the environment needs to be taken into account carefully, with emergency

path planning allowing for less restricted airspace usage. Planning and coordinating UAVs with different allowable and adjustable airspace access will be complex and needs to be planned for by the UAV operator. For the proposed agent-based model, the distribution centre and UAV operator will be represented by the same agent, being the operator agent.

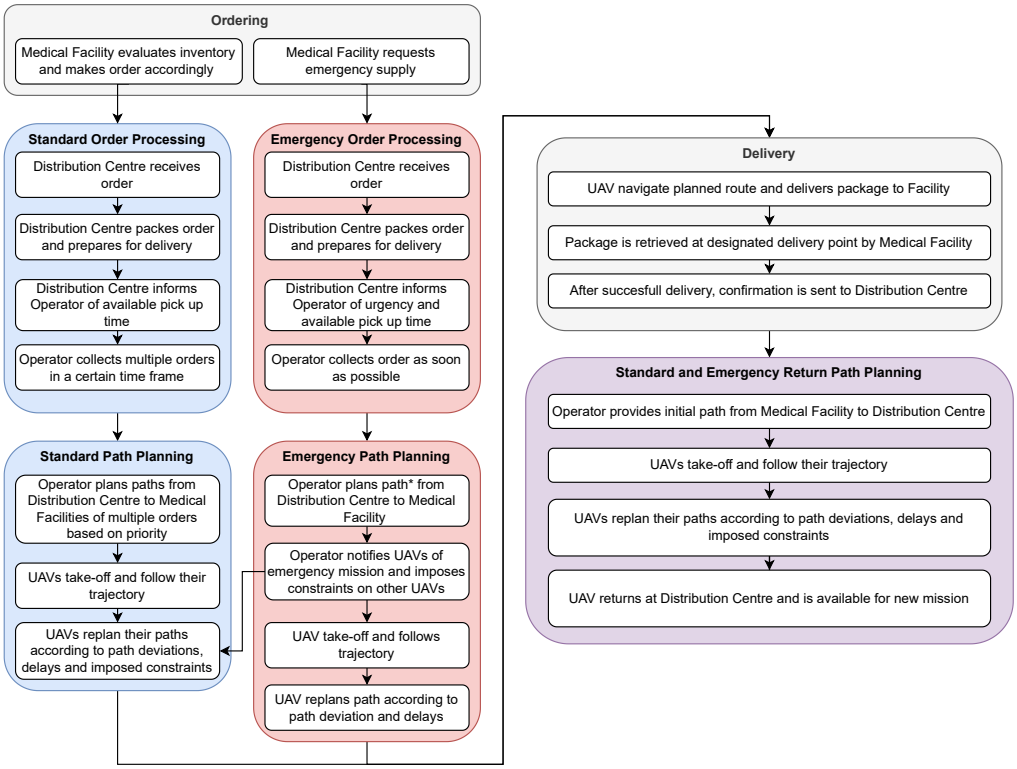


Figure 4.1: Preliminary overview of the proposed system focusing on ordering and delivering of medical supplies.

## 4.2. System Objectives

The current medical delivery supply chain is facing multiple challenges, including challenges regarding inventory management, timely delivery of (emergency) supplies, and controlling temperature. The extension of the current system by using drones for delivery of medical supplies can support inventory management and timely delivery of (emergency) supplies. The scope of this thesis will focus on real-time path planning in a complex dynamically adjustable urban environment for medical services. Therefore, the proposed system will have the following two main objectives:

1. **Ensure on-time delivery of (emergency) medical supplies by enabling fast UAV operations.**
2. **Enable UAV operations into urban environment respecting the geofences of a real-world environment.**

## 4.3. System Requirements & Assumptions

Based on the presented urban air mobility use case, as described in section 2.4, and the proposed system, as described in subsection 4.1.2 system requirements and assumptions will be made. Requirements and assumptions for the modeling of the UAV Medical Supply Delivery problem can be found in subsection 4.3.1 and Table 4.2 respectively. Most requirements and assumptions follow from information provided in the previous discussed chapters and sections. Both the requirements and assumptions fall under different categories, which are Delivery, Vehicle, Planning & Coordination, and Environment. The requirements and assumptions following from the proposed system will have a great influence of the planning & coordination of UAVs in urban environment. Planning & coordination can be achieved and modelled by use of Multi-Agent Pathfinding (MAPF) algorithms, which will be discussed in chapter 5. Based on these requirements, the best suited MAPF algorithm will be selected.

### 4.3.1. Requirements

An overview of the initial requirements for the proposed system, being UAV medical supply in urban environment, can be found in Table 4.1.

Table 4.1: Requirements for the modelling of the UAV Medical Supply Delivery problem

| Identifier | Category | Requirement | Section |
|---|---|---|---|
| REQ-DEL-01 | Delivery | *UAV Delivery shall only be considered for the delivery of medical supplies* | 2.4 |
| REQ-DEL-02 | Delivery | *UAV Delivery shall provide timely delivery of medical items* | 4.1 |
| REQ-VEH-01 | Vehicle | *The UAV shall have sufficient payload capacity to carry various medical supplies* | 2.2, 2.4 |
| REQ-VEH-02 | Vehicle | *Only UAVs that are be able to communicate to both the Operators as well as other UAVs in proximity shall be allowed in the environment* | 2.3 |
| REQ-VEH-03 | Vehicle | *The UAV shall be able to fullfill the time requirement required to deliver (emergency) medical supplies within the Rotterdam Area* | 2.2 |
| REQ-VEH-04 | Vehicle | *The UAV shall have VTOL capabilities* | 2.2 |
| REQ-PC-01 | Planning & Coordination | *The UAV Operator shall have to provide conflict-free paths to UAVs* | 3.1, 3.2 |
| REQ-PC-02 | Planning & Coordination | *The UAV shall have to be able to replan its path partially in real-time in case of new constraints and/or path deviations* | 4.1 |
| REQ-PC-03 | Planning & Coordination | *UAV path planning and coordination shall provide mission prioritisation based on urgency levels of delivery item* | 4.1 |
| REQ-PC-04 | Planning & Coordination | *Operator shall be able to provide UAV path planning and coordination for multiple agents* | 4.1 |
| REQ-ENV-01 | Environment | *The UAV Delivery System shall be able to operate in urban environments without collisions of UAVs* | 3.2 |
| REQ-ENV-02 | Environment | *The environment shall be modelled to incorporate geographical characteristics of the delivery area in 3D* | 3.1, 3.2 |
| REQ-ENV-03 | Environment | *Obstacles and no-fly zones shall be modelled as keep-out geofence to ensure safe operations* | 3.2 |
| REQ-ENV-04 | Environment | *No-fly zones shall be dynamically adjustable to allow for emergency missions* | 3.2 |

### 4.3.2. Assumptions

An overview of the initial assumptions for the modelling of the UAV medical supply problem can be found in Table 4.2.

Table 4.2: Assumptions for the modelling of the UAV Medical Supply Delivery problem

| Identifier | Category | Assumption | Section |
|---|---|---|---|
| ASS-DEL-01 | Delivery | *Delivery location of a customer (medical facility) will be located at predefined delivery areas* | 2.1, 4.1 |
| ASS-DEL-02 | Delivery | *Pick-up of delivery item will occur only at selected warehouses containing sufficient inventory of medical supplies to meet demand* | 2.1, 3.1 |
| ASS-DEL-03 | Delivery | *Delivery of package will occur using a cable-delivery system, allowing the drone to hover above the customer location* | 2.1, 2.2 |
| ASS-VEH-01 | Vehicle | *Iniatally, only Wingcopter 178 will be available within the UAS fleet; UAS characteristics will be modelled after Wingcopters technical specifications.* | 2.2 |
| ASS-VEH-02 | Vehicle | *Batteries of UAVs can be changed at delivery centres, where swapped batteries will be completely recharged* | 2.2 |

| | | | |
|---|---|---|---|
| ASS-VEH-03 | Vehicle | *The UAV will have sufficient battery life to complete delivery mission without recharging* | 2.2 |
| ASS-VEH-04 | Vehicle | *UAVs will be able to communicate to Operators at all times* | 2.3 |
| ASS-VEH-05 | Vehicle | *UAVs will be able to communicate to other UAVs (V2V communication) only in close proximity (up to 500 meters)* | 2.3 |
| ASS-VEH-06 | Vehicle | *UAVs will be able to meet requirements and regulations regarding transportation of medical items* | 4.1 |
| ASS-VEH-07 | Vehicle | *UAVs will not have any technical difficulties during operations, such as loss of communication or engine failure* | 4.1 |
| ASS-VEH-08 | Vehicle | *Operator-to-drone and drone-to-drone communication will not have a delay of more than 1 seconds.* | 2.3 |
| ASS-PC-01 | Planning & Coordination | *One UAV Operator will be able to plan and coordinate all task allocations and flight paths for all UAVs* | *2.3* |
| ASS-PC-02 | Planning & Coordination | *UAV will only know its own task and planned flight path; Only in close proximity of other UAVs flight plans will be shared to avoid conflicts* | 2.3 |
| ASS-PC-03 | Planning & Coordination | *UAV will know path constraints in case an emergency mission takes place* | 2.3 |
| ASS-PC-04 | Planning & Coordination | *Only UAV operations from one Operator will be taken into account* | 2.3, 4.1 |
| ASS-ENV-01 | Environment | *Influence of different weather conditions will not be taken into account (excluding wind conditions)* | - |
| ASS-ENV-02 | Environment | *Only statical objects (e.g. buildings) and 1 type of dynamical objects (other delivery drones) will be taken into account. No interaction with humans, cars, birds, etc. during flight.* | 3.2 |
| ASS-ENV-03 | Environment | *Urban airspace will be up to a maximum flight altittude of 120 meters above ground level.* | 2.3 |

# 5

# Multi Agent Pathfinding

Multi agent pathfinding (MAPF) is a growing area of research within the field of Artificial Intelligence (AI). The MAPF problem is to plan paths for multiple agents within a shared environment. This allows agents to reach their destination without the collision of agents. This chapter will foremost focus on classical MAPF algorithms. First, section 5.1 will provide an introduction into MAPF, by giving the problem definition, discussing the difference between centralised and distributed MAPF methods, and the different objective functions that can be used. Second, section 5.2 gives an overview of the to be discussed MAPF algorithms, where the algorithms are being split up based on optimality. For optimal algorithms, section 5.3 and section 5.4 will provide information on optimal reduction-based and search-based algorithms respectively. Regarding sub-optimal algorithms, rule-based and search-based algorithms will be discussed in section 5.5 and section 5.6 respectively. After a deep-dive into classical MAPF algorithms, section 5.7 will shortly elaborate on beyond classical MAPF algorithms incorporating large agents, kinematic constraints, and reinforcement learning. section 5.8 will present key insights obtained from previous work combining UAV path planning and MAPF algorithms. Next, a comparison of MAPF algorithms will be made in order to determine the most suitable MAPF algorithm to use for the medical delivery problem, which is presented in section 5.9. Lastly, based on the information presented within this chapter and chapter 2-chapter 4, a selection for a suitable simulation software tool will be made in section 5.10.

## 5.1. Introduction to Multi Agent Pathfinding

Multi agent pathfinding (MAPF) is the problem of finding paths for multiple agents within a shared environment, where agents have to reach their destination without having collisions between the agents or the environment [63, 96, 97]. In this section we will start with introducing the problem definition of classical MAPF, explaining the difference between centralised and distributed planning methods, and defining multiple possible objective functions.

### 5.1.1. Problem Definition

The classical MAPF problem with $k$ agents $(a_i, ..., a_{i+n})$ is defined by the tuple *(G, s, g)*, where G = (V, E) is an undirected graph, whose vertices the agent can occupy, $s : [1, ..., k] \rightarrow$ V is a function that maps the agent to a start vertex, and $g : [1, ..., k] \rightarrow$ V is a function that maps the agent to a goal or target vertex [96, 97]. For a classical MAPF it is assumed that time is discretised. For every time step $t$, the agent is located on one of the vertices and can perform a single action. An agent can perform two actions, which are to wait or to move. Waiting indicates that the agent will stay at its current location for one time step. Moving indicates that the agent will move from its current location to an adjacent location within the next time step [96]. A solution to the MAPF problem is valid once all agents reach their goal locations without collisions between agents. This is achieved by planning a path $p_i$ for agent $a_i$ without having a conflict with the paths of other agents. A conflict arises when two or more agents are planned to occupy the same location at the same time. If this conflict is not resolved, it will result in a collision. A conflict occurs if agents are planned to occupy the same vertex at the same time (Vertex conflict), or if agents are planned to transverse the same edge at the same time in the same direction (Edge conflict), or if agents are planned to swap locations in one time step (Swapping conflict) [97]. A swapping conflict is also indicated in literature as an edge conflict and will be from here

onward be indicated similarly. A joint plan denotes the set of plans or paths of single-agents, for each agent.

### 5.1.2. Centralised & Distributed Pathfinding

MAPF problems can be categorised into two distinct groups, which are centralised and distributed planning methods [90]. Centralised planning methods use a single central computing power, which needs to find a global solution for all $k$ agents. Centralised planning also encompasses the planning of paths where each agent plans its own path, but the knowledge is shared among the agents and centralised entity has control over all agents. For distributed planning methods, each agent is responsible for the planning of its own path. Distributed conflict resolution methods for agents can be prescribed, reactive or explicitly negotiated [81]. Both centralised and distributed MAPF methods have their benefits. For centralised methods, global optimal solutions can be guaranteed by using global constraints and interactions between agents. Also a high level of coordination between agents can be achieved, as complete information can be shared between agents. The benefits of distributed MAPF methods are that they are in general more scalable and suitable for problems with large number of agents. For centralised methods, the problem size can scale exponentially with the number of agents involved. Using a decentralised approach also allows for a more robustness, as agents will be able to adapt quickly to unexpected challenges.

The UTM Architecture proposed by the FAA and NASA [27], shows that a distributed planning method will be used, when considering planning and coordination for multiple UAS Operators within a shared environment. However per UAS Operator the planning and coordination will be centralised, especially in pre-flight conditions. During operations, when uncertainty is introduced, distributed planning methods can be utilised to resolve unforeseen conflicts. This work will focus on the operations of one UAS operator only, and therefore centralised methods are more suited to the problem at hand.

### 5.1.3. Objective Functions

In general, MAPF problems can have multiple valid solutions, which are based on the chosen objective function. The two most common objective functions are makespan and sum of costs. Makespan is denoted as the number of time steps required for all agents to reach their goal. Sum of costs is denoted as the sum of time steps for each agent to reach its goal or target. The goal of an agent is the desired state that an individual agent aims to achieve. The goal of an agent can encompass more than only reaching a certain location, but could also include to deliver a package or to perform certain actions. A target often refers to a location that an agent aims to navigate to. Reaching the target can be the goal of the agent, but can also be a sub-goal in order to achieve the final goal of the agent. Once an agent reaches its goal and the goal is to stay at the target it needs to be determined how staying at the target will influence the sum of costs. A possibility could be that if the agent waits at its goal, the sum of costs does not increase by the waiting time of that agent. [96, 97] Traditionally for classical MAPF, makespan is considered for reduction-based MAPF, while sum of costs is mostly used for search-based MAPF algorithms. Reduction-based and search-based algorithms will be explained in section 5.2-section 5.4. Other objective functions are for example flow-time, which is the objective to minimise the average length of paths of the agents [115], or to maximise the number of agents that reach their goal within a certain time span [64]. These objective functions are less common in classical MAPF problems, but do become more popular for Beyond Classical MAPF problems.

## 5.2. MAPF Algorithms Overview

MAPF algorithms can be classified using different approaches. This section will split classical MAPF algorithms based on optimally, which is optimal and (bounded) suboptimal MAPF algorithms. Optimal solutions for MAPF algorithms aim to minimize the objective function, as presented in subsection 5.1.3. Optimal methods guarantee e.g. the minimal cost required for each agent to reach their goal. (Bounded) Sub-optimal methods provide solutions that are not optimal, but do provide the benefit of being more computational efficient and scalable. Figure 5.1, shows an overview of the different classes of algorithms. The following sections will provide an overview of the different algorithms and will focus on why an algorithm would be fit or would not be fit to the problem at hand, which is UAV Medical Supply Delivery. Algorithms that are deemed not suitable will only be discussed briefly for completeness. section 5.3 will give an overview of the optimal reduction based MAPF solvers. section 5.4 will give an overview of the optimal search-based MAPF solvers, which will include a deeper insight into A*-based optimal algorithms, Increasing Cost Tree Search (ICTS) algorithms and Conflict Based-Search algorithms in subsection 5.4.1, subsection 5.4.2, and subsection 5.4.3 respectively.

Besides the optimal MAPF solvers, the Bounded Sub-Optimal solvers will be discussed in section 5.5 and section 5.6 for rule-based and search-based bounded sup optimal solvers respectively. section 5.6 will use a similar structure to section 5.4 when discussing the different algorithms.

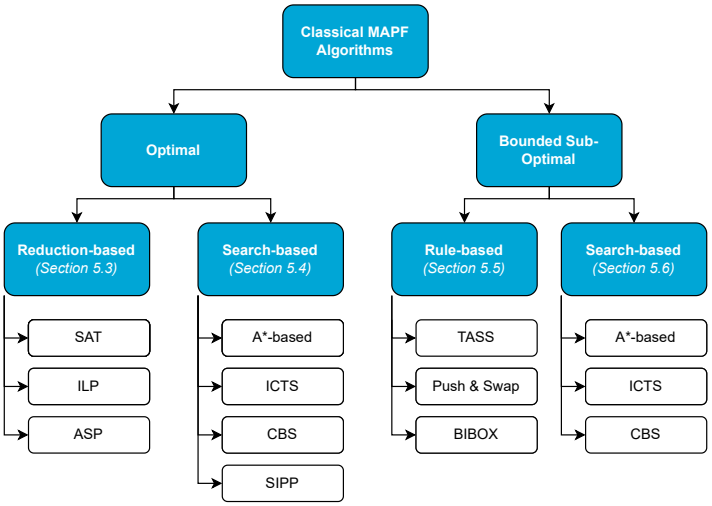


Figure 5.1: Overview of Classical MAPF algorithms

## 5.3. Optimal Reduction-based MAPF Algorithms

The aim of reduction-based MAPF algorithms is to decompose the problem into smaller sub-problems that can be solved using other well-studied problems. Three examples of this class, are boolean satisfiability (SAT) [46], Integer Linear Programming (ILP) [121] and Answer Set Programming (ASP) [25]. Reduction-based algorithms are complete for all MAPF instances. The objective function for this class is makespan, but can be modified to solve MAPF with other objective functions, such as sum of costs, optimally. Also adapting for (bounded) sub-optimally is possible, however both adaptations are not deemed trivial. It must be noted that for search-based algorithms with sum of costs objectives, it is deemed easily possible to makespan, which is an advantage if one wants to compare solutions using different objectives [63].

SAT Algorithms make use of boolean variables (True or False), which is used to express the graph, agents locations and constraints. The following constraints are applied, which are constraints to ensure valid starting and goal locations, constraints to ensure agents occupy only one vertex per time step, constraints to ensure one vertex is occupied by one agent per time step, and constraints to ensure correct movements. The optimal makespan can be found by increasing the total time until a satisfiable solution is found. [46, 100]

Integer Linear Programming (ILP) [121] can also be used to solve MAPF problems, and allows for easy adaptation of objective functions. ILP is a type of optimisation problem, with variables that consist only of integers. Both the objective function and constraints are linear. The benefit of using ILP, is that when implemented correctly can allow for a 100% success rate, however ILP does not scale well computationally for increasing size of either agents or environment.

Answer Set Programming (ASP) [25] models the path finding problem as a so-called program and tries to solve the problem by computing models. These comping models are called answer sets. The benefit of ASP is that it allows for a framework which can solve multiple path finding variations. It can also be applied to any sort of graphs. The downside is that the finding of optimal solutions, including constraints for collision types, does not computationally scale well.

The key strength of reduction-based solvers is to find solutions for small densely populated maps with many obstacles. However, for large maps reduction-based solver can be unable to find solutions, even with a small number of agents [46]. Even though the urban environment consist of areas made up with a high density of obstacles (e.g. buildings and trees), the environment itself will be large. It also expected that UAVs are able to pass each other within streets, by passing each other vertically and/or horizontally, depending on the street

characteristics.

## 5.4. Optimal Search-based MAPF Algorithms

The search-based optimal algorithm class is the one of the most prominent MAPF solvers. In general, search-based algorithms perform better on large sparsely populated maps, while having trouble finding a solution on small densely populated maps, which is the opposite for reduction-based algorithms. The search-based algorithms can be divided into three subclasses, which are Extensions of A*, Increasing Cost Tree Search (ICTS) and Conflict Based-Search, which are discussed in subsection 5.4.1, subsection 5.4.2, and subsection 5.4.3 respectively.

### 5.4.1. Extensions of A*

The A* algorithm is one of the most used path finding algorithms within the field of AI, and can both be applied to single agent path finding as well as multi agent path finding. First, single agent path finding will be discussed.

#### Single Agent Path Finding A*

A* can be used for a single agent to find the shortest path from a start node to a goal node on a graph [96]. Other well-known path finding algorithms are Dijkstra's Algorithm and Best-First-Search. The downside of Dijkstra's algorithm is that the algorithm explores a significant amount of nodes, as it will expand radial on a graph. However, it is guaranteed to find the shortest path. Greedy Best-First-Search makes use of an heuristic to determine which node should be explored next, which does not always result in the shortest path, especially when concave obstacles are involved [3], but is able to provide answers in a faster manner. It is not quantified how much faster it is able to provide a solution.

The A* algorithm uses a heuristic search algorithm that uses the following cost function, $f(n) = g(n) + h(n)$, where $g(n)$ is the current cost to reach node $n$ from the start node and $h(n)$ is a heuristic that estimates the cost of reaching the goal node from the current node $n$ [13, 96]. The common heuristic functions used for this algorithm are heuristic distances such as the Manhattan distance, Euclidean Distance, or excluding distance [31]. These heuristics calculate the distance from the current node to the goal node, without taking obstacles into account. The Manhattan distance is the absolute difference between two coordinates [49]. The Euclidean distance is the distance between two coordinates, when following a direct line between the two nodes [49].

A heuristic is admissible iff $h(n) \leq h^*(n)$, where $h^*(n)$ is a perfect heuristic and $h^*(n)$ is known for all nodes . If the heuristic is admissible, it is guaranteed to find an optimal solution. The pseudo-code of the A* algorithm can be found in Algorithm 1.

#### Multi-agent path finding A*

The above mentioned algorithm can also be adapted for MAPF, by running A* on the $k$-agent search space, where the states represent the different possibilities to place $k$ agents into all |V| vertices [28, 96]. An admissible heuristic is to sum the individual heuristics for each agent, using a distance heuristic, such as Manhattan or Euclidean distance. A more informed heuristic is the sum of individual costs, which is also admissible. For each agent $a_i$ the optimal path is calculated from its current node to the goal node of the specific agent. The sum of these costs over all agents is then used as the heuristic. This heuristic can be calculated during operations of the agent [91] or can be computed in a pre-processing phase [93]. The latter uses a breadth first search from the agents goal position to every free grid position [93].

One of the main drawbacks of using A* for MAPF is that the state-space is exponential in the number of agents. Also the branching factor of a given state can be exponential in the number of agents. The branching factor is the number of possible positions an agent can occupy in the next time step. Both drawbacks combined can lead to computational infeasability, especially when many agents are involved. In order to reduce the computational time of A*, multiple enhancements have been made. We will focus on the following 3 enhancements, which are Independence Detection (ID) [94], Operator Decomposition (OD) [93], and Enhanced Partial Expension (EPEA*) [35]. Also an A*-based algorithm called M* [109] will be explored.

