Scalable Conflict Detection and Resolution Methods for Safe Unmanned Aircraft Systems Traffic Management

by

Florence HO

Dissertation

submitted to the Department of Informatics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

The Graduate University for Advanced Studies, SOKENDAI
September 2020

Contents

Summary 4
Acknowledgements 5

Introduction **6**
Context 6
Research Objectives 7
Thesis Structure 8
Research Contributions 10

1 In-Flight Conflict Detection and Resolution **13**
1.1 Introduction 13
1.2 Related Works on In-Flight Conflict Detection and Resolution . . 15
1.3 Background: Optimal Reciprocal Collision Avoidance (ORCA) . 16
1.4 Problem Definition 19
1.4.1 Task Allocation 19
1.4.2 Conflict Detection and Resolution 20
1.5 Approach 21
1.5.1 Preparatory Phase: Task Allocation and Flight Path Generation 22
1.5.2 In-Flight Phase: Conflict Detection and Resolution (CDR) 23
1.5.3 Conflict Detection in Adapted ORCA 24
1.5.4 Conflict Resolution in Adapted ORCA 26
1.5.5 Empirical Tuning of Parameters in Adapted ORCA . . . 29
1.6 Simulation Scenarios 29
1.6.1 Super-Conflict Scenario 29
1.6.2 Real-world delivery Scenario 30
1.7 Experimental Results 32
1.7.1 Super-Conflict Scenario 32
1.7.2 Real-world Delivery Scenario 33
1.7.3 Scalability 36
1.8 Conclusions 36

Future Works for In-Flight CDR **37**

2 Centralized Pre-Flight Conflict Detection and Resolution 38
2.1 Introduction . . . 39
2.2 Related Works on Multi-Agent Path Finding (MAPF) . . . 41
2.3 Problem Formulation . . . 42
2.3.1 UTM Application Context . . . 42
2.3.2 Problem Definition . . . 42
2.4 Background: Conflict Based Search (CBS) . . . 44
2.5 Extension of CBS and ECBS for UTM Context . . . 46
2.5.1 Spatio-Temporal Pruning . . . 46
2.5.2 Extension to Heterogeneous Agents . . . 46
2.5.3 Incorporating Geometrical Computations . . . 46
2.5.4 Reformulation of Low Level Search . . . 48
2.6 Batch Processing . . . 50
2.7 Experiments . . . 51
2.7.1 Results for Adaptation to Heterogeneous Case . . . 51
2.7.2 Results for Comparing Different Approaches to the Processing of UAS Operations . . . 52
2.7.3 Scalability . . . 57
2.8 Conclusions . . . 57

3 Pre-Flight Conflict Detection and Resolution for UAV Integration in Shared Airspace: Sendai 2030 Model Case 58
3.1 Introduction . . . 58
3.2 Related Works . . . 61
3.2.1 Conflict Detection and Resolution Methods and Metrics . 61
3.2.2 Multi-Agent Path Finding . . . 62
3.3 The Need for Advanced Pre-Flight Conflict Detection and Resolution Methods . . . 62
3.3.1 Sendai 2030 Model Case . . . 63
3.3.2 Map and UAV Representation . . . 64
3.3.3 CDR Assessment . . . 65
3.3.4 Trajectory Based Operations Concept . . . 67
3.4 Pre-Flight CDR as a MAPF Problem . . . 68
3.4.1 Pre-Flight CDR Problem Model . . . 68
3.4.2 Ongoing Processing of UAS Operation Requests . . . 70
3.5 UAV Traffic Topology Analysis . . . 71
3.6 Air Traffic Complexity Metrics . . . 74
3.6.1 Number of UAVs in the Air . . . 74
3.6.2 Proximity Measures . . . 74
3.6.3 Number of Conflicts and Modified Trajectories . . . 75
3.7 Simulation Experiments . . . 75
3.7.1 Results for Air Traffic Complexity Metrics . . . 76
3.8 Conclusions . . . 79

4 Decentralized Pre-Flight Conflict Detection and Resolution 80
4.1 Introduction . . . 80
4.2 Related Works . . . 82
4.2.1 Multi-Agent Path Finding for Self-Interested Agents . . . 82
4.2.2 Automated Negotiation . . . 83
4.3 Problem Formulation . . . 84
4.3.1 Motivation . . . 84
4.3.2 Definitions . . . 85
4.3.3 UASSP Cost Function . . . 86
4.3.4 Fairness . . . 86
4.3.5 Optimality . . . 87
4.4 Pre-Processing Phase . . . 87
4.5 Decentralized Approaches . . . 88
4.5.1 Prioritization Approach . . . 89
4.5.2 Negotiation Approach . . . 90
4.5.3 Generalization to Multilateral Case . . . 93
4.6 Experiments . . . 95
4.6.1 Model Case Scenario . . . 95
4.6.2 Experimental Results . . . 97
4.7 Conclusions . . . 101

5 Scheduling in Pre-Flight Conflict Detection and Resolution 103
5.1 Introduction . . . 103
5.2 Related Works . . . 105
5.2.1 Multi-Agent Path Finding . . . 105
5.2.2 Traffic Scheduling . . . 105
5.3 Problem Formulation . . . 106
5.4 Approach . . . 107
5.4.1 Baseline Approaches . . . 107
5.4.2 Combination Approaches . . . 108
5.5 Experiments . . . 111
5.5.1 Model Case Scenario . . . 111
5.5.2 Experimental Results . . . 113
5.6 Conclusions . . . 114

Future Works for Pre-Flight CDR 115

Final Conclusions 116

Bibliography 119

List of Publications 128
Journal Papers . . . 128
Conference Papers . . . 128

Summary

The increasing demand for services performed by Unmanned Aerial Vehicles (UAVs), also called "drones", requires the conception of an Unmanned Aircraft System Traffic Management (UTM) system to ensure the safety and the efficiency of UAV operations in shared low altitude airspace. For this purpose, Conflict Detection and Resolution (CDR) methods that will resolve conflicts, i.e. predicted collisions, between UAVs need to be designed and developed.

In particular, the conception and deployment of a UTM system has recently prompted research on a multi-layered architecture. Two main layers can be distinguished: Pre-Flight CDR whereby conflicts are solved before UAVs takeoff by processing their submitted flight paths beforehand, and In-Flight CDR whereby conflicts are solved in real time when UAVs are flying, in case of unexpected events. In this research, we explore the conception and development of methods for these two layers.

First, we propose a scalable In-Flight CDR method based on Optimal Reciprocal Collision Avoidance (ORCA). ORCA is a state-of-the-art collision avoidance algorithm mainly used in a limited theoretical scope, for pedestrian simulations and ground robots. Thus, it does not address practical considerations that are necessary to the deployment of UAVs in shared airspace, such as navigation inaccuracies, communication overhead, and flight phases. Therefore, we extended the method to In-Flight CDR for UAVs, by including an advanced conflict detection mechanism to improve ORCA's scalability, and uncertainty parameters to address navigation inaccuracies.

Second, we propose a scalable Pre-Flight CDR method based on Enhanced Conflict-Based Search (ECBS). We address the Pre-Flight CDR problem by introducing a mapping to Multi-Agent Path Finding (MAPF), and we extend state-of-the-art MAPF algorithms to take into account the practical requirements of the UTM context. Different from existing approaches, we introduced a novel mapping to the UAVs case, that considers 3D space, heterogeneous agents with different sizes and speeds, and ongoing processing of UAV operations requests. We proposed a new method for conflict detection based on geometrical computations to address heterogeneous agents and improve the scalability of our method. Moreover, we addressed the ongoing processing of operation requests by proposing a "batch" processing based on ECBS and a First-Come First-Served (FCFS) processing based on the Cooperative A* algorithm. We also propose several extensions of our Pre-Flight CDR method to address UTM practical requirements such as a decentralized CDR using negotiation, and an efficient combination of re-planning and scheduling methods.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Prof. Helmut Prendinger for this unique opportunity.

Then, I thank Prof. Marc Cavazza whose valuable insights and suggestions supported this research.

I also thank Prof. Katsumi Inoue and Prof. Ken Hayami for their advice.

Many thanks to the lab members Ruben, Artur, and Bastien who supported this work as well.

I also want to thank the intern students who came to the lab and offered good companionship to me in boring times. Special thanks to Ana and Benjamin who dedicated their internship to assisting this research.

Finally, I thank my family for helping me reach this point in my life.

Last but not least, my experience living in Japan was certainly wonderful and I enjoyed living every day to the fullest. I would like to thank all the nice Japanese people that I met.

This research is the result of hard work associated with determination to learn and achieve the development of solutions. I am proud of what I was able to achieve and that I was able to overcome all the challenges and obstacles until now with what I had.

To conclude, this PhD experience allowed me to express my creativity and make use of my intuition in this great setting that is Japan. I hope to achieve even greater success and continue on making impactful contributions to the society in the near future!

Introduction

Context

The use of Unmanned Aerial Vehicles (UAVs) also called Unmanned Aircraft Systems (UASs), or simply "drones" to perform varied services such as goods delivery, surveillance, search and rescue and so on as shown in Fig.1, has been rapidly expanding in the last few years. Thus, the low-altitude air traffic is expected to grow significantly [44, 49]. Any 'conflict' [51], i.e. possibility of collision between UAVs, must be avoided. Several independent UAS Service Providers (UASSPs) will support UAS Operators to task UAVs with limited capacities. Therefore, the conception of an Unmanned Aircraft Systems Traffic Management (UTM) system [48, 49] is important to ensure the safe integration of UAVs in the low altitude airspace.

Unlike Air Traffic Management (ATM), the design of a comprehensive Unmanned Aircraft System Traffic Management (UTM) system for the deployment of UAVs in low altitude airspace is still under discussion. In particular, the conception and development of Conflict Detection and Resolution (CDR) methods is key for the integration of UAVs in shared airspace. The objective in the conception of these methods is (1) to enable safety by ensuring UAVs do not collide, i.e. resolving all conflicts, and (2) to provide efficiency in the determination of UAV flight paths such as scalability and optimality.

In recent years, NASA has initiated the design and conception of UTM [49, 70], and several baseline concepts have emerged. There is the common understanding that UTM will be composed of a multi-layered architecture [44] similar to Air Traffic Management (ATM) [51]. In particular, the International Civil Aviation Organization's (ICAO) Global Air Traffic Management Operational Concept [39] defines different conflict management or 'redundancy' layers.

Figure 1: Different UAVs applications.

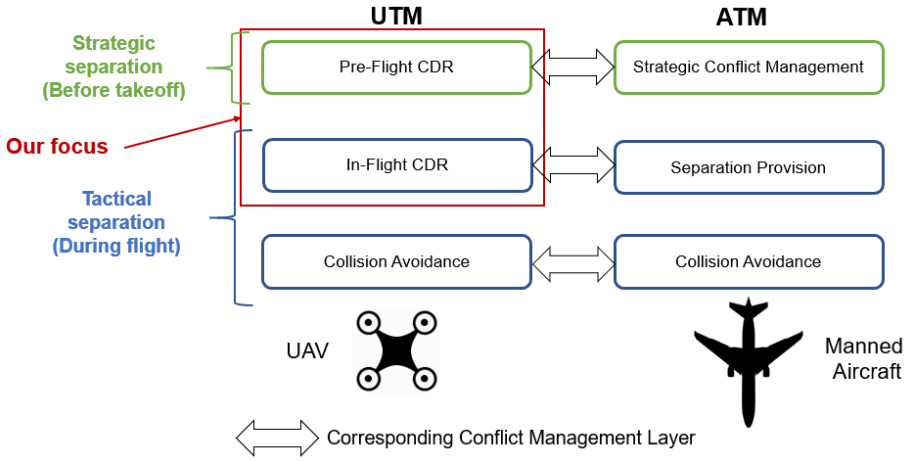


Figure 2: Conflict management layers in UTM and ATM.

These layers can be separated in three categories (see Fig. 2). Each of these distinct layers are applied during the different phases of a UAV operation processing, before the UAV takes off (strategic separation), and then during its actual flight (tactical separation) [39]:

- Pre-Flight CDR methods aim to provide conflict-free paths for all UAVs before their actual takeoff.
- In-Flight CDR methods ensure separation provision for the flight paths of all UAVs during flight, to account for dynamic events such as bad weather, emergency operations, and so on.
- Collision Avoidance methods, such as Sense and Avoid, are considered as a final fail-safe and are based on sensing technology onboard a UAV.

Research Objectives

In this thesis, we propose to address the different challenges of defining and conceiving efficient CDR methods for the In-Flight and Pre-Flight phases with respect to the requirements of the UTM context.

The originality of our work resides in the combination between existing theoretical models, algorithms that are used in limited settings and the practical requirements and concepts of the novel real world domain that is UTM. This novel combination leads to several extensions of existing algorithms and frameworks in the multi-agents systems domain.

Our challenge is thus to bridge the gap between the existing frameworks in the AI community and the novel real world domain of UTM. Therefore, our

Figure 3: A quadcopter UAV as considered in our work.

Property	**Pre-Flight CDR**	**In-Flight CDR**
Time of processing	Off-line	Online
UAV flight path consideration	Entire flight paths of all given UAVs	Portions of flight paths of the UAVs in conflict
Optimality	Global	Local
Resolution method (Baseline algorithm)	Multi-Agent Path Finding Algorithm (CBS)	Reactive Collision Avoidance Algorithm (ORCA)

Table 1: Comparison of Pre-Flight CDR and In-Flight CDR properties.

research objective is to conceive and evaluate efficient CDR methods that satisfy the practical requirements of the UTM domain.

We propose approaches that are:

- *Safe*: they ensure that no collision occurs between UAVs in shared low altitude airspace.
- *Scalable*: they can handle efficiently the increase in the number of UAVs considered, and for instance their performance increases linearly with the number of UAVs, while providing solutions that aim to minimize a defined objective, such as flight paths costs.

In this work, we assume that all UAVs are *quadcopters* as in Fig. 3, thus holonomic agents.

Thesis Structure

Pre-Flight CDR and In-Flight CDR are complementary methods as defined by ICAO's conflict management layers shown in Fig. 2. So they are two distinct processes with different properties as outlined in Table 1:

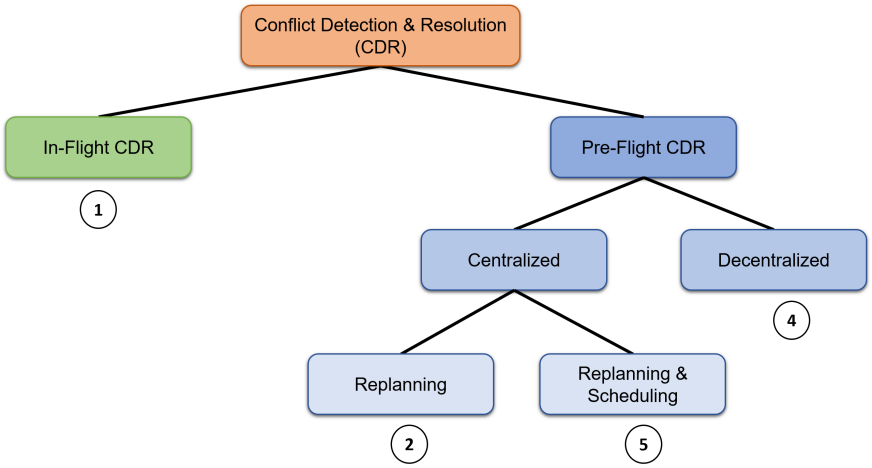


Figure 4: Structure of the dissertation with the distinction and relationships of the different parts. The In-Flight CDR and Pre-Flight CDR branches represent two distinct types of CDR methods with different properties. Chapter 3 is not represented in the tree as it is a broader version of Chapter 2.

- Pre-Flight CDR is applied off-line before UAVs' flight, and then In-Flight CDR is applied online while UAVs are flying.
- Since the Pre-Flight CDR process occurs before UAVs takeoff, it considers all given UAV flight paths and resolves all existing conflicts at once.

  In contrast, since the In-Flight CDR process occurs in real time while UAVs are flying, it considers locally detected conflicts between flying UAVs with predefined look-ahead times, thus locally resolves conflicts.
- Since Pre-Flight CDR allows for longer detection and resolution look-ahead times, it is not time critical and it uses sophisticated resolution capabilities that can prevent and mitigate downstream conflicts. With Pre-Flight CDR methods, it is thus possible to determine globally optimal solutions in terms of costs like delays for each UAV agent.

  In contrast, In-Flight CDR is more time critical as UAVs continuously move while processing. So the conflict-free solutions obtained are locally optimal with respect to costs such as induced delays, as In-Flight CDR focuses on resolving an incoming conflict detected within the look-ahead time window and do not necessarily prevent downstream conflicts from occurring.

Hence, Pre-Flight CDR and In-Flight CDR require different types of resolution methods to satisfy their respective properties. The methods used for In-Flight CDR would not be appropriate for Pre-Flight CDR and vice versa.

Namely, Pre-Flight CDR is conceived as a Multi-Agent Path Finding (MAPF) problem whereby all agents flight paths are considered off-line. On the other hand, In-Flight CDR requires a reactive collision avoidance method that can be applied online while UAVs are flying, with possibly small time window for conflict resolution.

Therefore, this thesis is divided into two distinct research parts, one for each phase as shown in Fig. 4, In-Flight CDR (Chapter 1) and Pre-Flight CDR (Chapter 2 to 5). They have different Related Works, background and baseline approach.

Research Contributions

Our research makes several contributions described in the following chapters:

- In **Chapter 1**, we focus on the conception of a scalable and practical method for In-Flight CDR, which refers to approaches that are applied in real-time to solve conflicts between UAVs that are about to collide during flight. We introduced an adaptation of Optimal Reciprocal Collision Avoidance (ORCA). ORCA is a state-of-the-art collision avoidance algorithm mainly used in a limited theoretical scope, for pedestrian simulations and ground robots. Thus, it does not address practical considerations that are necessary to the deployment of UAVs in shared airspace, such as navigation inaccuracies, communication overhead, and flight phases. Therefore, we extended the method to In-Flight CDR for UAVs, by including an advanced conflict detection mechanism to improve ORCA's scalability, and uncertainty parameters to address navigation inaccuracies. We evaluated our approach through simulations. First, by empirically tuning the considered ORCA parameters, we reduced the generated deviations from the initial flight path. Second, by simulating realistic UAV traffic for delivery, we determined a value for separation distance between UAVs that uses airspace efficiently.

- In **Chapter 2**, we propose a scalable Pre-Flight CDR method based on Enhanced Conflict-Based Search (ECBS). We address the Pre-Flight CDR problem by introducing a mapping to Multi-Agent Path Finding (MAPF), and we extend state-of-the-art MAPF algorithms to take into account the practical requirements of the UTM context. MAPF approaches have mostly been applied to solve comparable problems with ground robots. However, they were tested with simplifying assumptions that do not reflect important characteristics of many real-world domains, such as delivery by UAVs. Therefore, we introduced a novel mapping to the UAVs case, which considers 3D space, heterogeneous agents with different sizes and speeds, and ongoing processing of UAV operation requests. We proposed an efficient method for conflict detection based on geometrical computations to address heterogeneous agents and to improve the scalability of our method. Moreover, we addressed the ongoing processing of operation requests by

proposing a "batch" processing based on ECBS and a First-Come First-Served (FCFS) processing based on the Cooperative A* algorithm. We performed simulations based on a study on UAV usage predicted for 2030 in Sendai in Japan. For peak hour scenarios, batch processing proved more efficient, with a better throughput than FCFS processing. Thus, it is able to meet the requirements of timely and accurate response on delivery requests to users of such UTM services.

- In **Chapter 3**, we focus on a broader topic of the assessment of Pre-Flight CDR concepts within a realistic UAV traffic simulation. Few studies have examined realistic scenarios and requirements for the efficient deployment of a UTM system. In particular, we illustrate our study with the Sendai 2030 model case, a realistic projection of UAV usage for deliveries in one area in Japan. This model case considers up to 21,000 requests for UAV operations over a 13-hour service time, and thus poses a challenge for Pre-Flight CDR methods. Importantly, we demonstrate the necessity for advanced Pre-Flight CDR methods relying on 4DT (3D plus time Trajectories) in future UTM systems, with an empirical comparison of 3D and 4DT airspace reservation methods. We show that the 4DT approach allows a more efficient airspace usage than the 3D approach. Moreover, we propose a characterization and a comprehensive analysis of the UAV traffic topology for deliveries by UAVs and suggest several metrics to understand the complexity of low altitude air traffic in our model case.

- In **Chapter 4**, further for Pre-Flight CDR, we propose an extension of the MAPF model based on negotiation that allows several independent UAS Service Providers (UASSPs) to resolve the existing conflicts between their given UAV operations according to their individual costs in a decentralized way. UASSPs are independent entities, each with their own group of UAVs, and they will provide services, such as de-confliction services, to UAS operators who submit UAV operation (flight path) requests from customers. Recent discussions on UTM suggest that such centralized control might not be practical or desirable, and this may result in an "unfair" distribution of induced costs in terms of delays among UASSPs. In this context, Pre-Flight CDR must support the decentralized resolution of conflicts, whereby self-interested "agents" (UASSPs) communicate with each other to resolve conflicts among their UAV operations, with their own business objectives to maintain a certain service quality or satisfy operational constraints. Therefore, we propose a suitable approach based on a negotiation method to allow several UASSP agents to bargain between them a conflict-free solution for their UAV operations, whereby they agree on their individual costs computed in function of total delays and so on.

- Finally, in **Chapter 5**, we consider another extension for Pre-Flight CDR, whereby we introduce scheduling techniques, i.e. time-based resolution, into the MAPF model to improve the performance of Pre-Flight CDR

methods. The standard MAPF formulation provides limited assumptions whereby UAV agents' start times are fixed, and use a uniform speed on their path. Therefore, we hereby propose to relax those assumptions to improve the efficiency of MAPF solvers in particular in high density structured airspace. We introduce two methods for conflict resolution based on temporal resolution, that are takeoff scheduling and speed scheduling. In this chapter, the ECBS-based Pre-Flight CDR method introduced in the previous chapters is used, so it scales with the number of operations to process. Here, we propose to combine different conflict resolution techniques in an informed manner, unlike standard MAPF solvers that use path replanning by default. For this purpose, we introduced a conflict type distinction into the conflict detection step of ECBS that allows to determine the most efficient technique to be applied depending on the conflict type detected. This extension allows to improve the generated delays of the obtained solutions and to reduce the number of rejected UAV operations.

Each chapter is preceded by an introductory paragraph that explains how the chapter is related to the overall research. A list of publications of the research in this dissertation, in conference chapters and journal chapters can be found after the Bibliography at the end of the dissertation.

Chapter 1

In-Flight Conflict Detection and Resolution

For In-Flight Conflict Detection and Resolution (CDR), we introduce an adaptation of Optimal Reciprocal Collision Avoidance (ORCA), which is a state-of-the-art collision avoidance algorithm hitherto mainly used in a limited theoretical scope, to realistic UAV operations. Our approach, called Adapted ORCA, addresses practical considerations that are inherent to the deployment of UAVs in shared airspace, such as navigation inaccuracies, communication overhead, and flight phases. We validate our approach through simulations. First, by empirically tuning the ORCA parameters look-ahead time window and deconfliction distance, we are able to minimize the ORCA generated deviations from the nominal flight path. Second, by simulating realistic UAV traffic for delivery, we can determine a value for separation distance between UAVs that uses airspace efficiently.

1.1 Introduction

One of the preconditions for the successful real-world deployment of Unmanned Aerial Vehicle (UAV) fleets is the development of a safe and efficient Unmanned Aircraft System (UAS) Traffic Management (UTM) system [49]. In the near future, several independent UAS Service Providers (UASSPs) will task multiple UAVs with limited capacities to visit specific locations, and operate in low-altitude and possibly high-density airspace that is shared among the UASSPs.

For safety, the path of each UAV must avoid *static* obstacles, such as terrain elevation and no-fly zones, and *dynamic* obstacles, such as other UAVs controlled by other service providers. With dynamic obstacles, Conflict Detection and Resolution (CDR) methods become indispensable for safe air traffic. In UTM, and similar in ATM (Air Traffic Management), CDR methods refer to different levels of "redundancy" [44, 48, 49].