**Algorithm 1** A-star Algorithm taken from Candra, Budiman and Pohan [13]

```
1: open_list = set containing start
2: closed_list = empty set
3: start.g = 0
4: start.f = start.g + heuristic(start, goal)
5: while current is not goal do
6:     current = open_list element with lowest f cost
7:     remove current from open_list
8:     add current to closed_list
9:     for each neighbor not in closed_list do
10:        if neighbor not in closed_list then
11:            neighbor.f = current.g + heuristic(neighbor, goal)
12:            if neighbor is not in open_list then
13:                add neighbor to open_list
14:            else
15:                openneighbor = neighbor in open_list
16:                if neighbor.g < openneighbor.g then
17:                    openneighbor.g = neighbor.g
18:                    openneighbor.parent = neighbor.parent
19:            if open_list is empty then
20:                return false
21:            return backtrack_path(goal)
```

### Independence Detection (ID)

Independence Detection (ID) [94] is an enhancement to A*, which divides agents into independent groups. These groups are afterwards solved separately. Agents are independent if there is a solution for each group in which the solutions do not have any collisions. Independent groups can be created by first finding an optimal path for each agent separately and afterwards combining them in groups if conflicts are present. Combining is done for the first two agents that have a conflict and is resolved by a global search A* algorithm, e.g. A* + Operator Decomposition (OD) [94]. The computational time of solving the MAPF is dominated by the largest independent group of $k$' agents, and is therefore able to reduce the computational time $|O(V)^k|$ to $|O(V)^{k'}|$ [28].

### Operator Decomposition (OD)

Operator Decomposition (OD) [93] is an improvement to A*, which reduces the branching factor from ${b_{agent}}^k$ to $b_{agent}$. This is done by avoiding surplus nodes. Surplus nodes are nodes, whose cost is greater than the cost of the optimal solution, and are therefore not needed in order to compute the optimal solution. By avoiding to generate these nodes, a reduction of the computational effort can be obtained [35]. In the standard A* algorithm, every agent moves (or stays within) a state for every time step. OD only considers movement of the first agent, where the order of the agents is arbitrary, but fixed. Standard states are when no agent has been assigned a move. Intermediate states is when at least one agent has been assigned a movement. At the intermediate states, the movement of agents without assigned movements are considered only. This will result in the generation of new intermediate nodes. Assigning movement to the last unassigned agent in an intermediate state, results in a standard state. After a solution is found, the intermediate nodes in the open_list are not developed in standard nodes, resulting in a reduction of the surplus nodes [28, 93].

### Enhanced Partial Expension (EPEA*)

Enhanced Partial Expension (EPEA*) [35] similar to OD reduces the generation of surplus nodes to decrease the computational time. EPEA* avoids the generation of surplus nodes by using a priori [1] domain knowledge. EPEA* generates only nodes with $f(n_c) = f(n)$. Using a priori domain- and heuristic specific knowledge a list of operators is created, which leads to a list of nodes with the needed f-value, without creating surplus nodes. This is achieved by using an Operator Selection Function (OSF). The OSF is able to return for the expansion of node $n$, the set of child nodes with $f(n_c) = f(n)$, as well as able to store the minimum f-value

[1] a priori means *from what is before* in Latin. It means that a state can be derived from reasoning alone

with $f(n_c) > f(n)$. The latter is done as this node can be useful, but it cannot be proven in the current phase of the search. The first child nodes are classified as provably useful and the second nodes are classified as possibly surplus. Provably useful is defined as it can be mathematically be proven that the solution is useful. A possibly surplus node can become provably useful when node $n$ is re-expanded with a higher stored value. An example of the OSF is visualised in Figure 5.2. The goal node is located North-West of the current node. Using the Manhattan distance heuristic, the distance from the nodes located in each directions from the current nodes are calculated, which is visualised on the right-hand side of Figure 5.2. Moving increases the value by 1, which leads to a value of 4 for moving in a Northern or Western direction, while moving in Eastern or Southern direction leads to a value of 6. $f(n) = 4$, resulting in a value difference of +0 for the Northern and Western movement and +2 in Eastern and Southern direction, which is visualised on the left-hand side of Figure 5.2. [35]

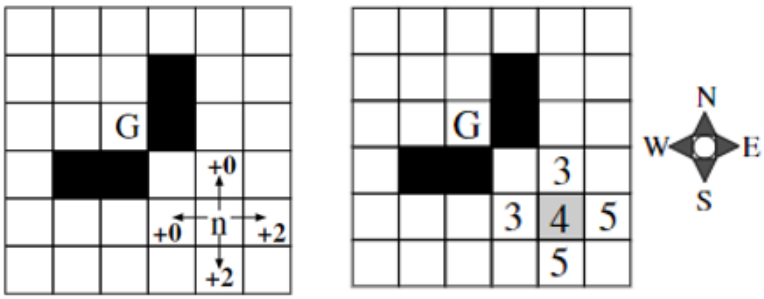


Figure 5.2: Example of a full-checking OSF implementation in EPEA* using the Manhatten distance heuristic, figure taken from Goldenberg et al. [35]. Left-hand side displays the difference between $f(n)$ and $f(n_c)$ and the right-hand side displays the f-value of the heuristic.

### M*

M* [109] is an A*-based algorithm, that also searches the k-agent search space, similar to A*. One of the main limitations of A* is the exponential branching factor. For M*, the branching factor and the dimensionality is dynamically changed based on the number of conflicts. Dimensionality is denoted as the number of agents that is not allowed to have conflicts [28]. The branching factor is changed in the following manner. Initially, when a node is expanded, the M* algorithm only generates a child node which is part of the optimal individual path of each agent $a_i$. This means that every agent $a_i$ will 'move' a node until a conflict is found between $q \geq 2$ agents [28]. For only the conflicting agents the nodes from the start node to the current node are re-expended, generating nodes for all the possible actions these agents can perform and finding a collision free path for these agents. The collisions itself are stored in a conflict set. For the other agents only the optimal path will be followed [96]. This process is repeated until all agents find their goal without collisions.

Several improvements to the M* algorithm were made. The first being Recursive M* (rM*) [109], which identifies disjoint conflict subsets and combines them. Afterwards paths are found for these sub problems recursively. Therefore, rM* has similarity to ID, as it identifies sub problems which can be solved independently [96]. Another improvement is the combination of rM* with OD, which is called ODrM* [29]. The rM* algorithm makes uses of OD for the underlying planner. It is used for first finding an optimal path for each agent $a_i$, and secondly for finding solutions for the conflict set. Ferner et al.[29] demonstrate that ODrM* outperforms both rM* and OD. A typical 8-connected grid world with 32x32 cells with a 20 percent change of a cell being marked as an obstacle was used to perform all tests. The start and goal position of each agent was chosen randomly and the number of agents was varied from 5 up to 60. A total of 100 simulations per agent number were performed, each with a random environment. 5 minutes were given to find a solution. The number of successful simulations as well as the mean time to find a solution were used to compare the algorithms. The time to find a solution is used as a performance indicator. It was demonstrated that ODrM* outperforms M*, as M* has a 20-30 percent lower success rate compared to ODrM*, when 25-30 agents are involved. Also combining OD with rM* leads to a 60 percent improvement in performance, for problems that include 20 agents.

In conclusion, based on the results from Ferner et al.[29] the combination of rM* and OD performs the best within the subclass of optimal search-based MAPF Algorithms, searching the k-agent state-space.

### 5.4.2. The Increasing Cost Tree Search

Another optimal search-based MAPF algorithm is the Increasing Cost Tree Search (ICTS) [88], which is different compared to the extensions of A*, as it does not search the k-agent space directly [96]. ICTS divides the MAPF problem into two problems based on the understanding that the complete solution for problem is built from individual solutions of the agents. An individual solution is the path from start to goal. The two problems are the minimal cost of each agents path to find an optimal solution per agent, and finding a set of non-conflicting paths for all agents, given their individual costs. These two problems are solved for ICTS in different levels of the algorithm, called the high-level search and low-level search. The goal of the high-level search is to find a minimal cost solution in a search space that spans combinations of cost of each agent. The low-level search acts as a goal test for the high-level search, where it searches for a valid solution under cost constraints given by the high-level search. [88]

The high-level search is performed on the increasing cost tree, which consist of nodes, a root of the tree, a successor function, and a goal test. First, every node $s$ consists of a $k$-vector including the individual path costs, denoted as $[C_1, C_2, ..., C_k]$. Node $s$ represents all possible complete solutions where the individual path cost of agent $a_i$ is equal to $C_i$. For every node within the same level, the total cost is the same. Second, the root of the ICT is the optimal cost of individual paths without taking other agents into account, similar to the sum of individual costs in A*, which is denoted as $[opt_1, opt_2, ..., opt_k]$. Third, each node will generate k successors, where each successor increments the cost of agent $a_i$, increasing the total cost function by 1. Lastly, an ICT node is a goal node if the individual path cost of each agent is equal to $C_i$ and there is a non-conflicting solution. The depth of the optimal goal node is denoted by $\Delta$. The depth of the search is dependent on the number of agents $k$, the topology of the searched map and the ratio of agents to vertices on the map. [88]

For the low-level search, all paths of cost $C_i$ are stored in a compact data-structure called multi-value decision diagram (MDD). The low-level searches the cross product of the MDDs to find a set of $k$ non-conflicting paths for the different agents. This can be searched using a depth-first branch-and-bound as this is not an optimisation problem. If a set is found, the low-level search returns *True*, and the search is stopped. If *False* is returned, the search continues to the next ICT node via the high-level search. [28, 88]

An extension in order to speed up ICTS is to use pruning techniques. The goal of pruning is to remove sections of the ICT, by quickly identifying subsets of single-agent plan costs for which there is no valid solution. If such a subset exists, the node can be declared a non-goal node and the high-level search can continue. Different pruning techniques exist, such as Simple pairwise/triple pruning, enhanced pairwise/triple pruning, and repeated enhanced/triple pairwise. Sharon et al.[88] demonstrate that the combination of ICTS + enhanced triple pruning (3E) perform best.

### 5.4.3. Conflict Based Search

Next, Conflict-based Search (CBS) [90] similar to ICTS [88] uses a two-level search in order find an optimal solution for all agents. CBS is used in a variety of sophisticated real-world scenarios, and will therefore be explained in greater detail, starting with the classical CBS.

#### Classical CBS

The CBS algorithm is not based on A*, and does not search the k-agent space directly [96], which is similar to ICTS. CBS uses the following definitions. Paths are only considered for a single agent from their respective start to goal vertex. A solution is the set of $k$ paths for the given set of $k$ agents. Furthermore, a constraint, denoted as a tuple $(a_i, v, t)$, is a vertex $v$, where agent $a_i$ is prohibited to be at time step $t$. A consistent path is a path of agent $a_i$ where all constraints are met for agent $a_i$ . A consistent solution is the set of $k$ consistent paths. A conflict, denoted as tuple $(a_i, a_j, v, t)$, is when agent $a_i$ and $a_j$ are occupying vertex $v$ at time step $t$. A solution is valid if all $k$ paths have no conflicts. A consistent solution can be invalid, if the paths have conflicts, even when meeting the own constraints of each agent. The main idea of CBS is to develop a set of constraints for each agent and find paths taking into account these constraints. When conflicts arise, new constraints are formed. CBS uses a two-level search algorithm, where the goal of the high-level search is to find conflicts and add constraints. The low-level search focuses on updating the agents path to be consistent with the new constraints. The high and low-level search will be explained in more detail below. [28, 90]

The high level of CBS searches the constraint tree (CT), which is a binary tree, where each node $N$ consist of the following data: *N.constraints, N.solution,* and *N.cost.* Regarding the set of constraints (*N.constraints*), the CT root, is an empty set. The child nodes in the CT inherit the constraints of the parent node and adds one new constraint for each agent. The solution (N.solution) is a set of $k$ paths. The paths for each agent $a_i$ are found using the low-level search. The total cost (*N.cost*) is the summation of all cost over all $k$ agents, which is denoted as the f-value of the node. The nodes in the CT are ordered based on their costs, using a best-first search. [28, 90]

A node in the CT is processed in the following manner. The low-level search is used given a list of constraints for a node $N$ and returns an optimal path for individual agents given their constraints. For finding the optimal path in the low-level search, any optimal path finding algorithm can be used, such as A*. Ties between nodes with the same f-value are broken in favor of paths with fewer conflicts. If a consistent path is found for each agent, the paths are validated by simulating the movements of the agents along their planned solutions. If all agents reach their goal without any conflict, $N$ is declared a goal node, the high-level search is stopped, and *N.solutions* is returned. If the solution is invalid, the node is declared a non-goal node and the high-level search continues. [28, 90]

Given a non-goal CT node, a conflict $(a_i, a_j, v, t)$ is resolved in the following manner, which is called as a split action. For a valid solution a constraint for either agent $a_i$ or $a_j$ is imposed for vertex $v$ at time step $t$. The node $N$, containing the conflict, is split into two child nodes, where each node contains either a constraint for agent $a_i$ or agent $a_j$. The low-level search is then afterwards only used for the agent with the imposed constraint. [28, 90]

The pseudo-code for CBS can be seen in Algorithm 2 for the high-level search, while the pseudo-code for the low-level search using simple A* can be seen in subsection 5.4.1 in Algorithm 1. Lines 1-7, and 20-22 belong to the classical CBS, while the colored lines belong to CBS improvements, which will be discussed later in this subsection. Algorithm 3 shows the pseudo-code for generating child nodes for CBS high-level search.

**Algorithm 2** CBS High-level Main Algorithm taken from Felner et al.[28]

```
1:  Init R with low-level paths for the individual agents
2:  insert R into open
3:  while OPEN not empty do
4:      N ← best from OPEN // lowest cost solution
5:      Simulate the paths in N and find all conflicts
6:      if N has no conflict then
7:          return N.solution // N is goal
8:      C ← find-cardinal/semi-cardinal-conflict(N) // (PC)
9:      if C is semi- or non-cardinal then
10:         if Find-bypass(N,C) // (BP) then
11:             Continue
12:     if should-merge(a_i, a_j) // Optional, MA-CBS then
13:         a_ij = merge(a_i, a_j)
14:         if MR active then
15:             Restart Search
16:         Update N.constraints()
17:         Update N.solution by invoking low-level(a_i j)
18:         Insert N back into OPEN
19:         Continue // go back to the while statement
20:     for each agent a_i in C do
21:         A ← Generate Child(N, (a_i, s, t))
22:         Insert A into OPEN
```

**Algorithm 3** CBS High-level Generate Child function taken from Felner et al.[28]

1: $A.constraints \leftarrow N.constraints + (a_i, s, t)$
2: $A.solution \leftarrow N.solution$
3: Update $A.solution$ by invoking $low\ level(a_i)$
4: $A.cost \leftarrow SIC(A.solution)$
5: **return** $A$

In order to test the performance of CBS with previously developed MAPF algorithms, Sharon et al. [90] demonstrate the success rate of EPEA*, ICTS + 3E (denoted as ICTS), and CBS on three different maps, each with their own topology. These benchmark maps are taken from the game Dragon Age: Origins (DAO). The first map, map den520d, has many open large space without bottlenecks. The second map, map ost003d, has a few open spaces and few bottlenecks. Lastly, the third map, brc202d, has few open spaces and many bottlenecks. The success rate of each algorithm per map can be seen in Figure 5.3. On all three algorithms ID is used to enhance performance. It can be seen that ICTS performs best on a map with large open spaces and few bottlenecks, while CBS performs in general best on maps with few open spaces and many bottlenecks. ICTS outperforms EPEA* on all maps, while CBS only outperforms EPEA* on the second and third map. The reason that ICTS outperforms CBS in large open spaces is that ICTS is more efficient at quickly expanding the search tree and therefore finding open spaces. This allows the algorithm to find solutions faster, as it is less focused on solving conflicts. However, for maps containing many bottlenecks an few open spaces, CBS will outperform ICTS as it is more efficient in resolving conflicts.

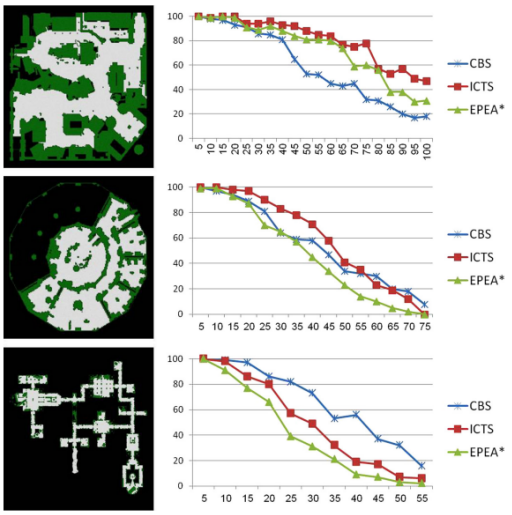


Figure 5.3: Success-rate CBS compared to ICTS and EPEA*, taken from Sharon et al.[90]

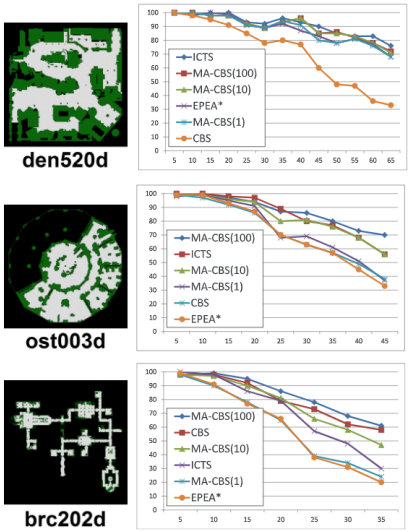


Figure 5.4: Success-rate of MA-CBS experiments using EPEA* as low-level solver, taken from Sharon et al.[89]

As classical CBS chooses to split up nodes arbitrarily, poor split up choices can increase the CT. Therefore several improvements were made to CBS, which will be discussed further in this section.

### Meta-agent CBS (MA-CBS)

The first improvement to CBS is Meta-agent CBS (MA-CBS) [89]. Classical CBS behaves poorly when there is a high rate of internal conflicts between agents. The number of conflicts could become so high, that using a coupled solver would lead to better results. Therefore MA-CBS adds the option to merge conflict agents into meta-agents instead of the split action. A meta-agent itself consist of $M$ agents and is treated as a single composite agent, whose state consists of a vector containing the locations of each agent. Meta-agents can be merged together with other (meta-)agents, however their node within the CT is never split. The pseudo-code for MA-CBS can be found in Algorithm 2, line (12-19). When agents are merged, the constraints are unified (line 16). Afterwards, the low-level search is invoked for the new meta-agent only (line 17). The merging of constraints must ensure the return of an optimal solution, which is why Sharon et al.[89] proposed a simple merge policy. Two agents $a_i$ and $a_j$ are merged into a meta-agent, denoted as $a_{ij}$, if the number of conflicts

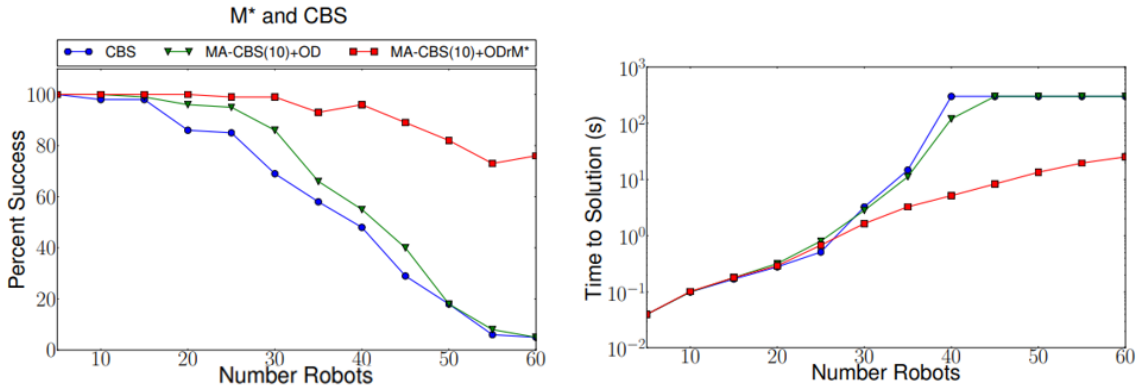


Figure 5.5: Comparison of simulation results between CBS, MA-CBS(10) + OD and MA-CBS(10)+ODrM*. The left plot shows the succes rate of found solutions within a 5-min time frame, the right plot shows the time to find a solution. Figure taken from Ferner et al.[29]

.

between the two agents exceeds a predefined parameter, denoted as *B*. The merge-policy parameter is denoted as MA-CBS(B). If this parameter is not preceded, the *should-merge* function (line 12) returns False and a normal split action is performed. Sharon et al.[89] demonstrate that MA-CBS outperforms classical CBS. Using three different maps, each with their own topology the success-rate of MA-CBS is tested. The number of *k* agents is increased from 5 to 40 in increments of 5. For each value of *k*, 100 instances are run. The results of the experiment can be seen in Figure 5.4. It must be noted that MA-CBS(0) is the same as A* (or equivalant extensions, such as EPEA*) + ID, while MA-CBS($\infty$) is the same as classical CBS. EPEA* has been chosen as the low level solver, as it outperforms classical A*.

The results in Figure 5.4 show that MA-CBS with non extreme values outperforms both MA-CBS(0) (which is EPEA* + ID, denoted as EPEA*) and MA-CBS($\infty$) for all three maps. In general if more agents are present within a map, denoted as map density, low values of B are more efficient (lower computational time). Furthermore, for maps with large open spaces and few bottlenecks, low values of B are also more efficient, while maps with many bottlenecks and few open spaces, high values of B are more efficient. Lastly, if a slow MAPF solver is used for the low-level search, high values of B are preferred. This is because the higher the B-value, the less often the low level search is invoked. The computational performance of the low-level search MAPF solver can be increased by using a ODrM* in combination with MA-CBS [29]. Using the same test environment as discussed in subsection 5.4.1 for M*, Ferner et al.[29] demonstrate that ODrM* using the MA-CBS framework outperforms both basic CBS and MA-CBS combined with the A* + OD coupled planner. The selected B-value for MA-CBS with ODrM* and MA-CBS with OD is 10. The results can be seen in Figure 5.5, where it is demonstrated that a success rate of 96% can be achieved for 40 agents, and almost 80% can be achieved for 60 agents. This compares to below 60% success rate for both CBS and MA-CBS(10)+OD for 40 agents and below far 20% for 60 agents. In a small environment with few agents the performance of all algorithms is comparable, but with more agents involved, hence more conflicts need to be resolved. This results in MA-CBS(10) + ODrM* outperforming both other algorithms.

### Merge and Restart

An improvement to MA-CBS [89] is to restart the search from the root node of the CT, when a merge decision is made. This improvement is called Merge and Restart (MR) [11]. A new root node is used, including the merged agents from the beginning. The pseudo code of this improvement can be found in Algorithm 2, in lines 14 and 15. MR is a simple to implement improvement, but does save computational effort, which will be elaborated on after discussing all improvements.

### Preferring Cardinal Conflicts

The second improvement to (MA-)CBS proposed by Boyarski et al. [11] is the splitting of nodes in the CT, when a cardinal conflict occurs. This improvement is called Preferring Cardinal Conflicts (PC). In general, conflicts can be divided into cardinal-conflicts, semi-cardinal conflicts and non-cardinal conflicts. For cardinal-

conflicts, when a conflict $(a_i, a_j, v, t)$ is split into constraint $(a_i, v, t)$ and $(a_j, v, t)$ and when invoking the low-level search, the cost for both agents' paths is increased including the constraint. Another definition is that optimal path of both agents $a_i$ and $a_j$ include location $v$ at time step $t$. Second, semi-cardinal conflicts are conflicts where only the cost of one agents' path is increased. Third, non-cardinal conflicts are conflicts where neither one of the two constraints lead to an increased cost. Using the definitions for cardinal conflicts, the improvement is included within the pseudo-code for Algorithm 2. In line 8, the conflicts are examined. If one of the constraints is cardinal, the node is split, indicating that the algorithm continues to line 20.

### Bypassing conflicts

The last improvement focuses on preventing the splitting of nodes, by bypassing the conflict. A conflict is bypassed by modifying the path of one the agents. This is improvement is called Bypassing Conflicts (BP) [12]. This especially is beneficial for conflicts that are semi-cardinal or non-cardinal. By modifying the path of one the agents instead of splitting the node the size of the CT is kept smaller, saving computational effort in the high-level search. The pseudo-code for BP can be seen in lines 10 and 11 in Algorithm 2. Two variants of BP are Peek at the Child (BP1) [12] and Deep Search for Bypasses (BP2) [12]. BP1 peeks at one of the two immediate children in the CT and tries to adopt their paths, while BP2 searches in a best-first manner through all children and tries to adopt the path with the lowest cost.

### Improved Conflict Based Search

The improvements MA-CBS, MR, PC and BP can all be optionally added to CBS separately. If all are combined together, the algorithm is called Improved CBS (ICBS). Boyarski et al. [11] use extensive experiment to compare different optimal MAPF algorithms. For the experiment the maps from DAO were used, where each map has a different topology. The selected MAPF algorithms are as follows. For ICBS, ICBS(25) (combination of MA-CBS+BP+PC+MR) is chosen. This is compared with MA-CBS(25)+BP [12], MA-CBS[89], EPEA*[35], and ICTS+p (ICTS + pruning) [88]. The results can be seen in Figure 5.6. First, from these results it can be concluded that ICBS outperforms other CBS variants for all graphs. ICBS also has in general the best performance regarding run time for all graphs. It can be noted that ICBS outperforms all algorithms in maps with few open spaces and many bottlenecks. However, when large open spaces are involved it has similar performance to EPEA* and ICTS. For previous versions of CBS, CBS always under-performed compared to ICTS and EPEA*, which can be seen in Figure 5.3 and Figure 5.4. It must be noted that no MAPF solver is best for all circumstances and all topologies and therefore the chosen solver, should be based on the use case.

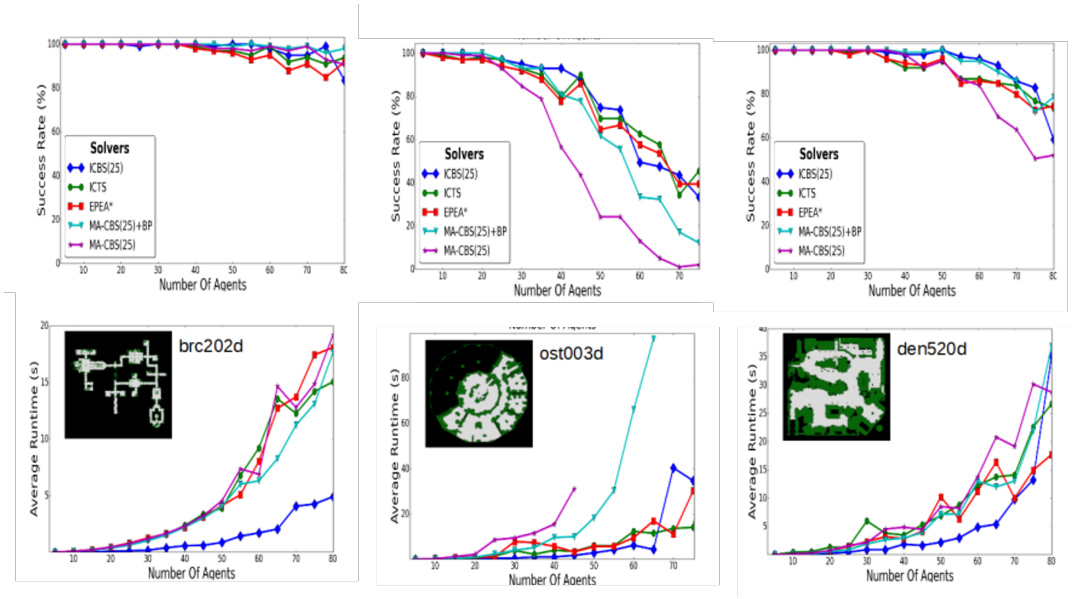


Figure 5.6: Comparison of simulation results between ICBS(25), ICTS, EPEA*, MA-CBS(25), and MA-CBS(25). The top plots show the success rate of found solutions for each map, the bottom plot shows the average runtime. Figure taken from Boyarski et al [11]

### 5.4.4. Safe Interval Path Planning

In the previous subsections, the discussed algorithms model the environment as if it is static. Dynamic obstacles, which in most cases are other agents, are turned into static obstacles, which can lead to computational inefficient planning methods. This is because it requires adding time as an additional dimension to the search space. In an online setting, every time a new agent is introduced, re-planning needs to take place, which decreases the computational performance even more. An algorithm that was developed for the planning of agents in dynamic environments is Safe Interval Path Planning (SIPP) [75]. SIPP does not have a state for every configuration and time step pair, but does use a grouping of collision free time steps. This grouping is called a safe-interval, and for a pair the configuration and the safe interval is used. This leads to a smaller search space, as the the maximum number of safe interval for any given configuration is at most the number of agents moving through the specific configuration. The safe interval is defined as the period of time in which there is no collision for a configuration. One time step before and after the safe interval a collision would occur. Only when a dynamic obstacle would not move through the configuration, there will not be a collision after the safe interval. The collision interval is the opposite of the safe interval. An example of a configuration with a timeline for safe intervals and collision intervals can be seen in Figure 5.7. The SIPP algorithm also makes use of a Dynamic Obstacle Representation, which tracks the dynamic obstacles (agents). This includes a list of all agents, where each agent has a radius and a trajectory. The trajectory exist of a list of points including state variables. The state variables specify the configuration, time, and point's uncertainty. The trajectory list also allows for more than one possible trajectories.

The planning exists of two phases, which are the Graph Construction and the Graph Search. The first phase exist of creating a timeline for each spatial configuration using the predicted trajectories of each agent. In the second phase, an A* search is run. The used A* algorithm has two alterations, which are the selection of successor states and how time variables are updated for states. For SIPP the states are expanded at the earliest time step possible for each location and time interval in order to ensure the maximum set of successors for that state. In short the overall algorithm divides the timeline of each agent into safe and collision time intervals. By calculating the next target point the agent can decide to move or stop at a certain location [114]. Wang et al, show that SIPP does not always work. This occurs in a situation when the path of one agent is a subset of of the path of another agent. As one agent is inclined to wait at a certain location until the collision interval has passed, the agent will end up in collision with the other agent. This specific example can be seen in Figure 5.8. The algorithm is suited for waiting actively to avoid obstacles, but does not have the ability to actively avoid agents. The SIPP algorithm similar to extensions of A* also be integrated with other MAPF solvers, such as CBS [4]. However the implementation is more complex compared to A*. Several extensions and integrations of SIPP exist, which are Any-Angle Pathfinding [119], Anytime SIPP [70], and SIPP with Kinodynamic Constraints [2].

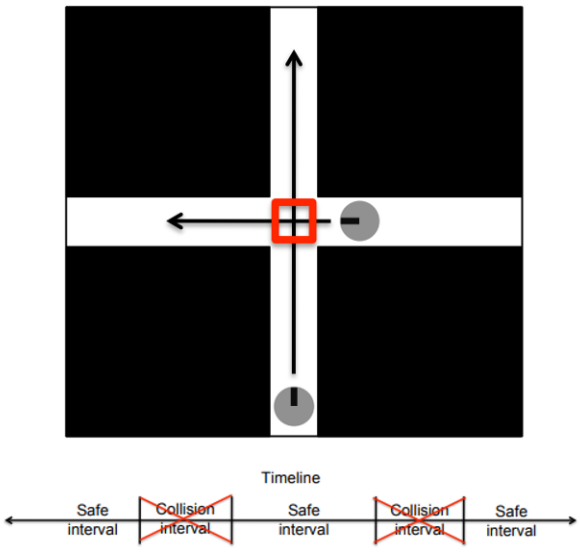


Figure 5.7: Environment containing two agents, where a configuration is highlighted. For the configuration a timeline is presented. Figure taken from Phillips and Likhachev [75]

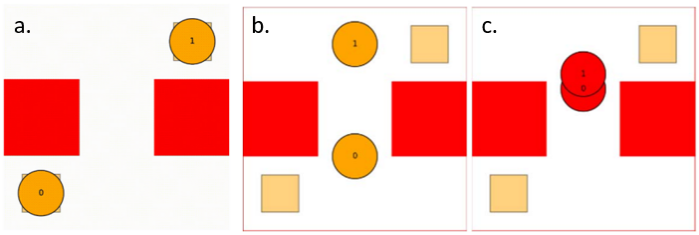


Figure 5.8: Unsuccesful demonstration of the SIPP algorith. Left figure (a.) shows the initial environment, middle figure (b.) shows agent 1 waiting for agent 0, right figure shows the inevitable collision. Figure taken from Wang et al. [114]

## 5.5. Rule-based MAPF Algorithms

Rule-based MAPF algorithms are a class of sub-optimal algorithms and use rules for specific agent-movements within a variety of scenarios. Compared to search-based algorithms, they do not include massive search. This leads to finding solutions in a relatively fast manner, but comes with the price of having solutions that are far from optimal. Another downside of rule-based algorithms is that they require special properties of the underlying graph in order to work. This can lead to being an inapplicable MAPF algorithm for many problems. The required properties are solver dependent and will be discussed below. However, due to both limitations, having solutions being far from optimal and being limited to specific graph types, these algorithms will only be touched upon shortly, as they are not deemed suitable to the modelling of UAM services. Recently developed algorithms are tree-based agent swapping strategy (TASS) [50], Push-and-Swap [62], and BIBOX [99].

TASS [50] is a complete algorithm for tree graphs only. A tree graph is a graph that is connected and acyclic, indicating that there are no cycles in the graph. This means that between two vertices there is a unique path between two vertices. Taking the urban environment into account, a tree-graph does not represent this environment and therefore this algorithm is not suitable.

Next, Push-and-Swap [62] is an algorithm that moves to its goal until it is unable to move further and uses unused vertices to reverse the location of two agents. The algorithm only works in graphs with $n$ vertices and maximum $n$-2 agents. Enhancements are Parallel-Push-and-Swap [83] and Push-and-Rotate [19]. The benefit of these algorithms is that it able to solve MAPF problems with many agents (100+) in a very fast manner, however generating solutions far from optimal.

Lastly, BIBOX [99] and its enhancement diBOX [10] are complete MAPF algorithms for bi-connected graphs only. This algorithm is very efficient on bi-connected graphs, but it cannot be used for traditional grid environments. For a traditional grid each vertex is connected to 4 other vertices. Similar to Push-and-Swap, the algorithm only works in graphs with $n$ vertices and maximum $n$-2 agents.

## 5.6. Bounded Sub-Optimal Search-based MAPF Algorithms

Even though optimal search-based optimal algorithms improved their computational performance by various extensions described in section 5.4, on large maps when many agents are involved, it can become infeasible for these algorithms to solve optimally in reasonable time. Bounded sub-optimal search based algorithms are algorithms that accepts a parameter $\epsilon > 0$ and returns a solution with a cost that is at most $1 + \epsilon$ of the optimal solution. $1 + \epsilon$ can also be denoted as bound $w$ in literature. Increasing the value of $\epsilon$ could possibly decrease the computational time to find a solution, while still allowing to find an approximate optimal solution. This section will use the same sub-classes as described in section 5.4 to discuss different algorithms. [96]

### 5.6.1. Extensions of A*

The A* algorithm and its extension are widely used in search-based MAPF algorithms, especially for the low-level search of CBS. Within literature many approximately optimal A*-based algorithms exist. This subsection will only describe the most promising algorithms, which are Weighted A* [76] and its variants, and Hierarchical Cooperative A* [92] and its variants.

#### Weighted A*

First, the most well-known A*-based algorithms that can be used both for individual as multi-agent planning is Weighted A* (WA*) [76]. It uses a best-first search that uses an evaluation function, $f(n) = g(n) + (1+\epsilon)h(n)$, to choose which node will be expanded. Within literature many enhancements exist to WA* for single-agent path finding, such as e.g. Optimistic Search [101], Explicit Estimation Search (EES) [102], and Dynamic Potential Search (DNS) [34].

The principle from WA*, to inflate the heuristic by adding a weight, can also be used for other variants of A*, such as M*. This is called inflated M* [110]. Inflating the heuristic for M* has two main benefits. The first one being that the inflated heuristic biases the search towards the ends of search tree, which are close to the goal. This is beneficial as the solutions are more likely to be found faster. Another benefit is that these nodes within the search tree will have smaller conflict sets in general, which is one of the downsides of optimal M*, as the re-expansion of vertices is exponential in the size of the conflict set [110].

#### Hierarchical Cooperative A*

Besides variations of the WA* algorithm, another prominent example if search-based sub-optimal algorithms is Hierarchical Cooperative A* (HCA*) [92]. HCA* is a distributed planner, where each agents path is planned one after each other, while having a certain hierarchy. After the path of an agent is determined, the path itself is denoted within a reservation table, that is shared with other agents. The entries within the reservation table are not allowed to be used by to-be-planned agents' paths. As only a small part of the reservation table will be accessed, an efficient implementation would be to store the grid locations and time steps using a hash table. Similar to A*, different heuristics can be used for the search, such as Euclidean or Manhattan distance.

One of the main limitations to HCA*, is that the prioritisation of agents highly influences the paths of other agents, and therefore could lead to far from optimal results. Windowed-HCA* (WHCA*) [92] only applies the reservation table, within a limited time window. This means that for each agent the cooperative search from its current position to the goal node is performed up to a certain fixed depth, resulting in a partial route. The abstract search is performed to full depth to ensure that agents are moving towards their goal. The agents reservation are therefore only considered during the planning window and are ignored in the rest of the search. During the movement of agents, the window will shift forwards, once all agents have reached a certain set point. After moving the window, the cooperative search will occur again. The order of prioritisation of agents can be altered per window shift, reducing the bias present in HCA*.Another benefit of WHCA* is that due to taking into account the reservation table for only the window duration, the number of possible conflicts between agents is reduced. WHCA* is an online MAPF solver, while even though optimality and completeness is sacrificed, it allows for real time path planning. This makes it a strong algorithm to be used for real time drone planning.

A promising extension of WHCA* is Conflict-Oriented WHCA*(CO-WHCA*) [9]. For WHCA*, agents could potentially have no conflicts during the first time window $W$ and move towards their goal position. However, if a conflict occurs just after the first window, at time step $W + 1$, it can become difficult for agents to resolve their conflict, especially if kinematic constraints are applied, which could result in a deadlock situation. If a deadlock situation occurs agents are not able to move and reach their goal location, making the solution incomplete. Therefore, CO-WHCA* places the window position around conflicts that were found in the abstract search, increasing the success-rate of the solver. Improving the prioritisation scheme will also lead to higher success-rates.

### 5.6.2. The Increasing Cost Tree Search

Regarding ICTS, there are no bounded sub-optimal ICTS algorthims for classical MAPF problems [96]. Due to the fact that the high-level search does not involve a heuristic, inflated A* variants cannot be used. However, within the current literature there does exist an approximately optimal variants of ICTS, in which agents are moving on a weighted graph, meaning that the edges have a non-unit cost. This allows for agents to be able to arrive at a vertex at different times compared to other agents [111]. A benefit of having weighted graphs is to allow for variable speed of agents. Including non-unit cost within ICTS, is denoted as extended ICTS (eICTS). In eICTS each nodes has a lower and upper bound within the ICTS. The high-level search is a best-first search on the lower bound. The low-level search looks for an optimal solution within these bounds. Looking for an

optimal version within the low-level search can be computational intensive and therefore sub-optimallity can be added in the low-level search. Adding also sub-optimallity to eICTS results in an approximately optimal version of eICTS called wICTS. [96, 111]

### 5.6.3. Conflict Based Search

The last optimal search-based MAPF solver that will be discussed a variations of CBS. Optimal CBS and its variants are used often in real-world scenarios, but do have the limitation that it is exponential in the number of conflicts and therefore limits the scalability of CBS. Multiple enhancements exist to CBS, were suboptimallity is included. This subsection will explore variants, where only the low-level search is altered, and where both the high and low-level search are altered.

#### Sub-optimal CBS low-level search

Improving the computational speed of CBS in the low-level search can be easily done by changing the existing optimal shortest path algorithm, such as A*, to a suboptimal variant of A*, such as WA* [76], inflated M* [110], EES [102], or DNS [34].

#### Greedy CBS

A variant of CBS, where suboptimallity is introduced in both the high and low-level search is Greedy-CBS (GCBS) [7]. In classical CBS both the high and low-level uses best-first search in order to find an optimal path for each agent in the low-level search and the lowest cost CT-node in the high-level search. GCBS uses the same framework, but allows for more flexible search. This results in finding valid solutions faster, but could result in suboptimal solutions. The main idea of the relaxation of the high-level search for GCBS is to prioritise CT nodes that seem closer to the goal, similar to inflated M* [110]. The reason is that these nodes will be likely to have less conflicts. GCBS uses conflict heuristics to find CT nodes that have less conflicts and therefore are more likely to lead to a goal node. Different conflict heuristics are e.g. Number of conflicts (counting conflicts per CT node), Number of conflicting agents (counting number of agents that have at least one conflict), and Number of Pairs (counting number of pair of agents that have at least one conflict). The relaxation of the low-level search is not done by implementing a variant of WA*, as these algorithms return long paths with many new conflicts, thus increasing the size of the CT. However, it is done by modifying the low-level search for agent $a_i$ to a best-first search, that then prioritises a path with minimum number of conflicts with paths of other agents. This means that if the oath of agent $a_1$ is planned, and the path of agent $a_1$ will be planned using the low-level search, the search would prefer a path that has no conflict with $a_2$, even if this sacrifice optimality. This indicates that GCBS is therefore not optimal, but is however complete if an upper bound $B$ exists on the cost of a valid solution and the algorithm will prune nodes in the CT, that have a higher cost than $B$.

#### Bounded Sub-optimal CBS

Two other promising sub-optimal variants are Bounded CBS (BCBS) [7] and Enhanced CBS (ECBS) [7]. Both algorithms use a focal search frame work for the high level search. Focal search maintains two two list of nodes, one being the classical OPEN list and one being a new FOCAL list. The FOCAL list contains a subset of nodes from OPEN, which nodes may lead to a solution that is approximately optimal. Another heuristic can be used to determine which nodes within FOCAL needs to be extended. This heuristic can be inadmissable (overestimating the cost of the path) and domain-dependent. Thus for focal search two arbitrary functions are used, denoted as $f_1$ and $f_2$. $f_1$ determines which nodes are in FOCAL. $f_2$ is a to-be-chosen heuristic determining the nodes to be extended in FOCAL. BCBS applies for the high level search the following focal-search. The cost of the CT node is determined by $g(n)$, and for the second heuristic the conflict heuristic from GCBS is used, denoted as $h_c(n)$. For the low-level search for the first function of the focal search, the regular $f(n)$ is used from the A* algorithm, which is $f(n) = g(n) + h(n)$. For the second function, the conflict heuristic is used. The bounds given for the high- and low level search for BCBS are denoted as BCBS($w_h, w_l$). If one of the bounds is equal to 1, instead of a focal search, an optimal search is used for either the high or low level search. BCBS($\infty$,$\infty$) is the same as GCBS as all nodes from OPEN will be used in FOCAL. Barer et al [7] proof that the solution is guaranteed to be less than or equal to $w \cdot C*$, where $w = w_h \cdot w_l$. The distribution of $w$ between $w_h$ and $w_l$ is not trivial. For BCBS($w_h, w_l$) $w_h$ and $w_l$ are fixed, which can be inefficient. To improve this, an enhancement to BCBS is proposed, called Enhanced CBS (ECBS) [7]. ECBS provides more flexibility within the focal search. ECBS makes use of the same low-level search as BCBS(1, $w$). The low-level search returns two values to high-level search, which is similar to BCBS the cost of the node and the lower

bound of the optimal solution of the entire problem. This allows for the algorithm to have more flexibility in the high level when the low-level returns node cost close to the lower bound. In conclusion, both BCBS and ECBS never expand nodes with cost higher than $w$ times the optimal solution, allowing for an increase in computational performance compared to CBS, with the maximum cost of w times the optimal solution

#### CBS with Priorities & Priority-based Search

Lastly, two variants of CBS which use prioritisation of agents to increase the computational performance are discussed. First, CBS with Priorities (CBSwP) [65] performs similar to CBS a best-first search for the high-level search. Within each CT node a priority ordering is stored and only child nodes are generated where the priority of the child node extends the priority of the parent node. Conflicts are resolved in order using preferring cardinal conflicts [11] to select which conflicts need to be resolved first.

Next, Priority-Based Search (PBS) [65] is a two-level algorithm, which can be used for prioritised planning. A priority tree (PT) is constructed within by performing a depth-first search on the high-level to dynamically construct a priority ordering. PBS chooses which agent should be given a higher prioritisation by using a greedy algorithm. A partial priority order is constructed until no collisions are found. This is achieved by efficiently backtracking and exploring other branches when there is no solution in the current branch. If two agents collide, an order pair is constructed by splitting a PT node, similar to CBSwP for CT nodes. No constraints are stored in PT nodes, which is different compared to CBSwP. If agent $a_i$ has a higher priority than agent $a_j$, no collisions exist between the two agents if PBS processes PT Node $N$. For PBS, a possibility exist to add an initial priority order, which the solution must respect. By default this list is empty. Different priority orderings can be used for PBS. Interesting examples are FIX, LH, SH, and RND. FIX is a PBS variant that is similar to CA* [92], using the order of the agents in the MAPF instance. LH has a fixed priority order, for which agents with the longest paths from start to goal vertex have the highest priority. SH uses an opposite priority ordering, thus the agent with the shortest path has the highest priority. Lastly, RND is different compared to the aforementioned priority orderings, as it initially runs PBS 10 times using a randomised priority ordering. After 10 times, the solution with the smallest cost is selected.

Experiments performed by Ma et al [65], demonstrate that PBS and its variants have a higher success rate with increasing number of agents, within the selected time limit, compared to classical CBS and CBSwP, due to the fastly increasing size of the CT for CBS and CBSwP. The benefit of CBSwP is increased performance compared to CBS, while finding almost optimal solutions. Standard PBS outperforms all PBS variants when considering optimallity, as it is able to obtain solutions only 4% worse than optimal. Further experiments performed for PBS using the DAO map, brc202d (few open spaces, many bottlenecks) demonstrate that PBS is also easily scalable with up to 600 agents. However, it must be noted that this map does not represent urban area topology, which often contains large open areas. Experiments performed using the DAO lak503d map (large open spaces, few bottlnecks) show up to 6 time faster runtimes compared to CBS and almost 5 times faster runtimes compared to CBSwP with 100 agents.

## 5.7. Beyond Classical MAPF

The previous sections only focused on classical MAPF algorithms, which all have three main assumptions in common [96], which are as follows:

- **Assumption 1:** Every action takes exactly one time step
- **Assumption 2:** Time is not continuous, but discretised into time steps.
- **Assumption 3:** Each agent only occuppies 1 vertex at every time step

These assumptions do not represent real-world scenarios, and therefore more sophisticated MAPF algorithms have been developed. All beyond classical MAPF algorithms focus on relaxing one or multiple of these assumptions. This section will only focus on a limited number of methods, which are deemed most useful for the UAV medical supply delivery problem. The methods that will be discussed are MAPF with large agents, MAPF with Kinematic Constraints and MAPF using Reinforcement Learning. The use of reinforcement learning for MAPF is an emerging field and considered to be state-of-the-art. The methods and techniques discussed in this section will not be considered for the trade-off in section 5.9. This is because the discussed techniques can often be applied to multiple MAPF algorithms and are therefore seen as possible extensions to the to-be selected algorithm.