- Pre-Flight CDR: here, conflicts are detected and resolved based on flight plans submitted to the UTM, before the actual flights. In ATM, such method is called Strategic Conflict Management.

- In-Flight CDR: because of changing weather conditions, or some emergency, sone conflict-free paths created Pre-Flight CDR might not be safe anymore. Therefore, In-Flight CDR methods are required that adapt UAVs movement in real-time, during the flight. In ATM, such techniques are called Separation Provision.

- Collision Avoidance: those methods provide a final failsafe and might be based on video processing or other sensing technology.

In this chapter, we will focus on In-Flight CDR methods. Please note that those methods might be centralized (as presented here), or decentralized, e.g., running onboard the vehicle.

Additionally, the design and specification of a complete UTM system will require many considerations [44,49], some of which are not yet determined, such as requirements for communication, application of dynamic airspace configuration, and so on.

Our implementation of the UTM system simulator comprises the following main components of a real UTM system.[1]

- UAS Service Provider (UASSP): all UAS Operators use a UASSP for task assignment (incl. path planning) of UAVs given service requirements;

- Core UTM: all UASSPs connect to a central Core UTM that hosts the CDR (Conflict Detection and Resolution) service for In-Flight CDR.

In this chapter, we focus on airspace shared among UAVs from different, independent UAS service providers, whereby all UAVs are assumed to be connected to the UTM via UASSPs. Moreover, in the low-altitude shared airspace, interactions with manned aircraft are limited, and airports are considered as no-fly zones for UAVs, so we do not consider manned aircraft in this setting. We adopt a partly simplified setting with respect to UAV types, and consider only one model of quadcopter for which we take realistic flight parameters into account.

The operational scenarios considered in this chapter are mainly the delivery of goods by UAVs from different UASSPs, including commercial and public services. For that purpose, we present a two-stage approach to UAVs delivery operations in shared airspace [34]. First, in the *preparatory phase*, for all UAVs connected to an UASSP, paths are planned that avoid collisions with static obstacles, such as terrain elevation and no-fly zones. Note that in this step, we do not consider the possibility of conflicts, even within the fleet of UAVs controlled by one UASSP. Second, in the *In-Flight phase*, as a standard approach

[1] This setup is also realistic in terms of current regulatory environment.

in UTM [44], the CDR method modifies trajectories of those UAVs, within or across fleets, that are at risk of collision.

While most of the recent works focus on adapting ORCA's computations to obtain collision-free maneuvers for UAVs, the novelty of our approach is in proposing a practical integration of ORCA as an In-Flight CDR mechanism to the UTM system. The Standard ORCA algorithm is usually applied in theoretical simulations with simplifying assumptions that do not hold in practice, such as instant velocity changes or perfect communication. So, the main challenge of our work is to effectively adapt Standard ORCA to address real-world constraints and thus make ORCA applicable to the UTM context.

We hereby assume that the same In-Flight CDR method is applied by all UAVs.

We also conduct simulations of a real area in Japan with realistic settings regarding the demand for UAVs and delivery tasks.

Specifically, this chapter makes three main contributions:

1. We are the first to adapt Standard ORCA to the practical UTM context, where quadcopter UAVs operate in shared low-altitude airspace. With our new version of ORCA, called Adapted ORCA, we address the communication overheads induced by Standard ORCA, the UAV navigation inaccuracies, and the distinct UAV flight phases, such as take-off and landing.

2. For super-conflict scenarios, we propose an empirical tuning of ORCA parameters, namely the look-ahead time window τ, and the deconfliction distance *dec_dist*, to reduce the path deviation generated by ORCA.

3. We also evaluate the performance of our approach in a series of realistic simulations based on real-world delivery scenarios. With the collected simulation data, we show that our approach ensures zero percent physical collisions.

1.2 Related Works on In-Flight Conflict Detection and Resolution

Kuchar and Yang [51] provide an extensive survey on In-Flight CDR methods for aircrafts and defined the notion of conflict as "an event in the future in which two or more aircrafts will experience a loss of minimum separation between each other" ([51], p. 4). Airspace segregation and corridor concepts are already applied for aircrafts [48] and can also reduce the number of conflicts. However, the possibility of conflicts is still non-negligible in the presence of dynamic events such as wind, tracking errors, or flight delays. Several works centered on UAV deployment have recently emerged, such as the ORCHID project [8, 67], which focuses on optimally assigning tasks to UAVs in a disaster response context. However, their work targets only one single UAS Operator, who is in control of the entire UAV fleet, whereas we focus on an airspace shared among multiple

independent service providers, so that conflicts cannot be ruled out beforehand. [9, 47] and [69] proposed a coupling of task allocation and path planning for UAVs. Yet, they did not address collision avoidance between moving UAVs. Moreover, they only consider a 2D context, thus assuming UAVs keep a constant altitude. By contrast, we take into consideration 3D elements, based on a realistic elevation map.

Regarding UAV collision avoidance methods, several approaches have been proposed. [86] apply a speed change method to solve conflicts but are limited by frontal conflicts and uncertainties. In a similar way, [95, 97] use speed and heading changes restricted to 2D; hence they do not consider any altitude change. In the AgentFly project, [78, 79] propose decentralized algorithms for collision avoidance based on game theory, but they do not consider a realistic UTM setting with task allocation or static obstacles. With priority-based techniques [15, 44], UAVs are ordered into a sequence and planning is done one-by-one, such that each UAV avoids collisions with the higher-priority UAVs. This greedy approach tends to perform well in uncluttered environments, but it is in general incomplete and often fails in dynamic environments. With knowledge-based methods [61], a lookup table of conflict resolution is generated offline with the use of a Markov decision process [61] and then applied online. The generation usually requires large computation time.

Recently, the computation of collision-avoiding velocities with the use of velocity obstacles shows widespread adoption [2, 16, 33, 41, 44, 73]. Among them, Optimal Reciprocal Collision Avoidance (ORCA) [85] is the prominent approach.

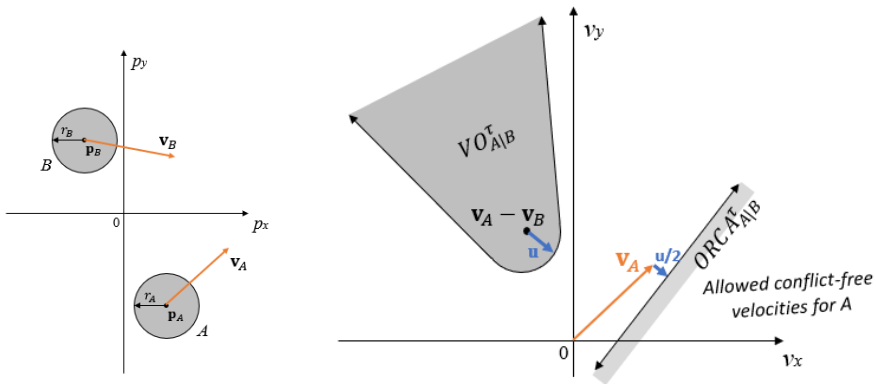
In our UTM context, we focus on developing In-Flight CDR methods for UAVs. ORCA is a method originally applied in simulations of collision avoidance for holonomic agents, such as pedestrians or ground robots. In this chapter, we propose an adaptation of ORCA, so it can be incorporated into a multi-layer CDR approach.

Several other works have recently also applied ORCA to the case of UAVs by adapting its computations. For instance, Alonso-Mora et al. [3] have incorporated motion equations for quadcopters into ORCA's computations to obtain more precise collision-free maneuvers for UAVs. Yet, all these works focus on designing a local collision avoidance method without considering aspects of the UTM context, such as automated UAVs that carry out missions, position uncertainty, or safe separation between UAVs. By contrast, we propose a novel version of ORCA that supports CDR in a realistic UTM context.

1.3 Background: Optimal Reciprocal Collision Avoidance (ORCA)

We hereby present a description of the Optimal Reciprocal Collision Avoidance (ORCA) [85] algorithm on which we base our In-Flight CDR approach.

ORCA relies on the concept of velocity obstacles computed for each agent.



(a) Example with two agents A and B in conflict with respective current velocities, positions, and radii.

(b) ORCA computations for agent A with respect to agent B, the velocity obstacle $VO^{\tau}_{A|B}$ and the half-space of collision-free velocities $ORCA^{\tau}_{A|B}$.

Figure 1.1: Example of ORCA computations of conflict-free velocity.

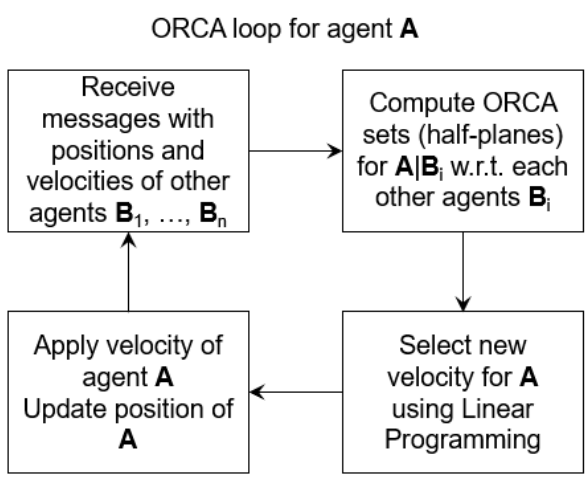


Figure 1.2: ORCA loop for each agent.

The velocity obstacle [27] of an agent induced by a moving obstacle is the set of all velocities that will result in a collision between the given agent and a moving obstacle within some fixed time interval into the future, assuming that the dynamic obstacle maintains a constant velocity. If the agent applies a velocity within the region corresponding to the velocity obstacle, then the agent and the moving obstacle will collide. If the velocity chosen is outside the velocity obstacle, then a collision will be avoided.

Let agent A of current velocity v_A and radius r_A and agent B of current velocity v_B and radius r_B in conflict as shown in Fig. 1.1 (a). Thus, a geometric representation of a velocity obstacle $VO_{A|B}$ for agent A with respect to agent B corresponds to a cone computed from agents A and B radii and positions as shown in Fig. 1.1 (b).

However, the velocity obstacle approach alone does not work well for local collision avoidance within a group of agents where each agent is actively changing its velocity to avoid the other agents, since it assumes that other agents may not change their velocities.

If all agents were to use velocity obstacles to choose a new velocity, there would be oscillations in the motion of the agents between successive time steps. If each agent chooses the new velocity closest to its current velocity, then the agents will automatically pass each other on the same side.

So, rather than choosing velocities closest to their current velocities, in ORCA, the agents must select the velocity closest to their preferred velocity v_A^{pref}, usually the velocity directed from each agent towards its goal position.

Thus, the ORCA algorithm augments the velocity obstacle with a half-plane $ORCA^{\tau}_{A|B}$ for an agent A with respect to an agent B in the time window τ as shown in Fig. 1.1. This half-plane defines a set of allowed velocities that are both collision-free for a fixed time window τ, and will ensure that the motion of the agents will be smooth.

As shown in Fig. 1.1, let u be the vector from the relative velocity $v_A - v_B$ of the agents A and B to the closest point on the boundary of the truncated velocity obstacle for agent A induced by agent B. Let n be the outward normal of the boundary of the velocity obstacle at $v_A - v_B + u$. So u is the smallest change required to the relative velocity of agents A and B to avoid a collision.

Incorporating reciprocity, each agent adjusts its velocity by at least $\lambda \cdot u$ where here $\lambda = \frac{1}{2}$ to avoid the collision in a reciprocal way for each agent.

Therefore, the velocities allowed by ORCA are in a half-plane in the direction of n starting at the point $v_A + \lambda \cdot u$.

So the half-plane $ORCA^{\tau}_{A|B}$ of allowed velocities for agent A is formally defined as:

$$ORCA^{\tau}_{A|B} = \{v|(v - (v_A^{opt} + \lambda \cdot u)) \cdot n \geq 0\} \tag{1.1}$$

Where v_A^{opt} can usually be chosen as v_A the current velocity of agent A.

Then, we consider $ORCA^{\tau}_{A}$ the set of velocities that are allowed for A with respect to *all* other agents $B_1 \dots B_n$. $ORCA^{\tau}_{A}$ is the intersection of the half-planes of permitted velocities induced by each other agent $B_1 \dots B_n$.

Next, the agent selects a new collision-free velocity v_A^{new} for itself that is closest to its preferred velocity v_A^{pref} among all velocities inside the region of permitted velocities:

$$v_A^{new} = \underset{v \in ORCA_A^{\tau}}{\arg\min} \; ||v - v_A^{pref}|| \tag{1.2}$$

The algorithm returns the velocity in $ORCA_A^{\tau}$ that is closest to v_A^{pref} with the use of linear programming to solve Eq. 1.2.

Then, for each agent, ORCA iterates through four steps as indicated in Fig. 1.2:

1. The given agent receives the positions and velocities of all other agents.
2. It computes the ORCA half-planes with respect to each other agents.
3. It then selects the new velocity with the use of linear programming.
4. It applies the new velocity and updates its new position.

This cycle is repeated at each time step of the agent's execution of its assigned path.

1.4 Problem Definition

In this section, we define the overall problem. Our main objective is to ensure safe UAV operations in shared airspace, i.e. no collisions between any UAVs or with static obstacles, and compliance with UTM requirements while providing minimum evasive behavior in case of conflict.

The environment is assumed to be known and the state space is in three dimensions. Hence a location will have three coordinates: latitude, longitude and altitude relative to the elevation from mean sea level. UAVs are allowed to fly between legal bounds that are hereby fixed to $min_{alt} = 30$ meters (m) for the minimum altitude, and $max_{alt} = 150$m for the maximum altitude relative to the elevation of a point of given latitude and longitude coordinates.

Our problem is directly motivated by several practical applications like delivery or surveillance tasks. We assume the dynamic situation where an operator in charge of a given fleet of UAVs continuously receives tasks requests at regular time intervals.

1.4.1 Task Allocation

We consider the sub-problem of task allocation for each UAV fleet. In this sub-problem, the objective is to minimize the total sum of traveled distances for all considered UAVs of the fleet.

For each UAV fleet, we define:

- A set of m UAVs $U = \{1, \ldots, m\}$: each UAV $i \in U$ has an initial location l_i considered as the first take-off point and also the final location, since typically each UAV has to return to its hub once its mission is completed.

- The battery capacity of each UAV $D_i > 0$ which allows the UAV to travel a maximum distance.
- The payload capacity of each UAV $P_i > 0$ for carrying goods.
- A set of n tasks $T = \{T_1, \ldots, T_n\}$: each task $T_j \in T$ has a required payload $p_j \geq 0$ and a given location.
- The cost function we use in the following is defined as the distance to travel from one given task location T_i to the location of another task T_j: $d(T_i, T_j)$.

We denote the path of a UAV i as being the set of p ordered waypoints: $\pi_i = \{(x_1, y_1, z_1), \ldots, (x_p, y_p, z_p)\}$. The altitude constraints are defined as:

$$\begin{aligned} \forall (x_k, y_k, z_k) \in \pi_i, elevation(x_k, y_k) + min_{alt} \leq z_k \\ z_k \leq elevation(x_k, y_k) + max_{alt} \end{aligned} \tag{1.3}$$

The following assumptions apply to the sub-problem:

- Each UAV returns to the same initial location of departure.
- Each task is considered as a destination location and is reached exactly once by a UAV.
- The total distance traveled does not exceed the UAV's battery capacity.
- The payload demand does not exceed a UAV payload capacity.

These are the assumptions used in our task allocation algorithm.

1.4.2 Conflict Detection and Resolution

As previously mentioned, a *conflict* is the predicted loss of minimum separation between two or more aerial vehicles [51].

In the UTM context, similar to the ATM context, no physical collisions are ever acceptable. For that purpose, the actual body radius of each UAV is enlarged by adding extra layers that represent (i) different uncertainty margins, such as navigation system error and flight technical error, and (ii) the minimum separation distance determined as a UTM airspace constraint (see Fig. 1.3).

To accurately determine the individual radii, we introduce an expression that incorporates a first layer that we call the $Constructor$ layer, whose value can be retrieved from UAV manufacturers data (see Eq. 1.4). It includes the actual size of a UAV, $BodyRadius$, and several navigation error factors relating to operational risk as listed in [45], such as the Navigation System Error NSE, which is the difference between the true position of a UAV and its displayed position, the Flight Technical Error FTE, which is the difference between the required flight path and the displayed position of a UAV, and the $SafetyFactor$, an arbitrary safety margin coefficient.

$$Constructor = (BodyRadius + NSE + FTE) \cdot SafetyFactor \tag{1.4}$$

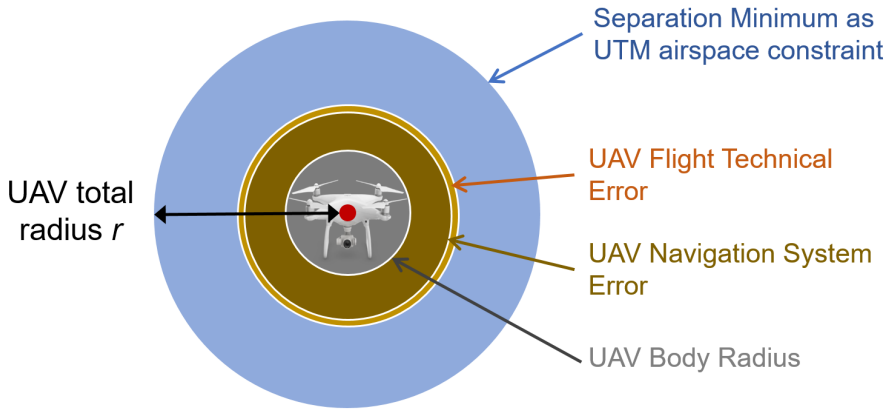


Figure 1.3: Safety layers considered for the determination of the total separation distance.

Then, a second safety layer called $SeparationMinimum$ is added, which is an arbitrary distance whose value is evaluated in this chapter. Such layer is also adopted by the UTM regulation [21, 49]. In the case of quadcopters, which can be considered as holonomic agents, the impact of using a specific PID controller and an adaptive fuzzy method would be minimal and have little influence on our CDR approach.

Finally, the *UAV-specific radius* r is shown in Eq. 1.5.

$$r = Constructor + 0.5 \cdot SeparationMinimum \tag{1.5}$$

Now, a *near-collision* happens when the spheres representing the separation minima of all involved UAVs actually intersect.

Conflict resolution is triggered when a conflict is predicted and has to ensure that there is no loss of minimum separation, i.e., the separation minima of UAVs never intersect.

1.5 Approach

While recent works using UAVs focus on the specific computations of collision-free maneuvers, the novelty of our work is the integration of ORCA as a CDR mechanism in the UTM context, which features special requests that need to be addressed.

We first present our two-stage approach used for UAV operations in shared airspace. An example of the overall pipeline is shown in Fig. 1.4. In the *preparatory phase*, for each fleet of UAVs connected to an UASSP, paths are

generated that avoid collisions with static obstacles, such as terrain elevation or no-fly zones. In this step, we do not consider the possibility of conflicts, even within each fleet of UAVs controlled by an UASSP. Then, in the *In-Flight phase*, as a standard approach in UTM [21,44], our CDR method modifies the nominal trajectories for UAVs of all fleets, to avoid loss of minimum separation.

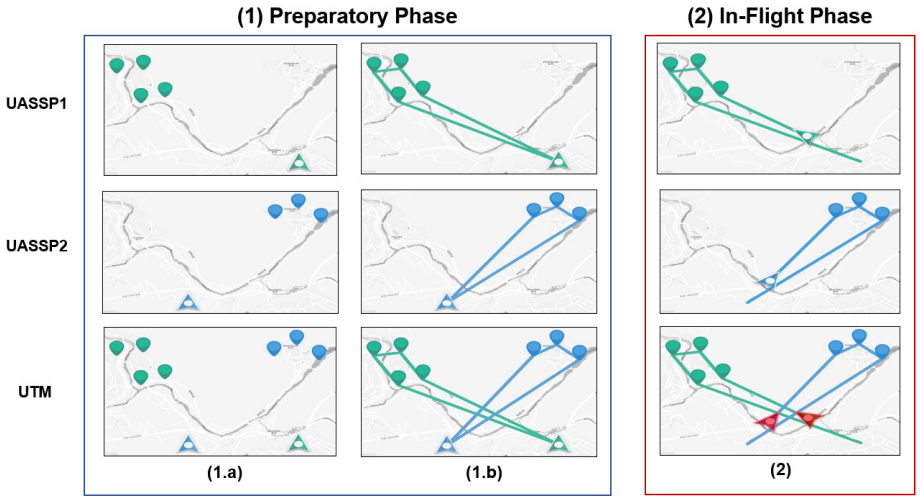


Figure 1.4: Example application of our two-stage approach with 2 UASSPs. *Preparatory Phase*: (1.a) Tasks requests are received by each UASSP at regular time intervals and path costs are precomputed with 2.5D path planning; (1.b) Tasks are allocated and paths are generated for each UAV; *In-Flight Phase*: (2) UAVs start flying in shared airspace and all UAVs transmit their telemetry to the Core UTM system that hosts the In-Flight CDR algorithm that resolves conflicts for all UAVs in the given area.

1.5.1 Preparatory Phase: Task Allocation and Flight Path Generation

We present our *Preparatory* procedure (see Fig. 1.4), which aims at generating a feasible sequence of tasks for each UAV according to a defined cost function.

The simultaneous computation of task allocation and path planning can quickly become intractable even for relatively small problem instances [69]. However, due to scalability considerations, an optimal planner is not necessary. To improve the efficiency of the computations, we first prune unlikely allocations with a soft clustering approach [10]. Such soft clustering is advantageous, as it encodes uncertainties on data, namely the unknown real distance to be computed by path planning, and allows tasks to belong to more than one cluster.

As mentioned in Section 1.2, we use Theta* [59] for any-angle path planning to match with quadcopters motion. For each pair of locations of the same cluster, we compute all paths with Theta* and their associated costs. We use a generated elevation map, thus the search space is comparable to a 2.5D representation and each node in the grid contains its latitude, longitude, and elevation. Also, no-fly zones can be incorporated by putting an infinite cost in the evaluation function. The corresponding waypoints and costs for the paths between every pair of task locations are stored in a lookup table for each UAV.

Finally, we apply Tabu Search (TS), a metaheuristic that efficiently solves large sized optimization problems [30]. TS is an improved version of the Local Search which incorporates a memory of a fixed number of recent moves used, called "tabu list". The size of the tabu list is dynamically adjusted in function of the improvement of the current solution [30]. With the real path costs previously computed by Theta*, TS determines the sequences of tasks to be allocated to each UAV with respect to the constraints presented in Section 1.4.1 and detailed in Algorithm 1.

One important step in the TS metaheuristic is the neighbourhood selection process. This step determines the strategy to guide successively the search process towards an optimum. In our case, a move means assigning a task from one UAV to another, or changing the order of assignment between two tasks assigned to the same UAV.

This layered approach between task allocation and path planning prevents from a combinatorial growth in the number of possible permutations with the number of UAVs and tasks. The result is a set of task allocations and individual flight paths for each UAV, which is compatible with their real-world speed, payload and battery life.

1.5.2 In-Flight Phase: Conflict Detection and Resolution (CDR)

The flight paths associated to task allocations are calculated without any consideration of potential airspace conflicts between UAVs. Hence there is a need to implement suitable In-Flight CDR mechanisms.

In this section, we introduce our CDR process called *Adapted ORCA*, which improves over the existing ORCA algorithm [85] hereby referred to as *Standard ORCA*. Importantly, Adapted ORCA is compatible to real-world UAV deployment. The proposed centralized method can be complemented by decentralized on-board CDR methods as a redundancy mechanism.

We first introduce the ORCA parameter notation in Table 1.1. For a detailed description of the ORCA velocity computations, we refer to [85] .

Standard ORCA has been proven (i) to guarantee collision-free motion for the given time window τ and (ii) to generate a local minimum deviation path [85]. Our proposed CDR method, Adapted ORCA, conserves the theoretical properties of Standard ORCA, while supporting its application to real-world UAV deployment.

Algorithm 1: Task allocation process with Tabu Search

Data: T: tasks ; U: UAVs
Result: $z_{best} \leftarrow (\mathscr{T}_i, \Pi_i)_{\forall i \in U}$: solution as a sequence of tasks and associated paths for each UAV i
$(C_i)_{\forall i \in U} \leftarrow FuzzyClustering(U, T, \gamma)$; `// Cluster of tasks locations for UAV i`
$M \leftarrow PathPlanning((C_i)_{\forall i \in U})$; `// Lookup table containing all costs and paths computed by Theta* between pairs of tasks locations for UAV i`
Generate an initial feasible solution z_0 of cost(z_0) with a greedy search;
$z_{best} \leftarrow z_0$;
$z \leftarrow z_0$;
TabuList $\leftarrow \emptyset$;
iter $\leftarrow 0$;
while (iter $< Max_{iter}$) **do**
  Find the best solution z' of the current neighbourhood with move $m' \notin TabuList$ and $cost(z')$ in M;
  TabuList.push(m');
  if $cost(z') \leq cost(z_{best})$ **then**
    $z_{best} \leftarrow z'$;
  if $TabuList.size > MaxTabuSize$ **then**
    TabuList.removeFirst();
  $z \leftarrow z'$;
  iter++;

In the following, we describe in detail the features of our Adapted ORCA as a CDR approach and discuss the differences with Standard ORCA.

1.5.3 Conflict Detection in Adapted ORCA

In this section, we describe our conflict detection process that improves upon Standard ORCA. As previously mentioned, a conflict is defined as a predicted loss of minimum separation.

In Standard ORCA, each agent applies the velocity computed by the algorithm at each time step. Velocity obstacles are computed at each time step to detect possible conflicts as shown in Fig. 1.1 and Fig. 1.2, and a new velocity is computed with respect to these constraints. However, this approach results in a possible overhead of computations and communication. If there is no conflict, the same velocity as the current velocity agent is computed.

Thus, we propose to avoid this overhead by introducing a conflict detection mechanism that is composed of two steps: a shallow step and a detailed step.The mechanism is also described in Algorithm 2.

Parameter	Description
Preferred velocity v^{pref}	Velocity of a UAV directed towards its goal if there was no conflict
Time step Δt	Time step of each ORCA iteration which is set to the same value as the UAV telemetry update rate (5Hz)
Time horizon τ	Arbitrarily fixed time window in which the velocity computed by ORCA is guaranteed to be collision-free.
Radius r	Radius of the sphere surrounding a UAV
Separation distance $sep\_dist = 2r$	Center to center point distance between 2 UAVs which is used to declare a *near* collision in case of violation
Deconfliction distance dec_dist	Center to center point distance between 2 UAVs which is used to filter unlikely conflicts.
Reciprocity coefficient λ	Percentage of the repartition in deviation between 2 UAVs in conflict: $\lambda = 0.5$ for each UAV in case of equal reciprocity
ORCA velocity v^{ORCA}	Velocity computed by ORCA algorithm at each time step, which is guaranteed to be collision-free for a fixed time horizon τ

Table 1.1: ORCA parameters

Shallow Step in Conflict Detection

The shallow step identifies potential conflicts by the use of a fixed distance called *deconfliction distance* dec_dist, which is a center to center point distance between two UAVs. It is a basic threshold distance, also originally used in Standard ORCA to filter agents that are too far apart, such that at time t, if the positions of two UAVs A and B are such as: $dist(p_A(t), p_B(t)) > dec\_dist$, no computations are made and thus any potential conflict is ignored. Based on the physical interpretation of dec_dist, we can define a lower bound on its value as follows:

$$dec\_dist \geq sep\_dist + \tau \cdot (2 \cdot MaxSpeed) \tag{1.6}$$

with $MaxSpeed$ the given maximum speed of each UAV.

However, the use of the deconfliction distance dec_dist alone cannot identify false-positives, i.e., a potential conflict was declared, but the UAVs would never collide, e.g., because they fly in parallel. We refine the conflict detection phase to address this issue.

Detailed Step in Conflict Detection

At time t, if the shallow step identified a potential conflict, we perform a more precise conflict detection computation over the τ time window by comparing the difference in magnitude between the velocity v^{ORCA} and the preferred velocity

Algorithm 2: Adapted ORCA

```
Data: U: UAVs
for u_i ∈ U do
    while p_i != Goal_i do
        Update p_i;
        for u_j (j ≠ i) ∈ U do
            // Conflict Detection
            if dist(p_i, p_j) ≤ dec_dist_i then
                if ||v_i^ORCA − v_i^pref|| > ε then
                    // Conflict Resolution
                    if u_i.State == u_j.State == In-Flight then
                        λ_{i/j} = 0.5;
                    else if u_i.State == In-Flight && u_j.State != In-Flight then
                        λ_{i/j} = 1;
                    else
                        λ_{i/j} = 0;
                    iter = 0
                    while iter ≤ ΔT do
                        v_i ← v_i^ORCA;    // Apply corresponding ORCA computations to UAV i
                        iter++;
```

of the UAV v^{pref}, which is also the current velocity, as in Eq. 1.7. This assumes that a UAV will keep its current velocity during the time window τ, which is generally the case as delivery UAVs tend to travel at constant speeds for most of their journey and external effects such as wind operate on a larger timescale than our sampling rate.

$$\text{if } (||v^{ORCA} - v^{pref}|| > \epsilon) \text{ then } inConflict = true \tag{1.7}$$

In case of conflict, the velocity computed by ORCA is transmitted to the given UAVs to avoid a loss of minimum separation in τ seconds, else nothing is transmitted and the UAV keeps its current velocity.

1.5.4 Conflict Resolution in Adapted ORCA

Following the conflict detection phase, the conflict resolution phase is triggered. ORCA computes collision-free velocities iteratively and transmits them to the given UAVs at a fixed rate (see Fig. 1.2). Note that ORCA avoids any knock-on or "domino" effect since it takes into account all the UAVs that are in conflict with each other during the time window τ in the area considered. A new collision-free velocity is computed with the intersection of the collision-free spaces for each UAV involved in a conflict at each iteration of Adapted ORCA.

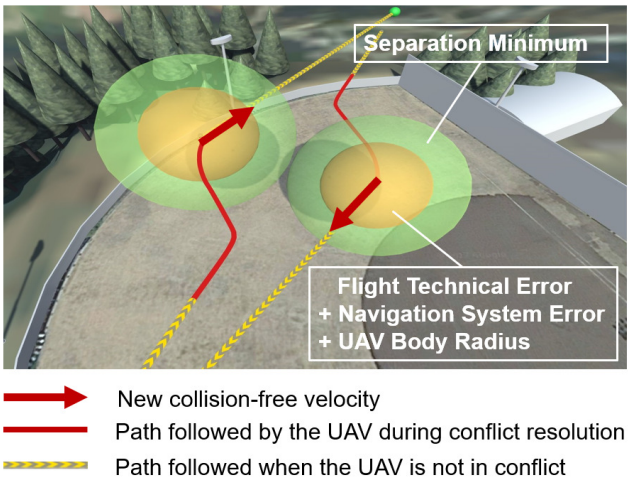


Figure 1.5: Example of Adapted ORCA solving a conflict between 2 UAVs.

Next, we turn to explaining concepts that have been newly integrated to Adapted ORCA, and were not present in Standard ORCA.

Start and End of Conflict Resolution

While conflict detection runs permanently, conflict resolution is a conditional event triggered by conflict detection. This creates an alternative behavior between a non-conflict state where a UAV follows its initially planned trajectory, and a conflict state, where a UAV follows newly generated instructions to avoid potential loss of minimum separation.

So, we formalize the notion of "start and end of conflict resolution", i.e., when and how a path should be modified by ORCA. The start of the CDR step is dependent on the fixed *dec_dist* parameter that defines the distance from where a UAV will receive ORCA velocities to change its original trajectory in case of conflict, as shown in Fig. 1.5.

When the condition in Eq. 1.7 turns false, i.e., the velocity computed by ORCA v^{ORCA} for a UAV in conflict becomes sufficiently close to its preferred velocity v^{pref}, the conflict is declared as solved and the UAV resumes its initial path towards its next waypoint.

In order to not create an oscillating behavior between non-conflict and in conflict states, this check is done after an arbitrarily fixed minimum conflict duration ΔT, i.e., number of ORCA iterations.

UAV Flight States

UAVs can have three flight states. (1) *Take-off*, (2) *In-Flight*, and (3) *Landing*. Because Standard ORCA was initially applied to ground robots, the take-off and

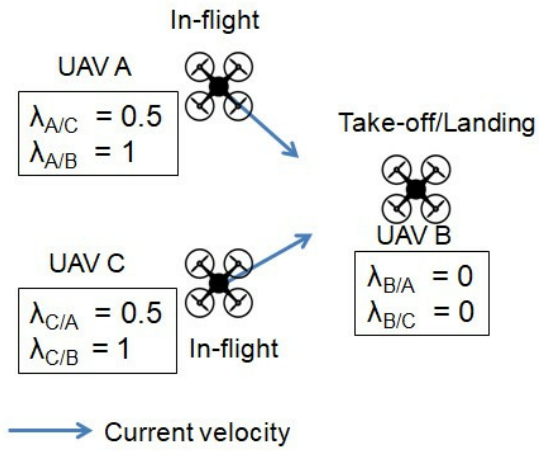


Figure 1.6: Example of three UAVs in conflict in a top down view. The state (In-Flight, take-off or landing) of each UAV is indicated together with the corresponding values of λ. The velocity of UAV B is not shown as it has vertical direction.

landing phases were not considered. There may be conflicts occurring between UAVs that are In-Flight and UAVs that are taking-off or landing.

In the UTM context, those specific phases of a UAV flight plan cannot be interrupted. Adapted ORCA addresses this situation by dynamically adapting the reciprocity coefficient λ parameter to each type of conflict situation considering the status (In-Flight versus take-off or landing) of each UAV as also described in Algorithm 2. When a UAV is in take-off or landing state, it should not be deviated as a rule, since these phases can require specific and more constrained maneuvers. So the UAV in In-Flight state takes full responsibility to avoid lack of minimum separation.

For this purpose, the value of the λ reciprocity coefficient of ORCA has to be dynamically changed depending on the conflict situation, since it constrains the set of permitted velocities when computing the solution velocity v^{ORCA} for a UAV (Fig. 1.6). Currently, we do not consider scheduling to avoid such conflicts.

UAV Acceleration Constraints

UAVs have physically imposed limits both on their speed and on their acceleration. Hence, we add a constraint taking into account the maximum acceleration $MaxAcc$, so that the new velocity v generated by ORCA is reachable:

$$||v^{ORCA} - v(t)|| \leq MaxAcc \cdot \Delta t \tag{1.8}$$

1.5.5 Empirical Tuning of Parameters in Adapted ORCA

Standard ORCA has several parameters that can affect its performance. Specifically, the look-ahead time window τ associated with the deconfliction distance *dec_dist* can affect the performance of an ORCA based CDR method because it influences the velocity computations, and thus the deviation trajectory.

Thus, unlike Standard ORCA, which arbitrarily fixes the values of these parameters, Adapted ORCA can be empirically tuned to perform optimally in certain scenarios. Section 1.7 will introduce a super-conflict scenario and demonstrate such empirical fine-tuning.

1.6 Simulation Scenarios

To determine the potential of a real world deployment of our CDR approach, we developed a simulation platform that simulates realistic service UAV operations. We use Monte Carlo simulations to provide a statistical evaluation of our method. Those simulations allow to assess the extent of the losses of minimum separation by varying the values of the minimum separation. Since real-world flight experiments are not practical at this time, Monte Carlo simulations are the most viable method to validate our CDR approach.

The following control parameters were used both in the super-conflict scenario and the real-world scenario. First, the total separation distance $sep\_dist = 2r$ (assuming same-type vehicle) with possible values of UAV-specific radius r as presented in Eq. 1.5 are determined with:

- $BodyRadius = 0.3$m (size of a DJI Phantom 4 UAV)
- $NSE = 2.0$m (GPS standard accuracy)
- $FTE = 0.5$m (empirically measured for DJI Phantom 4)
- $SafetyFactor = 1.1$ (arbitrarily fixed)

Hence, the resulting $Constructor$ layer has 3m radius.

Then, the $SeparationMinimum$ values hereby tested range from 4m to 24m. Thus, applying Eq. 1.5, we consider a range of values for r from 5m to 15m in the following simulations.

Second, for the time window τ, the optimal value to trigger our CDR method is empirically determined in our simulations. We have been experimenting with values between 4s to 10s.

Third, for the deconfliction distance *dec_dist*, the value of must satisfy the lower bound in Eq. 1.6, and thus we consider a range of values from 30m to 45m in our experiments.

1.6.1 Super-Conflict Scenario

We created extreme conflict scenarios in which up to 10 UAVs placed in antipodal positions and located in a small area of 200m $\times$ 200m are heading to

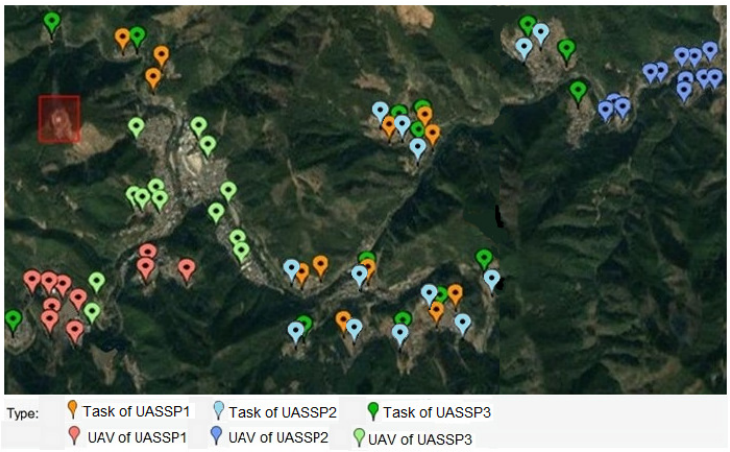


Figure 1.7: Map visualization of the Okutama area showing the UAVs initial take-off locations and task locations used in our simulations. There are fleets of three UAS Service Providers (indicated by red, blue, green) with 35 UAVs in total. UAVs are dispatched to three distinct zones that have commercial and/or health facilities. A total of 39 task locations are positioned in isolated areas that are within the range of the UAVs of each fleet. There is also one no-fly zone (red square to the left).

a single collision point. We aim to derive performance parameters for ORCA, whereby the values of *dec_dist* and τ efficiently trigger conflict resolution and show that deviation from the initial trajectory can even be reduced.

In this scenario, the ϵ parameter used for conflict detection in Eq. 1.7 is fixed to 0.05, and the minimum conflict duration ΔT is fixed to 10 iterations of ORCA algorithm.

1.6.2 Real-world delivery Scenario

We propose experimental scenarios based on a real world setup of the specific region of Okutama, a rural area in Japan as shown in Fig. 1.7.

We performed our simulations in an area of about $8km^2$. The rural area of Okutama can be a first potential candidate for real-world deployment since airspace in urban areas is not available under current legislative requirements.

Our scenarios were conceived based on a real-world preliminary study conducted on Okutama. We collected statistics on various aspects of the region such as age range, population density and repartition. With these figures and through various interviews with local authorities, also considering the topography of the area, we have designed our delivery scenarios as shown in Fig.1.7.

Consequently, our study reflects realistic expectations for UAV service in a given airspace. For instance, the number of simultaneous UAVs in the airspace was derived from the expected number of deliveries, average flying time and

hours of operation. The considered initial take-off locations for all UAVs are derived from the locations of shops, hospitals and other service facilities in the area.

In those scenarios, several UASSPs are in the same area, each in charge of their own fleet of UAVs. We fix each UAV capacity (26 min of maximum flight duration as a lower bound with a maximum speed of 5m/s, and a maximum acceleration of 3m/s^2), and maximum payload (1kg), as defined by current quadcopters manufacturers specifications.

The UAVs used in our simulations are DJI Phantom 4 copters have an average battery capacity of 26 min. We put a margin of 10 % on the actual battery consumption limit in the generated flight paths of each quadcopter to ensure that there is sufficient battery in case of deviation during the flight. The reliability of these parameters have been tested in real world experiments with experimental flight tests.

The telemetry update time step Δt is fixed at 0.2s. We located each task so that at least one UAV of the corresponding fleet can reach it as shown in Fig. 1.7. We also added the constraint that two UAVs' initial locations and tasks locations cannot be closer to each other less than the defined sep_dist.

During the simulations, batches of tasks among the possible task locations of Fig. 1.7 are randomly generated for each fleet at fixed time intervals. We ran 100 simulation samples for each experiment.

Each simulation represents a 4-hour service scenario, and we assume a fixed small amount of time for battery recharging every time the UAV returns to its initial location. Simulations were ran with an accelerated timeframe so as to reduce simulation time.

In order to avoid a 'consumption 'of UAVs by collisions that would hinder the production of simulation results, we determine that whenever a near-collision takes place (where no CDR is active), those UAVs disappear, and "spare" UAVs are automatically created at the same initial take-off locations of the involved UAVs, but at a different time for each, to not potentially recreate the same collision.

In the scenarios, UAVs are tasked mostly to transport small, light and valuable items, such as medicine, equipment and specific food items. There are three UAS Service Providers (see Fig. 1.7):

- UASSP1 (UAVs positions in red): Health related delivery and transport (medicines, biological samples) (10 UAVs; 12 tasks locations)
- UASSP2 (UAVs positions in blue): Daily items and food (12 UAVs; 13 tasks locations)
- UASSP3 (UAVs positions in green): Daily items and food (13 UAVs; 14 tasks locations)

All terrain information is provided via the elevation map generated by the existing Mapzen Terrain Tile Service.

The quadcopters flight motions are simulated by our own in-house simulator which uses data acquired from physical trials. At the moment, available UAV

simulators are single-computer stand-alone systems that do not scale well and are thus impractical for our broader purpose of simulating large-scale UTM scenarios. Hence, we built our own distributed networked simulator that scales simulations up to thousands of concurrent UAVs.

All algorithms previously described are implemented in Java and we ran our experiments on a 2.9 GHz Intel Core i5-4210 CPU with 16 GB RAM.

1.7 Experimental Results

1.7.1 Super-Conflict Scenario

The two metrics used in the high density experiment are:

- Average time optimality per UAV. This study aims to evaluate the optimality, or degradation of optimality, in UAVs' evasive behavior.
- Deviation in distance. This study investigates the amount of detour required to avoid loss of minimum separation.

Average time optimality per UAV is formalized as follows. For a UAV j, time optimality degradation is defined as the ratio of the time it would have taken for j to reach its goal location if there was no conflict t^{ideal} to the actual total time taken t^{actual}. Then, the average time optimality per UAV with N referring to the total number of UAVs considered is defined as:

$$Opt_j = \frac{t^{ideal}}{t^{actual}} \; ; \; AverageTimeOptimality = \frac{\sum_{j=1}^{N} Opt_j}{N}$$

Deviation in distance, similar to time optimality, measures the difference between the actual distance traveled and the ideal distance a UAV would have traveled based on their initial (optimal) paths.

In our first study, we will investigate the influence of τ and *dec_dist* parameters on ORCA resolution. The value of the look-ahead time horizon τ, and by extension *dec_dist*, influence the quality of the resolution.

Since those parameters are used in the computation of the new velocity by ORCA (in case of conflict), our hypothesis is that there is a value or interval of values of τ and hence *dec_dist* for which the deviation from an initial trajectory is minimal while ensuring no loss of minimal separation.

We propose to determine those values empirically through our simulated experiments. We fix MaxSpeed to 5m/s and vary τ with *dec_dist* and the separation radius r values. In Fig. 1.8 and 1.9, we show that choosing the appropriate values of τ and *dec_dist* can minimize the deviation with the considered UAVs.

In the case of super-conflict scenarios, these values allow us to initiate Adapted ORCA velocity computations with more informed values than choosing an arbitrary value.

With a larger radius r, the UAVs are then considered to be larger moving agents, so the deviation in trajectory is larger, as shown in Fig. 1.9. Further,

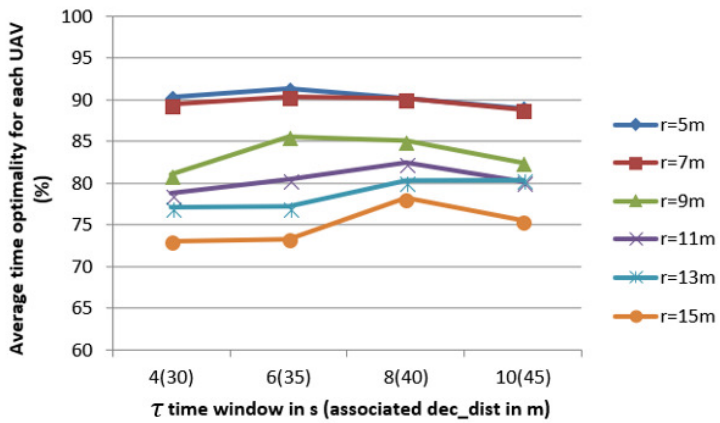


Figure 1.8: Evolution of average time optimality for Adapted ORCA in extreme conflict scenarios with MaxSpeed = 5m/s and different values of r relative to τ and associated *dec_dist* values.

the average time optimality in Fig. 1.8 is inferior for the values of r ranging from 9m to 15m compared to the 5m and 7m values.

We also observe that a larger r value requires a larger τ with the associated *dec_dist* value to minimize the deviations from the initial trajectories. For instance, for r = 9m, the highest time optimality is reached for a value of τ = 6s (with *dec_dist* = 35m), whereas for r = 11m, it is reached for τ =8s (with *dec_dist* = 40m).

Otherwise, for smaller values of τ, we observe a larger degradation in time optimality, since UAVs in conflict do not have sufficient time to react and tend to get slower. For higher values of τ, the UAVs tend to react too early and hence take unnecessary detour.

1.7.2 Real-world Delivery Scenario

In the delivery scenario, we study three types of setups:

- Frequency of loss of minimum separation with and without CDR. This study provides an empirical estimation of the likelihood of safety issues in shared airspace.
- Influence of the separation radius on occurrences of loss of minimum separation with and without CDR. Note that only the value for separation minimum is varied, while we take the entire separation radius to allow center point to center point distance measurement.
- Influence of the distribution of UASSPs' fleets on loss of minimum separation with and without CDR.

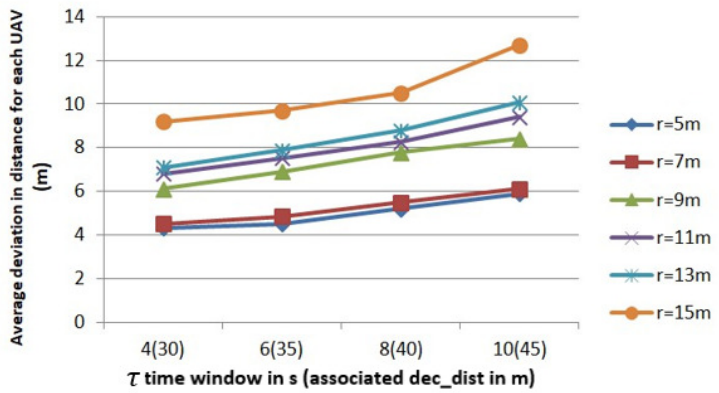


Figure 1.9: Evolution of average deviation in distance for Adapted ORCA in extreme conflict scenarios with MaxSpeed = 5m/s and different values of r relative to τ and associated *dec_dist* values.

First, we investigate the frequency of loss of minimum separation with and without CDR. We fix the number of fleets (UASSPs) to 3, the total number of possible task locations to 39 and vary the total number of UAVs from 15 to 35 among the possible locations shown in Fig. 1.7. The maximum number of UAVs is derived from the preliminary UTM studies in [44] given the rural area and its population density.

We first simulate scenarios without the use of our CDR method and report the near-collisions (violations of minimum separation distance *sep_dist*) that occurred.

In Fig. 1.10, considering a separation radius r of 15m, as expected, the total number of near-collisions observed in all the simulations increases with the number of UAVs in the same area, hence with a higher density. More precisely, the number of near-collisions observed increases in a quadratic manner.

As shown in Fig. 1.10, these realistic simulations also assessed the safety provided by our CDR approach, Adapted ORCA, in particular with the separation distance to ensure no loss of minimum seperation (no near-collisions) and hence 0% of physical collisions.