### 5.7.1. MAPF with Large Agents

For classical MAPF methods, the geometry of agents is ignored, resulting in agents being represented by a point, occupying only one vertex in the graph $G$. In reality, an agent will occupy more than one vertex at every time step, excluding any virtual boundaries involved for an agent. Li et al [57] propose Multi-Agent Path Finding for Large Agents (LA-MAPF). LA-MAPF is harder to solve than MAPF, as agents are more likely to have conflicts. The geometrical shape of an agent is defined by $S_i$. An example of a geometrical shape of an agent at vertex $v = (3,3)$ with an $1.5 \times 1.5$ square shape, can be described as follows: $S_1(v) = \{(x^{(1)}, x^{(2)}) | 0 \leq x^{(j)} - v^{(j)} \leq 1.5, j = 1,2\}$. The configuration space of agent $a_i$, is the sub-graph $G_i = (V_i, E_i)$. The sub-graph is part of the graph in which the agent is able to move. The geometric shape of each agent is assumed to be fixed and centred around a reference point. It is assumed that the geometrical shape cannot undergo transformations, such as rotations. The definitions of conflicts is also generalised. A vertex conflict is represented by a five-element tuple $(a_i, a_j, u, v, t)$ and an edge conflict is represented by a seven-element tuple, $(a_i, a_j, u_1, u_2, v_1, v_2, t)$.

Li et al [57] adapt CBS for LA-MAPF, by changing the conflict detection function initially. However, if normal splitting of nodes is used, many intermediate CT Nodes are generated, which is inefficient. This can be resolved by adding multiple constraints in a single CT node expansion, resulting in a new algorithm called Multi-Constraint CBS (MC-CBS). The constraints added resolve multiple related conflicts in a single CT node expansion, resulting in a smaller cost tree. A vertex or edge constraint for agents $a_i$ and $a_j$ are mutually disjunctive iff any pair of conflict free paths satisfies at least one of the two imposed constraints. This means that there does not exist a conflict free path, in which both imposed constraints are violated. A set of two constraints is mutually disjunctive iff each constraint in one set is mutually disjunctive with every constraint in the other set, which helps with constraint propagation. Two child nodes are generated when a conflict is resolved, including the existing constraint set as well as two additional constraint sets, denoted as $C_1$ and $C_2$, which include core constraints to resolve the conflict (similar to classical CBS) and enhanced constraints that ensure that both constraint sets are mutually disjunctive. Different strategies can be employed to choose the constraint sets. Examples of constraint set building methods are the asymmetric approach, symmetric approach, and the maximisation of costs of child nodes. The asymmetric approach only adds a constraint to the left child node of size one $\{(a_i, u, t)\}$ and a large constraint set $\{(a_j, v', t | a_i, a_j, u, v', t)\}$ to right child node. The symmetric approach adds the same constraint set to both agents by choosing a point $p$ inside the overlap area. All vertices the shape of the agent occupies are included in the constraint set, including point $p$. The maximisation of cost of the child nodes uses the insight from ICBS using preferring cardinal conflicts, as described in subsection 5.4.3. For MC-CBS, a search for the pair of constraints with highest possible cost is conducted. Li et al [57] demonstrate that MC-CBS outperforms classical CBS up to three times for different map dimensions and topologies.

### 5.7.2. MAPF with Kinematic Constraints

Besides, the possibility that agents can occupy more than vertex at every time step, another extension to MAPF is to incorporate the kinematic constraints of agents during path finding. UAVs are highly maneuverable, but are not able to stop immediately in real-world scenarios. Therefore, extending MAPF to include kinematic constraints allows us for modelling of real-world constraints.

Hönig et al. [44] introduce MAPF-POST in order to allow for the modelling of kinematic constraints. For classical MAPF, the move actions of agents are only depended on their current location. However, for MAPF-POST, the speed and orientation of the agent are taken into account using state parameters. To incorporate speed and direction, the underlying graph becomes directed, which enforces agents to transverse certain edges in one direction only. First, a collision-free MAPF plan is constructed for all agents, similar to classical MAPF. Afterwards, the MAPF plan is converted to a data structure called the Temporal Plan Graph (TPG). TPG is a directed acyclic graph, $G = (V, E)$, where each vertex $v \in V$ represents an agent entering a location, which is called an event. Each edge $(v, v')$ represents a temporal precedence between events $v$ and $v'$, where event $v$ needs to be scheduled before event $v'$. By use of two types of temporal precedences imposed by the TPG, agents will not collide if agents execute a plan-execution schedule. The first type precedence is that each agent will enter locations in the order given by the MAPF plan. The second type is that for each pair of agents that enter the same location, the order in which the agents enter the location is enforced from the MAPF plan. The TPG does not discretise time and specifies a partial order among all events. This allows for the incorporation of maximum velocity and imperfect plan execution of agents. After constructing the TPG, the next step

is augmenting the graph. This is done by adding additional vertices to provide guaranteed safety distances between all agents. These additional vertices are called safety markers. The following step is to convert the TPG into a Simple Temporal Network (STN). STN is a directed acyclic graph $G' = (V', E')$, where each vertex represents an event and each edge is a temporal constraint between two events. The constraints have a lower and upper bound, to ensure that event $v$ is scheduled no later than event $v'$. Using STN allows for the modelling of both minimum and maximum speeds for agents. The final step is to calculate the plan-execution schedule, which can be done by using graph-based optimisation or linear programming. Using MAPF-POST allows for the integration of kinematic constraints and can be used for multiple MAPF algorithms.

### 5.7.3. Reinforcement Learning for MAPF

The last beyond MAPF algorithm category that will be discussed is the use of reinforcement learning for MAPF. With Artificial Intelligence (AI), and reinforcement learning (RL) as a sub-field, being a hot topic these last years, the use of RL can offer benefits to MAPF algorithms. RL uses rewards and/or penalties as feedback mechanism to train agents to interpret its environment and make decisions afterwards. Making use of RL allows agents to potentially plan more efficient routes, especially in environments were many agents are involved and scalability can only be achieved by using sub-optimal algorithms. An overview of recent work on multi agent reinforcement learning for path planning is presented in Table 5.1. It can be noted from recent work that various reinforcement learning methods are used, mainly focusing on small environments. For 2D environments, the number of agents can be up to 1024 [17, 84], but for 3D environment the maximum number of agents is 16 [86]. 16 UAV agents could be sufficient for modelling the UAV Medical Supply Delivery, however Semnani et al. [86] only model this on a very small grid (8 X 8 X 4), which is not representable for the problem at hand. To our knowledge, no prior work exist that uses reinforcement learning for MAPF in large 3D environments. Therefore, the use of reinforcement learning for the UAV medical supply delivery problem is currently outside of the scope of this thesis, due the fact that the current methods are not mature enough to allow to be used in large open environments incorporating large scale UAV operations.

Table 5.1: Overview of papers using reinforcement learning for MAPF

| Authors | Year | Modelling Method | Map | Max. # Agents |
|---|---|---|---|---|
| Wang et al. [113] | 2021 | Deep Reinforcement Learning (Decentralised) | 2D Grid 100 X 100 | 4 UAV Agents |
| Damani et al. [17] | 2021 | Reinforcement and imitation learning (Decentralised) | 2D Grid 160 X 160 | 1024 Agents |
| Semnani et al. [86] | 2020 | Reinforcement learning and Force-based motion planning (Decentralised) | 3D Grid 8 X 8 X 4 | 16 agents |
| Qie et al. [79] | 2019 | Reinforcement learning Target assignment and path planning | 2D | 5 UAV Agents |
| Li et al. [59] | 2019 | Q-learning (Reinforcement learning) (Decentralised) | 3D | 4 UAV Agents |
| Sartoretti et al. [84] | 2019 | Reinforcement and imitation learning (Decentralised) | 2D Grid 160 X 160 | 1024 Agents |

## 5.8. MAPF Algorithms used for UAV path planning

After elaborating on both optimal and sub optimal classical MAPF algorithms, as well as looking at beyond classical MAPF algorithm extensions, this section will discuss papers using MAPF for UAV path planning and coordination. An overview of related work can be seen in Table 5.2. It must be noted that only limited papers are available that combine MAPF and UAV Operations, with F. Ho [40–42] contributed the most in terms of written papers. Therefore, the used algorithms are biased towards the choice of the author. This section will only discuss the most relevant segment of two papers. First, in which is the conflict resolution via re-planning, take-off scheduling and speed adjustment by Ho et al. [42], and the modelling of large agents and spatio-temporal pruning by Ho et al. [40].

Table 5.2: Overview of papers using MAPF algorithms for the planning and coordination of UAVs

| Authors | Year | Modelling Method | Map Dimensions [km] | Key Observations |
|---|---|---|---|---|
| Liu et al [60] | 2023 | Multi-CBS (Centralised) | 500 x 500 | Sparse A* used for low level CBS |
| Choudhury et al [15] | 2021 | ECBS (Centralised) | 12 x 12<br>20 x 20 | Task allocation via VRP |
| Ho et al. [42] | 2021 | ECBS (Centralised) | 12.8 x12.8 | Conflict Resolution via replanning, takeoff schedulling, speed adjustment |
| Ho et al. [41] | 2020 | ECBS (Decentralised) | 12.8 x 12.8 | Conflict Resolution via prioritisation and negotiation |
| Ho et al. [40] | 2019 | CBS, ECBS, CA* (Centralised) | 12.8 x 12.8 | Modelling large agents using computational geometry method<br>Spatio-Temporal Pruning |

### 5.8.1. Conflict Resolution

First, Ho et al [42] have developed a strategic pre-flight conflict detection and resolution (CDR) using a MAPF Model. The selected MAPF algorithm is ECBS. ECBS itself is briefly explained in subsection 5.6.3. Besides the standard re-planning MAPF algorithm use to resolve conflicts, take-off scheduling and speed adjustment are introduced as additional methods to resolve conflicts. The introduction of both methods allows to relax two assumptions made in standard MAPF problems. First, the start time can be delayed, instead of having a fixed start time of an UAV, making the start time variable. Next, the incorporation of speed adjustments, relaxes the assumption that an agent has a constant speed during the flight path. The agent will be able to fly at maximum speed, with the option of decreasing its speed on given flight path segments. Re-planning, take-off scheduling and speed adjustment can be used to solve different collision types, but all have its drawbacks. An overview of this is given in Table 5.3. The aforementioned CDR methods are tested on a 12.8 km by 12.8 km map representing Tokyo with the number of UAV operations ranging from 500 to 3000 within 1 hour. The simulation demonstrate that re-planning and take-off scheduling offer only minor differences compared to using all three CDR methods, concerning the average delay per operation. It must be noted that this algorithm can only be used in an offline setting, as all operations are delivered in one batch. For the UAV medical supply delivery problem, CDR in an online setting would allow to also model emergency deliveries. This one of the main use cases of medical supply delivery and therefore has great importance.

Table 5.3: Conflict Resolution methods proposed by Ho et al [42], the conflict types they can resolve and their drawbacks.

| Conflict Resolution | Conflict Type Ability | Drawbacks |
|---|---|---|
| *Re-planning* | *Head on*<br>*Cross/Pursuit* | *Additional battery consumption required*<br>*Spatial modifying flight plan*<br>*Generate infeasible operations, especially for start/goal collisions* |
| *Take-Off Scheduling* | *Head on*<br>*Cross/Pursuit*<br>*Close to start/goal* | *More delays for head on conflicts due to late start time*<br>*High total task completion time* |
| *Speed Adjustment* | *Cross/Pursuit* | *Generate infeasible operations, especially for start/goal collisions* |

### 5.8.2. Modeling Large UAV Agents

Next, Ho et al [40] present an extension of CBS and ECBS that allows the modelling of large UAV agents with diverse geometry, allowing agents to occupy more than one voxel (volumetric pixel, representing a 3D volume). The straightforward approach would be to consider all the voxels an agent would occupy at each time step and determine if two agents have at least one voxel in common between time step $t$ and time step $t+1$. For large maps with many agents involved this method becomes computational expensive, as at every time step all intersected voxels need to be determined as well as verifying if none of the intersected voxels is occupied twice. This also leads to agents occupying more space than actual needed. This can be seen in the left figure (a) in Figure 5.9. A more efficient method is the so called computational geometry method that makes use of geometrical considerations to detect conflicts. This method makes use of conflict intervals, similar to SIPP, as discussed in subsection 5.4.4. The time to collision $t_C$ within the time interval $[t_a, t_b]$ is computed to detect a conflict between agents whose paths segments overlap and can be obtained by solving Equation 5.1 [40], with $v_i$ and $v_j$ the velocity of the agents and $p_i$ and $p_j$ the positions of the agents at the given time step.

$$\begin{aligned} \left\| p_i(t_C) - p_j(t_C) \right\|^2 &= \left(r_i + r_j\right)^2 \\ \text{i.e.} \quad \left(p_i(t_a) + v_i \cdot t_C - \left(p_j(t_a) + v_j \cdot t_C\right)\right)^2 &= \left(r_i + r_j\right)^2 \end{aligned} \tag{5.1}$$

If the roots of Equation 5.1 [40] are real and positive a collision has occurred, where the time of the collision is $t_C$ =Min($t_{\text{root1}}$, $t_{\text{root2}}$). The conflict interval is given in Equation 5.2 [40]. A visual representation of the computational geometry method is shown in the right figure (b) of Figure 5.9.

$$I_C = [t_C; \text{Min}(t_b; \text{Max}(t_{\text{root}\,1}; t_{\text{root}\,2}))] \tag{5.2}$$

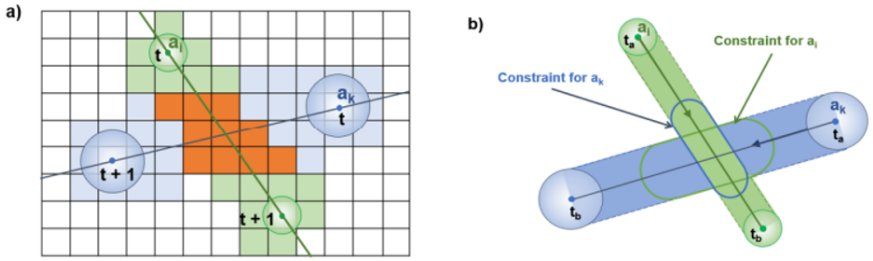


Figure 5.9: Left figure (a) displays conflict interval for the voxels intersection method. Right figure (b) displays conflict interval for the computational geometry method. Figures taken from Ho et al. [40]

### 5.8.3. Spatio-Temporal Pruning

Besides the computational geometry method, Ho et al [40] propose another method to reduce the computational effort for conflict detection and resolution. This method is called Spatio-Temporal Pruning. The search space contains $\frac{N(N-1)}{2}$ states in order to detect any conflict between all pairs of operations. Spatio-Temporal pruning reduces the search space, as two operations might have no time or spatial intersection. For each operation $O_i$ a subset potential conflicts is determined, $O_i^{\text{conflict}}$. This is done by considering spatial and temporal information of the given operation, for which the pseudocode is given in Algorithm 4. The aforementioned conflict detection for large agents is afterwards performed on the reduced set of operations.

**Algorithm 4** Spatio-Temporal Pruning algorithm, which defines the subsets of agents in potential conflicts, taken from Ho et al [40]

**Data:** $O$ set of operations, $O_i$ with associated $T_i$ time interval and agent size $r_i$
**Result:** $\forall O_i \in O, O_i^{\text{Conflict}} \subseteq O$
**for** $O_i \in O$ **do**
  *# Determine if temporal overlap between $O_i$ and $O_j$*
  **if** $T_i \cap T_j \neq \emptyset$ **then**
    *# Determine if spatial overlap during common time interval*
    **if** *ShortestDistance*$(O_i, O_j, T_i \cap T_j) \leq r_i + r_j$ **then**
      Determine if the potential conflict is on the outbound and/or return path of each operation;
      Add $O_i$ in $O_j^{\text{conflict}}$ and $O_j$ in $O_i^{\text{conflict}}$

## 5.9. Comparison of MAPF Algorithms

This section will give an overview of the state-of-the-art classical MAPF Algorithms based on the solver classes defined in section 5.2. The algorithms have been extensively discussed in section 5.3-section 5.6. In order to determine the best suited MAPF algorithm for the UAV Medical Delivery Problem, a trade-off is performed. This trade-off is performed on the following criteria, which are ranked in order of importance. This includes a weighted score as well, to demonstrate its importance. If applicable, a reference is made to the requirements of the proposed system for medical UAV delivery, presented in section 4.3.

- **Performance Urban Area:** This criteria relates to how well the algorithm performs, focusing on success-rate, for map topologies representing urban areas (*REQ-ENV-01*). This will mainly consist of large open

areas, few bottlenecks, with possibility for large number of involved agents (*REQ-PC-04*). The success-rate is the percentage of solutions found within a 5 minute time frame, as e.g. shown on the lefthand side of Figure 5.6.

- **Suitable for online use:** This criteria considers, if the algorithm is suited for Online Use, which is needed for the real-time re-planning of UAVs due to path deviations, delays, and new imposed constraints (*REQ-PC-02, REQ-PC-03*).
- **Suitable for Prioritisation:** To allow for mission prioritisation of emergency missions (*REC-PC-03*), the algorithm should be suited to incorporate prioritisation, which is judged by this criteria.
- **Completeness:** Finding a solution for any configuration of agents is a desired property (*REQ-PC-01*), which is assessed by this metric. It only takes into account completeness for grids representing an urban environment (*REQ-ENV-01*).
- **Computational Efficiency:** This measure assesses the required run time needed in order to return a collision-free solution, needed to allow for real-time performance (*REC-PC-02*)
- **Complexity of Implementation:** This is a measure to assess the complexity involved with implementing the algorithm
- **Scalability:** This measure considers how well the algorithm can maintain computational efficiency when the number of agents is increased (*REQ-PC-04*) or environment size is enlarged (*REQ-ENV-02, REQ-ENV-03*).
- **Optimality:** The optimality guarantees can be divided into 3 main categories, which are no optimal solutions guaranteed, (bounded) sub-optimal solutions guaranteed, and optimal solutions guaranteed. For (bounded) sub-optimal algorithms a distinction can be made, between non-adjustable and adjustable bounds. Optimal solutions will result in shorter delivery time and more efficient routes.
- **Maturity:** This criteria evaluates the level of maturity involved for a MAPF algorithm, demonstrating how well-developed and tested the algorithm is.

In order to reduce the overall size of the trade-off, a selection state-of-the-art MAPF algorithms has been made from different solvers classes, described in section 5.3-section 5.6. It must be noted optimal-reduction based MAPF algorithms and rule-based MAPF algorithms have not been included within the trade-off. Rule-based MAPF algorithms are not selected due to the fact that adapting the objective function to sum-of-cost is not trivial, and the algorithms are only suitable for small, dense graphs. This is the opposite topology of a graph representing an urban environment. Next, discussed rule-based MAPF algorithms require special properties for the underlying graph in order to work, making it inapplicable for modelling the UAV medical delivery problem. For example, TASS [50] is only complete for tree graphs and BIBOX [99] requires bi-connected graphs. An overview of the selected MAPF algorithms included key characteristics are presented in Table 5.4.

For the purpose of comparing and selecting the most suitable MAPF algorithm for the problem at hand, a weighted trade-off is performed, which can be seen in Table 5.5. The performance in urban area, suitability for online use and suitability for prioritisation are deemed most important for the delivery of medical supplies in urban environment. This is due to the fact that online use and prioritisation allow for planning of emergency missions. These criteria have therefore a weighted score of 3. Completeness, computational efficiency, complexity of implementation and scalability are afterwards seen as important criteria and have received a score of 2. Optimality of the obtained solution is deemed less important, as optimal solutions are often time-consuming, which can lead to infeasible methods for real-world scenarios. Also the maturity of the algorithm is deemed less important. Mature algorithm offer the benefit of having multiple available extensions, but could be less suitable for the problem at hand. A score of either 1, 2 or 3 is assigned to each criteria for all algorithms.

It can be seen in Table 5.5 that CO-WHCA* is deemed the most suitable MAPF algorithm for the delivery of medical supplies in urban environment. The key strengths of this algorithm is that it can be used for online planning and replanning of operations, allows for mission prioritisation, is computationally efficient, and is easily scalable. This comes with the cost of providing no guarantees regarding the optimality and completeness of the solution. By strategically placing the planning windows around conflicts, deadlock situations can be minimised. Also by adjusting the prioritisation scheme, better solutions can be obtained. CO-WHCA* makes use of a two level search in order to plan the paths of agents. First, a centralised planning method is used to find possible conflicts and provide the initial path to the UAV agent. Afterwards, re-planning is possible if necessary. PBS is deemed the second most suitable MAPF algorithm. The algorithm provides no guarantees regarding its completeness, although does provide near optimal solutions. The algorithm is less

suited for online use and is deemed more difficult to implement compared to CO-WHCA*. A possible extension of CO-WHCA* could be to make use of PBS for the first level conflict detection and resolution. This could possibly decrease the cost of the solution by planning more optimal paths in the initial phase, while still allowing for prioritisation of agents. However, the combination of both algorithms does increase the complexity of implementation. It must be noted that also different trade-off methods, such as using equal weights for all criteria and/or using assigned scores based on a 1-3-9 scale, do not affect the final trade-off results of CO-WHCA* and PBS.

Table 5.4: Overview of selected state-of-the-art MAPF algorithms

| **MAPF Algorithm** | **Optimallity** | **Characteristics** |
|---|---|---|
| A* + OD + ID | Optimal | *Well studied algorithm best suited for open map topologies with small number of agents.*<br>*Relatively easy to implement algorithm, used for several low-level search algorithms.*<br>*Search space grows exponential with number of agents.* |
| ODrM* | Optimal | *Increased computational performance compared to A* + OD + ID, but more complexity involved for implementation.*<br>*Algorithm is not as extensively studied and implement for low-level search as A* (+OD + ID).* |
| ICTS | Optimal | *Efficient algorithm for finding solutions for maps with large open areas, by quickly finding open spaces for agent movements.*<br>*Algorithm only offers extension using pruning techniques, less mature than A* extensions and CBS.* |
| ICBS | Optimal | *Well-studied algorithm for finding solutions for maps with few open areas and many bottlenecks.*<br>*Cost tree grows exponential with number of agents for classical CBS.*<br>*CBS extensions allow for improved performance in open environments, with the cost of adding complexity for implementation.* |
| SIPP | Optimal | *Algorithm suited for dynamic environments, allowing for multiple possible trajectories of dynamic obstacles.*<br>*Algorithm suited for actively waiting but not actively avoiding obstacles.*<br>*Suitable for use in low-level search, implementation less trivial compared to A* extensions.* |
| (CO)-WHCA* | Suboptimal | *Well-studied and easy to implement decoupled algorithm.*<br>*Allows for prioritisation of agents and online use.*<br>*No optimallity and completeness guarantees.*<br>*Efficient algorithm for large open areas, with several options to avoid deadlocks.* |
| ECBS | Bounded Suboptimal | *Increased computational performance compared to CBS, with adjustable optimallity bound. Allows for slightly increased cost.*<br>*Distribution of bound w over high and low-level search is not trivial, and difficult to predict how it affects its performance.* |
| PBS | Bounded Suboptimal | *Increased computational performance compared to CBS, with results near optimal.*<br>*Allows for prioritisation of agents, including user-defined order* |

Table 5.5: MAPF algorithm trade-off using weighted scores

| | **Weight** | **A* + OD + ID** | **ODrM*** | **ICTS** | **ICBS** | **SIPP** | **CO-WHCA*** | **ECBS** | **PBS** |
|---|---|---|---|---|---|---|---|---|---|
| *Performance Urban Area* | **3** | 2 | 2 | 3 | 2 | 1 | 2 | 2 | 2 |
| *Suitable for Online Use* | **3** | 1 | 1 | 1 | 1 | 1 | 3 | 1 | 2 |
| *Suitable for Prioritisation* | **3** | 2 | 2 | 1 | 1 | 1 | 3 | 1 | 3 |
| *Completeness* | **2** | 3 | 3 | 3 | 3 | 2 | 1 | 3 | 1 |
| *Computational Efficiency* | **2** | 1 | 2 | 1 | 1 | 3 | 3 | 3 | 3 |
| *Complexity of Implementation* | **2** | 3 | 2 | 2 | 2 | 2 | 3 | 1 | 2 |
| *Scalability* | **2** | 1 | 2 | 1 | 1 | 3 | 3 | 3 | 3 |
| *Optimality* | **1** | 3 | 3 | 3 | 3 | 3 | 1 | 2 | 2 |
| *Maturity* | **1** | 3 | 2 | 1 | 3 | 1 | 3 | 2 | 2 |
| **Total Score** | | **37** | **38** | **33** | **32** | **33** | **48** | **36** | **43** |

## 5.10. Simulation Software

Lastly, In order to simulate the path planning and coordination of UAVs in the urban environment, as described in the previous chapters, an agent-based modelling software tool needs to be selected. This tool allows us to simulate the complex system by representing the individual agents as well as their interactions. From this, emergent properties and patters can be analysed, giving important insights. The benefit of using ABM, is that allows us to model complex and non-linear systems by defining agent characteristics and behaviours. An agent based model comprises of three elements, which are the agents, the environment, and the imposed rules. The agents will be able to act autonomously in response to their surrounding in order to achieve their internal goal. The environment will be modelled using a well-defined topology, using for instance a square or hexagonal grid. An urban environment can be modelled using Geographic Information System (GIS) data to replicate existing buildings. The behavioral rules can be modelled after the real world actors on which the agents are based. [5]

This section will give an overview of ABM tools that could be used for the UAV Medical Supply Delivery Problem. It is beneficial if the tool is both open-source and allows to program in Python in order to allow the thesis to be finished in time, without the need of learning a new programming language. An overview of four simulation tools is given in Table 5.6. The tools will be discussed in a more detail below.