Let us look at the situation where no CDR method is in place. Most of the near-collisions that happened were In-Flight, but near-collisions while one UAV is taking-off or landing also happened to a fewer extent (Table 1.2) All cases of loss of minimum separation was observed in those scenarios were between 2 UAVs only.

Moreover, Table 1.3 shows that most near-collisions happened between UAVs of fleets from different UASSPs. Based on the minimum cost objective of the task allocation, it is unlikely that a conflict occurs between UAVs of the same UASSP, unless there is a large number of tasks concentrated on a particular area, and visiting them all would exceed the capacity of one UAV.

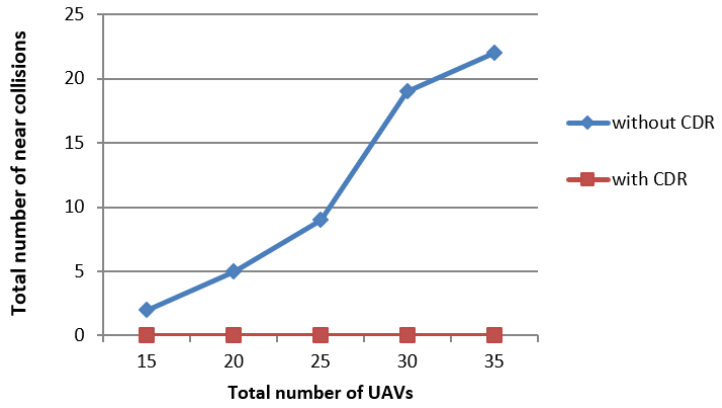


Figure 1.10: Impact of CDR on the number of total near-collisions (loss of minimum separation) considering $r = 15$m.

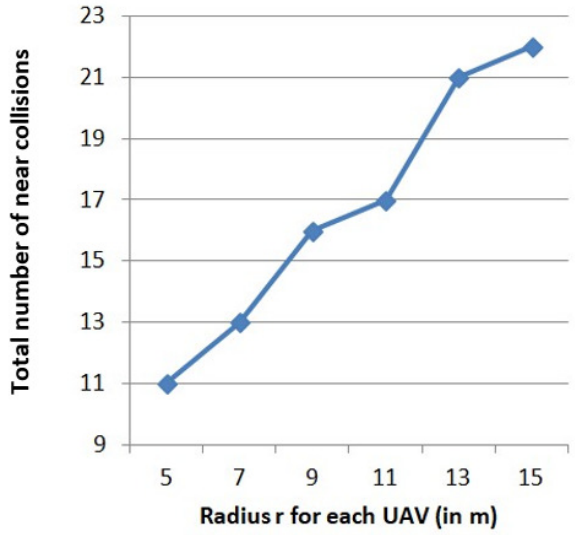


Figure 1.11: Total number of near-collisions relative to the radius r without CDR.

Between UAVs in In-Flight state	94 %
Between UAVs in In-Flight and take-off/landing state	6 %

Table 1.2: Average % of near-collisions in UAVs' states (with no CDR method applied).

Between UAVs of the same fleet	11.1 %
Between UAVs of different fleets	88.9 %

Table 1.3: Average % of near-collisions within same fleet and between different fleets (with no CDR method applied).

Number of fleets	3	4	5	6
Number of collisions	22	22	23	25

Table 1.4: Total number of near-collisions with different number of fleets (with no CDR method applied).

Second, we study the influence of the separation radius r on the number of near collisions, i.e., violations of minimum separation. We fix the number of fleets (UASSPs) to 3, the total number of UAVs to 35 and the total number of possible task locations to 39, as shown in Fig. 1.7. Then, we analyze the number of near-collisions for the different values of $r = 5, \ldots, 15$m, as presented in Fig. 1.11. The total number of near-collisions increases with the considered radius r.

Given the rationale adopted by UTM regulations, even for the minimum value of $r = 5$m, near-collisions happen if no CDR method is in place. The use of a CDR method, such as Adapted ORCA, prevents such violations.

Finally, we investigate the influence of the distribution of fleets (UASSPs) on loss of minimum separation. We keep the same scenario with 35 UAVs in total, with $r = 15$m, and the 39 existing tasks locations in total. Here, we vary the number of UASSPs from three UASSPs with 12 or 13 UAVs each, to six UASSPs with 5 or 6 UAVs each, so that there are 35 UAVs in total. We observe in Table 1.4 that the number of near-collisions increases with the number of UASSPs in the same area. Hence, losses of minimum separation would happen with a larger number of UASSPs rather than having more UAVs in each UASSPs.

1.7.3 Scalability

In our setting, Adapted ORCA exhibits fast runtimes (100ms in average), on the same order of magnitude as Standard ORCA, which has already shown high scalability with applications involving thousands of agents. Moreover, the scalability of the ORCA method is further improved by the incorporation of the conflict detection filtering steps introduced in Section 1.5.3 that allows to reduce the amount of computations that Standard ORCA would normally require.

1.8 Conclusions

Future UAV-based services will require a fully implemented UTM system to ensure safe and efficient operations in low-altitude airspace. UAS Operators will choose from several UASSPs to use flight operation services such as automated

task allocation and generation of collision-free flight paths. Yet, the Conflict Detection and Resolution (CDR) method hosted by the Core UTM is needed to solve In-Flight conflicts among UAVs of different, independent service providers.

The contribution of this chapter is a new CDR method based on ORCA, called Adapted ORCA, which is a practical and effective CDR mechanism for a realistic UTM operational context.

To validate our CDR method, we designed and implemented a simulation platform that runs realistic service UAVs operations. First, we looked at extreme-conflict simulations, and conducted an analysis of ORCA parameters that can affect its solution quality. The simulations yielded optimized values for the key parameters look-ahead time window and deconfliction distance. Those values can be reused in similar situations, such as disaster situation, to optimize the performance of our CDR method based on Adapted ORCA.

Second, we performed extensive simulations on realistic scenarios based on a real world study. With the sample collected, we obtained safety parameters, i.e., the frequency of loss of minimum separation, if no CDR method is implemented. With those simulated scenarios, we were able to study and evaluate the impact of the minimum separation distance value in a realistic set of operations in a rural environment, with the purpose of ensuring physical safety between all UAVs. Therefore, we showed that our proposed In-Flight CDR method is able to ensure the safety of UAVs flights.

Third, our Adapted ORCA presents an improved scalability and processing time from Standard ORCA, i.e. thousands of agents in less than a second [85], since we reduce the amount of computations needed for each UAV agent with the incorporation of conflict detection filtering steps. Thus, the scalability of our approach with respect to the number of agents would still be enough to handle the expected number of UAVs in the low altitude airspace such as Okutama, which is in the low hundreds of UAVs.

Future Works for In-Flight CDR

Future work for In-Flight CDR will address some simplifying assumptions in this chapter:

- We currently assume homogeneous UAV fleets consisting entirely of same-type quadcopters. Here, we plan to work on an heterogeneous airspace with UAVs having different dynamics and capabilities, including plane-type UAVs. This will require the study of a generalized In-Flight CDR method.

- We plan to address situations where automated UAVs may encounter human-controlled aircraft, such as hobby drones, or helicopters.

Chapter 2

Centralized Pre-Flight Conflict Detection and Resolution

In the context of Unmanned Aircraft System Traffic Management (UTM), it is expected that Unmanned Aerial Vehicle (UAV) operators submit flight paths planned by service providers to avoid collisions with static obstacles, such as terrain elevation and no-fly zones. Here, we assume that the UTM system processes the paths of all UAVs given their properties such as speed, size, start time, and so on. In the In-Flight phase, as a redundancy mechanism, while UAVs populate the shared airspace, their trajectories can be adapted due to emergent events, such as bad weather or emergency operations, using In-Flight Conflict Detection and Resolution (CDR) methods. One important element, or redundancy, for safe and efficient UAV operation is Pre-Flight CDR methods that generate conflict-free paths for UAVs before their actual flight. In this chapter, we focus on this phase. Multi-Agent Path Finding (MAPF) has already been successfully applied to comparable problems with ground robots. However, most MAPF methods were tested with simplifying assumptions which do not reflect important characteristics of many real-world domains, such as delivery by UAVs where heterogeneous agents need to be considered, and new requests for flight operations are received continuously. In this chapter, we extend the CBS and ECBS algorithms to efficiently incorporate heterogeneous agents with computational geometry and we reduce the search space with spatio-temporal pruning. Moreover, our work introduces a "batching" method into ECBS to address increased amounts of requests for delivery operations in an efficient manner. We compare the performance of our "batching" approach in terms of runtime, throughput and solution costs to a "first-come first-served" approach. Our scenarios are based on a study on UAV usage predicted for 2030 in a real area in Japan. Our simulations indicate that our proposed ECBS based "batching" approach is more time efficient than incremental planning based on Cooperative

A, and hence can meet the requirements of timely and accurate response on delivery requests to users of such UTM services.*

2.1 Introduction

The main challenge in the UTM context [48, 49] is to design efficient Conflict Detection and Resolution (CDR) approaches [44], whereby a *conflict* is defined as "an event in the future in which two or more aircrafts will experience a loss of minimum separation between each other" [51]. Similar to the ATM concept [44,51], we can distinguish two phases in the design of CDR methods for UTM: a Pre-Flight phase and an In-Flight phase. In-Flight CDR methods will ensure conflict-free paths for all UAVs during flight. Pre-Flight CDR methods, on the other hand, aim to provide conflict-free paths to all UAVs before actual take off.

In the design of future UTM systems, it is envisioned that UAV operators submit flight paths that are pre-planned by service providers to avoid collisions with static obstacles, such as terrain elevation and no-fly zones. Then the UTM system processes the paths of all UAVs given their properties such as speed, size, start time, and so on. Then, in the In-Flight phase, as a redundancy mechanism, while UAVs populate the shared airspace, their trajectories can be adapted due to emergent events, such as bad weather or emergency operations, using In-Flight CDR methods. While many In-Flight CDR methods have been recently developed in the UTM context [3,34,42,86,96], Pre-Flight CDR methods remain mostly unexplored for this domain.

Thus, in this chapter, we aim to advance the development of Pre-Flight CDR methods. We consider quadcopters as service UAVs that are assigned to fly anytime during a service day from start locations (UAV hubs) to given task or goal locations, then return back to their initial locations. Moreover, due to the variation of UAVs in size and speed, we conceive the heterogeneity of the future integrated airspace.

Pre-Flight CDR relies on the concept of 4D trajectory-based operations, i.e., a sequence of waypoints that consists of 3D coordinates and associated timestamps that a UAV passes. Pre-Flight CDR can be represented as a Multi-Agent Path Finding (MAPF) problem. In MAPF, several agents must avoid collisions while moving from given start locations to goal locations. MAPF has mostly been studied in 2D environments, for a variety of applications such as video games and Amazon warehouse ground robots [55]. Further, the typical MAPF setting is a "one-shot" problem where all agents start simultaneously and all have a distinct pair of start and goal locations. It also assumes homogeneous agents which all have the same size and are all contained inside each cell of the given grid map, and all move by one cell at each time step.

Our low altitude airspace domain can be characterized by:

- UAV operations have different start times, hence newly computed flight paths must take into account the existing (already approved) flight paths;

- UAVs fly from their respective hubs to specific locations and then have to return to these hubs, which introduces a precedence relationship;

- UAVs are heterogeneous agents in terms of different sizes and speeds.

In this chapter, we propose to extend the MAPF framework to the UAV operations scenario for Pre-Flight CDR.

Existing MAPF solvers have been demonstrated to be optimal for the sum of individual costs objective and complete such as Conflict-Based Search (CBS) [75] or bounded suboptimal such as Enhanced CBS (ECBS) [6]. Differently, incremental techniques that plan agents sequentially in a predefined order, have also been proposed, such as Cooperative A* (CA*) [77] and are known to be unbounded suboptimal and incomplete.

In the UTM context, one suggestion is that UAS (Unmanned Aircraft System) operators submit their flight plans on the previous day, which corresponds to a "one-shot" approach. However, this approach would be impractical for timely delivery by logistics companies. Another suggestion is the "first-come first-served" (FCFS) approach, where each incoming request is calculated in order. This approach could be assimilated to the CA* method, which generates unbounded suboptimal solutions with no guarantee for a solution. Therefore, we propose to process "batches" of UAS operation requests, as simultaneous planning provides the ability to replan paths without any ordering constraints.

Then, we evaluate the scalability of our proposed approaches in terms of number of operations to process. For this purpose, we will compare CA* and ECBS approaches in terms of computation time, throughput and solution costs.

This chapter presents two main contributions:

- We extend the MAPF formulation to the context of UTM (Unmanned Aircraft System Traffic Management), specifically deliveries by UAVs in shared airspace. We introduce a new formalization of the problem which takes into account (i) flights with different start times, including return paths after delivery, and (ii) agents (UAVs) of different size and speed.

- We address heterogeneous agents by incorporating geometrical computations, rather than simple voxel intersection, into the internal conflict detection process of our considered MAPF algorithms. Further, we introduce a spatio-temporal pruning that reduces the search space.

We propose an extension of CBS and ECBS that introduces a "batching" method to process incoming flight requests. This method offers advantages to the FCFS approach, if timely response to the delivery user is important. We evaluate and compare these techniques based on a realistic projection of drone usage in 2030, referring to a real city in Japan.

2.2 Related Works on Multi-Agent Path Finding (MAPF)

Cooperative Multi-Agent Path Finding (MAPF) is known as NP-hard to solve optimally for the sum of individual costs (SIC) objective [98]. In the UTM context, low-altitude airspace is often seen as a common resource. So the SIC objective aligns with the aim to minimize the air traffic. While we adopt this global objective, other works on MAPF use combinatorial auctions [4] or taxation schemes [13] to consider self-interested agents.

A*-based MAPF solvers search the joint state space, treating a configuration of several agents as a state. Such optimal algorithms include Enhanced Partial Expansion A* (EPEA*) [31], Independence Detection and Operator Decomposition (OD-ID) [81], Increasing Cost Tree Search (ICTS) [76], M* [87] and Conflict Based Search (CBS) [14, 25, 75]. Other optimal approaches use Integer Linear Programming [98]. In contrast, suboptimal approaches have been developed to provide a better performance in runtime as opposed to optimal approaches. The bounded suboptimal variant of CBS, Enhanced CBS (ECBS) [6] is among the most efficient approaches. Other suboptimal solvers make use of the existence of traffic flows with a well-defined physical structure and obstacle density in given instances by introducing flow annotation structures, such as Flow Annotation Replanning (FAR) [90] and ECBS+HWY [18].

We can also distinguish suboptimal search-based solvers that work in a prioritized way, but are incomplete and unbounded such as CA* [77], where the agents are planned one after the other according to a predefined order. Other unbounded suboptimal techniques use specific movement rules to solve MAPF instances [22, 53, 91]. However, all these approaches originally target "one-shot" scenarios, thus solved before all agents start to move. In this chapter, we will study a different version of MAPF where paths of agents are added over time, while other agents may be following previously generated plans. In our case, as a practical requirement, we do not allow replanning of previously processed paths since it requires real time communication and may incur some overhead in the online modification of paths.

Recently, few works have proposed different extensions of the MAPF framework to address real world settings. All these works address different aspects in isolation. Yet, they did not tackle heterogeneous agents in terms of different sizes and speeds. We hereby propose a setting that includes many of the realistic features proper to the UTM context. Few MAPF algorithms have been described for non-unit time step domains such as in [88] where the ICTS algorithm is extended to the non-unit cost domain. [93, 94] have incorporated any-angle pathfinding into the paths of each agent but their work is still in a 2D homogeneous setting.

In the context of the Amazon warehouses, [56] proposed a lifelong algorithm within the pickup and delivery setting for agents having tasks to reach in an ongoing way. They introduce a decentralized approach focusing on task allocation. In our context, we assume that every agent is associated with a given

start and goal, so there is no need for task allocation. In contrast, we study a centralized setting where agents service a large environment distinct from the Amazon use case.

2.3 Problem Formulation

2.3.1 UTM Application Context

In the low-altitude airspace context, we address a 3D scenario where restrictions are put on the flight altitude related to the elevation of the terrain. UAVs are allowed to fly between legal bounds that are hereby fixed to $min_{alt} = 90$ meters (m) for the minimum altitude, and $max_{alt} = 150$m for the maximum altitude, relative to the elevation from mean sea level of a point of given latitude and longitude coordinates. Hence, there is a 60m altitude range. In our work, we consider a 3D grid map composed of voxels with 30m edge size. We consider quadcopters UAVs that have holonomic motion, thus can move in any direction or hover.

In this chapter, we consider service scenarios such as delivery, where UAVs fly from predefined hubs to service locations, i.e., other hubs or homes. The assignment of those locations is done independently by service providers. So, the scope of this chapter does not include the allocation of tasks to the agents, unlike in [38, 54]. The role of the UTM System is to provide conflict-free paths for all submitted flight requests from all UAV operators.

2.3.2 Problem Definition

An instance of our problem is composed of N operations $O = \{O_1, \ldots, O_N\}$ which are each performed by agents that are UAVs, and an undirected graph $G = (V, E)$ which is a 26-neighbor cubic grid allowing diagonal moves. Agents can *move* along an edge of G or can *wait* on a vertex of G.
An agent a_i assigned to perform $O_i \in O$ is characterized by:

- A *radius* r_i: each agent is represented by a sphere of given radius r_i, and a center position p_i. In the UTM context, similar to the ATM context, no physical collisions are ever acceptable, so the radius of each UAV is enlarged with extra layers to ensure physical separation between UAVs.

- A *speed* sp_i: the given speed of a_i which is considered uniform on the whole path, as in the UTM context, this would be a constraint to ensure conflict-free paths generation [68].

- A *start* s_i and *goal* g_i locations: an operation O_i is composed of a pair of paths, an outbound path and a return path. The outbound path goes from a hub location s_i to a delivery location g_i and the return path is assumed to be symmetrical to the outbound path. In-between reaching the delivery location and returning to the hub location, we assume a fixed duration δ_i, and that agents do not remain in the space once they reach

Properties	Standard MAPF	Extended MAPF
Agent size	Same size for all, inside one voxel: $\forall a_i, r_i = r$	Different sizes, more or less than one voxel: $\forall a_i, r_i > 0$
Agent speed	1 voxel per time step for all: $\forall a_i, sp_i = sp$	Different speeds: $\forall a_i, sp_i > 0$
Agent start time	Simultaneous start for all: $\forall a_i, t_i^s = t^s = 0$	Different start times: $\forall a_i, t_i^s > 0$
Path	Unique path $(s_i; g_i)$ for each agent: $\forall a_i \neq a_j, s_i \neq s_j, g_i \neq g_j$ and no return	Agents can have the same starts and/or goals with different start times: $\exists a_i \neq a_j, t_i^s \neq t_j^s, s_i = s_j, g_i = g_j$ and all have to return
Conflict	Agents moving to the same vertex: $p_i(t) = p_j(t)$, or exchanging vertices at the same time step: $p_i(t) = p_j(t+1)$ and $p_j(t) = p_i(t+1)$	Any case of loss of minimum separation: $dist(p_i, p_j) \leq r_i + r_j$

Table 2.1: Extending the MAPF framework.

their destination or hub, since we consider that each UAV lands when reaching its assigned location.

- A *start time* $t_i^s > 0$: the time at which the operation must start, hence when the agent takes off.

In the UTM context, we want to prevent any violation of the minimum separation distance between two agents a_i and a_j, i.e., the sum of their respective radius, $r_i + r_j$. Hence, the constraint in our formulation is defined as follows: $\forall t, dist(p_i(t), p_j(t)) > r_i + r_j$. Moreover, since each agent's operation includes an outbound path and a return path, we must ensure that the precedence relation between the outbound path and the return path is always satisfied. The objective we hereby adopt remains the same as in standard MAPF, i.e., to minimize the sum of individual costs: $\min \sum_{O_i \in O} T_i$, with T_i the total cost of the operation O_i, which is the total duration of O_i. A solution consists of conflict-free paths for all N operations such that no violation of minimum separation occurs.

In Table 2.1, we describe how our formulation extends the MAPF framework. Note that here, the grid discretization serves to represent static obstacles and allow path planning, unlike in standard MAPF where it also indicates the position of each agent at each time step.

2.4 Background: Conflict Based Search (CBS)

We hereby describe CBS [75] and then its bounded suboptimal variant ECBS [6], on which we base our Pre-Flight CDR approaches.

CBS is a two-level search algorithm, and is complete and optimal with respect to the sum of individual costs.

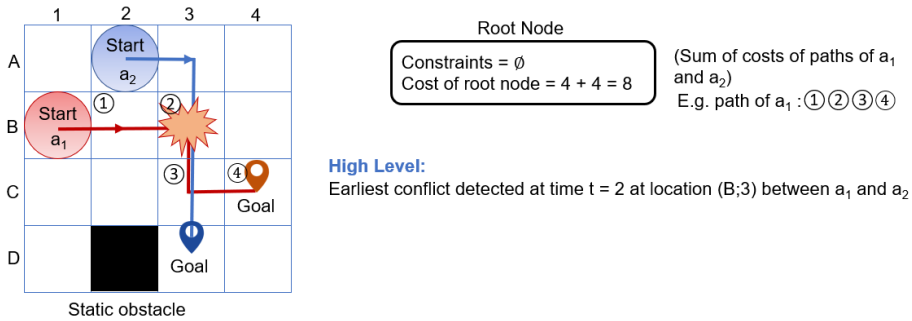


Figure 2.1: Example: CBS initial step, the earliest conflict is detected at the high level in the root node with the initial paths of the two agents.

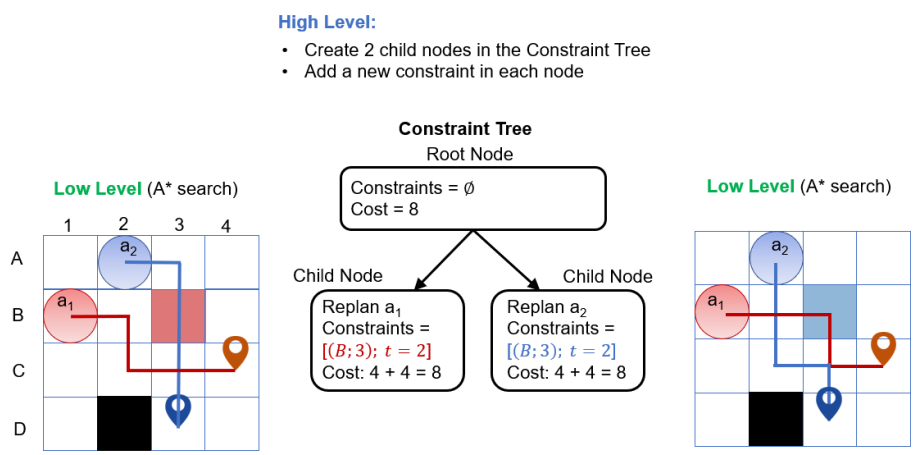


Figure 2.2: Example: Generation of children nodes in the constraint tree with a relative constraint for each agent.

CBS alternates between a *high level* and a *low level*.

The *high level* of CBS searches the binary Constraint Tree (CT). Each node of the CT contains: (1) a set of constraints imposed on the agents, meaning that an agent a_i cannot occupy a vertex v at a time step t; (2) a single solution that satisfies all constraints; and (3) the cost of solution, which is the sum of

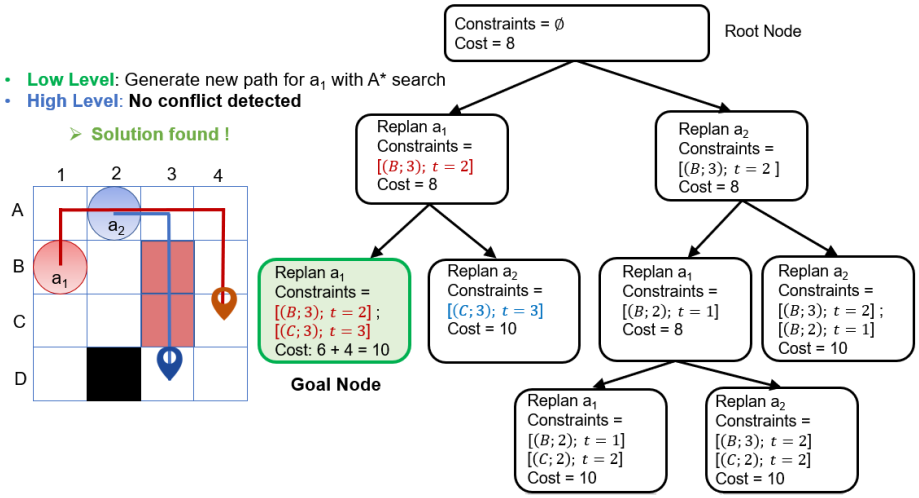


Figure 2.3: Example: Solution found in CBS.

the path costs of all agents. The root node of the CT contains an empty set of constraints as shown in our example in Fig. 2.1.

For each CT node n generated by the high level, a *low level search* which is an A* search, is invoked. For an individual agent, it generates a path that satisfies all constraints in node n imposed on the agent and provided by the high level. Once a CT node n is chosen for expansion by a best-first search, the solution is checked for any conflicts between the agents along their planned paths. If node n is conflict-free, then it is a goal node and CBS returns its solution as in Fig. 2.3. Otherwise, as in Fig. 2.2, n is split into two child nodes, each with an additional constraint on only one of the two agents involved in the earliest conflict for each child node, and thus, only the path of this agent will be replanned.

Then, ECBS [6] is a bounded suboptimal variant of CBS with similar principle. It is bounded by a factor $w > 1$ of the optimal solution cost. w is a parameter that can be arbitrarily fixed. The difference is that in the high and low level of ECBS, Focal search [62] is used instead of a Best-First search and an A* search. Focal search is a suboptimal variant of A* where a FOCAL list is maintained alongside the OPEN list with a second usually inadmissible heuristic that estimates the number of conflicts for instance. This is used to minimize the number of conflicts to solve. The FOCAL list contains a subset of the entries in the OPEN list, such that the cost of the entries in the FOCAL list are within a constant factor w of the best cost in the OPEN list.

2.5 Extension of CBS and ECBS for UTM Context

For the UTM context, we need to adapt and extend CBS and ECBS in three ways. First, we reduce the search space of CBS and ECBS in our problem domain by introducing a spatio-temporal pruning process. Second, we propose to incorporate a continuous-time conflict detection based on computational geometry, to integrate heterogeneous agents. Finally, we accommodate for the situation of ongoing requests for deliveries. Here, we will introduce a batch processing approach.

Note that in our case, initial paths for all agents are planned independently by UAS service providers and then submitted to the UTM system. As a result, a set of possibly conflicted paths is produced.

The following changes introduced in the remainder of this section apply to both algorithms, CBS and ECBS, even though only CBS is noted.

2.5.1 Spatio-Temporal Pruning

To detect any conflicts between all pairs of operations, the search space contains $\frac{N(N-1)}{2}$ states. Thus, the amount of computations increases significantly with the number of operations and the iterations of the process during the search. Moreover, in our context, two operations might have no time or no spatial intersection, i.e. no possibility of conflict. So, we propose to reduce the search space of potential conflicts by introducing a pruning process. We determine and associate a subset $O_i^{Conflict}$ of operations in potential conflict for each operation O_i by considering the temporal and spatial information of the given instance as shown in Algorithm 3. Then conflict detection (described below) is performed only within the reduced set of identified operations and common paths segments.

2.5.2 Extension to Heterogeneous Agents

In standard MAPF, all agents move by one voxel at each time step and there can only be two types of conflicts: vertex and edge conflict, as mentioned in Table 2.1. In our extended MAPF, where agents have different sizes and speeds, these properties do not hold anymore. Agents can occupy more or less than one voxel at each time step, and several voxels can be crossed between two successive time steps, which can lead to possible conflicts in between time steps. The use of straightforward discretization for conflict detection is thus inefficient. Therefore, we propose a conflict detection mechanism that relies on geometrical computations, and encodes continuous-time conflict detection.

2.5.3 Incorporating Geometrical Computations

We present a reformulated conflict detection mechanism that we incorporate into CBS. We need to consider all possible cases of conflicts as represented

Algorithm 3: Spatio-Temporal Pruning: Define the subsets of agents in potential conflict

```
Data: O set of operations, O_i with associated T_i time interval and agent
      size r_i
Result: ∀O_i ∈ O, O_i^Conflict ⊆ O
for O_i ∈ O do
    for O_j ∈ O do
        /* Determine if temporal overlap between O_i and O_j    */
        if T_i ∩ T_j ≠ ∅ then
            /* Determine if spatial overlap during common time
               interval                                          */
            if ShortestDistance(O_i, O_j, T_i ∩ T_j) ≤ 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^Conflict and O_j in O_i^Conflict;
```

in Fig. 2.4. All conflicts can be classified into these categories that can be considered as generalizations of vertex and edge collisions of standard MAPF.

A straightforward approach would be to consider all voxels that an agent intersects at each time step. Since an agent does not necessarily move by only one voxel at each time step, a discretization in time steps of the path of each agent according to their respective speeds must also be performed. Then, a conflict is detected between two agents between two successive time steps t and $t+1$ if they both have at least one voxel in common within their respective set of voxels as in Fig. 2.5. This method may be computationally expensive in practice, (1) due to the determination of the occupied voxels set of each agent, and (2) the intersection verification done at each time step for these sets possibly containing several voxels. Moreover, it induces an approximation in the actual space occupied by an agent and can lead to false positives in the detection of conflicts.

Thus, we introduce a more effective method to detect conflicts based on geometrical considerations. To detect a conflict between agents whose paths segments have temporal overlap, we compute the "time to collision" t_C within the common time interval $[t_a; t_b]$. t_C is obtained by solving the following quadratic equation with v_i and v_j the velocities and p_i and p_j the positions at any given times of agents a_i and a_j respectively:

$$\|p_i(t_C)-p_j(t_C)\|^2 = (r_i+r_j)^2 i.e. \quad (p_i(t_a)+v_i\cdot t_C-(p_j(t_a)+v_j\cdot t_C))^2 = (r_i+r_j)^2 \tag{2.1}$$

If there are real positive roots for the equation in the range of the interval $[t_a; t_b]$, we have obtained the time to collision $t_C = Min(t_{root1}; t_{root2})$. Thus, a *conflict interval* is defined, and it represents the time interval during which a violation

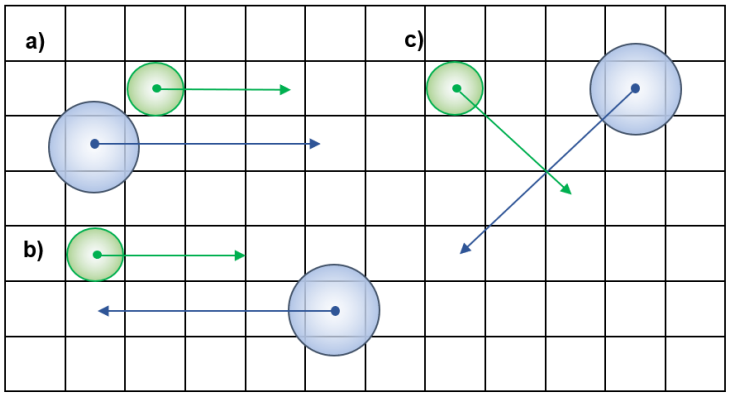


Figure 2.4: Types of conflicts between two agents of different sizes and speeds. a) Pursuit conflict, where the agents move closely towards the same direction ; b) Head-on conflict, where the agents move into opposite directions ; c) Intersection conflict, where the agents cross paths.

of the separation distance between two agents occurs,

$$I_C = [t_C; Min(t_b; Max(t_{root1}; t_{root2}))]$$

.

2.5.4 Reformulation of Low Level Search

Having redefined the conflict detection step, we now present the changes made to the low level search of CBS. First, we redefine the *constraint representation* used in CBS. In standard MAPF, a constraint is only a given vertex $v \in V$ that cannot be occupied at a time step t. In extended MAPF, since conflicts cannot be limited to discrete cases anymore, the constraints are expanded to a set of occupied space that cannot be crossed within the *conflict interval* during which a conflict was detected. We denote $\mathscr{V}(a_i, I_C)$ the space occupied by an agent during the conflict interval. With our geometrical considerations, it is a "capsule" volume shown in Fig. 2.5 characterized by a line segment and the associated radius of the agent. If a_i and a_j are in conflict in I_C, then the constraints tuples $(a_i, \mathscr{V}(a_j, I_C), I_C)$ and $(a_j, \mathscr{V}(a_i, I_C), I_C)$ are associated to a_i and a_j respectively.

In the low level search, the path of one agent in conflict is replanned with A* to satisfy all constraints associated to this agent. As shown in Algorithm 4, at each node expansion, if there exists a constraint such that the associated conflict interval I_C and $[t_{current}; t_n]$ overlap, then we perform a conflict detection computation within the common time interval as in Equation. 2.1. If a conflict is detected, then the move is not considered in the generation of the new path for the agent. When planning the conflict-free paths for all operations, we also

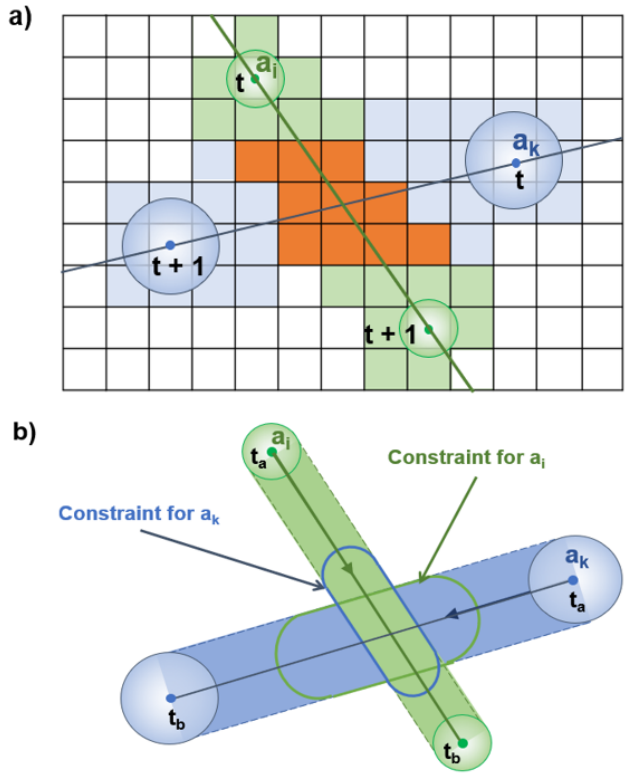


Figure 2.5: a) Conflict interval for the voxels intersection method (in red); b) Conflict interval for the computational geometry method (indicated as constraints).

Algorithm 4: Reformulated Low Level Search in Extended CBS: Node expansion

```
Data: Node current to expand for agent i with Constraints_i
for node n ∈ Neighbors of current do
    /* Check if reachable neighbor w.r.t static obstacles   */
    if n not reachable then
        Remove n from Neighbors;
    else
        for constraint c ∈ Constraints_i do
            if I_c ∩ [t_current; t_n] ≠ ∅ then
                /* Check time to conflict in the common interval
                   */
                Compute t_C in common time interval between I_c and
                  [t_current; t_n];
                if in Conflict then
                    Remove n from Neighbors;
```

need to ensure that the agents do not collide with any static obstacles such as elevation or no-fly zones. Static obstacles are represented via the voxels of the grid that are considered as blocked. In A*, the g function is used to exclude or select nodes for expansion. Here, we also improve the computational efficiency of our approach by pruning the search space and using geometrical computations instead of considering the intersection with each voxel independently.

2.6 Batch Processing

Our problem starts with an empty airspace where no agent has begun to fly yet, and our algorithm must solve a first given MAPF instance containing a set of submitted operations. So, the first MAPF instance is always a one-shot instance. Then, while the agents execute their generated plan, a new set of operations appears which might be in conflict with the already accepted operations. We refer to each given set of operations as a *batch* containing a finite number of operations. Hence, we need to consider two types of conflicts that can occur for an agent from a later batch, which are (1) conflict with an agent from a previous batch or (2) conflict with an agent from the same batch. For (1), since we assume that previously accepted operations cannot be modified anymore, we consider the paths of these operations as spatio-temporal obstacles that have to be avoided. So, a first step of detecting and solving this type of conflicts is performed, then the existing spatio-temporal obstacles are considered when replanning for conflicts within the same batch. For (2), as described in 2.4, (E)CBS considers conflicts between agents of the same batch and generates two child nodes for each agent of the conflict.

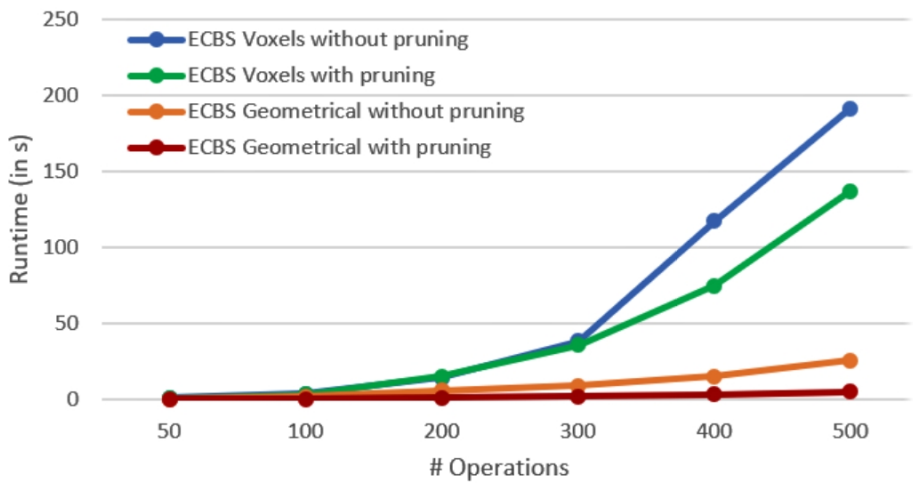


Figure 2.6: Average runtime for ECBS on $100 \times 100 \times 4$ grid map with 5% obstacle density.

2.7 Experiments

In this section, we describe the simulations and obtained results. First, we evaluate the efficiency of our extension to accommodate heterogeneous agents. Here, we compare the use of computational geometry and spatio-temporal pruning with a straightforward voxels intersection approach. Second, we compare the performances of CBS and ECBS with "batch" processing of UAS operations to a CA* based approach which exemplifies FCFS processing, in a real world use case. We will not provide numerical results for the "one-shot" approach, since it is impractical in the UTM context, where delivery requests of users should be confirmed in a timely fashion.

As a baseline dimension, each voxel of the grid map represents 30m in the real world. The radius value attributed to each agent ranges from 15m to 30m and the attributed speed from 15m/s to 18m/s. The starting time of each agent is set uniformly at random from the interval [1s; 1800s], so several agents may start at the same time. The approaches are implemented in Java and run on a 3.2GHz Intel Core i7-8700 desktop with 16 GB RAM.

2.7.1 Results for Adaptation to Heterogeneous Case

In these experiments, the aim is to evaluate the efficiency of our geometrical method and spatio-temporal pruning to address instances with heterogeneous agents for CBS, ECBS, and CA*. So, we perform a Monte Carlo simulation on a $100 \times 100 \times 4$ grid with 5% obstacle density, which corresponds to likely no fly zones on a peripheral (non-urban) area. For each number of operations ranging from 50 to 500, we create 30 instances by randomly generating start locations

with associated goal locations for each agent. Fig. 2.6 shows the runtimes for ECBS over the different number of operations considered, the observed trends being the same for CBS and CA*. The use of computational geometry and spatio-temporal pruning provides solutions significantly faster than a straightforward use of voxel intersection.

2.7.2 Results for Comparing Different Approaches to the Processing of UAS Operations

Experimental Setup

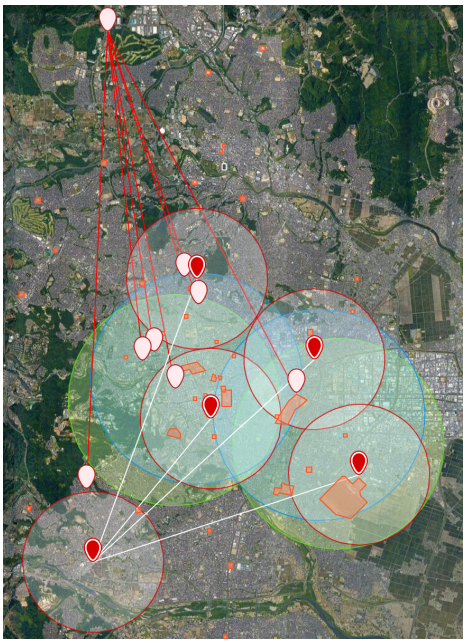


Figure 2.7: Map of Sendai, Japan, including candidate hub locations and service areas (circles). The white and red icons indicate the hub locations for Hub-to-Hub operations for company C and D respectively

.

Type of delivery	Company	#Hubs	Vicinity radius	# Deliveries per day
Hub-to-Home	A	2	3000 m	6,578 – 9,866
	B	2	3000 m	4,385 – 6,578
	C	5	2000 m	2,192 – 3,289
Hub-to-Hub	C	5 (1 main hub)	-	747 – 1,494
	D	8 (1 main hub)	-	8

Table 2.2: Parameters in the Sendai 2030 model case.

We base our scenarios on the data obtained from the study conducted by a consulting firm that projects service UAV deployment in 2030 in one region in Japan. The study is part of a large-scale governmental project on designing, specifying, and simulating the future UTM system, sponsored by *Anonymized.* We consider a 14.35 km $\times$ 17.10 km area. So we use a grid map of dimensions 478 $\times$ 570 voxels, which is delimited in altitude by a 2 voxels range (hence 60m range) relative to the elevation. The positions of static obstacles, i.e., blocked voxels, is fixed according to the given elevation map of the region and there are 41 no-fly zones. Based on the study, we consider three logistics companies that provide deliveries of goods and a Red Cross blood center that distributes blood samples in packages in the area. We distinguish two types of deliveries:

(1) Hub-to-Home deliveries: these are deliveries from hubs to service locations (homes) in a given service radius. Within these areas, we randomly distribute the service locations, and the minimum path length is fixed to 300m.

- Company A: 24 hubs, vicinity radius = 1500m, from 6,578 deliveries to 9,866 deliveries per day
- Company B: 24 hubs, vicinity radius = 1500m, from 4,385 deliveries to 6,578 deliveries per day
- Company C: 5 hubs, vicinity radius = 2000m, from 2,192 deliveries to 3,289 deliveries per day

(2) Hub-to-Hub deliveries: these are deliveries where specific hubs are connected to each other, and operations can only be conducted between those specific locations.

- Company A: 5 hubs, including one main hub from where all operations depart and four other hubs, from 747 deliveries to 1494 deliveries per day
- Company D: 8 hubs, including one main hub from where all operations depart and seven other hubs, 8 deliveries per day

One day of service represents 13 hours, from 8 a.m to 9 p.m. The total demand is estimated as up to 13,910 operations per day in normal season and

as up to 21,235 operations per day in busy season. Fig. 2.7 indicates the positions of the hubs and their associated vicinity area with different color for each company. In the study upon which our simulations are based, the frequency of actual deliveries is currently assumed to be uniformly distributed over this time frame. There is no indication in the study on when the delivery requests occur. However, we will also consider "peak" times with larger amounts of service requests and larger amounts of UAV traffic.

Unlike most existing works on MAPF techniques, such as warehouses where the size of the grid map is small and the occupancy by static obstacles is high [55], our scenario reflects a realistic use case (prepared by a major consulting company), where the environment size is large and has low density in terms of static and velocity obstacles. In terms of static obstacles occupancy, there is less than 1% blocked voxels of the total flyable volume of the area including no-fly zones and satisfying the altitude constraints. In terms of occupancy by dynamic obstacles (UAVs), the average number of UAVs in the air according to the study is about 200 UAVs at any time. Hence, by considering an average radius of 22m, we estimate that the average occupancy by UAVs at anytime is about 0.3% of the total flyable space. For a Hub-to-Home delivery the maximum flight time (for one way) is just a little over two minutes (2km distance with 15m/s speed), and most (one way) flights are just one minute. The topology of our flight paths differs from existing works in MAPF as all agents start from the same location (hub) but at different times as shown in Fig. 2.7.

User Scenario

We consider a service scenario where users (customers) request goods, such as food or other small items (under 5.5kg), which can be delivered by UAV. Such requests can refer to a delivery "as soon as possible" (ASAP), or a delivery for a specific later time, e.g., 7 p.m., on the same or next day. After putting the request, the user expects a response from the system almost immediately, such as "Your food item will arrive in 22 minutes", or "Your item will arrive tomorrow at 7:05 p.m.". This is an improvement over today's situation, where ASAP deliveries often have to change time estimates due to traffic, etc. Deliveries for specific future times often only provide a time window, such as "Your item will be delivered between 7 p.m. and 9 p.m.".

By 2030, we expect a better service for customers. Since customers want the timely response, the time efficiency of algorithms becomes very important. After a user submits the request for a good, it is processed by the UTM system. In simplified terms, a UAS service provider will first plan a flight path that avoids terrain and other static obstacles. This is a problem of single-agent planning and using A*-based search, it takes a few milliseconds for flight path up to 2 km. Then the UTM system performs Pre-Flight CDR. After the result is known – "Accept", "Accept with Modification", or "Reject" – the user is informed about the exact delivery time via the UAS service provider in a short time after putting the delivery request.

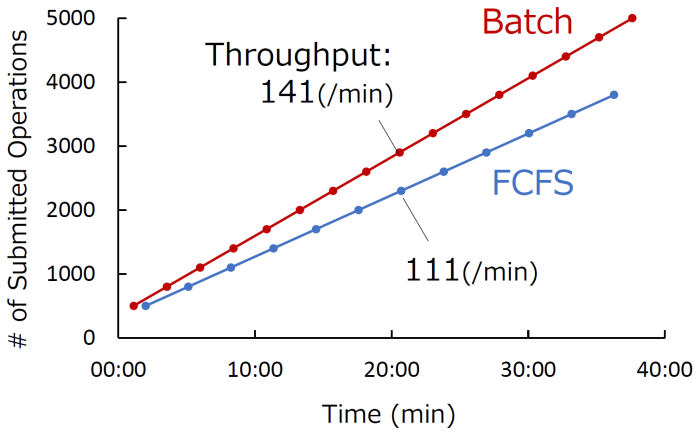


Figure 2.8: Comparison of throughput in function of the number of operations submitted between batch processing (ECBS) and FCFS processing (CA*).

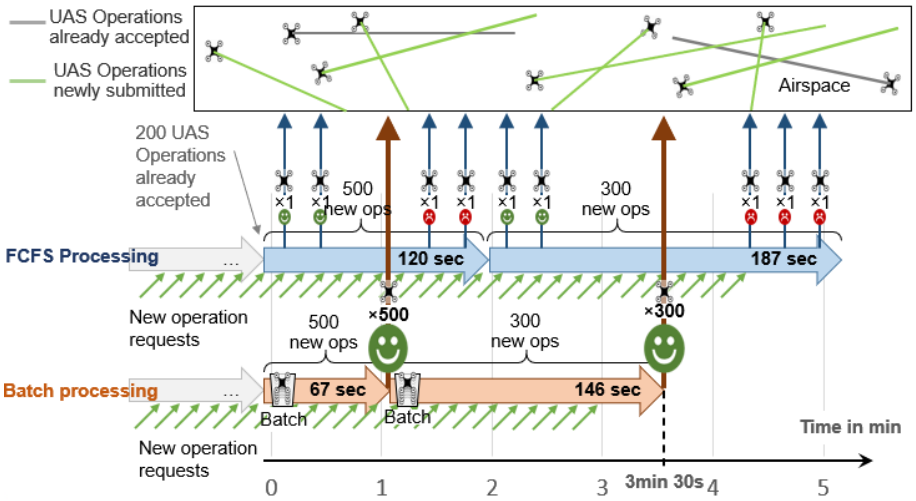


Figure 2.9: Representation example of throughput in function of the number of operations submitted between batch processing (ECBS) and FCFS processing (CA*).

Analysis

In our experiments, we study runtime and solution costs of FCFS (hereby labeled as CA*) and batch processing (hereby labeled as ECBS). Note that the

computational geometry and spatio-temporal pruning are hereby used in all approaches compared. Fig. 2.8 shows the results that are averaged over all iterations in our experiments. We fix a time limit of 10 min for each run. When a run takes more than 10 min in more than 15 iterations, we mark it as an excess in time limit and represent it with a dash line. We run 30 iterations for each experiment. The suboptimality bound w for ECBS is fixed to 1.5.