Table 5.6: Overview of Simulation Tools adapted from Antelmi et al [5] applicable to the UAV Medical Delivery Problem

| Simulation Tool | Programming Language | Simulation Environment | Visualisation | Open Source | Documentation | Phase |
|---|---|---|---|---|---|---|
| **AgentPy** | Python | Grid, continuous, network, GIS | 2D | Yes | Good | Deployed |
| **Bluesky** | Python | Continuous | 2D | Yes | Good | Deployed |
| **SoSID** | Python | Grid, Continuous | 2D | No | Limited | Development |
| **MESA** | Python | Grid, continuous, network, GIS | 2D, 3D | Yes | Extensive | Deployed |
| **NetLogo** | Netlogo, Python | Grid, continuous, network, GIS | 2D, 3D | Yes | Extensive | Deployed |

First, Agentpy [32] is an open-source Python library that can be used for developing and analysing ABM using Jupyter Notebooks, making it an easy to use tool. It combines both having a user-friendly interface with writing code in Python. Even though AgentPy is well suited for modelling and analysis of complex systems, it only allows for the modelling of 2D environments, making it less suitable to model the problem at hand. [5]

Second, Bluesky [43] is different to Agentpy, as it is not a global ABM tool. The tool is developed to simulate, visualise, and analyse air traffic operations using a Python framework. Bluesky was initially developed for the simulation of conventional aviation, but has been extended to allow for the integration of UAM services. This includes drone-specific route following modes, geofencing, and geovectoring. UAM services have been modelled in Vienna for the Metropolis II project [68], demonstrating that could also be used for the Medical Delivery Problem in Rotterdam. The visualisation is in 2D, but BlueSky can be used for the modelling of 3D operations.

Third, SoSID [51] is a python modelling and simulation toolkit designed for the rapid development of case studies of Systems of Systems Inverse Design (SoSID) developed by DLR. The tool is built on-top of Mesa. Systems of Systems is defined as a set of systems that interact with each other as well as enable capabilities that are unique compared to individual components of systems. SoSID allows aircraft designers to quantify how top-level requirements will influence the operational environment. Currently two case studies have been verified using the system, which are wildfire suppression and collaborative design of UAM System of Systems. The tool itself is currently still in development, but could allow easy modelling of UAM operations, as an ABM simulation is specifically designed for this use case.

Fourth, Mesa [66] is an agent-based modelling tool, which has an open-source Python Library. It can be used for creating, visualising and analysing simulations and is one of the most used ABM libraries, due to its ease of use and its accessibility. Several extensions to Mesa libraries exist, such as Mesa-Geo [112] and Mesa-3D allowing for incorporating GIS data and 3D visualisation into the software framework. Different simulation

environments can be used, such as a square or hexagonal grid or a continuous map. It must be noted that for Mesa the standard space is 2D. A 3D grid structure can potentially be modelled in Mesa, by extending the current framework, however this is not trivial. [5]

Lastly, Netlogo [103] is considered the standard platform for developing ABM, and allows for integration with many programming languages, including Python. Netlogo can be used to model 3D visualisations as well as incorporating GIS data usage, also allowing to use visual programming language, making it very easy to use. However, the downside to this is that Netlogo does have significant limitations regarding its model complexity. [5]

A trade-off between the different simulation tool has been performed, which can be seen in Table 5.7. AgentPy has the benefit of being an easy to use ABM tool, as it can be used in Jupyter Notebooks. However, it does not allow for the modelling of 3D, making the tool infeasible to use. Next, SoSID is also deemed infeasible to use as the tool is currently still in development, and is not easy to use due to its lack of documentation. Another downside is that is not open-source. Lastly, Netlogo is also deemed not suitable, as it does not allow the modelling of complexity that is needed within this thesis. BlueSky and MESA are both deemed suitable tools to use for the modelling of the UAV Delivery problem. The tools have comparable trade-off scores, however integrating 3D operations in Mesa is less trivial compared to BlueSky. Furthermore, Bluesky is specifically designed for air traffic operations. The Metropolis II project [68] has demonstrated successful modelling of UAM operations in Vienna, showing proof of concept. Therefore, BlueSky has been selected as simulation tool.

Table 5.7: Trade-off table for the selection of Simulation tool

| **Simulation Tool** | **Maturity** | **Ease of Use** | **Allow Complexity** | **Allow 3D Modelling** |
|---|---|---|---|---|
| *AgentPy* | + + | + + | + - | - - |
| *BlueSky* | + + | + - | + + | + + |
| *SoSID* | - - | - - | + - | + + |
| *MESA* | + + | + - | + + | + - |
| *NetLogo* | + + | + + | - - | + + |

# 6

# Research Proposal

This chapter will elaborate on the research proposal of path planning for medical urban air mobility services. The proposal is based on the extensive background information provided in the previous chapters. First, the research gaps will be discussed in section 6.1. Next, the research objective will be presented in section 6.2. An overview of all research questions and their sub-questions are given in section 6.3.

## 6.1. Research Gaps

Urban Air Mobility (UAM) is a rapidly emerging industry, which focuses on the transportation of passengers and cargo within the urban environment using Unmanned Aerial Vehicles (UAVs). In response to the increasing demand of UAM services, methods are required for autonomous flight planning and coordination of multiple UAVs in order to ensure safe and efficient operations. One of the most promising UAM services, is the delivery of medical supplies. Delivery of medical supplies has the benefit of having a high social acceptance and offering benefit to society [22]. UAVs have only limited payload capacity, making it well suited to delivery of medical supplies, such as medicine, emergency (blood) supplies, and vaccines. Medical supply delivery via UAVs has already been in operations in rural environment since 2016 offered by Zipline [123]. Integrating a similar system within a 3D urban environment introduces new challenges and constraints, adding complexity to the path planning and coordination of UAVs. Placement of restricted and limited access areas, such as imposed no-flying zones, recreational areas, or urban obstacles heavenly influence the path planning of UAVs. Virtual boundaries can be created by using keep-out geofencing methods. To allow for medical emergency missions in urban environment, keep-out geofences need to be dynamically adjustable for selected, e.g. emergency, UAVs only. Combining this with the need of prioritisation and real-time re-planning of emergency missions an extra layer of complexity is added. Multi-agent Pathfinding is well suited to ensure safe operation, as it allows for path planning and coordination of multiple agents within a shared environment.

This thesis will add in the following manner to research. The modelling of UAV operations using multi-agent pathfinding, focusing on 3D and real-world urban environment, is relatively unexplored. This work will focus on translating CO-WHCA* from 2D to 3D, including possible novel extensions and adaptations. Besides, within literature the focus for medical supply delivery has been on developing vehicle routing models, also called drone delivery, for last mile delivery [53, 69, 80]. These models are often small in scale and do not represent the travelled path by UAVs accurately. Lastly, this work will be innovative in the way dynamically adjustable geofencing will be developed and evaluated for no-fly zones based on mission prioritisation and the insights that can be obtained from it. Therefore the three main research gaps are as follows:

1. **Multi-agent Pathfinding in 3D and Real-world Environment is relatively unexplored**
2. **The travelled path of UAVs for drone delivery models is often not represented accurately**
3. **The use of dynamically adjustable no-fly zones (geofences) is relatively unexplored**

## 6.2. Research Objective

Following from the research gaps, described in section 6.1, the primary focus of this work is to develop a drone delivery model in a 3D environment representing the travelled paths of UAVs. The selected use case will be the delivery of medical supplies in the Rotterdam Area. A more accurate representation of the real world environment can be obtained by using dynamically adjustable no-fly zones. This allows for the incorporation of emergency missions with more direct paths, similar to how HEMS operate. Therefore, the main research objective of this work is specified as follows:

**Main Research Objective**

*To develop and evaluate an online path planning and coordination mechanism in 3D for a cooperative fleet of autonomous UAVs delivering medical supplies in the Rotterdam Area with dynamically adjustable geofences*

## 6.3. Research Questions

This section will give an overview of the main research questions as well as related sub questions needed to obtain the main research objective. If applicable, the sub-questions will be linked to the requirements and assumptions made in section 4.3.

1. **How can the operational environment for medical UAV delivery be modelled?**
   - What are the key variables and parameters that need to be taken into account when modelling the operational environment for medical UAV delivery? *(REQ-ENV-01, REQ-ENV-02)*
   - What assumptions can be made regarding the locations of medical facilities and warehouses in Rotterdam?
   - How can a complex 2D city structure be modelled into a simplified 3D environment? *(REQ-ENV-02)*
   - How can the airspace structure and no-fly zones within Rotterdam be modelled? *(REQ-ENV-03, REQ-ENV-04)*
   - How can dynamically adjustable (no-)fly zones be modelled to allow for UAV separation and emergency missions? *(REQ-ENV-03, REQ-ENV-04)*
2. **How can different actors in UTM be represented in an agent based model?**
   - What are relevant operational requirements and constraints for medical UAV delivery that need to be taken into account for each agent? *(REQ-DEL-01, REQ-DEL-02)*
   - How will UTM communication between different agents (Authority, Operator, UAV, Customer) take place? *(REQ-VEH-02)*
   - Which information needs to be shared between UAVs to allow for successful (re-)planning and coordination? *REQ-PC-01 – REC-PC-04*
   - How can UAV performance (e.g. speed, range, payload) be modelled? *(REQ-VEH-01, REQ-VEH-03, REQ-VEH-04)*
3. **How can demand for medical supplies for different medical facilities be modelled?**
   - Which data can be utilised to model the demand of medical supplies for medical facilities?
   - Which medical items will be suitable for UAV delivery and which operational constraints will need to be taken into account? *(REQ-DEL-01, REQ-DEL-02, REQ-VEH-01)*
4. **How can an online path planning and coordination mechanism be developed to allow for efficient and safe operations in urban environment?**
   - How can multi-agent path finding be used for modelling 3D UAV operations? *(REQ-PC-01 – REC-PC-04)*
   - How can safe separation between UAVs as well as (virtual) obstacles be taken into account during path finding? *(REQ-PC-01, REQ-PC-02)*

- How can mission prioritisation be incorporated in the path (re-)planning of UAVs? *(REQ-PC-03)*
- How can uncertainty (e.g. minor delays, weather) be incorporated into path (re-)planning of UAVs? *(REQ-PC-03)*

5. **How can the performance of the implemented path planning and coordination mechanism be analysed and explained?**

   - Which metrics and performance indicators can be utilised to evaluate the performance of the path planning and coordination model? *(REQ-PC-01 – REQ-PC-04)*
   - How does the selection of warehouses and medical facilities influence the obtained results? *(ASS-DEL-01, ASS-DEL-02)*
   - How can the model be used to provide insights into the modelling of other UAM services?

# III

## Supporting work

# 1

# Path Planning and Coordination Specifications

In this chapter, we briefly discuss the different functions present in the proposed path planning and coordination mechanism. The selected method is Windowed Cooperative Safe Interval Path Planning (WC-SIPP). This chapter will focus on the newly introduced or adjusted function, which differentiates our method from the original SIPP method, introduced by Philips and Likhachev [75].

## 1.1. findPath

The findPath function is called in WC-SIPP, when a path of a new agent needs to be planned. This function takes as input the starting position, time, speed, heading of the agent. The graph representation of the environment is also taken into account. In our approach, the start position is dependent on the planning window and therefore taking into account the current speed and heading is important to ensure that only feasible motions are performed by the UAV agent. The starting positions of agents could therefore be at the warehouse location or at a vertex in the environment, when already in operation. The pseudo-code of this function can be found in Algorithm 5. Besides the different input information, the function is not altered compared to the original findPath function from Philips and Likhachev [75].

## 1.2. getSuccessors

The getSuccessors function is used when finding a path for an agent using SIPP. Successors are created in the graph search to find a path from a start position to a goal position, with each successor representing a state that can be reached from the current state by taking a valid action. For this function several adjustments are made compared to the original function proposed by Philips and Likhachev [75]. First, for each configuration, it is checked if a motion is feasible considering the kinematic constraints. This is done to ensure that the trajectory is also valid considering the kinematics of the UAV agent. If the motion is considered to be infeasible given the kinematic constraints, the configuration is disregarded. Kinematic constraints are for example that a motion is not feasible given the time to accelerate/decelerate based on the edge length or having a too high turn speed. Furthermore, the function is adapted to include that the facility positions are seen as safe spaces. Lastly, it is checked if a collision occurs and if the separation distance is violated during flight.

## 1.3. possibleMoves

During the generation of successors for a state, possible moves from the current state to a new state are generated. In our model, a distinction can be made for the current position being on a vertex or on an edge. If the current position is a vertex, the valid neighbours will be neighbouring edge positions. The agent is not allowed to wait, so therefore the current position is not considered as a possible successor. For each successor, the state is defined by its location and speed. UAV agents are only allowed to change speed at vertex positions. Therefore, when generating successors for a vertex position, the number of possible successors is the number of neighbouring edges times the number of different travelling speeds. In our case, the UAV has a minimum speed of 2 $[m/s]$ and a maximum speed of 18 $[m/s]$ with an acceleration of 2 $[m/s^2]$ [21]. This indicates that

**Algorithm 5** Function *findPath*

**Input:**
  $n_{start}$: start position
  $t_{start}$: start time
  $V_{start}$: start speed
  $\psi_{start}$: start heading
  $G(V,E)$: Graph Urban Environment
**Output:**
  $P_{a_i}$: Path for agent $a_i$
**Function:**
$OPEN = \emptyset$, $CLOSED = \emptyset$
$s_{start} = (n_{start}, t_{start}, V_{start}, \psi_{start})$
f($s_{start}$) = h($s_{start}$)
insert $s_{start}$ into $OPEN$
**while** $s_{goal}$ is not expanded **do**
  $s \leftarrow$ state from $OPEN$ with minimal $f$-value
  remove $n$ from $OPEN$, insert $n$ to $CLOSED$
  $successors = getSuccessors(s)$
  **for** each $s'$ in $successors$ **do**
    **if** $s'$ was not visited before: **then**
      f(s') = g(s') = $\infty$
    **if** g(s') > g(s) + c(s,s'): **then**
      g(s') = g(s) + c(s, s')
      f(s') = g(s')+ h(s')
      insert s' into $OPEN$ with
**return** path

**Algorithm 6** SIPP Function getSuccessors

**Input:**
  current state $s$, Graph Urban Environment $G(V,E)$, Safe Intervals for each cfg $SI$,
  Separation List $SL$
**Output:**
  $successors$
**Function:**
$successors = \emptyset$
**for** each $motion$ in possibleMotions: **do**
  cfg = configuration of $motion$ applied to $s$
  $t_{motion}$ = time to perform $motion$
  **if** cfg is not a feasible motion considering kinematic constraints: **then**
    **Continue**
  $t_{start}$ = time(s) + $t_{motion}$
  $t_{end}$ = endTime(SI(s)) + $t_{motion}$
  **for** each safe interval SI in cfg: **do**
    **if** startTime(SI(s)) > $t_{end}$ or endTime(SI(s)) < $t_{start}$ **then**
      **Continue**
    t = earliest arrival time at cfg during interval SI
    **if** position in cfg is not a facility position: **then**
      **if** collision at t for cfg: **then**
        **Continue**
      **if** separation distance is violated for motion: **then**
        **Continue**
    s' = state of cfg with SI and t
    insert s' into successors
**return** $successors$

per neighbouring edge, 9 different speed options are available. If the current state position is an edge, only one possible successor is generated, which is the upcoming node position with the current edge speed.

After generating possible motions, it is checked if the motion is also feasible considering the kinematic constraints. First, it is checked if the motion is possible considering the turn speed restrictions. At a node, the heading change is calculated. If the turn speed is larger than the allowed turn speed based on the heading change, the successor is disregarded. Second, we verify if the possible motion is feasible considering the acceleration of the UAV. For example, if the current state has a speed of 18 $[m/s]$ and the successor state has a speed of 2 $[m/s]$ with an edge length of only 20 meters, the possible motion is deemed to be infeasible. This as the UAV is not able to decelerate to 2 $[m/s]$ during the length of this edge. Third, if the successor state would result in a motion where the altitude is changed, it is checked if the altitude can be reached within the time of travelling on the edge. If not, the successor is disregarded.

## 1.4. checkSeparation

In the process of generating successors, there is a check to ensure that active UAV agents maintain separation during potential movements. This is done using the checkSeparation function. The separation distance between UAVs is 32 meters in horizontal direction and 25 feet in vertical direction. The UAS operator keeps track of each UAV of its position during flight. The positions are discretised per time step of 1 second and contain the location expressed in latitude, longitude and altitude. The positions of all active UAVs are stored in a separation list. During the generation of successors for each feasible motion considering the kinematic constraints, it is checked if during that motion the separation distance is violated. This means, for example, that if a motion on an edge would take 10 seconds to perform, for each second during that motion, the distance to other active UAVs would be calculated. If the distance between UAVs during this motion is less than the defined separation distance, the successor is disregarded. It must be noted that this process is computationally heavy, as for every possible successor the separation distance between UAVs needs to be calculated for every second of the performed motion. For future research, more advanced methods should be introduced to decrease the amount of calculations needed to ensure separation.

## 1.5. updateDynamicObstacles

After paths of agents with a higher priority are planned, the paths of these agents are stored as dynamic obstacles. The combined list of dynamic obstacles is used to determine the safe intervals for both vertices and edges in order to avoid collisions while planning paths for agents with lower priorities. As agents are able to spend a longer period of time on an edge, it is important to implement safe intervals on edges as well. This in order to avoid conflicts where agents transverse the same edge at the same time in opposite directions. In our model, agents are able to follow each other on the same edge if they are travelling in the same direction. This is allowed as long as the distance between these agents is larger than the safety separation distance. Therefore, a safety interval is only imposed in one direction, being the opposite travel direction of the UAV agent.

# 2

# Overview of Scenarios

In this chapter, we present an overview of various scenarios and their corresponding independent variable settings. This chapter organizes the scenarios accounting for each experiment, which are detailed in subsequent sections.

*The remainder of this page is intentionally left blank due to formatting reasons*

## 2.1. Experiment A Scenarios

Table 2.1: Overview of scenarios used for experiment A, including all independent variable settings

| Scenario | Urban Environment Type | Demand Rate | Urgency Distribution for Packages | Priority Planning Order | Number of Unexpected Obstacles | Number of Layers | Window Size |
|---|---|---|---|---|---|---|---|
| 1 | Small | 2 | 25 | MTRT | 0 | 3 | 20 |
| 2 | Small | 4 | 25 | MTRT | 0 | 3 | 20 |
| 3 | Small | 6 | 25 | MTRT | 0 | 3 | 20 |
| 4 | Large | 2 | 25 | MTRT | 0 | 3 | 20 |
| 5 | Large | 4 | 25 | MTRT | 0 | 3 | 20 |
| 6 | Large | 6 | 25 | MTRT | 0 | 3 | 20 |
| 7 | Small | 2 | 25 | MTRT | 0 | 3 | 30 |
| 8 | Small | 4 | 25 | MTRT | 0 | 3 | 30 |
| 9 | Small | 6 | 25 | MTRT | 0 | 3 | 30 |
| 10 | Large | 2 | 25 | MTRT | 0 | 3 | 30 |
| 11 | Large | 4 | 25 | MTRT | 0 | 3 | 30 |
| 12 | Large | 6 | 25 | MTRT | 0 | 3 | 30 |
| 13 | Small | 2 | 25 | MTRT | 0 | 3 | 40 |
| 14 | Small | 4 | 25 | MTRT | 0 | 3 | 40 |
| 15 | Small | 6 | 25 | MTRT | 0 | 3 | 40 |
| 16 | Large | 2 | 25 | MTRT | 0 | 3 | 40 |
| 17 | Large | 4 | 25 | MTRT | 0 | 3 | 40 |
| 18 | Large | 6 | 25 | MTRT | 0 | 3 | 40 |

## 2.2. Experiment B Scenarios

Table 2.2: Overview of scenarios used for experiment B, including all independent variable settings

| Scenario | Urban Environment Type | Demand Rate | Urgency Distribution for Packages | Priority Planning Order | Number of Unexpected Obstacles | Number of Layers | Window Size |
|---|---|---|---|---|---|---|---|
| 1 | Small | 2 | 25 | MTRT | 0 | 3 | 40 |
| 2 | Small | 4 | 25 | MTRT | 0 | 3 | 40 |
| 3 | Small | 6 | 25 | MTRT | 0 | 3 | 40 |
| 4 | Large | 2 | 25 | MTRT | 0 | 3 | 40 |
| 5 | Large | 4 | 25 | MTRT | 0 | 3 | 40 |
| 6 | Large | 6 | 25 | MTRT | 0 | 3 | 40 |
| 7 | Small | 2 | 25 | MTRT | 0 | 1 | 40 |
| 8 | Small | 4 | 25 | MTRT | 0 | 1 | 40 |
| 9 | Small | 6 | 25 | MTRT | 0 | 1 | 40 |
| 10 | Small | 2 | 25 | MTRT | 0 | 5 | 40 |
| 11 | Small | 4 | 25 | MTRT | 0 | 5 | 40 |
| 12 | Small | 6 | 25 | MTRT | 0 | 5 | 40 |
| 13 | Large | 2 | 25 | MTRT | 0 | 1 | 40 |
| 14 | Large | 4 | 25 | MTRT | 0 | 1 | 40 |
| 15 | Large | 6 | 25 | MTRT | 0 | 1 | 40 |
| 16 | Large | 2 | 25 | MTRT | 0 | 5 | 40 |
| 17 | Large | 4 | 25 | MTRT | 0 | 5 | 40 |
| 18 | Large | 6 | 25 | MTRT | 0 | 5 | 40 |