In our real world use case, we assume that peaks occur at certain hours, such as lunch or dinner times. In this case, 1000s of user requests are submitted in a small time window, which also creates more UAV traffic. We propose to compare the performances of ECBS with "batch" processing and CA*-based processing for FCFS, since CBS with batch processing being an optimal algorithm, is the method with the longest runtime overall.

Runtime and Throughput. The difference between ECBS with batch processing and CA* indicates that the former allows a significant gain in time compared to the latter when processing large number of operations submitted. So, processing UAS operation requests in "batches" can be advantageous over a method that processes each UAS operation one after the other. However, the advantage only applies to situations where a large number of requests has to process. If requests drop in with low frequency, CA*-based is clearly more suitable, as ECBS would have to "wait" until the batch is filled.

Figure 2.8 shows that batch processing with our adapted ECBS leads to a higher throughput, that is the number of operations processed during a fixed time interval, on average than FCFS approach based on CA*, assuming peak times with a large number of submitted operations. On the other hand, if UAS operation requests are sparse, FCFS would be more practical than having to wait for a batch to fill up.

Deviation and solution costs. The deviation is hereby defined as the difference in distance costs between the initially submitted flight path and the possibly replanned conflict-free flight path. Our results show that the average deviation is small for all algorithms, and our ECBS batch processing provides a slightly better cost in average. Note that UAVs have speeds of up to 18m/s, so even 40m mean deviation corresponds to just a few seconds delay in average.

Rejection rate. We define the rejection rate metric as the percentage of UAS operations rejected among those submitted. The meaning of "rejection" is that no solution was found for the instance in question. Overall, our ECBS batch processing mechanism shows lower rejection rates than CA*. Non-solvable cases can be a situation where a UAV from an already accepted operation reaches its destination location at a certain time and another UAV from a newly submitted operation starts its flight at a close time from the same location. In this case, no solution can be found for the latter UAV. This rate not only increases with the number of submitted operations, but also with the number of already accepted operations, since these are considered as spatio-temporal obstacles in both approaches.

2.7.3 Scalability

Overall, our proposed batch processing method based on ECBS is shown to be scalable with the number of UAV agents, as we incorporated a spatio-temporal pruning step (in 2.5.1) and geometrical computations (in 2.5.3) to efficiently handle heterogeneous UAV agents (see 2.7.1). We showed the scalability in terms of the number of UAV operations for our methods for Pre-Flight CDR by comparing FCFS processing to Batch processing in high demand instances.

2.8 Conclusions

We hereby address the Pre-Flight CDR (Conflict Detection and Resolution) problem as a key component of an UTM (Unmanned Aircraft System Traffic Management) system. For this purpose, we propose an extended formulation of MAPF (Multi-Agent Path Finding) to address heterogeneous agents with different start times and return paths in the service UAV situation. To our knowledge, we are the first to introduce an extension of CBS, and its suboptimal variant ECBS, to address the UTM setting of UAS operations for delivery in a realistic scenario.

To efficiently handle heterogeneous agents, we incorporate a mechanism relying on spatio-temporal pruning and computational geometry into CBS and ECBS. Then, we present a comparative study of FCFS (first-come first-served), encoded by Cooperative A* (CA*) and a "batching" approach, encoded by CBS and ECBS. Importantly, our results are based on a study that projects drone usage in a real area in Japan in 2030. While other works rely on 2D maps with various degrees of obstacles or agents, we decided to focus on a use case as realistic as possible. Our results suggest that our proposed ECBS with batch processing has better runtime performance and solution cost than CA*-based FCFS for a high number of simultaneous requests for airspace reservation in a realistic setting of drone usage. This is important as users of a UAV delivery system want feedback on their request for a delivery as soon as possible, as for any other online service. Further, the incompleteness of CA* means that no solution is found. If the value is 1%, it might amount to an average of around 200 "rejections" over the course of a day, given 21,000 deliveries per day. However, if the number of incoming requests is not sufficiently large, there is no reason to fill up the batch. In this case, CA* is more suitable.

Therefore, in this chapter, we proposed a Pre-Flight CDR method based on the ECBS algorithm that is scalable with the number of UAV operations to process.

Chapter 3

Pre-Flight Conflict Detection and Resolution for UAV Integration in Shared Airspace: Sendai 2030 Model Case

The increasing demand for services performed by Unmanned Aerial Vehicles (UAVs) requires the simulation of Unmanned Aircraft System Traffic Management (UTM) systems. In particular, Pre-Flight Conflict Detection and Resolution (CDR) methods need to scale to future demand levels and generate conflict-free paths for a potentially large number of UAVs before actual takeoff. However, few studies have examined realistic scenarios and the requirements for the UTM system. In this chapter, we focus on the Sendai 2030 model case, a realistic projection of UAV usage for deliveries in one area in Japan. This model case considers up to 21,000 requests for Unmanned Aircraft Systems (UAS) operations over a 13 hour service time, and thus poses a challenge for the Pre-Flight CDR methods. Therefore, we propose an airspace reservation method based on 4DT (3D plus time Trajectories) and map the Pre-Flight CDR problem to a Multi-Agent Path Finding (MAPF) problem. We analyze the air traffic topology of deliveries by UAVs, and discuss several metrics to better understand the complexity of air traffic in the Sendai model case.

3.1 Introduction

With the emerging use of Unmanned Aerial Vehicles (UAVs) for operations such as goods delivery, surveillance, search and rescue, and agricultural moni-

toring, the low-altitude air traffic is expected to grow significantly in the coming years [44, 49]. Several independent UAS (Unmanned Aircraft System) Service Providers (UASSPs) will support UAS Operators to task UAVs with limited capacities to execute a variety of tasks. In particular, commercial delivery by UAVs is expected to become a widespread service. Any 'conflict' [51], i.e., possibility of collision between UAVs, must be avoided. Therefore, an Unmanned Aircraft Systems Traffic Management (UTM) system [48, 49] has to incorporate Conflict Detection and Resolution (CDR) methods.

Proposed UTM concepts have a multi-layered architecture [44] similar to Air Traffic Management (ATM) [51]. For this purpose, the International Civil Aviation Organization's (ICAO) Global Air Traffic Management Operational Concept [39] defines different conflict management or 'redundancy' layers. These layers can be separated in three categories (see Fig. 3.1). Each of these distinct layers are applied during the different phases of a UAV operation processing, before the UAV takes off (strategic separation), and then during its actual flight (tactical separation) [39]:

- Pre-Flight CDR methods aim to provide conflict-free paths for all UAVs before their actual takeoff.
- In-Flight CDR methods ensure separation provision for the flight paths of all UAVs during flight, to account for dynamic events such as bad weather, emergency operations, and so on.
- Collision Avoidance methods, such as Sense and Avoid, are considered as a final fail-safe and are based on sensing technology onboard a UAV.

However, unlike ATM, the conception of a comprehensive UTM system is still under discussion. First concepts of operations have been released by NASA [49], yet the design and the integration of CDR methods into the UTM system needs more investigation. While many Sense and Avoid and In-Flight CDR methods have been recently proposed and studied [3, 28, 35, 42, 86, 96], Pre-Flight CDR methods remain mostly unexplored. Therefore, this chapter will focus on Pre-Flight CDR methods and their evaluation.

In Pre-Flight CDR, which is equivalent to Strategic Conflict Management in ATM as represented in Fig. 3.1, the conflict detection and resolution process occurs before the given UAVs take off and fly their scheduled flight path. Therefore, in this chapter, we only consider planned flight paths for UAVs and not UAVs flying in real time. Hence, Pre-Flight CDR does not include any real time considerations, i.e., surveillance issues and data communication from UAVs. These considerations belong to the In-Flight CDR phase which occurs after the Pre-Flight CDR phase, once UAVs take off and fly their flight plan. In In-Flight CDR, real-time requirements are important to effectively provide conflict detection and resolution maneuvers by In-Flight CDR methods in case of unexpected changes to the flight trajectories.

In [37], we proposed to model the Pre-Flight CDR problem as a Multi-Agent Path Finding (MAPF) problem. In MAPF, agents must avoid collisions while

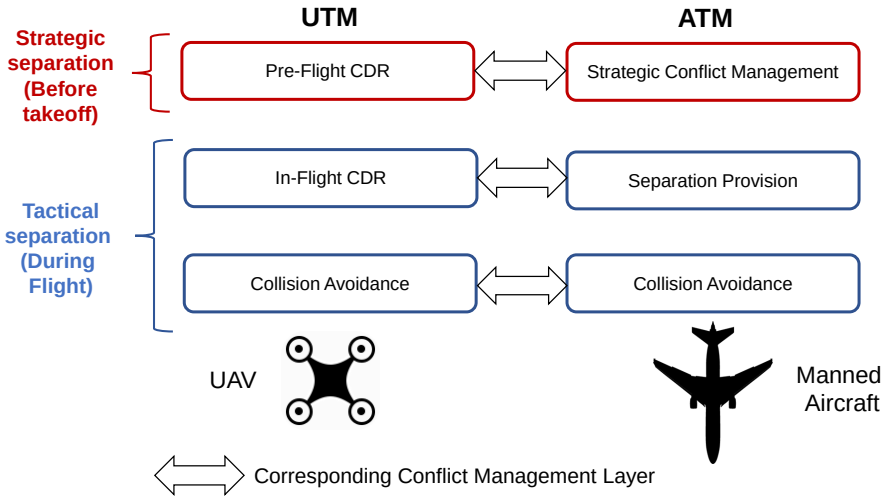


Figure 3.1: Conflict management layers in UTM and ATM.

following paths with given waypoints in space and time from given start locations to goal locations. However, the standard MAPF setting is limited and not directly applicable to the UTM domain. For instance, the standard formulation describes a "one-shot" problem, where all agents start simultaneously and all have a distinct pair of start and goal locations. It also assumes homogeneous agents that all have the same size, are contained inside cells of the given grid map, and all move by one cell at each time step. So we included heterogeneous agents and ongoing UAV operations processing in the formulation of the MAPF problem. In this chapter, in addition to the extension of the MAPF model formulation to Pre-Flight CDR, we conduct a deeper investigation into the Sendai model case and formulate a set of metrics to analyze the complexity of low-altitude air traffic.

This chapter presents the following main contributions:

- We analyze the necessity for advanced Pre-Flight CDR methods in future UTM systems. This necessity is motivated by the characteristics of the Sendai 2030 model case and an empirical comparison of 3D and 4DT (3D plus time Trajectories) airspace reservation methods.

- We characterize and analyze the UAV traffic topology for deliveries by UAVs and suggest several metrics to understand the complexity of air traffic in our model case.

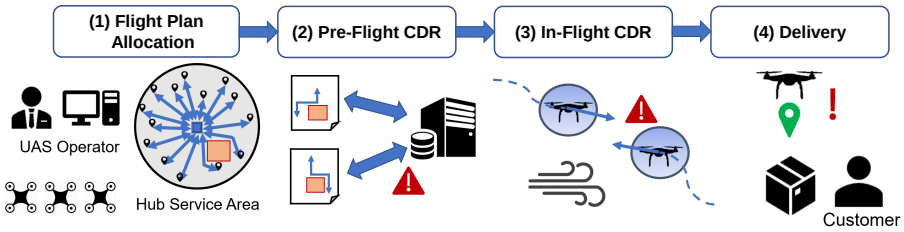


Figure 3.2: Flowchart of the different steps from operation request submission to delivery. (1) Operations requests are submitted by each independent UASSP; (2) Conflicts between all operations are solved in the Pre-Flight CDR phase which is the focus of this chapter; (3) When en-route, conflicts between flying UAVs can be solved with In-Flight CDR in case of dynamic events occurring; (4) A UAV reaches its delivery location, and returns to its hub afterwards.

3.2 Related Works

3.2.1 Conflict Detection and Resolution Methods and Metrics

In the ATM domain, extensive studies have been conducted for the conception and integration of CDR methods to all safety layers [51]. Differently, since UAVs have distinct capabilities, there are different challenges in the design of a UTM architecture. Following NASA's initiative [48], there currently exist several studies and works on the conception of a UTM system to enable the safe integration of UAVs in low-altitude shared airspace. Most of these works address the integration of In-Flight CDR methods [35, 42] or Sense and Avoid methods [28, 72, 78].

[44] introduce a multi-layered architecture based on a taxonomy for CDR, but they mainly address the conception of the In-Flight phase [43], which is assessed through Monte Carlo simulations. Also, the conception and evaluation of In-Flight CDR methods with simulations based on real world scenarios is discussed in [35, 96]. [68] introduce 4DT (3D plus time Trajectories) into UAV trajectory modeling, and they focus on the uncertainties and technical errors in UAV navigation. However, none of these works have presented practical methods to proactively de-conflict UAV traffic before the UAVs take off, i.e., Pre-Flight CDR methods.

By contrast, high altitude air traffic is already well analyzed, and ATM systems have been deployed for years [5]. In the evaluation of ATM systems, a variety of metrics have been proposed to characterize the complexity of air traffic. However, due to the presence of air traffic controllers and pilots, these studies mostly rely on human workload as one of the defining factors of traffic complexity [17, 50, 64].

However, in the UTM context, new metrics must be defined to meet the challenges of increased automation and new traffic patterns. Studies on the

predicted low-altitude air traffic are very recent. In particular, [32] proposed metrics to assess the density of the predicted low-altitude air traffic in Paris. Nevertheless, their study is focusing only on the In-Flight CDR situation. Further, their simulation scenarios are randomized traffic patterns that do not realistically represent the situation of deliveries by UAVs.

3.2.2 Multi-Agent Path Finding

The Multi-Agent Path Finding (MAPF) problem has been mainly studied without real world deployment considerations, with many works proposing approaches to solve instances for 2D video game benchmarks [26,75]. In the MAPF setting, agents located in a graph must move according to the given graph representation to their goal locations without colliding with each other for each move. However, new directions were established recently to handle complex real-world scenarios, such as robot path finding in Amazon Robotics warehouses [55, 56]. Here, agents are split into groups and a set of tasks is given to each group [38,54]. The objective is to allocate tasks to each agent while providing conflict-free paths.

[56] introduce a decentralized approach and focus on ongoing task allocation. In the UTM context, we assume that every agent is assigned a given start and goal location. So, we do not consider the task allocation process in our work.

[83] also propose an online version of MAPF solvers, where agents can replan their paths to solve conflicts when new agents appear. In our context, we do not allow replanning of previously generated paths in the Pre-Flight phase for practical reasons, i.e., UAS operations that are accepted cannot be modified anymore, and since UAVs may belong to different UASSPs, this would potentially impact several UASSPs.

3.3 The Need for Advanced Pre-Flight Conflict Detection and Resolution Methods

UAS Operators will use UAS Service Providers (UASSPs) to integrate UAS operations into low-altitude airspace, which is shared among several independent UASSPs. Unlike the human-centered ATM system, the UTM system is an automated system that must ensure conflict-free paths for a large number of UAVs, with the steps depicted in Fig. 3.2, UAVs will be tasked to perform deliveries anytime during a day.

To demonstrate the need for advanced Pre-Flight CDR methods, we first introduce the Sendai 2030 model case as a realistic scenario of future deliveries by UAVs and the representations considered in our study. Then, we motivate the necessity of Trajectory Based Operations (TBO) to handle the expected scale of UAS operations requests.

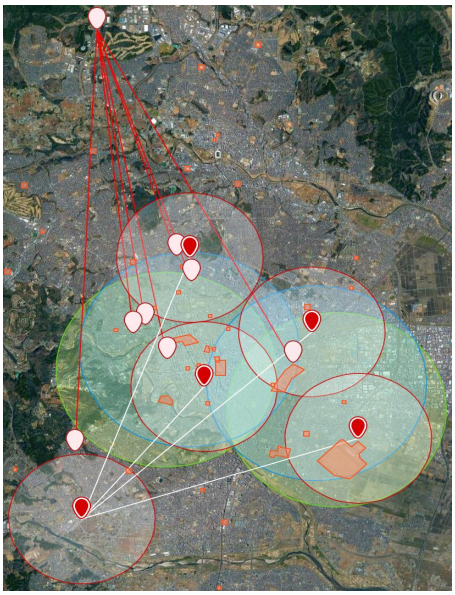


Figure 3.3: Map of Sendai, Japan, including candidate hub locations and service areas (circles). The white and red icons indicate the hub locations for Hub-to-Hub operations for company C and D respectively.

3.3.1 Sendai 2030 Model Case

This section describes the Sendai 2030 model case, which is based on a study that aims to project UAV service demand in 2030 in Sendai, Japan. The study was conducted by a consulting company, as part of the NEDO UTM Project, a large-scale governmental project on designing, simulating and specifying the UTM system.

The dimensions of the considered area in the given region are of 14.35 km×17.10 km. The study specifies that UAVs are allowed to fly between 90 m and 150 m, relative to the elevation of the terrain, hence, there is a 60 m altitude range.

Type of delivery	**Company**	**#Hubs**	**Vicinity radius**	**# Deliveries per day**
Hub-to-Home	A	2	3000 m	6,578 – 9,866
	B	2	3000 m	4,385 – 6,578
	C	5	2000 m	2,192 – 3,289
Hub-to-Hub	C	5 (1 main hub)	-	747 – 1,494
	D	8 (1 main hub)	-	8

Table 3.1: Parameters in the Sendai 2030 model case.

The scenarios considered by this study are delivery services provided by UAVs that fly from hubs locations to designated service locations. The study considers three major logistics companies (hereby anonymized as A, B and C, see Table 3.1) that provide deliveries of goods such as mail and package delivery, as well as a Red Cross blood center D which collects blood samples between medical centers.

The different hubs locations are set by the study considering the expected demand in the area. The assignment of the service locations is assumed to be done independently by UAS Operators within a given service radius around a hub.

Figure 3.3 depicts positions of hubs and their associated service area with a different color for each company.

We distinguish two types of deliveries (see Table 3.1 for concrete values derived from the study of the consulting firm).

- *Hub-to-Home*: deliveries performed from hubs to service locations (homes) located within a given service radius. Inside these areas, we fix the minimum flight path length to 300 m.
- *Hub-to-Hub*: deliveries performed between given pairs of hubs only.

The study estimates that, by 2030, the UAVs performing these delivery services will be able to carry up to 5.5 kg of payload, fly at 23 m/s maximum horizontal speed, ascend at 10 m/s and descend at 3 m/s. The expected operation time is 27 minutes.

In this study, one day of delivery service represents 13 hours, from 8 am to 9 pm. According to customers' demand in the region, weight of deliveries, and expected capabilities of UAVs, the study estimated that there is a total demand of up to 13,910 operations per day in normal season and up to 21,235 operations per day in busy season considering the given companies altogether[1].

In this chapter, we assume that each of the given companies uses its own UASSP.

3.3.2 Map and UAV Representation

The representation of our space is a 3D grid map composed of cells with 30 meters edge length. Thus, according to the Sendai model case area dimensions, we consider a 3D grid map of dimensions 478×570 cells, which is delimited in altitude by a 2 cells range (60 m range) relative to the elevation. The positions of static obstacles, i.e., blocked cells, is fixed according to the given elevation map of the region and there are 41 distinct no-fly zones fixed by the study.

We consider quadcopter UAVs that have holonomic motion, and can move in any direction at any time, or hover. In CDR, each UAV is conceived as a sphere of "total" radius r, which is composed of several layers as shown in Fig. 3.4 and Eq. 3.2, including the physical size of the UAV *BodyRadius*, the

[1] UAS reliability and risk assessment safety were not included in the survey provided by the consulting firm.

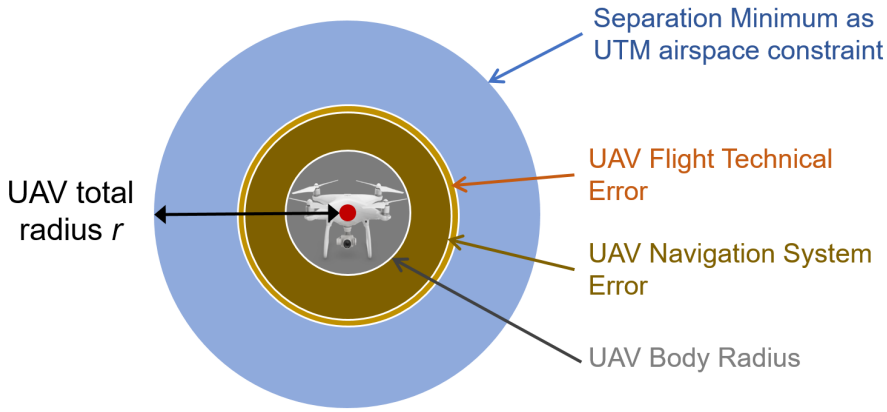


Figure 3.4: The different layers considered in the accumulated radius r for each UAV.

UAV navigation system error NSE (i.e. uncertainty in actual position), UAV flight technical error FTE (i.e. uncertainty in deviation from straight line of flight), and the separation minimum $SeparationMinimum$ defined by the UTM regulator. As in ATM, these parameters relate to operational risk [45].

$$Constructor = BodyRadius + NSE + FTE \tag{3.1}$$

$$r = Constructor + 0.5 \cdot SeparationMinimum \tag{3.2}$$

Although we do not consider the precise kinematics to model each UAV trajectory, our simulations are hereby informed by real-world data provided by the study. The study provided different existing UAVs specs in the Sendai 2030 Model Case such as wind stability, range, payload, and autonomy. The reliability of these parameters has been tested in real world experiments with experimental flight tests. With these real-world data, we determined the different accumulated radii values r for each considered UAV, and thus we hereby consider values ranging from 15 m to 30 m.

3.3.3 CDR Assessment

Next, we want to estimate the number of conflicts that would have to be handled by an In-Flight CDR method, if no Pre-Flight CDR is applied.

Each agent is represented by a sphere of given radius r_i, and a center position p_i. To avoid a conflict, we need to prevent any violation of the minimum separation distance between two agents a_i and a_j, i.e., the sum of their respective radii, $r_i + r_j$. Hence, we define the following constraint to ensure there is

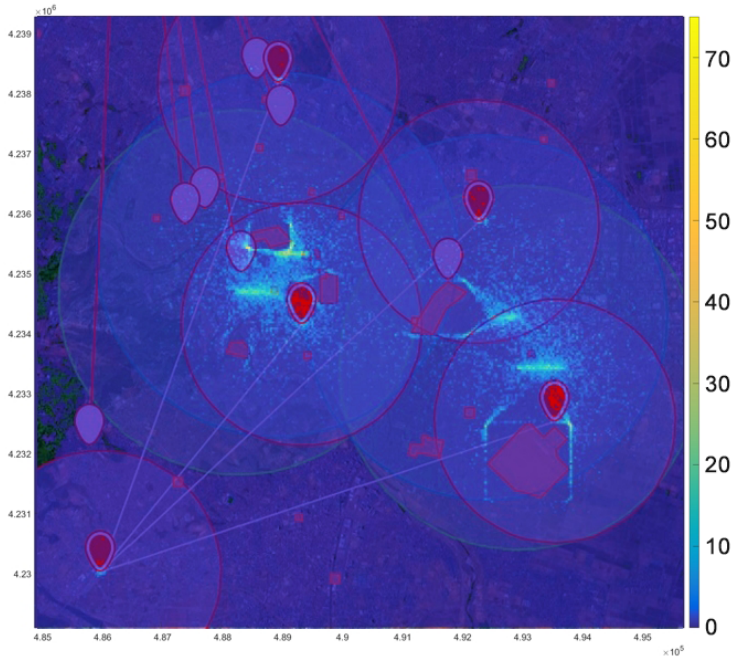


Figure 3.5: Visualization of the distribution of conflicts in the Sendai 2030 model case. Brighter points have more conflicts.

no conflict:

$$\forall t, dist(p_i(t), p_j(t)) > r_i + r_j \tag{3.3}$$

The following study calculates the frequency of conflicts in the Sendai 2030 model case, assuming no conflict resolution is applied. We assume that each UAV has a total radius set to 30 m. Table 3.2 shows that a high number of conflicts occur for different numbers of operations submitted during a 13 hours service day, assuming no CDR methods are applied. Figure 3.5 visualizes the distribution of conflicts for a simulated instance of 19,573 operations submitted during one day.

We observe that more than 80% of the reported conflicts occur "enroute" (during flight), i.e., at least 50 m away from the takeoff/landing location or delivery location. Hence, an In-Flight CDR method would need to be frequently activated to handle the conflicts. Therefore, Pre-Flight CDR methods are an important "redundancy" layer to ensure the safety of the airspace.

Moreover, we analyzed the number of UAVs involved in the same conflict, from two to four UAVs (see Table 3.2). The majority of conflicts involve two UAVs, which can be seen as simple case of pairwise resolution. However, there is a non negligible number of 3-way and even 4-way conflicts, which would require more complex resolution methods.

# Operations	**20,681**	**17,935**
# Conflicts	17,124	14,183
Average # Conflicts per Operation	0.828	0.790
2-way Conflicts (% of the total #)	16,705 (97.55%)	13,804 (97.33%)
3-way Conflicts (% of the total #)	417 (2.44%)	377 (2.66%)
4-way Conflicts (% of the total #)	2 (0.01%)	2 (0.01%)

Table 3.2: Number of conflicts in two instances of simulating deliveries in the Sendai 2030 model case.

3.3.4 Trajectory Based Operations Concept

In the new generation of ATM systems, the concept of Trajectory Based Operations (TBO) is expected to be deployed [74]. TBO consists of a coordination of four dimensional (3D plus time) trajectory (4DT) predictions and executions, whereby an aerial vehicle must follow given waypoints in space and time. With this "strategic 4D trajectory de-confliction" [74], TBO is expected to reduce air traffic complexity, by reducing the number of conflicts to be solved tactically, i.e. during flight, and its efficiency has been assessed independently in recent works [66, 89].

Similarly, in the proposed strategic deconfliction phase for UTM, our Pre-Flight CDR method solves conflicts between operations before the associated UAVs take off.

We hereby assume that Pre-Flight CDR is performed by a central entity which receives the operations of multiple UAS Service Providers (UASSPs). Since the Pre-Flight CDR process occurs before the given UAVs take off, whereby conflicts are predicted and resolved relying on 4DT, surveillance functions are not included in this phase as UAVs are not actually flying in this step. Once the predicted conflicts are solved, the central entity communicates the modified 4D trajectories to the UASSPs, which transmit the flight plans to their respective UAS Operators. In the Pre-Flight CDR phase the efficiency of the communication medium is not critical, and we hereby assume that communication is done via Internet as described by NASA UTM [71].

In the conception of the Pre-Flight phase for an UTM system, we will assess the efficiency of the 4DT paradigm in comparison to simpler options such as the use of 3D trajectory reservation.

Figure 3.6 shows the concept of 4DT and 3D conflict detection (CD). With 3D trajectory reservation, the entire trajectory of each UAV is considered as a spatial obstacle for other operations during its entire flight duration.

Next, we assess the effectiveness and practicality of the use of 4DT and 3D trajectory reservation in terms of the number of UAS operations rejected, which is an important practical consideration for the UTM system (see Fig. 3.7). We hereby use a simple first-come first-served (FCFS) approach, implemented by the Cooperative A* (CA*) algorithm [77], to plan all UAS operation requests

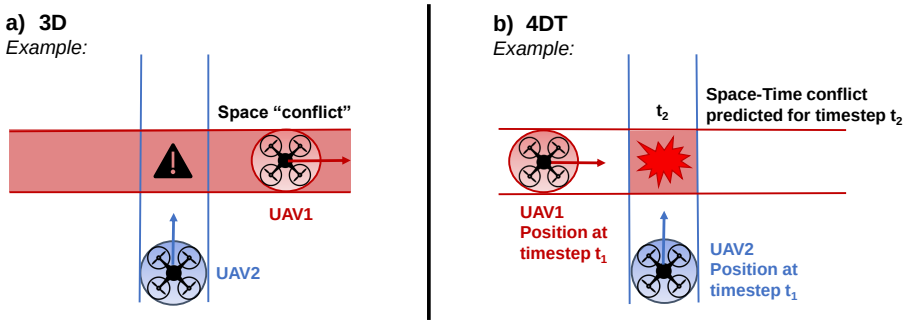


Figure 3.6: Examples of conflict detection (CD) in 2D for clarity of the representation: a) in 3D, in this example, UAV1 and UAV2 flight paths intersect spatially but there is no actual conflict, and the entire flight path of UAV1 is considered as an obstacle for UAV2; b) in 4DT, in this example, an actual conflict is detected between UAV1 and UAV2 flight paths in 4DT, as UAV1 and UAV2 cannot be at the same location at the same time.

in the simulations. Considering a 1 hour time window in normal and busy season in the Sendai 2030 model case, the percentage of rejected operations is significantly higher when using 3D trajectory reservation than 4DT in both CD (no resolution) and CDR cases, as the reserved trajectories occupy more space in an inefficient manner as there is not necessarily an actual conflict as shown in Fig.3.6 a).

4DT CD is also impractical as it rejects almost 50% of the operations, and the UAS Operator has to resubmit every other UAS operation. In contrast, 4DT CDR presents a very low rejection rate, whereby we observed that most of the rejected operations are due to conflicts that occur close to hubs.

3.4 Pre-Flight CDR as a MAPF Problem

First, we define the Multi-Agent Path Finding (MAPF) problem for Pre-Flight CDR. Then we discuss strategies to handle ongoing processing of UAV operation requests.

3.4.1 Pre-Flight CDR Problem Model

A MAPF problem can be extended to a Pre-Flight CDR problem as follows: An instance of our problem is composed of N operations $O = \{O_1, \ldots, O_N\}$, which are performed by agents (UAVs), and an undirected graph $G = (V, E)$, which is a 26-neighbor cubic grid allowing diagonal moves. Agents can 'move' along an edge of G or can 'wait' on a vertex of G. An agent a_i assigned to perform O_i is characterized by:

- A *radius* r_i: as in Sect. 3.3.1;

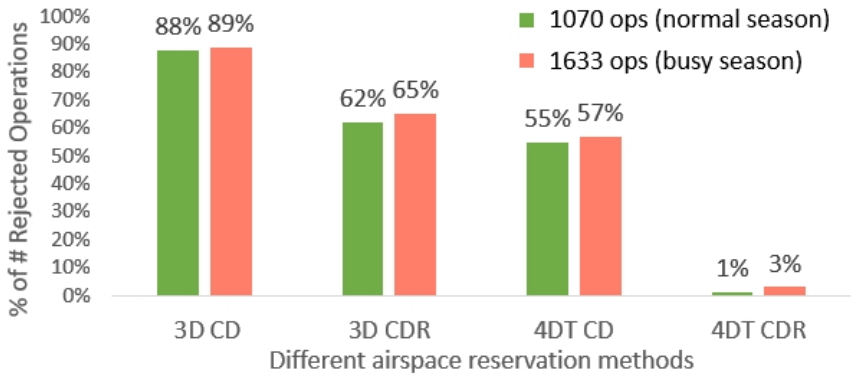


Figure 3.7: Percentage of rejected UAS operations for each airspace reservation method.

- A *speed* sp_i: the given speed of a_i is considered uniform on the whole path. In the UTM context, this would be a constraint to ensure conflict-free paths generation [68];
- A *start* s_i and *goal* g_i location: an operation O_i is composed of a pair of paths, an outbound path and a return path. The outbound path leads from a hub location s_i to a delivery location g_i, and the return path is assumed to be symmetrical to the outbound path. We assume a fixed duration δ_i for the duration between when a UAV lands to deliver some good and start returning to its hub location;
- A *start time* $t_i^s > 0$: the time at which the operation must start, hence when the agent takes off.

In the UTM context, we define the altitude constraints for any given point in space of coordinates (x, y, z) and which belongs to the path of an operation as:

$$elev(x, y) + alt_{min} \leq z \leq elev(x, y) + alt_{max} \tag{3.4}$$

With $elevation(x, y)$ the elevation value of the point of coordinates (x, y) in a path, and alt_{min} and alt_{max} the fixed altitude bounds.

Moreover, since each agent's delivery operation includes an outbound path and a return path, we must ensure that the precedence relation between the outbound path and the return path is satisfied.

In the UTM context, low-altitude airspace is often seen as a common resource shared among cooperative agents that would accept any deviation from their initial path to minimize the overall air traffic. Thus, the objective we hereby adopt remains the same as in standard MAPF, i.e., to minimize the sum of individual costs: $\min \sum_{O_i \in O} T_i$, with T_i the total cost of the operation O_i,

which is the total distance or the total duration of an operation. A solution consists of conflict-free paths for all N operations such that no violation of minimum separation occurs.

Unlike high altitude airplanes, UAVs come in a wide variety of different sizes, speeds, kinematics and so on. We consider "heterogeneous" UAVs with different sizes and speeds, whereby speeds range from 15m/s to 18m/s and sizes range from 15m to 30m.

Here, we only consider quadcopters which are holonomic agents, and thus can directly follow any given trajectory without particular motion constraints. Hence, the precise integration of kinematics is not in the scope of this chapter. To address heterogeneous agents, our conflict detection process based on the computation of a Time To Collision (TTC) [24, 37] allows to accurately detect conflicts between UAVs of different sizes and speeds by considering their 4D trajectories. We incorporated geometrical computations [24, 37] into A*-based MAPF solvers such as Enhanced Conflict Based Search (ECBS) and Cooperative A*. Moreover, in the conflict detection step, we apply a spatio-temporal pruning to filter out operations that may not overlap in flight times or may not intersect spatially.

As described in Section 3.1, in the Pre-Flight CDR phase, the conflict detection computations are performed considering the whole flight plans of UAVs before takeoff. Thus, there are no real time requirements for the computations of conflict detection in Pre-Flight CDR, unlike In-Flight CDR methods where such computing issue would be critical.

3.4.2 Ongoing Processing of UAS Operation Requests

With the steps depicted in Fig. 3.2, UAVs in the Sendai 2030 model case are tasked to perform deliveries anytime during the day. This requires ongoing processing of UAS operation requests.

We present two possible strategies to process incoming requests for UAV operation: (1) first-come first-served (FCFS) processing and (2) batch processing. For FCFS processing, we adapt the Cooperative A* algorithm [77]. For batch processing, we adapt the Enhanced Conflict Based Search (ECBS) algorithm [6], which is a state-of-the-art MAPF solver.

FCFS Processing with Cooperative A*: A simple baseline approach commonly adopted in traffic management is the use of first-come first-served (FCFS) processing. Here, every time an operation is submitted, it is processed in the order of arrival, while previously processed operations are considered as spatio-temporal obstacles for later operations. This is equivalent to the concept of the Cooperative A* method that uses a reservation table for each given agent planned in order [77]. This type of approach is proven to be unbounded sub-optimal (the solution cost can be very far from the optimal) and incomplete (a solution might not be found even though it exists).

Batch Processing with ECBS Algorithm: Batch processing has been shown useful in domains such as in data processing [99].

Let us assume our problem starts with an empty airspace and a first MAPF instance contains a set of UAS operations. So, the first MAPF instance is always a "one-shot" instance. Then, while the agents execute their generated plan, a new set of UAS operations appears that might be in conflict with the already accepted UAS operations. We refer to each given set of UAS operations as a *batch* B_i that contains a certain number of delivery operations.

Hence, two types of conflicts can occur for an agent from a later batch. It can be in conflict with another operation (1) from the previous batch or (2) from the same batch.

(1) Since we assume that previously accepted operations cannot be modified, we consider the paths of these operations as spatio-temporal obstacles. So, a first step of detecting and solving these type of conflicts is performed, and the existing spatio-temporal obstacles are considered when replanning for conflicts in (2).

(2) ECBS considers conflicts between agents of the same batch and solves conflicts between them.

ECBS is complete and suboptimal within a fixed bound w with respect to the sum of individual costs. More details on the functioning of the algorithm can be found in [6].

3.5 UAV Traffic Topology Analysis

To understand the difficulty of Pre-Flight CDR encoded as a MAPF problem, we need to characterize the properties of our space. [26] note that evaluating the complexity of some MAPF instance is not straightforward.

Our environment is a 3D space where agents can change altitude while satisfying elevation constraints. The static obstacles, i.e. no-fly zones, have an occupancy of 1% of the total flyable volume of the area, but are concentrated in a smaller area of the region.

Further, we estimate that the average occupancy by UAVs at anytime is about 0.3% of the total flyable space, assuming a total separation radius of 22 m. Even though this percentage might seem quite low, the proportion of UAVs in the total flyable volume is not a sufficient metric to indicate whether an instance is dense or not. As depicted in Fig. 3.3, many operations are concentrated in a smaller area of the region, and the major part of the considered space is practically empty.

Our situation is different from standard MAPF problems with ground vehicles that can consider a high number of static obstacles but less agents. In particular, we can distinguish two types of ground traffic situations commonly used in the literature, notably in the MAPF field:

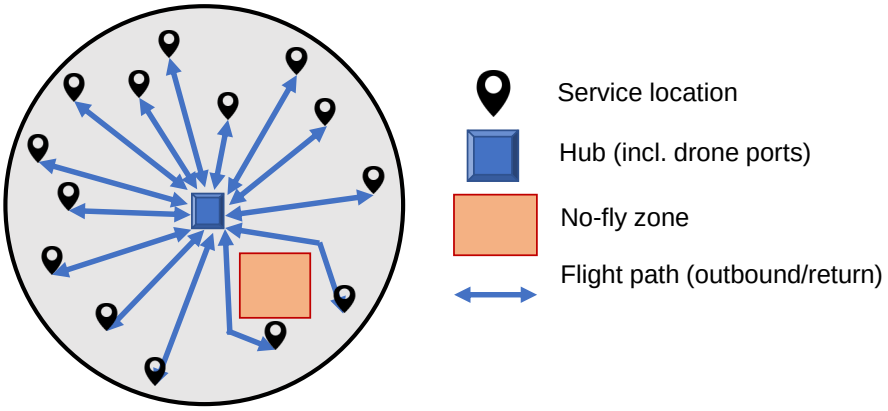


Figure 3.8: Schematic topology of UAVs' paths topology in our delivery scenario.

- *Uniform random instances.* In these instances, agents are positioned randomly in the space to produce various encounter situations. This relates to Monte Carlo simulations which aim to validate a given method by stressing the simulation conditions in repeatedly generated random samples. Thus, these instances are not meant to reflect a realistic traffic situation, but are rather a method to assess the robustness of CDR techniques. While evaluating the performance of such techniques in these conditions is important, it is also paramount to study and analyze realistic traffic scenarios. To the best of our knowledge, we are the first to propose an analysis of realistic UAV traffic patterns. In particular, for delivery use cases, the traffic topology has a specific pattern that we hereby describe unlike Monte Carlo random patterns.

- *Amazon warehouses.* In this instance, automated ground robots are tasked to navigate to inventory pods and move them from their storage locations to packing stations. Thus, a "corridor-like" traffic is induced, where agents move in limited directions in a cluttered environment.

In these 2D scenarios, the number of agents are limited and mostly uniformly distributed in space. Grids range from 8×8 to 100×100 cells for largest simulated warehouses, and the number of agents depend on the size of the warehouse that can range from just ten to a few hundreds [38, 55].

In the case of UAV air traffic, knowing the number of vehicles in the same area alone conveys only partial information on the complexity of air traffic. For example, the two patterns in Fig. 3.9a) have the same number of agents, but the different traffic patterns lead to an organized traffic flow with no conflicts in one case ("Stream"), and to a conflict-laden situation in the other one ("Uniform random").

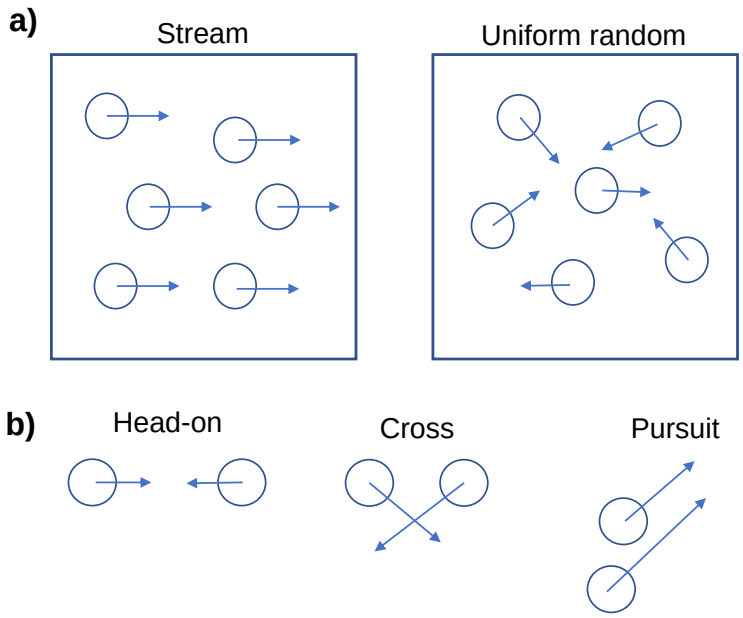


Figure 3.9: a) Different traffic patterns with same number of agents; b) Possible conflicts configurations.

Our scenario is also different from the existing benchmark scenarios for MAPF in terms of agent movement. Our use case induces a specific path topology, where UAVs are flying from the same hub location to several given service locations, as shown in Fig. 3.8. Thus, for each hub, we can observe a "flower-like" traffic pattern, where UAVs are concentrated around the same hub location, which can be a source of several conflicts.

Several hubs can have their service area overlap with each other. Therefore, we can distinguish different configurations of conflicts which can occur in our context, as shown in Fig. 3.9b). A *head-on* conflict can occur while one UAV is departing from a hub and another UAV is returning to the same hub. A *cross* conflict can occur when two UAVs of the same hub are tasked in intersecting paths or when the service areas of two hubs overlap, thus allowing paths to cross. A *pursuit* conflict can occur when two UAVs, either from the same hub or from two close hubs, are going in similar directions but each has a different speed.

3.6 Air Traffic Complexity Metrics

The complexity and the understanding of the air traffic is fundamental for the evaluation of any manned or unmanned air traffic management system. Therefore, we describe several metrics to characterize the air traffic complexity. [32] describe metrics that rely on proximity measures between UAVs in a limited setting. Instead, we propose a richer set of metrics in the context of Pre-Flight CDR.

3.6.1 Number of UAVs in the Air

We can first distinguish two metrics regarding the number of UAVs in the air: (1) the average number of UAVs in the air during a given time interval (e.g. 1 h), and (2) the average number of UAVs that are flying simultaneously at any time. Both can be seen as a first simple indicator of air traffic complexity.

3.6.2 Proximity Measures

As previously stated, measuring the number of UAVs in the air only provides partial information on an instance complexity. Quantifying the interactions between UAVs allows for a more accurate understanding of whether an instance is more complex than another as depicted in Fig. 3.9. If no UAV has to change its nominal path then the complexity is low, while on the other hand, if UAVs have to constantly avoid each other, then the complexity is high. Thus, an important indicator that we hereby propose is to measure the proximity between UAVs to indicate their interactions.

[32] proposes to compute airspace density metrics by randomly selecting a UAV and computing its proximity to others at that given time step. This method aims to evaluate how frequent a UAV would have to apply the given In-Flight CDR mechanism. Their study relies on a large number of samples.

For the Pre-Flight case, we propose a different methodology, which does not take random snapshots. It is deterministic as we take into account the flight paths of all given UAVs. We describe and quantify how close UAVs move toward each other. These metrics indicate (1) how close the closest UAV is on average, and (2) how many UAVs are in the immediate vicinity of a UAV on average [32]. In Pre-Flight CDR, the paths are entirely projected in 4D before all UAVs depart, i.e., we already assume conflict-free paths.

Average Minimum Closing Time: This indicates how close a UAV ever gets to another UAV. For each UAV, also called "ownship", we compute the time of the closest point of approach t_{CPA} between the ownship and the other UAVs, and we average over all ownships:

$$t_{CPA} = \frac{(p_i(t) - p_j(t))(v_i - v_j)}{|v_i - v_j|^2} \tag{3.5}$$

with v_i and v_j the velocities, and p_i and p_j positions for UAVs i and j at each time step t.

Average Number of Close UAVs: For each ownship, we count the number of UAVs that have a closing time of less than 15 seconds of the ownship during its entire flight path. It indicates the number of UAVs that the given ownship gets close to during its flight. The 15 second proximity threshold is based on [32] as a reference value for different types of aircrafts.

Average Number of Close UAVs per Time Step: For each ownship, we divide the previous metric by the duration of its flight. This provides an estimate of the number of UAVs that the given ownship gets close to, at anytime.

3.6.3 Number of Conflicts and Modified Trajectories

First, the number of conflicts detected in a scenario is an important indicator of the complexity of air traffic. The larger the number of conflicts, the more difficult it is to compute new paths that satisfy all safety constraints. Note that for batch processing, we distinguish two types of conflicts (see Sect. 3.4.2). For clarity, this metric will only count the number of conflicts detected within the same batch (i.e., high level nodes in ECBS).

Second, we count the number of UAS operations that had to be replanned, thus modified from their originally submitted (nominal) flight paths, whereby one trajectory might need modification to solve multiple conflicts. Note that this measure also includes paths that were only modified because of conflicts with previously accepted operations considered as spatio-temporal obstacles.

3.7 Simulation Experiments

In this section, we describe our simulation experiments.

- We compare the throughput of UAS operations of "batch" processing with ECBS to FCFS (first-come first-served) processing with Cooperative A* (CA*).
- We estimate the complexity of air traffic in the Sendai 2030 model case with our proposed metrics.

All approaches are implemented in Java and run on a 3.2GHz Intel Core i7-8700 desktop with 16 GB RAM. For each experiment, we generated 30 different instances.

In the Sendai 2030 model case study, the frequency of deliveries is assumed as uniformly distributed over the 13 hours service time frame. The study does not specify when the delivery requests occur. However, we can assume "peak" times with large amounts of service requests.

Airbus Altiscope Paris Case	Sendai 2030 Model Case
In-Flight case (TCAS), UAVs might need to deviate	Pre-Flight case, UAVs just follow the predetermined paths
1 "ownship" considered at the center of the given area	All "ownships" considered, and average
Random snapshots at certain time steps during the simulations	Snapshots at all time steps for each flight path during the simulations, and average
Random start and goal locations, different traffic patterns: uniform random, quadrant, stream, etc.	Specific hub locations with flower-like topology, distribution of service locations in a given vicinity range
10 km×10 km and 300 m altitude range area	14.35 km×17.10 km and 60 m altitude range area
∼2,000 flights per hour, ∼500 UAVs anytime	∼1,000 operations (2,000 flights) per hour, ∼200 UAVs anytime

Table 3.3: Comparison of the features in Paris case study [32] and Sendai 2030 model case.

3.7.1 Results for Air Traffic Complexity Metrics

# Ops submitted	Avrg minimum closing time (in seconds)	Avrg # of close UAVs	Avrg # of close UAVs per time step
100	28	0.7	0.003
300	15.5	2.1	0.007
500	12.1	3.9	0.013
1000	7.5	8.2	0.028
1500	6.2	11.7	0.053

Table 3.4: Results for the proximity metrics in the Sendai 2030 model case.

In this section, we present the results for air traffic complexity metrics described in Sect. 3.6.

Number of UAVs in the Air

First, we estimate from the Sendai 2030 model case that the average number of UAVs flying in the airspace within a 1 hour time interval is 1070 UAVs in normal season and 1633 UAVs in busy season. Second, the average number of UAVs simultaneously flying at any time is estimated to be about 200 UAVs.

Table 3.3 compares our Sendai 2030 model case to the Airbus Paris model case.

Proximity Measures

The average minimum closing time becomes smaller than 15 seconds for more than 500 submitted UAS operations (see Table 3.4). This indicates that from this amount of submitted operations, all UAVs tend to fly closer to each other on average, and thus the traffic can be considered as dense given this threshold.

The values for average number of close UAVs indicate that for 500 UAS operations submitted, a UAV flies at proximity of an average of 4 others during its entire flight path, and as expected the number of encounters increases with the number of operations. Taking into account the total duration of a flight, a UAV gets close to less than 1 other UAV on average at all times with respect to the 15 seconds threshold.