## 2.3. Experiment C Scenarios

Table 2.3: Overview of scenarios used for experiment C, including all independent variable settings

| Scenario | Urban Environment Type | Demand Rate | Urgency Distribution for Packages | Priority Planning Order | Number of Unexpected Obstacles | Number of Layers | Window Size |
|---|---|---|---|---|---|---|---|
| 1 | Small | 2 | 25 | MTRT | 0 | 3 | 40 |
| 2 | Small | 4 | 25 | MTRT | 0 | 3 | 40 |
| 3 | Small | 6 | 25 | MTRT | 0 | 3 | 40 |
| 4 | Large | 2 | 25 | MTRT | 0 | 3 | 40 |
| 5 | Large | 4 | 25 | MTRT | 0 | 3 | 40 |
| 6 | Large | 6 | 25 | MTRT | 0 | 3 | 40 |
| 7 | Small | 2 | 25 | MTRT | 5 | 3 | 40 |
| 8 | Small | 4 | 25 | MTRT | 5 | 3 | 40 |
| 9 | Small | 6 | 25 | MTRT | 5 | 3 | 40 |
| 10 | Large | 2 | 25 | MTRT | 5 | 3 | 40 |
| 11 | Large | 4 | 25 | MTRT | 5 | 3 | 40 |
| 12 | Large | 6 | 25 | MTRT | 5 | 3 | 40 |
| 13 | Small | 2 | 25 | MTRT | 10 | 3 | 40 |
| 14 | Small | 4 | 25 | MTRT | 10 | 3 | 40 |
| 15 | Small | 6 | 25 | MTRT | 10 | 3 | 40 |
| 16 | Large | 2 | 25 | MTRT | 10 | 3 | 40 |
| 17 | Large | 4 | 25 | MTRT | 10 | 3 | 40 |
| 18 | Large | 6 | 25 | MTRT | 10 | 3 | 40 |
| 19 | Small | 2 | 25 | MTRT | 15 | 3 | 40 |
| 20 | Small | 4 | 25 | MTRT | 15 | 3 | 40 |
| 21 | Small | 6 | 25 | MTRT | 15 | 3 | 40 |
| 22 | Large | 2 | 25 | MTRT | 15 | 3 | 40 |
| 23 | Large | 4 | 25 | MTRT | 15 | 3 | 40 |
| 24 | Large | 6 | 25 | MTRT | 15 | 3 | 40 |
| 25 | Small | 2 | 25 | MTRT | 20 | 3 | 40 |
| 26 | Small | 4 | 25 | MTRT | 20 | 3 | 40 |
| 27 | Small | 6 | 25 | MTRT | 20 | 3 | 40 |
| 28 | Large | 2 | 25 | MTRT | 20 | 3 | 40 |
| 29 | Large | 4 | 25 | MTRT | 20 | 3 | 40 |
| 30 | Large | 6 | 25 | MTRT | 20 | 3 | 40 |

## 2.4. Experiment D Scenarios

Table 2.4: Overview of scenarios used for experiment D, including all independent variable settings

| Scenario | Urban Environment Type | Demand Rate | Urgency Distribution for Packages | Priority Planning Order | Number of Unexpected Obstacles | Number of Layers | Window Size |
|---|---|---|---|---|---|---|---|
| 1 | Small | 2 | 25 | MTRT | 0 | 3 | 40 |
| 2 | Small | 4 | 25 | MTRT | 0 | 3 | 40 |
| 3 | Small | 6 | 25 | MTRT | 0 | 3 | 40 |
| 4 | Large | 2 | 25 | MTRT | 0 | 3 | 40 |
| 5 | Large | 4 | 25 | MTRT | 0 | 3 | 40 |
| 6 | Large | 6 | 25 | MTRT | 0 | 3 | 40 |
| 7 | Large | 2 | 10 | MTRT | 0 | 3 | 40 |
| 8 | Large | 4 | 10 | MTRT | 0 | 3 | 40 |
| 9 | Large | 6 | 10 | MTRT | 0 | 3 | 40 |
| 10 | Large | 2 | 50 | MTRT | 0 | 3 | 40 |
| 11 | Large | 4 | 50 | MTRT | 0 | 3 | 40 |
| 12 | Large | 6 | 50 | MTRT | 0 | 3 | 40 |
| 13 | Large | 2 | 10 | RT | 0 | 3 | 40 |
| 14 | Large | 4 | 10 | RT | 0 | 3 | 40 |
| 15 | Large | 6 | 10 | RT | 0 | 3 | 40 |
| 16 | Large | 2 | 25 | RT | 0 | 3 | 40 |
| 17 | Large | 4 | 25 | RT | 0 | 3 | 40 |
| 18 | Large | 6 | 25 | RT | 0 | 3 | 40 |
| 19 | Large | 2 | 50 | RT | 0 | 3 | 40 |
| 20 | Large | 4 | 50 | RT | 0 | 3 | 40 |
| 21 | Large | 6 | 50 | RT | 0 | 3 | 40 |
| 22 | Large | 2 | 10 | R | 0 | 3 | 40 |
| 23 | Large | 4 | 10 | R | 0 | 3 | 40 |
| 24 | Large | 6 | 10 | R | 0 | 3 | 40 |
| 25 | Large | 2 | 25 | R | 0 | 3 | 40 |
| 26 | Large | 4 | 25 | R | 0 | 3 | 40 |
| 27 | Large | 6 | 25 | R | 0 | 3 | 40 |
| 28 | Large | 2 | 50 | R | 0 | 3 | 40 |
| 29 | Large | 4 | 50 | R | 0 | 3 | 40 |
| 30 | Large | 6 | 50 | R | 0 | 3 | 40 |

# 3

# Variability of Simulation

One of the key considerations in experimental design when considering a multi-agent system is determining the appropriate number of experimental runs to obtain meaningful results. This is especially important when randomness is introduced in scenarios. This is the case in our work, as packages are requested by the healthcare facility agent using a Poisson Distribution. Therefore, the coefficient of variation (CV) is used as a statistical measure to asses the variability within a dataset relative to its mean. The stabilisation of the coefficient of variation indicates that a sufficient number of simulations is performed to assess the distribution of the model output. The coefficient of variation can be calculated using Equation 3.1.

$$CV = \frac{\sigma(o)}{\mu(o)} \tag{3.1}$$

Every scenario is characterised by a unique subset of independent variables, and carried out a total of 100 simulation runs. For all scenarios the same set of random seeds is used in order to ensure uniformity in inputs across scenarios, allowing for direct comparisons between them. However, it is important to note that due to the implementation of our path planning and coordination mechanism bottlenecks could occur within the simulations. The occurrence of a bottleneck makes the simulation unsuccessful. To ensure a fair comparison between results across different scenarios, we adhered to using identical sets of seeds. This means that for certain experiments, the number of simulations available for comparison is reduced. Therefore, we opted to select 100 simulation runs per scenario, as this resulted in a constant coefficient of variance. Only for experiment C, a total of 200 simulation runs per scenario was necessary to ensure stability of the coefficient of variance. Considering the size of our study only the most variable scenarios per experiment are shown, where the variability was observed visually.

## 3.1. Experiment A

The coefficient of variation for different KPIs is shown in subsequent figures, considering the small environment with a demand rate of 2 and a window size of 20 for experiment A.

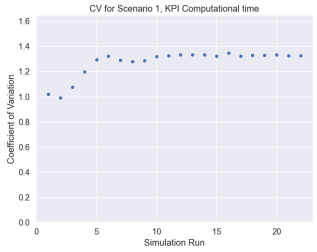


Figure 3.1: Coefficient of variation for experiment A, KPI Computational Time

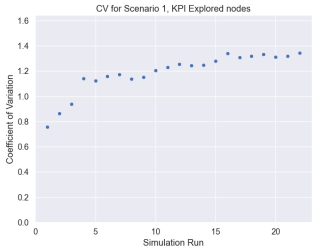


Figure 3.2: Coefficient of variation for experiment A, KPI Explored Nodes

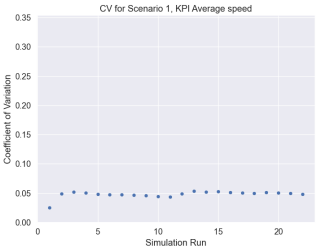


Figure 3.3: Coefficient of variation for experiment A, KPI Average Speed

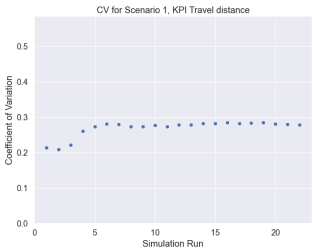


Figure 3.4: Coefficient of variation for experiment A, KPI Traveled Distance

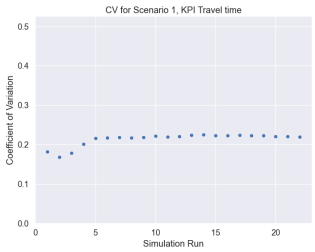


Figure 3.5: Coefficient of variation for experiment A, KPI Travel Time

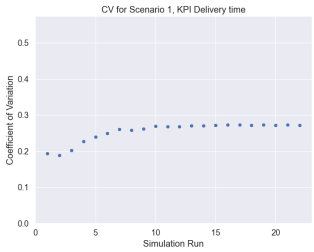


Figure 3.6: Coefficient of variation for experiment A, KPI Delivery Time

## 3.2. Experiment B

The coefficient of variation for different KPIs is shown in subsequent figures, considering the small environment with a demand rate of 4 and 1 available layer.

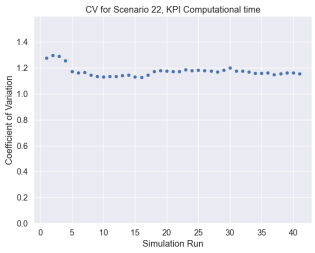


Figure 3.7: Coefficient of variation for experiment B, KPI Computational Time

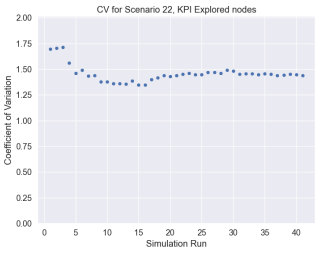


Figure 3.8: Coefficient of variation for experiment B, KPI Explored Nodes

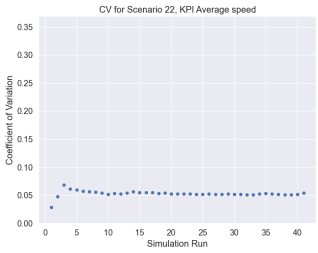


Figure 3.9: Coefficient of variation for experiment B, KPI Average Speed


Figure 3.10: Coefficient of variation for experiment B, KPI Traveled Distance

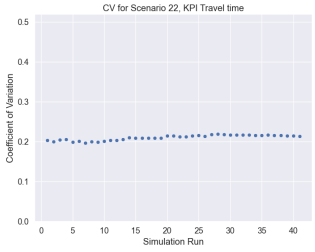


Figure 3.11: Coefficient of variation for experiment B, KPI Travel Time

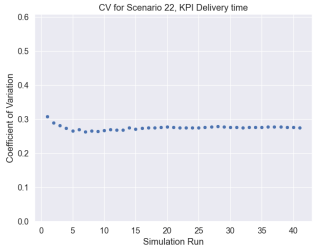


Figure 3.12: Coefficient of variation for experiment B, KPI Delivery Time

## 3.3. Experiment C

The coefficient of variation for different KPIs is shown in subsequent figures, considering the small environment with a demand rate of 2, with 0 unexpected obstacles for experiment C.

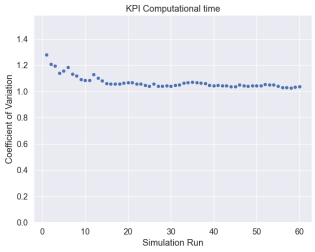


Figure 3.13: Coefficient of variation for experiment C, KPI Computational Time

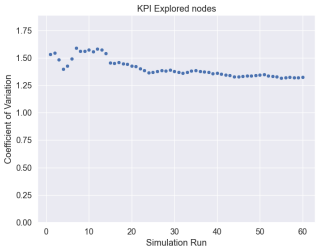


Figure 3.14: Coefficient of variation for experiment C, KPI Explored Nodes

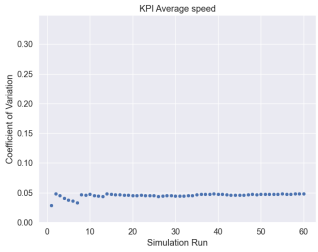


Figure 3.15: Coefficient of variation for experiment C, KPI Average Speed

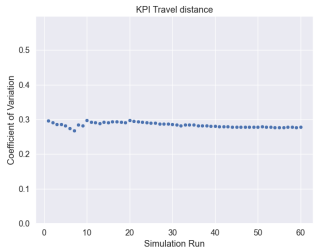


Figure 3.16: Coefficient of variation for experiment C, KPI Traveled Distance

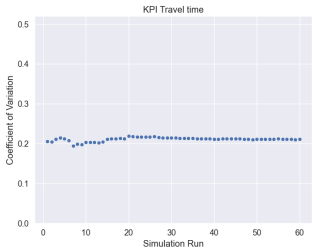


Figure 3.17: Coefficient of variation for experiment C, KPI Travel Time


Figure 3.18: Coefficient of variation for experiment C, KPI Delivery Time

## 3.4. Experiment D

The coefficient of variation for different KPIs is shown in subsequent figures for the large environment, with a request time priority planning order, 50% urgency distribution, and a demand rate of 2.

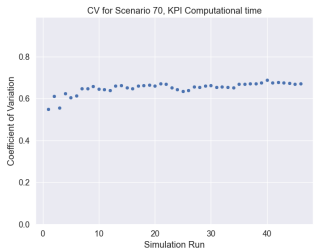


Figure 3.19: Coefficient of variation for experiment D, KPI Computational Time

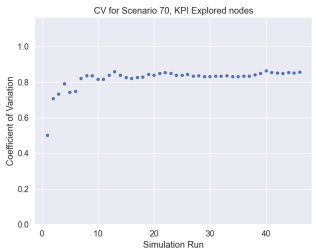


Figure 3.20: Coefficient of variation for experiment D, KPI Explored Nodes

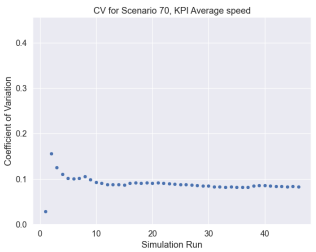


Figure 3.21: Coefficient of variation for experiment D, KPI Average Speed

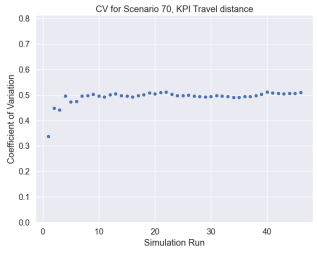


Figure 3.22: Coefficient of variation for experiment D, KPI Traveled Distance

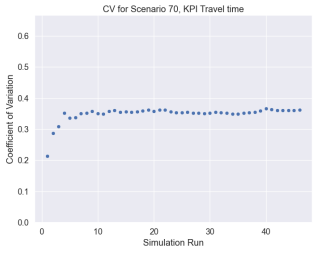


Figure 3.23: Coefficient of variation for experiment D, KPI Travel Time

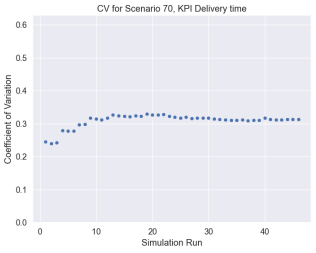


Figure 3.24: Coefficient of variation for experiment D, KPI Delivery Time

# 4

# Additional Experiment A Results

In this chapter, additional results are presented for experiment A. The aim of this experiment is to investigate the impact of the window sizes used for WC-SIPP on the performance of the algorithm.

## 4.1. Performance WC-SIPP

In our research paper, we presented the computational time and explored nodes for various window sizes for the large environment. In this section, the computational time and explored nodes for various window sizes and demand rates are visualised in Figure 4.1 and Figure 4.2 considering the small environment. Here it is illustrated that for every demand scenario, a window size of 40 offers better computational time performance compared to the other window sizes. The average speed KPI is also shown in Figure 4.3 and Figure 4.3 for the small and large environments respectively. Here a difference in average speed can be observed for different window sizes. A statistical analysis is performed using the Kruskal Wallis statistical test, which demonstrates that there is a significant difference considering the computational time, explored nodes and average speed. This is true for all demand rates and environments, except for 2 scenarios. This is for the small environment with a demand rate of 6 for the computational time KPI, and for the large environment with a demand rate of 2 for the average speed KPI. The p-values can be seen in Table 4.1.

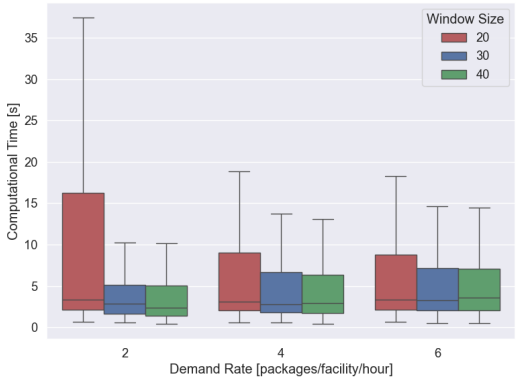


Figure 4.1: Computational time with varying window sizes and demand rates for the small environment

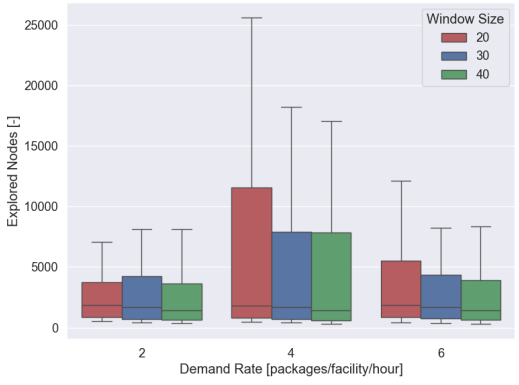


Figure 4.2: Explored nodes with varying window sizes and demand rates for the small environment

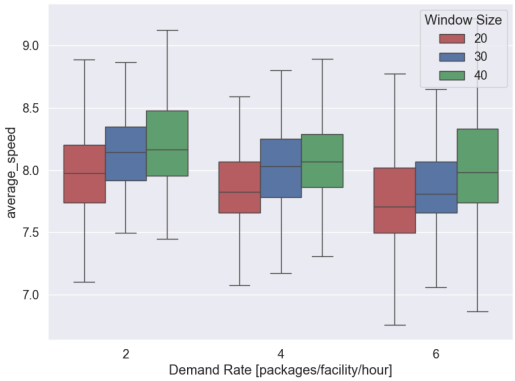


Figure 4.3: Average speed with varying window sizes and demand rates for the small environment

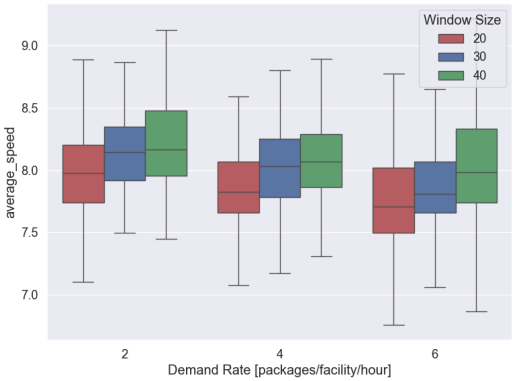


Figure 4.4: Average speed with varying window sizes and demand rates for the small environment

Table 4.1: Statistical analysis comparing the performance of the algorithm for window sizes 20, 30, and 40, varying the environment type and demand rate. Values given in the table are p-values obtained from the Kruskal Wallis statistical test

| **Environment** | Small | | | Large | | |
|---|---|---|---|---|---|---|
| **Demand Rate** | 2 | 4 | 6 | 2 | 4 | 6 |
| **KPI** | | | | | | |
| Computational Time | 3.4E-06 | 6.11E-03 | 0.62 | 3.7E-08 | 1.9E-12 | 3.7E-14 |
| Explored Nodes | 1.0E-04 | 5.1E-19 | 6.7E-11 | 1.4E-07 | 5.0E-15 | 1.4E-25 |
| Travel Distance | 0.94 | 0.87 | 0.99 | 0.86 | 0.66 | 0.70 |
| Travel Time | 0.98 | 0.99 | 0.94 | 0.92 | 0.74 | 0.86 |
| Average Speed | 4.59E-07 | 3.56E-43 | 1.64E-24 | 0.34 | 1.32E-10 | 1.03E-23 |
| Delivery Time | 0.99 | 0.99 | 0.99 | 0.97 | 0.76 | 0.75 |

Following the Kruskal-Wallis test, the Vargha-Delaney test is used to analyse the effect size of the observed significant difference. An overview of the A-values is given in Table 4.2, which shows for the computational time a negligible to small effect. For the explored nodes a small effect is observed, and for the average speed, a negligible to moderate effect is seen.

Table 4.2: Effect size analysis comparing the performance of the algorithm for window sizes 20 and 40, varying the environment type and demand rate. Values given in the table are A-values obtained from the Vargha-Delaney statistical test

| **Environment** | Small | | | Large | | |
|---|---|---|---|---|---|---|
| **Demand Rate** | 2 | 4 | 6 | 2 | 4 | 6 |
| **KPI** | | | | | | |
| Computational Time | 0.65 S | 0.56 N | 0.51 N | 0.62 S | 0.61 S | 0.59 S |
| Explored Nodes | 0.62 S | 0.62 S | 0.62 S | 0.62 S | 0.61 S | 0.49 S |
| Average Speed | 0.34 S | 0.30 M | 0.32 M | 0.47 N | 0.43 S | 0.38 S |

# 5

# Additional Experiment B Results

This chapter will provide supporting results from experiment B, where the focus has been on investigating the scalability and performance of multi-layer path planning.

## 5.1. Performance WC-SIPP

In our research paper, we presented the increase in computational time and explored nodes for an increasing number of available layers for the large environment. In Figure 5.1 and Figure 5.2 the computational time and explored nodes for an increasing number of available layers for the small environment can be seen. Similar to the large environment, it is demonstrated that for an increasing number of layers, the computational time and explored nodes are significantly increasing.

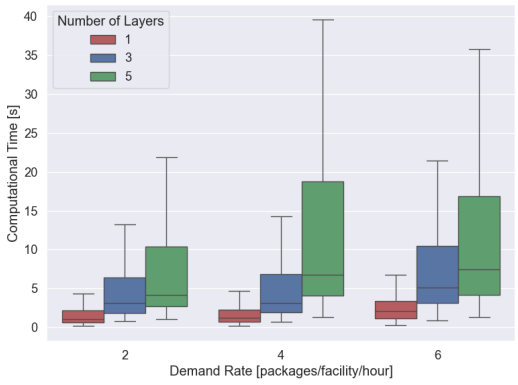


Figure 5.1: Computational time with varying number of layers and demand rates for the small environment

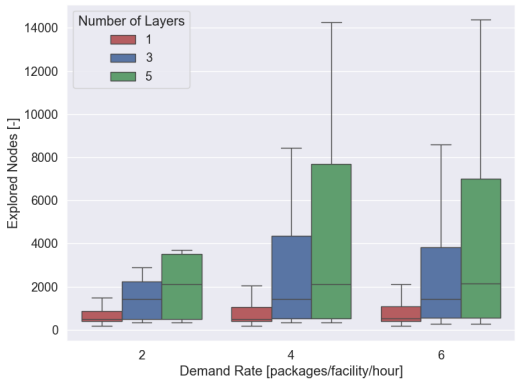


Figure 5.2: Explored nodes with varying numbers of layers and demand rates for the small environment

Furthermore, the computational time for both the small and large environment is visualised in Figure 5.3 and Figure 5.4 respectively using a contour plot. These plots give a comprehensive view of how the computational time varies for different combinations of layers and demand rates. It can be seen that for path planning in multiple layers, the number of layers has a larger influence on the computational time compared to the demand rate. Especially for the large environment, the computational time is increasing rapidly when the number of layers increases. This underlies the conclusion that more efficient methods need to be considered for larger and more complex environments, as otherwise multi-agent path planning in 3D environments could become infeasible.

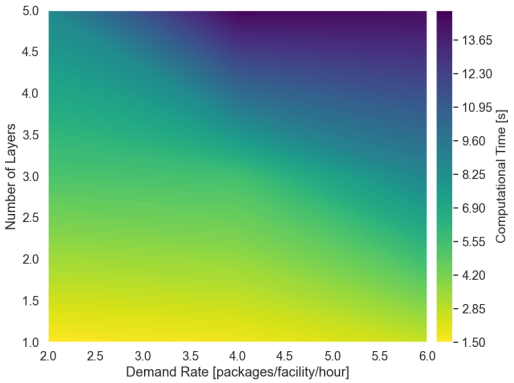


Figure 5.3: Computational time with varying number of layers and demand rates for the small environment

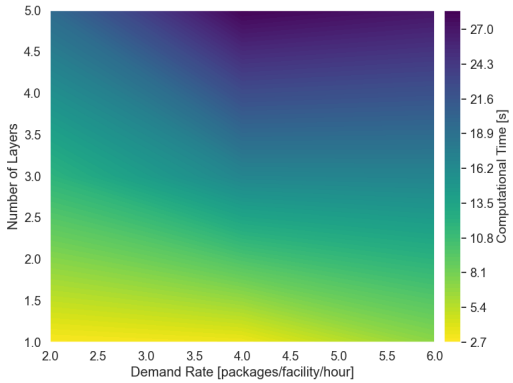


Figure 5.4: Computational time with varying number of layers and demand rates for the large environment

Following from these insights, the Vargha-Delaney effect size test is used to analyse the effect size increasing the number of layers has on the computational time, explored nodes and average speed. The A-values for comparing 1 and 3 layer environments, can be found in Table 5.1. The A-values for comparing 3 and 5 layers can be found in Table 5.2. Here it can be seen that when considering the computational time, a large effect is observed when increasing the number of layers from 1 to 3 and from 3 to 5. For the explored nodes, a moderate to large effect is observed. This is in line with the graphical representation of the computational time and explored node with an increasing number of layers. For the average speed, a negligible effect is observed. So even though a statistically significant difference is measured when increasing the number of layers, the effect on the average speed is not significant, when considering its effect. This difference in average speeds for a varying number of available layers per demand rate is visualised in Figure 5.5 and Figure 5.6 for the small and large environments respectively. These figures support that even though differences are statistically significant, the differences are marginally. Therefore, the increased speed does not lead to a significant improvement in travel time or delivery time.

Table 5.1: Statistical analysis comparing the performance of the algorithm for the small and large environment with 1 and 3 layers, varying the demand rates. Values given in the table are A-values obtained from the Vargha-Delaney effect size test.

| **Environment** | Small | | | Large | | |
|---|---|---|---|---|---|---|
| **Demand Rate** | 2 | 4 | 6 | 2 | 4 | 6 |
| **KPI** | | | | | | |
| Computational Time | 0.83 L | 0.81 L | 0.82 L | 0.91 L | 0.94 L | 0.79 L |
| Explored Nodes | 0.69 M | 0.69 M | 0.69 M | 0.70 M | 0.70 M | 0.70 M |
| Average Speed | 0.54 N | 0.52 N | 0.55 N | 0.53 N | 0.56 N | 0.56 N |

Table 5.2: Statistical analysis comparing the performance of the algorithm for the small and large environment with 3 and 5 layers, varying the demand rates. Values given in the table are A-values obtained from the Vargha-Delaney effect size test.

| **Environment** | Small | | | Large | | |
|---|---|---|---|---|---|---|
| **Demand Rate** | 2 | 4 | 6 | 2 | 4 | 6 |
| **KPI** | | | | | | |
| Computational Time | 0.098 L | 0.06 L | 0.11 L | 0.03 L | 0.02 L | 0.11 L |
| Explored Nodes | 0.27 M | 0.27 M | 0.27 M | 0.25 L | 0.26 L | 0.25 L |
| Average Speed | 0.43 N | 0.46 N | 0.44 N | 0.46 N | 0.43 L | 0.43 S |

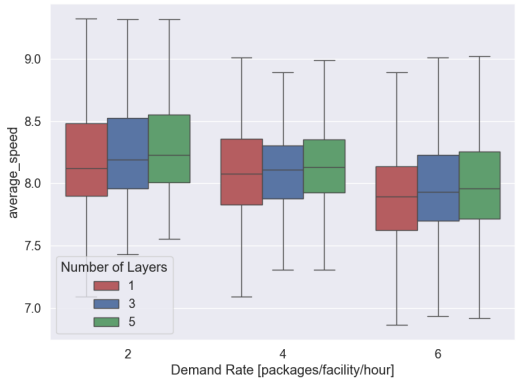


Figure 5.5: Average speed with varying number of layers and demand rates for the small environment

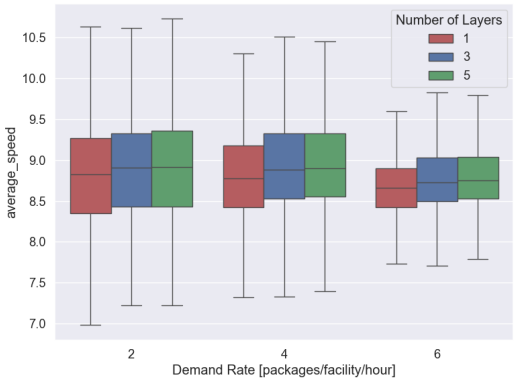


Figure 5.6: Average speed with varying number of layers and demand rates for the large environment

*The remainder of this page is intentionally left blank due to formatting reasons*

## 5.2. Heatmaps

This section gives insight into the flown routes for UAV agents, which are presented using heat maps. In this section only the results for 1 layer and 3 layer environment are shown, to demonstrate the difference between operating in 1 layer and 3 layer environment. This is shown for both the small and large environments. As can be seen for both environments, the bottom layer is utilised most by UAVs during operations. The second and third layers are predominantly used to avoid collisions at frequently visited areas.

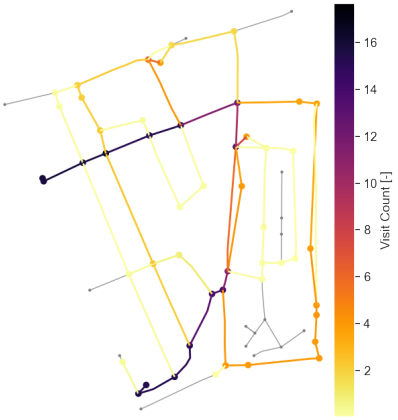


Figure 5.7: Heatmap of the flown paths for the small environment with 1 available layer, for all demand scenarios, showing the number of times a node and edge is visited. Layer 1 is shown

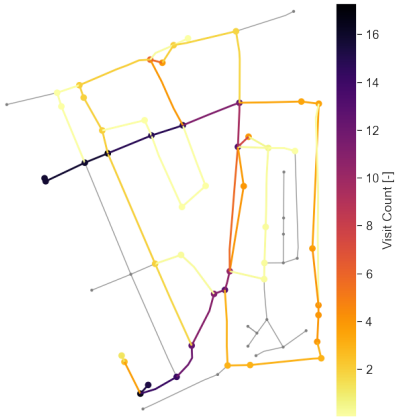


Figure 5.8: Heatmap of the flown paths for the small environment with 3 available layers, for all demand scenarios, showing the number of times a node and edge is visited. Layer 1 is shown

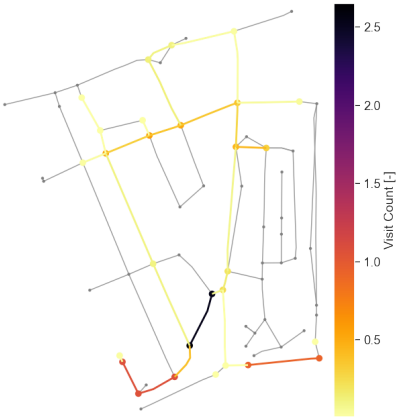


Figure 5.9: Heatmap of the flown paths for the small environment with 1 available layer, for all demand scenarios, showing the number of times a node and edge is visited. Layer 2 is shown

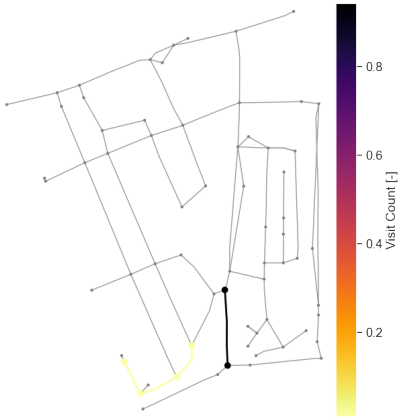


Figure 5.10: Heatmap of the flown paths for the small environment with 3 available layers, for all demand scenarios, showing the number of times a node and edge is visited. Layer 3 is shown

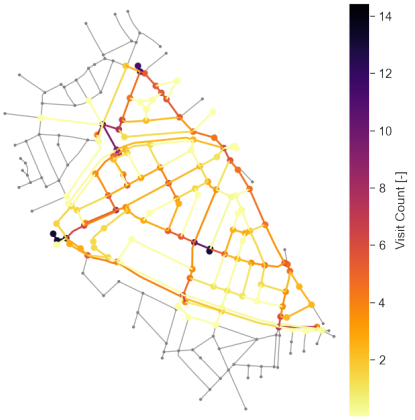


Figure 5.11: Heatmap of the flown paths for the large environment with 1 available layer, for all demand scenarios, showing the number of times a node and edge is visited. Layer 1 is shown

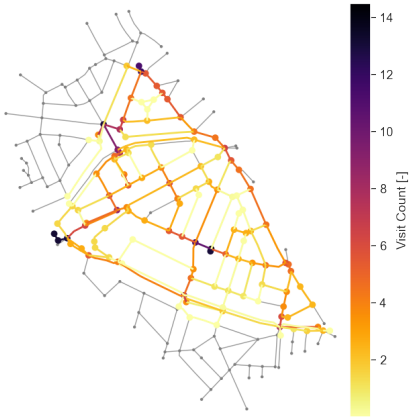


Figure 5.12: Heatmap of the flown paths for the large environment with 3 available layers, for all demand scenarios, showing the number of times a node and edge is visited. Layer 1 is shown

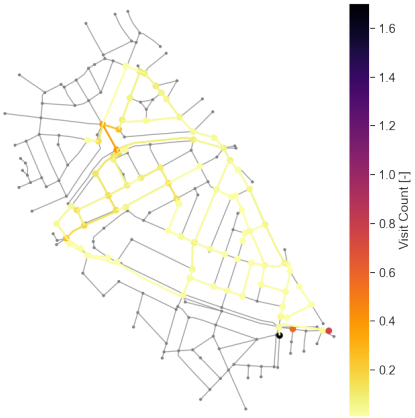


Figure 5.13: Heatmap of the flown paths for the large environment with 1 available layer, for all demand scenarios, showing the number of times a node and edge is visited. Layer 2 is shown

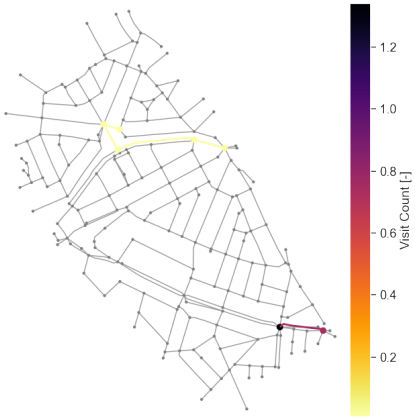


Figure 5.14: Heatmap of the flown paths for the large environment with 3 available layers, for all demand scenarios, showing the number of times a node and edge is visited. Layer 3 is shown

# 6

# Additional Experiment C Results

This chapter will provide supporting results from experiment C, where the aim of the experiment has been to assess the effectiveness of the online path planning and coordination mechanism to plan paths in dynamic environments. Before showing the additional results, the unexpected obstacle locations for both environments are illustrated.

## 6.1. Unexpected Obstacles Placement

The figures in this section present the selected vertices at which unexpected obstacles can be introduced. The majority of the unexpected obstacles are located in the bottom layer, as this layer is mostly utilised by the UAV agents. The locations were selected after conducting an analysis, where the most frequently used vertices by UAVs were observed. Vertices that are direct neighbours of warehouse or healthcare facility locations are not selected as possible locations where unexpected obstacles can appear. This choice has been made to avoid infeasible operations. The locations of unexpected obstacles for the small environment in layers 1 and 2 are shown in Figure 6.1 and Figure 6.2 respectively. For the large environment, the unexpected obstacles in layers 1 and 2 are illustrated in Figure 6.3 and Figure 6.4 respectively.

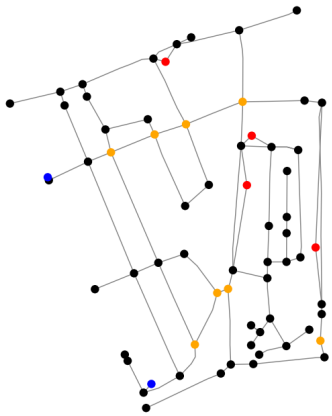

Figure 6.1: Graph representation for the small environment with blue nodes being warehouse locations, red nodes being healthcare facilities and orange nodes being locations where unexpected obstacles could be introduced for layer 1.

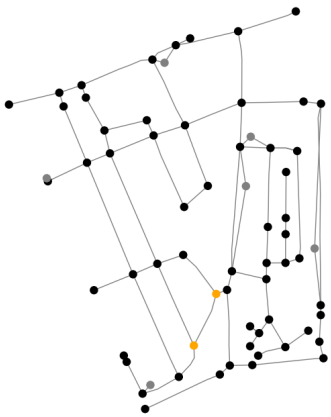

Figure 6.2: Graph representation for the small environment with orange nodes being locations where unexpected obstacles could be introduced for layer 2

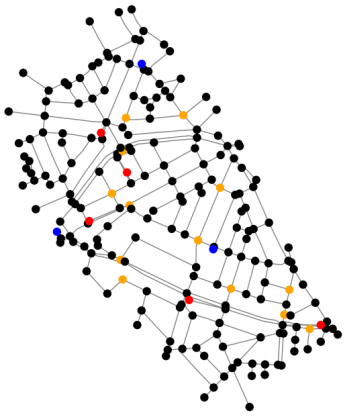

Figure 6.3: Graph representation for the large environment with blue nodes being warehouse locations, red nodes being healthcare facilities and orange nodes being locations where unexpected obstacles could be introduced for layer 1.

Figure 6.4: Graph representation for the large environment with orange nodes being locations where unexpected obstacles could be introduced for layer 2

## 6.2. Performance WC-SIPP

For the third experiment, unexpected obstacles are introduced, where the number of unexpected obstacles ranges from 0 to 20. Using the Kruskal Wallis statistical test, a significant difference was measured for the computational time for both the small and large environments. Using the Vargha-Delaney effect size test, the effect size of the measured difference was negligible. The computational time for both the small and large environment for varying numbers of unexpected obstacles under different demand rates are shown in Figure 6.5 and Figure 6.6 respectively. From these figures, it can be clearly seen that differences are marginal. This is also the case for the average speed in the small environment, which can be seen in Figure 6.7.

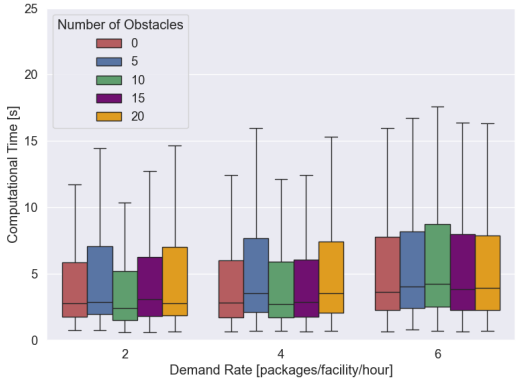


Figure 6.5: Computational time with varying number of unexpected obstacles and demand rates for the small environment

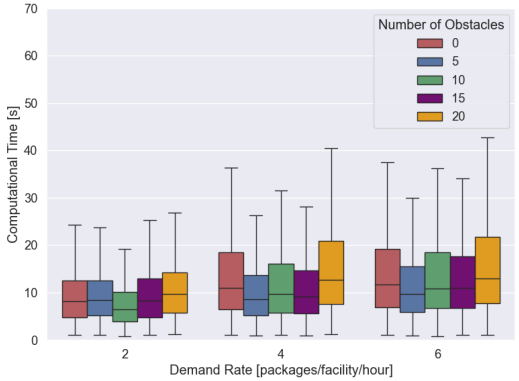


Figure 6.6: Computational time with varying number of unexpected obstacles and demand rates for the large environment

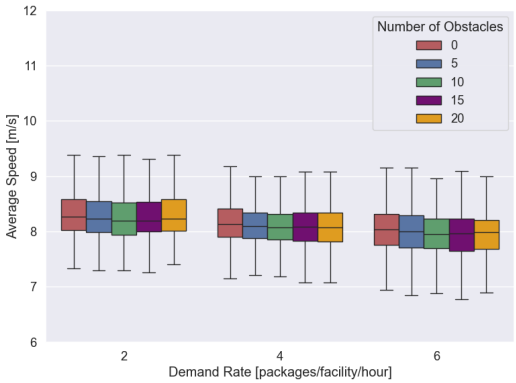


Figure 6.7: Average speed with varying number of unexpected obstacles and demand rates for the small environment

# 7

# Additional Experiment D Results

This chapter will provide supporting results from experiment D, where the focus was on investigating the effect of different priority planning ordering methods to assess the performance of the path planning algorithm.

## 7.1. Comparison urgent versus standard mission types

First, this section demonstrates the effect of different urgency distribution for the mission type and request time priority planning order. The subsequent figures underline the finding that for different urgency distribution rates, no significant difference can be found when comparing KPIs from urgent missions to standard missions. For illustration purposes, only one scenario has been selected, which is for a demand rate of 6.

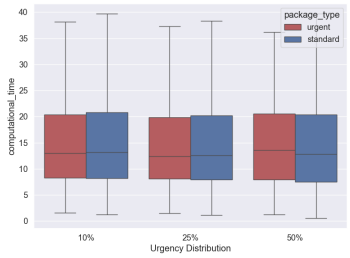


Figure 7.1: Computational time with varying urgency distributions comparing urgent and standard missions for a demand rate of 6, and large environment

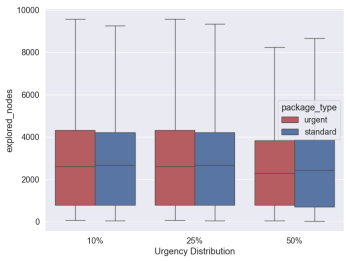


Figure 7.2: Explored nodes with varying urgency distributions comparing urgent and standard missions for a demand rate of 6, and large environment

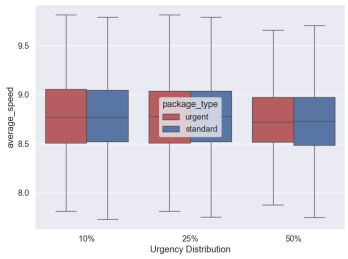


Figure 7.3: Average speed with varying urgency distributions comparing urgent and standard missions for a demand rate of 6, and large environment

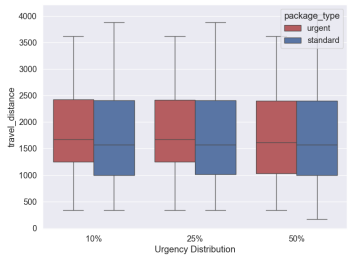


Figure 7.4: Travel distance with varying urgency distributions comparing urgent and standard missions for a demand rate of 6, and large environment

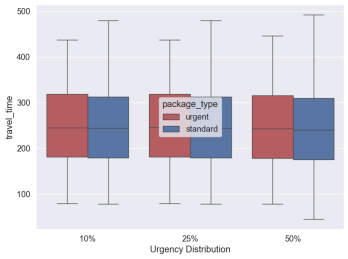


Figure 7.5: Travel time with varying urgency distributions comparing urgent and standard missions for a demand rate of 6, and large environment

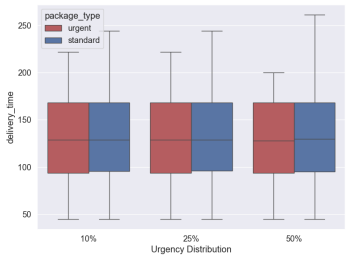


Figure 7.6: Delivery time with varying urgency distributions comparing urgent and standard missions for a demand rate of 6, and large environment

## 7.2. Comparison of Priority Planning Order

Second, additional results are provided to demonstrate the effect of different priority planning orders. The subsequent figures underpin the finding that for different priority planning ordering methods, no significant difference can be found when comparing KPIs from urgent missions to standard missions. For illustration purposes, only one scenario has been selected, which is for a demand rate of 6, and urgency distribution of 50%.

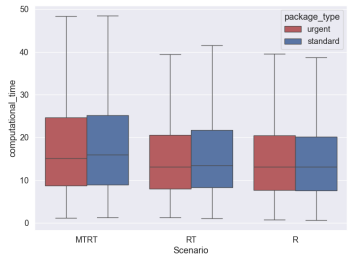


Figure 7.7: Computational time with varying priority planning orders comparing urgent and standard missions for a demand rate of 6, urgency distribution of 50%, and large environment

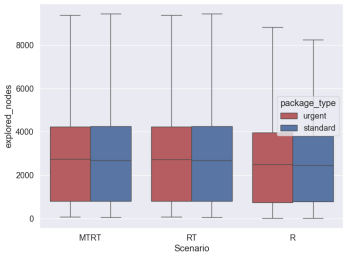


Figure 7.8: Explored Nodes with varying priority planning orders comparing urgent and standard missions for a demand rate of 6, urgency distribution of 50%, and large environment

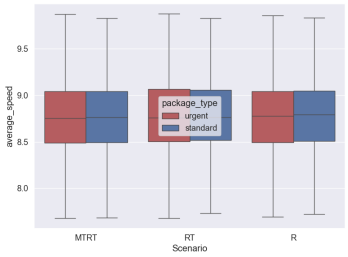


Figure 7.9: Average Speed with varying priority planning orders comparing urgent and standard missions for a demand rate of 6, urgency distribution of 50%, and large environment

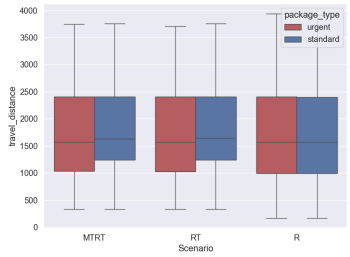


Figure 7.10: Travel distance with varying priority planning orders comparing urgent and standard missions for a demand rate of 6, urgency distribution of 50%, and large environment

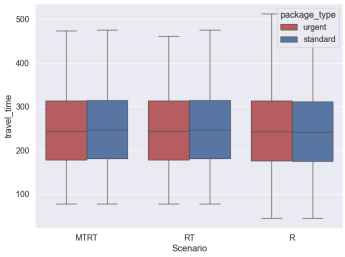


Figure 7.11: Travel time with varying priority planning orders comparing urgent and standard missions for a demand rate of 6, urgency distribution of 50%, and large environment

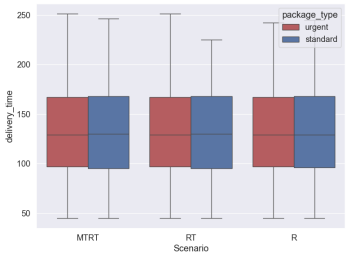


Figure 7.12: Delivery time with varying priority planning orders comparing urgent and standard missions for a demand rate of 6, urgency distribution of 50%, and large environment

# Bibliography


[1] M. Ahmadimanesh, A. Tavakoli, A. Pooya, and F. Dehghanian. Designing an optimal inventory management model for the blood supply chain: synthesis of reusable simulation and neural network. *Medicine*, 99(29), 2020.

[2] Z. Ali and K. Yakovlev. Safe interval path planning with kinodynamic constraints. *arXiv e-prints*, pages arXiv–2302, 2023.

[3] P. Amit. Introduction to a*, May 2014. URL http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html.

[4] A. Andreychuk, K. Yakovlev, E. Boyarski, and R. Stern. Improving continuous-time conflict based search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11220–11227, May 2021.

[5] A. Antelmi, G. Cordasco, G. DAmbrosio, D. De Vinco, and C. Spagnuolo. Experimenting with agent-based model simulation tools. *Applied Sciences*, 13(1), 2023. ISSN 2076-3417.

[6] AT&T. Taking 5g to new heights, 2022. URL https://about.att.com/story/2022/5G-drone-program.html. Accessed on 09-05-2023.

[7] M. Barer, G. Sharon, R. Stern, and A. Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Proceedings of the International Symposium on Combinatorial Search*, volume 5, pages 19–27, 2014.

[8] Blade Urban Air Mobility Inc. Get to the airport in 5 minutes., 2023. URL https://www.blade.com/airport. Accessed on 25-04-2023.

[9] Z. Bnaya and A. Felner. Conflict-oriented windowed hierarchical cooperative a. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3743–3748. IEEE, 2014.

[10] A. Botea, D. Bonusi, and P. Surynek. Solving multi-agent path finding on strongly biconnected digraphs. *Journal of Artificial Intelligence Research*, 62:273–314, 2018.

[11] E. Boyarski, A. Felner, R. Stern, G. Sharon, E. Shimony, O. Bezalel, and D. Tolpin. Improved conflict-based search for optimal multi-agent path finding. *24th International Joint Conference on Artificial Intelligence*, 2015.

[12] E. Boyrasky, A. Felner, G. Sharon, and R. Stern. Dont split, try to work it out: Bypassing conflicts in multi-agent pathfinding. *Proceedings of the International Conference on Automated Planning and Scheduling*, 25(1):47–51, Apr. 2015.

[13] A. Candra, M.A. Budiman, and R.I. Pohan. Application of a-star algorithm on pathfinding game. *Journal of Physics: Conference Series*, 1898(1):012047, jun 2021.

[14] A. Chakrabarty, C.A. Ippolito, J. Baculi, K.S. Krishnakumar, and S. Hening. Vehicle to vehicle (v2v) communication for collision avoidance for multi-copters flying in utm–tcl4. In *AIAA Scitech 2019 Forum*, page 0690, 2019.

[15] S. Choudhury, K. Solovey, M.J. Kochenderfer, and M. Pavone. Efficient large-scale multi-drone delivery using transit networks. *Journal of Artificial Intelligence Research*, 70:757–788, 2021.

[16] A. Cornell, B. Kloss, DJ Presser, and R. Riedel. Tdrones take to the sky, potentially disrupting last-mile delivery, 2023. URL https://www.mckinsey.com/industries/aerospace-and-defense/our-insights/future-air-mobility-blog/drones-take-to-the-sky-potentially-disrupting-last-mile-delivery. Accessed on 25-04-2023.

[17] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti. Primal$_2$: Pathfinding via reinforcement and imitation multi-agent learning - lifelong. *IEEE Robotics and Automation Letters*, 6(2):2666–2673, 2021.

[18] M. de Bok and L. Tavasszy. An empirical agent-based simulation system for urban goods transport (mass-gt). *Procedia Computer Science*, 130:126–133, 2018. ISSN 1877-0509. The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops.

[19] B. de Wilde, A.W. ter Mors, and C. Witteveen. Push and rotate: a complete multi-agent pathfinding algorithm. *Journal of Artificial Intelligence Research*, 51:443–492, 2014.

[20] Dji. Dji mavic 2 enterprice, 2023. URL https://www.dji.com/nl/mavic-2-enterprise. Accessed from https://www.dji.com/nl/mavic-2-enterprise on 02-05-2023.

[21] DJI. Matrice 600 specs, 2024. URL https://www.dji.com/nl/matrice600. Accessed on 28-03-2024.

[22] European Union Aviation Safety Agency (EASA). Study on the societal acceptance of urban air mobility in europe. Technical report, EASA, 03 2021.

[23] European Union Aviation Safety Agency (EASA). Easy access rules for unmanned aircraft systems. Technical report, EASA, 09 2022.

[24] Margaret Eichleay, Emily Evens, Kayla Stankevitz, and Caleb Parker. Using the unmanned aerial vehicle delivery decision tool to consider transporting medical supplies via drone. *Global Health: Science and Practice*, 7:500–506, 2019.

[25] E. Erdem, D. Kisa, U. Oztok, and P. Schüller. A general formal framework for pathfinding problems with multiple agents. *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013*, 07 2013.

[26] H. Eskandaripour and E. Boldsaikhan. Last-mile drone delivery: Past, present, and future. *Drones*, 7 (2), 2023. ISSN 2504-446X.

[27] Federal Aviation Administration. Unmanned aircraft system traffic management: Concept of operations v2.0. Technical report, FAA, 03 2020.

[28] A. Felner, R. Stern, S. E. Shimony, E. Boyarski, M. Goldenberg, G. Sharon, N.R. Sturtevant, G. Wagner, and P. Surynek. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *Symposium on Combinatorial Search*, 2021.

[29] C. Ferner, G. Wagner, and H. Choset. Odrm* optimal multirobot path planning in low dimensional search spaces. In *2013 IEEE International Conference on Robotics and Automation*, pages 3854–3859, 2013.

[30] B. Fletcher. Wingcopter 198 specifications, 2021. URL https://www.fiercewireless.com/5g/t-mobile-s-5g-off-to-drone-races. Accessed on 09-05-2023.

[31] D. Foead, A. Ghifari, M.B. Kusuma, N. Hanafiah, and E. Gunawan. A systematic literature review of a* pathfinding. *Procedia Computer Science*, 179:507–514, 2021. ISSN 1877-0509. 5th International Conference on Computer Science and Computational Intelligence 2020.

[32] J. Foramitti. Agentpy: A package for agent-based modeling in python. *Journal of Open Source Software*, 6(62):3065, 2021.

[33] Gemeente Rotterdam. Drones, 2023. URL https://www.rotterdam.nl/drones. Accessed on 12-06-2023.

[34] D. Gilon, A. Felner, and R. Stern. Dynamic potential searcha new bounded suboptimal search. In *Proceedings of the International Symposium on Combinatorial Search*, volume 7, pages 36–44, 2016.

[35] M. Goldenberg, A. Felner, R. Stern, G. Sharon, N. Sturtevant, R.C. Holte, and J. Schaeffer. Enhanced partial expansion a*. *J. Artif. Int. Res.*, 50(1):141187, may 2014. ISSN 1076-9757.

[36] Google Earth. Map showing rotterdam, 2023. URL earth.google.com/web/. Accessed on 12-06-2023.

[37] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4): 12–18, 2008.

[38] Health Strategies Consultancy LLC. Follow the pill: Understanding the us commercial pharmaceutical supply chain, 2005.

[39] K. Heineke, B. Kloss, and R. Riedel. The future of air mobility: Electric aircraft and flying taxis, 2023. URL https://www.mckinsey.com/featured-insights/the-next-normal/air-taxis. Accessed on 25-04-2023.

[40] F. Ho, A. Goncalves, A. Salta, M. Cavazza, R. Geraldes, and H. Prendinger. Multi-agent path finding for uav traffic management: Robotics track. *Autonomous Agents and Multiagent Systems 2019*, 2019.

[41] F. Ho, R. Geraldes, A. Gonçalves, B. Rigault, B. Sportich, D. Kubo, M. Cavazza, and H. Prendinger. Decentralized multi-agent path finding for uav traffic management. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):997–1008, 2020.

[42] F. Ho, A. Gonçalves, B. Rigault, R. Geraldes, A. Chicharo, M. Cavazza, and H. Prendinger. Multi-agent path finding in unmanned aircraft system traffic management with scheduling and speed variation. *IEEE Intelligent Transportation Systems Magazine*, 14(5):8–21, 2021.

[43] J. Hoekstra and J. Ellerbroek. Bluesky atc simulator project: an open data and open source approach. *International Conference for Research on Air Transportation*, 06 2016.

[44] W. Hoenig, T. K. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig. Multi-agent path finding with kinematic constraints. *Proceedings of the International Conference on Automated Planning and Scheduling*, 26(1):477–485, Mar. 2016.

[45] J. Holden and N. Goel. Uber elevate: Fast-forwarding to a future of on-demand urban air transportation. Technical report, Uber, 10 2016.

[46] M. Husár, J. Švancara, P. Obermeier, R. Barták, and T. Schaub. Reduction-based solving of multi-agent pathfinding on large maps using graph pruning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, page 624632, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450392136.

[47] K. Jacobs, S. Warner, M. Rietra, L. Mazza, J. Buvat, A. Khadikar, S. Cherian, and Y. Khemka. The last-mile delivery challenge: Giving retail and consumer product customers a superior delivery experience without impacting profitability. Technical report, Capgemini, 01 2019.

[48] Joby Aviation. Electric aerial ridesharing, 2023. URL https://www.jobyaviation.com/. Accessed on 25-04-2023.

[49] Sudip Karki and Hari Sagar Ranjitkar. Comparison of a*, euclidean and manhattan distance using influence map in ms. pac-man, 2016.

[50] M. Khorshid, R. Holte, and N. Sturtevant. A polynomial-time algorithm for non-optimal multi-agent pathfinding. In *Proceedings of the International Symposium on Combinatorial Search*, volume 2, pages 76–83, 2011.

[51] S. Kilkis, P.S. Prakasha, N. Naeem, and B. Nagel. A python modelling and simulation toolkit for rapid development of system of systems inverse design (sosid) case studies. In *AIAA Aviation 2021 Forum*, page 3000, 2021.

[52] J. Kim and E. Atkins. Airspace geofencing and flight planning for low-altitude, urban, small unmanned aircraft systems. *Applied Sciences*, 12(2), 2022. ISSN 2076-3417.

[53] P. Kitjacharoenchai, B. Min, and S. Lee. Two echelon vehicle routing problem with drones in last mile delivery. *International Journal of Production Economics*, 225:107598, 2020. ISSN 0925-5273.

[54] J. Koetsier. Wing drone delivery in 2024: capable of handling tens of millions of deliveries for millions of consumers, March 2023. URL https://www.forbes.com/sites/johnkoetsier/2023/03/11/wing-drone-delivery-in-2024-capable-of-handling-tens-of-millions-of-deliveries-for-millions-of-consumers/. Accessed on 25-04-2023.

[55] K. Korosec. Zipline is now the national drone service provider for rwanda, December 2022. URL https://techcrunch.com/2022/12/15/zipline-is-now-the-national-drone-service-provider-for-rwanda/. Accessed from on 25-04-2023.

[56] D. Kritchanchai and W. Meesamut. Developing inventory management in hospital. *International Journal of Supply Chain Management*, 4(2):11–19, 2015.

[57] J. Li, P. Surynek, A. Felner, H. Ma, T. K. Kumar, and S. Koenig. Multi-agent path finding for large agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7627–7634, Jul. 2019.

[58] Lilium GmbH. Introducing the first electric vehicle take-off and landing jet, 2023. URL https://lilium.com/jet. Accessed on 25-04-2023.

[59] X. Liu, Y. Liu, Y. Chen, and L. Hanzo. Trajectory design and power control for multi-uav assisted wireless networks: A machine learning approach. *IEEE Transactions on Vehicular Technology*, 68(8):7957–7969, 2019.

[60] X. Liu, Y. Su, Y. Wu, and Y. Guo. Multi-conflict-based optimal algorithm for multi-uav cooperative path planning. *Drones*, 7:217, 03 2023.

[61] J. Lloyd and J. Cheyne. The origins of the vaccine cold chain and a glimpse of the future. *Vaccine*, 35 (17):2115–2120, 2017. ISSN 0264-410X. Building Next Generation Immunization Supply Chains.

[62] R. Luna and K.E. Bekris. Efficient and complete centralized multi-robot path planning. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3268–3275, 2011.

[63] H. Ma. Graph-based multi-robot path finding and planning. *Current Robotics Reports*, 3:1–8, 09 2022.

[64] H. Ma, G. Wagner, A. Felner, J. Li, T.K. Kumar, Satish, and S. Koenig. Multi-agent path finding with deadlines: Preliminary results. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 20042006, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[65] H. Ma, D. Harabor, P.J. Stuckey, J. Li, and S. Koenig. Searching with consistent prioritization for multi-agent path finding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7643–7650, Jul. 2019.

[66] D. Masad and J. Kazil. Mesa: an agent-based modeling framework. In *14th PYTHON in Science Conference*, volume 2015, pages 53–60. Citeseer, 2015.

[67] S.A.H. Mohsan, M. Khan, F. Noor, I. Ullah, and M. Alsharif. Towards the unmanned aerial vehicles (uavs): A comprehensive review. *Drones*, 6, 06 2022.

[68] A. Morfin Veytia, C.A. Badea, J. Ellerbroek, J. Hoekstra, N. Patrinopoulou, I. Daramouskas, V. Lappas, V. Kostopoulos, A. Vidosavljevic, J. van Ham, E. Sunil, P. Alonso, J. Terrazas, D. Bereziat, A. Vidosavljevic, and L. Sedov. Metropolis ii: Benefits of centralised separation management in high-density urban airspace. *SESAR Innovation Days*, 12 2022.

[69] M. Moshref-Javadi, A. Hemmati, and M. Winkenbach. A truck and drones model for last-mile delivery: A mathematical model and heuristic approach. *Applied Mathematical Modelling*, 80:290–318, 2020. ISSN 0307-904X.

[70] V. Narayanan, M. Phillips, and M. Likhachev. Anytime safe interval path planning for dynamic environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4708–4715. IEEE, 2012.

[71] Port of Rotterdam. Drone port of rotterdam: U-space airspace prototype; whitepaper. Technical report, Port of Rotterdam, 2023.

[72] "Port of Rotterdam". Eerste drone-vertiport in nederland in gebruik genomen, 05 2023. URL https://www.portofrotterdam.com/nl/nieuws-en-persberichten/eerste-drone-vertiport-in-nederland-in-gebruik-genomen. Accessed on 13-06-2023.

[73] M. Pachayappan and B. Sundarakani. Drone delivery logistics model for on-demand hyperlocal market. *International Journal of Logistics Research and Applications*, 0(0):1–33, 2022.

[74] D. Paresh. Amazons drone delivery dream is crashing, 2023. URL https://www.wired.com/story/crashes-and-layoffs-plague-amazons-drone-delivery-pilot/. Accessed on 27-05-2023.

[75] M. Phillips and M. Likhachev. Sipp: Safe interval path planning for dynamic environments. In *2011 IEEE international conference on robotics and automation*, pages 5628–5635. IEEE, 2011.

[76] I. Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3):193–204, 1970. ISSN 0004-3702.

[77] PostNL. Bezorging van medische goederen aan huis, 2023. URL https://www.postnl.nl/zakelijke-oplossingen/health/medische-goederen-aan-huis/. Accessed on 11-05-2023.

[78] QGIS Development Team. *QGIS Geographic Information System*. Open Source Geospatial Foundation, 2009. URL http://qgis.osgeo.org.

[79] H. Qie, D. Shi, T. Shen, X. Xu, Y. Li, and L. Wang. Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning. *IEEE Access*, 7:146264–146272, 2019.

[80] B. Rabta, C. Wankmüller, and G. Reiner. A drone fleet model for last-mile distribution in disaster relief operations. *International Journal of Disaster Risk Reduction*, 28:107–112, 2018. ISSN 2212-4209.

[81] M. Ribeiro, J. Ellerbroek, and J. Hoekstra. Review of conflict resolution methods for manned and unmanned aviation. *Aerospace*, 7(6), 2020. ISSN 2226-4310.

[82] Rijksoverheid and LVNL. Official no-fly zones provided by the dutch central government, 2023. URL https://map.godrone.nl/. Accessed on 12-06-2023.

[83] Q. Sajid, R. Luna, and K. Bekris. Multi-agent pathfinding with simultaneous execution of single-agent primitives. In *Proceedings of the International Symposium on Combinatorial Search*, volume 3, pages 88–96, 2012.

[84] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T.K. Kumar, S. Koenig, and H. Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019.

[85] O. Schneider, S. Kern, F. Knabe, I. Gerdes, D. Delahaye, A. Vidosavljevic, P. van Leeuwen, D. Nieuwenhuisen, E. Sunil, J. Hoekstra, and J. Ellerbroek. Metropolis urban airspace design. Technical report, DLR, 10 2014.

[86] S.H. Semnani, H. Liu, M. Everett, A. de Ruiter, and Jonathan P. How. Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3221–3226, 2020.

[87] X. Shang, G. Zhang, B. Jia, and M. Almanaseer. The healthcare supply location-inventory-routing problem: A robust approach. *Transportation Research Part E: Logistics and Transportation Review*, 158: 102588, 2022. ISSN 1366-5545.

[88] G. Sharon, R. Stern, M. Goldenberg, and A. Felner. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 195:662–667, 01 2011.

[89] G. Sharon, R. Stern, A. Felner, and N. Sturtevant. Meta-agent conflict-based search for optimal multi-agent path finding. In *Proceedings of the 5th Annual Symposium on Combinatorial Search, SoCS 2012*, Proceedings of the 5th Annual Symposium on Combinatorial Search, SoCS 2012, pages 97–104, December 2012. ISBN 9781577355847. 5th International Symposium on Combinatorial Search, SoCS 2012 ; Conference date: 19-07-2012 Through 21-07-2012.

[90] G. Sharon, R. Stern, A. Felner, and N.R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015. ISSN 0004-3702.

[91] D. Silver. Cooperative pathfinding. *Proceedings of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference, AIIDE 2005*, pages 117–122, 01 2005.

[92] D. Silver. Cooperative pathfinding. In *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, volume 1, pages 117–122, 2005.

[93] T. Standley. *Finding Optimal Solutions to Cooperative Pathfinding Problems*, page 173178. AAAI'10. AAAI Press, 2010.

[94] T. Standley and R. Korf. Complete algorithms for cooperative pathfinding problems. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, page 668673. AAAI Press, 2011. ISBN 9781577355137.

[95] S.H.W. Stanger, N. Yates, R. Wilding, and S. Cotton. Blood inventory management: Hospital best practice. *Transfusion Medicine Reviews*, 26(2):153–163, 2012. ISSN 0887-7963.

[96] R. Stern. *Multi-Agent Path Finding – An Overview*, pages 96–115. Springer International Publishing, Cham, 2019. ISBN 978-3-030-33274-7.

[97] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, E. Boyarski, and R. Barták. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, pages 151–159, 06 2019.

[98] E. Sunil, J. Hoekstra, J. Ellerbroek, F. Bussink, D. Nieuwenhuisen, A. Vidosavljevic, and S. Kern. Metropolis: Relating airspace structure and capacity for extreme traffic densities. In *11th USA/EUROPE Air Traffic Management R&D Seminar*, 06 2015.

[99] P. Surynek. A novel approach to path planning for multiple robots in bi-connected graphs. In *2009 IEEE International Conference on Robotics and Automation*, pages 3613–3619, 2009.

[100] P. Surynek. Towards optimal cooperative path planning in hard setups through satisfiability solving. In *PRICAI 2012: Trends in Artificial Intelligence*, pages 564–576, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[101] J. Thayer and W. Ruml. Faster than weighted a*: An optimistic approach to bounded suboptimal search. *ICAPS 2008 - Proceedings of the 18th International Conference on Automated Planning and Scheduling*, pages 355–362, 01 2008.

[102] J. Thayer and W. Ruml. Bounded suboptimal search: A direct approach using inadmissible estimates. *IJCAI International Joint Conference on Artificial Intelligence*, pages 674–679, 01 2011.

[103] S. Tisue and U. Wilensky. Center for connected learning and computer-based modeling northwestern university, evanston, illinois. *NetLogo: A Simple Environment for Modeling Complexity, Citeseer*, 1999.

[104] H. Ullah, N.G. Nair, A. Moore, C. Nugent, P. Muschamp, and M. Cuevas. 5g communication: An overview of vehicle-to-everything, drones, and healthcare use-cases. *IEEE Access*, 7:37251–37268, 2019.

[105] UPS. Ups operates first ever u.s. drone covid-19 vaccine delivery, 2023. URL https://about.ups.com/us/en/our-stories/innovation-driven/drone-covid-vaccine-deliveries.htmls. Accessed on 25-04-2023.

[106] Velomedi. Bezorgservice voor apotheken, 2023. URL https://velomedi.nl/. Accessed on 11-05-2023.

[107] Verizon. 5g-powered drone monitoring, 2023. URL https://www.verizon.com/business/resources/5g/5g-business-use-cases/autonomous-machines/drone-monitoring/. Accessed on 09-05-2023.

[108] Volocopter GmbH. Volocity: The air taxi that's a cut above, 2023. URL https://www.volocopter.com/solutions/volocity/. Accessed on 25-04-2023.

[109] G. Wagner and H. Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *Proceedings of (IROS) IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3260 – 3267, September 2011.

[110] G. Wagner and H. Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015. ISSN 0004-3702.

[111] T.T. Walker, N.R. Sturtevant, and A. Felner. Extended increasing cost tree search for non-unit cost domains. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 534–540. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[112] B. Wang, V. Hess, and A. Crooks. Mesa-geo: A gis extension for the mesa agent-based modeling framework in python. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on GeoSpatial Simulation*, GeoSim '22, page 110, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450395373.

[113] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo. Multi-agent deep reinforcement learning-based trajectory planning for multi-uav assisted mobile edge computing. *IEEE Transactions on Cognitive Communications and Networking*, 7(1):73–84, 2021.

[114] X. Wang, Z. Yan, and L. Zhong. Centralized and decentralized methods for multi-robot safe navigation. In *2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, pages 150–159. IEEE, 2022.

[115] J. Weise, S. Mai, H. Zille, and S. Mostaghim. On the scalable multi-objective multi-agent pathfinding problem. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020.

[116] Wing Aviation LLC. Better delivery., 2023. URL https://wing.com. Accessed on 25-04-2023.

[117] Wingcopter. Wingcopter 178 specifications, 2021. URL https://wingcopter.com/wp-content/uploads/2021/02/Technical-Details-Wingcopter-178-Heavy-Lift-A-Delivery-Variant-1-1.pdf. Accessed on 02-05-2023.

[118] Wingcopter. Wingcopter 198 specifications, 2023. URL https://wingcopter.com/wingcopter-198#specs. Accessed on 02-05-2023.

[119] K. Yakovlev and A. Andreychuk. Any-angle pathfinding for multiple agents based on sipp algorithm. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27, pages 586–594, 2017.

[120] G. Yang, X. Lin, Y. Li, H. Cui, M. Xu, D. Wu, H. Rydén, and S. Redhwan. A telecom perspective on the internet of drones: From lte-advanced to 5g. *ArXiv*, 03 2018.

[121] J. Yu and S.M. LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177, 2016.

[122] Y. Zeng, Q. Wu, and R. Zhang. Accessing from the sky: A tutorial on uav communications for 5g and beyond. *Proceedings of the IEEE*, 107(12):2327–2375, 2019.

[123] Zipline. Welcome to the best delivery experience not on earth., 2023. URL https://www.flyzipline.com. Accessed on 25-04-2023.

[124] Zorgkaart Nederland. Zorgaanbieders in rotterdam, 2023. URL https://www.zorgkaartnederland.nl/. Accessed on 08-06-2023.