Number of Conflicts and Modified Trajectories

# Ops already accepted	**# Ops submitted**	**ECBS**	**CA***
		# Paths modified (# High level nodes expanded)	**# Paths modified**
200	100	50 (14)	55
	300	146(72)	160
	500	282 (258)	339
400	100	67 (13)	67
	300	183 (70)	175
	500	344 (275)	339
700	100	80 (10)	81
	300	202 (75)	208
	500	388 (272)	399

Table 3.5: Proportion of paths modified and number of conflicts (high level nodes expanded in ECBS).

We start with the results on the number of conflicts. In the simulations, our adapted ECBS algorithm was used to de-conflict operations, and without loss of generalization, no previously accepted operations are considered. The UASSPs are the ones considered for each company from Sect. 3.3.1, hence there are 4 UASSPs.

In these experiments, we aim to analyze the spatial distribution of conflicts, first, between UASSPs which have several hubs, then, between the different hubs without distinction of UASSP.

Figure 3.10 and Fig. 3.11 show that conflicts mostly occur between UAVs starting from the same hub (Inter-Hub), and thus same UASSP (Inter-UASSP). This result suggests that a "localized" use of a Pre-Flight CDR method on each UASSP level could reduce the computational effort to de-conflict UAS operations across independent UASSPs.

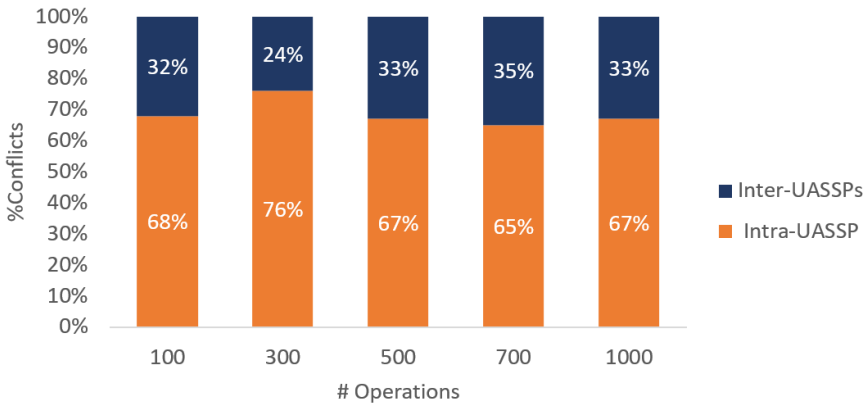


Figure 3.10: Average % of conflicts for UASSPs.

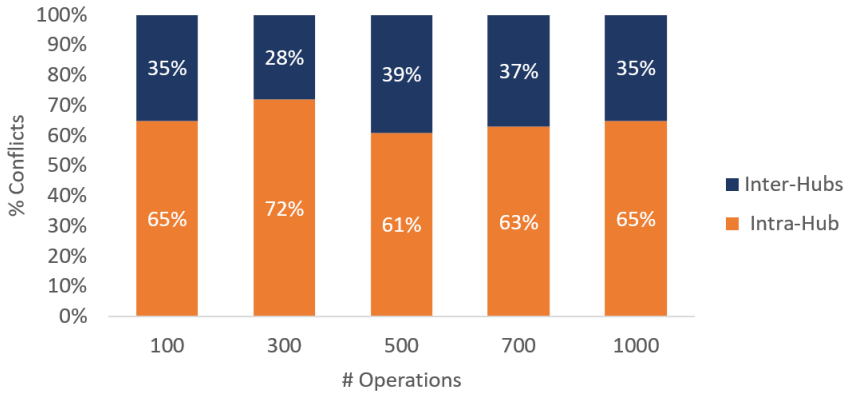


Figure 3.11: Average % of conflicts for Hubs.

Figure 3.12: Distribution of conflicts.

Table 3.5 shows the results for the number of modified UAS operation trajectories. First, we observe that around 50% UAS operations on average are involved in at least one conflict, so that their nominal flight path had to be modified. This information can be seen as one indicator of air traffic complexity. Second, the results for ECBS and CA* are almost identical.

3.8 Conclusions

The Pre-Flight CDR phase is entirely performed before UAVs take off by considering their submitted flight plans. However, there exists little experience on the actual demand and properties of UAS operations in a realistic scenario. Therefore, in this chapter, we studied the Sendai 2030 model case, which was created by a consulting company to understand the demand for UAV deliveries in 2030. The demand reported in the study suggests that conflicts between UAVs would be very frequent and scalable CDR methods need to be developed.

Our solution to solve the Pre-Flight CDR problem is based on the concept of Trajectory Based Operations, where each UAS operation is seen as a 4DT (3D plus time Trajectory), and UAVs become space-time obstacles for other UAVs. This concept was compared to the simpler 3D concept, where each UAS operation reserves the entire airspace needed to execute the operation. The results indicate that a simple 3D is not practical as a large number of UAS operation requests would have to be rejected.

Since we focus on deliveries by UAVs scenarios, it is important to understand the difference between our MAPF formulated problem and other MAPF problems. Occupancy by static obstacles in the Sendai model case is very low compared to other 2D problem instances. However, the model case has specific properties such as altitude constraints based on elevation and the delivery scenario induces a specific "flower-like" air traffic pattern, which is not seen in other MAPF instances.

Therefore, we defined several general metrics to study air traffic complexity, including the number of UAVs in the air, proximity measures, and the number of conflicts. At this moment, the results are hard to interpret, as comparable data is not, or only sparsely, available [32]. Our simulations also showed a localization of the conflicts within UASSPs in particular areas of the region.

Chapter 4

Decentralized Pre-Flight Conflict Detection and Resolution

In the previous chapters, we assumed that the UTM system is a centralized entity in charge of Conflict Detection and Resolution (CDR). However, recent discussions on UTM suggest that such centralized control might not be practical or desirable. Therefore, in this chapter, we explore Pre-Flight CDR methods whereby independent UAS Service Providers (UASSPs) with their own interests, communicate with each other to resolve conflicts among their UAV operations—without centralized UTM directives. We propose a novel Multi-Agent Path Finding (MAPF) model that supports the decentralized resolution of conflicts, whereby different 'agents', here UASSPs, manage their UAV operations. We present two approaches: (1) a prioritization approach and (2) a simple yet practical pairwise negotiation approach where UASSPs agents determine an agreement to solve conflicts between their UAV operations. We evaluate the performance of our proposed approaches with simulation scenarios based on a consultancy study of predicted UAV traffic for delivery services in Sendai, Japan, 2030. We demonstrate that our negotiation approach improves the "fairness" between UASSPs, i.e. the distribution of costs between UASSPs in terms of total delays and rejected operations due to replanning is more balanced when compared to the prioritization approach.

4.1 Introduction

The definition of UTM regulations has generated growing interest in the development of Conflict Detection and Resolution (CDR) methods to ensure separation between UAVs. Similar to Air Traffic Management (ATM), UTM will employ three redundancy layers that are applied in different phases of a UAV's

flight [44]: Pre-Flight CDR, In-Flight CDR and Collision Avoidance. This redundancy principle is based on the guidelines of the International Civil Aviation Organization (ICAO) [39].

On the one hand, the development of Collision Avoidance and In-Flight CDR methods have recently received significant attention [3, 28, 35, 42, 86, 96]. On the other hand, only few works have presented practical methods to proactively deconflict UAV traffic before UAVs take off, i.e. Pre-Flight CDR methods [36, 37].

In this chapter, we focus on the conception of Pre-Flight CDR methods that aim at solving conflicts between UAV operations before their takeoff. We assume four dimensional (3D plus time) trajectory (4DT) executions, whereby UAVs must follow waypoints at given timesteps.

The Pre-Flight CDR process, also called "strategic deconfliction" [70] will first provide 4DT flight plans determined before takeoff. Then, in case of disturbances, i.e. dynamic events that would disturb the plans generated by Pre-Flight CDR, In-Flight CDR will be triggered in real-time while UAVs are flying. Pre-Flight CDR is necessary to reduce the amount of conflicts that In-Flight CDR would have to process otherwise. We have previously evaluated the impact of such situation [36]. If Pre-Flight CDR was not applied, the complexity of deconfliction for In-Flight CDR would increase, and possibly jeopardize the safety objective. Thus, efficiency and safety are increased through the application of Pre-Flight CDR combined with In-Flight CDR, whereby the latter addresses disturbances.

It has been previously shown that the Pre-Flight CDR problem can be formulated as a Multi-Agent Path Finding (MAPF) problem [37]. However, the standard MAPF formulation does not capture real world considerations such as those of UTM. First, most existing works in MAPF assume a centralized setting, where each agent executes a unique path determined by a central computing entity. Second, they assume that a global cost is minimized over all agents, whereby all agents are cooperative such that no agent is prioritized over another, and any agent may be instructed to take a longer path to avoid conflicts.

UTM assumes that different UAS Service Providers (UASSPs), each with their own group of UAVs, will provide services, such as de-confliction services, to UAS operators who submit UAV operation requests from customers [70]. UASSPs are independent entities with their own business objectives to maintain a certain service quality or satisfy operational constraints. For instance, if the flight path of a UAV operation significantly deviates from its nominal path to avoid conflicts, or if a UAV operation request is rejected, it could negatively affect customers' expectations and require some compensation.

UASSPs aim to provide conflict-free paths while minimizing their own costs in terms of the amount of delays and rejections on their given UAV operations. In the UTM context, a decentralized Pre-Flight CDR process is motivated by practical requirements defined by regulating authorities such as NASA [70], whereby UASSPs communicate among themselves to resolve conflicts between their respective UAV operations. Moreover, UAV operations will have associated valuations to reflect the different levels of importance attributed by customers.

In this context, decentralized approaches for Pre-Flight CDR are necessary and beneficial because they allow UASSPs to deconflict their operations according to costs as determined by themselves. At the same time, UASSPs can protect private information that defines their views in terms of costs for replanning each operation.

In this chapter, we present a novel MAPF model for the Pre-Flight CDR problem with the characteristics shown in Table 4.1. In particular, we support a notion of "fairness" that relates to the distribution of costs in terms of delays and rejected operations between UASSPs. On the other hand, the standard MAPF formulation makes no distinction between UASSPs in the resolution process. Therefore, the resolution of conflicts may result in largely different impacts on each UASSP, as there is no provision to ensure a balanced distribution of delays and rejected operations among UASSPs. By contrast, our proposed approach provides "fairness" among UASSPs, which is a core consideration in UTM.

To the best of our knowledge, we are the first to propose a decentralized MAPF process for Pre-Flight CDR, where UASSP agents with their individual costs deconflict their flight paths before takeoff. This chapter makes the following main contributions:

- We formulate an extended MAPF model, where UASSPs plan collision-free paths for their given UAV operations and aim to minimize their own cost objective. We define these costs based on independent practical considerations such as induced delays and rejections.
- We introduce a decentralized MAPF algorithm that supports resolution with negotiation between UASSPs representing UAV operations.

We compare our approach, which uses a bilateral negotiated resolution, to an approach using prioritization instead of negotiation. With prioritization, UASSPs determine an order between themselves, according to which they will replan their whole set of UAV operations. Here, the lower placed UASSP will replan all its operations while considering all of the higher placed UASSP's operations as fixed spatio-temporal obstacles. We will consider two strategies to determine the order between UASSPs: a randomized strategy and a cost-based strategy. We evaluate our approaches on scenarios based on a realistic study of predicted demand of UAV delivery operations for 2030 in Sendai, Japan.

4.2 Related Works

4.2.1 Multi-Agent Path Finding for Self-Interested Agents

The Multi-Agent Path Finding (MAPF) problem has been extensively studied in many works proposing approaches to solve instances mostly for 2D benchmarks [26, 55]. The existing methods generally assume a centralized setting where a single computing entity finds a solution considering all given agents, with a global objective to minimize, such as the sum of costs or the makespan. To

UTM Standard MAPF problem concept	**UTM Novel MAPF problem concept**
Centralized	Decentralized
Agent = UAV (flight path)	Agent = UASSP entity managing several UAV operations (flight paths)
UAV operations are all of same importance	Each UAV operation has a valuation (low, medium, high)
Cooperative agents	Self-interested agents
Minimize "sum of costs" (durations of flight paths) over all UAV agents → Global optimization	Minimize individual costs (delays, etc) for each UASSP agent → Pareto optimization

Table 4.1: MAPF concepts for Pre-Flight CDR.

minimize these global objectives, optimal and suboptimal approaches have been proposed [6, 75, 76, 81, 82].

Recently, new directions were established to handle real-world scenarios, notably for ground robot path finding in Amazon Robotics warehouses [55]. Several variants of the MAPF formulation have been proposed [38, 54, 56] that consider teams of agents and focus on allocating tasks to each agent while providing conflict-free paths. In this chapter, we assume that all UAV operations are submitted with already defined start and service locations. So, in our work, we do not address the task allocation process. Instead, we consider distinct teams of UAV agents, whereby each team is represented by an UASSP agent that aims to minimize its own cost function.

Few works have considered a decentralized setting in MAPF [11, 29, 63], where agents communicate between themselves to find a conflict-free solution. However, these works consider the standard global optimization setup where agents, each with a unique path, communicate to resolve conflicts to minimize the total solution costs without expressing any self-interest.

In [4], the authors introduce a mapping to combinatorial auctions, and in [13], a taxation mechanism is proposed that incentivises self-interested agents to optimize social welfare.

In all these existing approaches, a mediating entity, such as an auctioneer, is needed to process all preferences of agents. By contrast, a decentralized process is required for the UTM architecture [70], in which all UASSP agents communicate between themselves to resolve conflicts of their operations.

4.2.2 Automated Negotiation

Several works in the Air Traffic Management (ATM) domain have proposed decentralized CDR approaches that use automated negotiation, mostly for the In-Flight CDR phase [65, 80, 92]. In [92], the authors applied a monotonic con-

cession protocol where each aircraft can concede to the other aircraft whether it will fly a higher cost trajectory to resolve a conflict. However, these negotiation protocols assume a common cost function known by all agents or require each agent to explicitly communicate their preferences to others.

In [65], a decentralized negotiation based In-Flight CDR approach is proposed where a pair of aircrafts can negotiate their maneuvers to solve a conflict. Their short-term deconfliction approach only processes a local immediate conflict in each negotiation process, and computes simple response trajectories to the immediate conflict, thus not mitigating downstream conflicts. So the induced deviation from the initial flight path might not be minimal for a given flight path and thus it provides a sub-optimal solution in terms of total deviation.

Both of these works on In-Flight CDR support concessions to create a balance in the costs of resolutions between aircrafts.

In contrast, we resolve conflicts in a strategic way by processing flight paths, i.e. operations, of UAVs before their actual takeoff. Thus, unlike In-Flight CDR, the Pre-Flight CDR process does not have strict real-time requirements since all flights are processed within a certain amount of time ahead of takeoff.

Other works such as [58] focus on the design of automated negotiation approaches by modeling a utility function for each user with improved preference elicitation processes. [57] introduce a self-interested approach for MAPF where each self-interested agent is assumed to negotiate one individual path. However, this approach might not be scalable or practical when there is a large number of conflicts between paths, as a large amount of communications would be required to reach a conflict-free solution for all paths.

By contrast, our work aims at solving a novel MAPF problem that considers UASSP agents with given sets of flight paths. We focus on the definition of a practical Pre-Flight CDR approach based on negotiation, rather than based on the process of users' preference elicitation.

4.3 Problem Formulation

In this section, we formalize our model that extends the standard MAPF formulation with UASSP agents that are in charge of deconflicting their given set of UAV operations. Each UASSP agent aims to minimize its individual cost function, which takes into account the delays induced for each UAV operation with a given valuation, and the number of operations rejected.

4.3.1 Motivation

In the context of UTM, using a decentralized process is a practical requirement defined by regulating authorities [70].

Existing centralized methods in the standard MAPF framework consider a global objective, and thus seek globally optimal solutions without distinguishing agents' individual costs (see Table 4.1).

By contrast, in the context of low altitude airspace traffic management, each UASSP agent's individual interest or priority is an important realistic consideration. Thus, a decentralized method allows us to address self-interested agents and maintain privacy about their preferences, unlike a centralized method.

Moreover, applying a decentralized resolution at the UASSP level instead of the UAV level has two main advantages:

1. It addresses practical requirements: If we resolve conflicts on the UAV level, UAVs would communicate between themselves to solve their conflicts in a decentralized way in the Pre-Flight phase. However, in the UTM system architecture defined by NASA UTM [70], it is required that only UASSP agents communicate between themselves to deconflict their operations in the Pre-Flight phase. Further, UASSPs are economic entities that have their own preferences in terms of deconfliction costs. They are responsible for conflict resolution of their operations.

2. It reduces communication costs: If UASSPs communicate to deconflict their operations, yet the processing would be for each UAV operation on a "per conflict" basis, it would require a significantly high amount of communications between UASSPs to resolve potentially large numbers of conflicts. Thus, UASSP level resolution allows us to avoid this communication overhead by processing several conflicts at once.

4.3.2 Definitions

We distinguish between a *UAV agent* (a UAV operation) that performs a flight path, and a *UASSP agent* that manages a group of UAV agents (UAV operations), as shown in Table 4.1. In this formulation of the Pre-Flight CDR problem, a UASSP agent communicates with other UASSP agents to determine the resolution for their respective group of UAV agents.

We define an agent as a UASSP that is responsible for a set of UAV agents (UAV operations). We consider K given UASSPs agents. Each UASSP agent $(A_k)_{k\in[1;K]}$ has to resolve its own MAPF problem instance defined by a set of N_k UAV operations $\Omega_k = (a_i)_{i\in[1;N_k]}$ that are each performed by a UAV. Each operation represents a flight path with a fixed start location and a goal location. We consider an undirected graph G which is a 26-neighbor 3D cubic grid map. UAVs can 'move' along an edge of G or can 'wait' on a vertex of G.

Each operation a_i has an attributed valuation v_i that reflects its importance, which can be "low", "medium", or "high", depending on customer requirement. Thus several operations may have the same valuation.

Each UAV assigned to perform an operation a_i is represented by a sphere of given radius r_i, and a center position p_i. For any pairs of UAV operations a_i and a_j, any violation of the minimum separation distance, i.e. the sum of the respective radii of the associated UAVs $r_i + r_j$, must be prevented. Hence, we define the following constraint to ensure there is no conflict at any timestep t:

$$\forall t, dist(p_i(t), p_j(t)) > r_i + r_j \tag{4.1}$$

In the UTM context, we also define altitude constraints for the flight paths of UAVs by reference to coordinates (x, y, z):

$$elevation(x, y) + alt_{min} \leq z \leq elevation(x, y) + alt_{max} \tag{4.2}$$

$elevation(x, y)$ refers to the terrain elevation value of the point of coordinates (x, y) in a path, measured from mean sea level, and alt_{min}= 90 m and alt_{max}= 150 m are fixed altitude bounds relative to elevation. Hence, there is a 60 m altitude range for UAV operations. A solution of the given MAPF instance for A_k consists of conflict-free paths for all UAV operations in Ω_k.

4.3.3 UASSP Cost Function

Unlike the standard MAPF formulation, each UASSP agent A_k has its own independent objective which is to minimize its own cost value C_k^v. That value represents the service degradation due to conflict resolution on its given operations of same valuation, as shown in Eq. 4.3. An operation is rejected when no resolution is found to solve a conflict or when the generated deviation exceeds the given battery autonomy. The operation is then removed from the solution set, i.e. rejected. A rejection is considered as a more important penalty than an added delay, so the whole rejected path duration is considered as the incurred cost.

We define the cost function C_k^v for each UASSP agent A_k for all of its operations of same valuation $v \in \{low; medium; high\}$ as:

$$C_k^v(\Omega_k) = \sum_{a_i \in \Omega_k^v} c(a_i) \tag{4.3}$$

$$c(a_i) = \begin{cases} T_i' - T_i \quad if \quad path \quad is \quad replanned \quad (delay) \\ T_i \quad if \quad path \quad is \quad rejected \\ 0 \quad else \end{cases} \tag{4.4}$$

where $\Omega_k^v \subseteq \Omega_k$ is the subset of operations of A_k with same valuation v, $c(a_i)$ represents the individual cost for each operation a_i with initial flight duration T_i and flight duration obtained after replanning T_i'.

4.3.4 Fairness

Fairness is a core consideration in the UTM context, as several economic and competitive entities are involved to provide services to customers in the shared low altitude airspace. We distinguish two notions of "fairness" according to the type of agent, for a *UAV agent* and for a *UASSP agent*, and we address both notions:

- *UAV agent (operation) level*: the order of valuations must be respected among all operations. Here, we address this issue by prioritizing higher valued UAV operations when in conflict, i.e., they become spatio-temporal obstacles for the lower valued operations which in turn must be replanned.

- *UASSP agent level*: we conceive of fairness as the more balanced distribution of costs, such as total delays and rejected operations, among all UASSP agents for all operations of same valuation.

Fairness at the UAV operation level simply prioritizes according to the given valuations of the operations. On the other hand, fairness at the UASSP agent level requires the conception of a CDR approach that allows UASSP agents to determine a mutual solution that they would consider as "fair".

4.3.5 Optimality

We distinguish two notions of optimality whereby each reflects a different objective:

1. Optimality in terms of total deviation caused by resolving conflicts for all given UAV agents of an UASSP agent: This relates to the difference in the total durations between the conflict-free flights and the initial flights of the given UAV agents (delays). The optimality hereby refers to the solution that minimizes the deviation caused for all UAV agents' flight paths, i.e., the solution that minimizes the "sum of costs" objective, as in standard MAPF.

2. Optimality in terms of fairness for all UASSP agents: This concerns the *distribution* of costs related to deviations among different UASSP agents. The optimality hereby refers to the solution that maximizes fairness, i.e., the distribution of deviations costs among UASSP agents.

Here, we focus on fairness among different UASSP agents, which is a novel objective for optimality.

Note that in the UTM context, it is important to make a trade-off between theoretical optimality of a solution and practical constraints, such as scalability.

4.4 Pre-Processing Phase

This section describes the initial steps that each UASSP agent performs independently before processing Pre-Flight CDR with other UASSPs. For this purpose, we address specific properties of the configuration of UAV delivery service scenarios. Each UASSP agent is responsible for a given set of hubs. Each UAV assigned to perform an operation departs from a given hub and returns to the same hub once its mission is completed. We assume that each hub can service a predetermined limited surrounding area ('service area') with a fixed radius as represented in Fig. 4.1, whereby specific service locations are assigned to each UAV. Therefore, the hubs of a UASSP agent may have overlapping service areas with other hubs pertaining to other UASSPs, where conflicts can occur between UAV operations of different UASSPs.

First, each UASSP processes its own operations so that its own paths are initially conflict-free, as shown in Fig. 4.1. Then, each UASSP agent A_k only

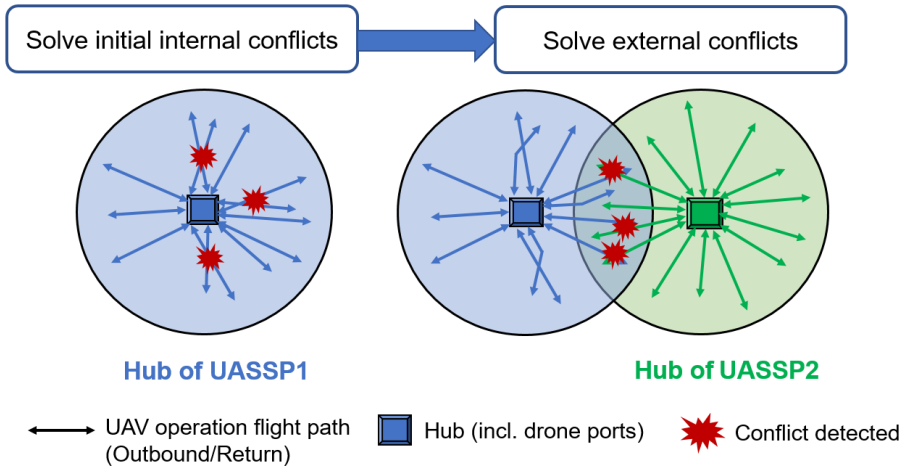


Figure 4.1: Flowchart of the Pre-Flight CDR process for a UASSP agent: (1) Each UASSP agent first solves all existing conflicts among its own set of operations, using a standard MAPF solver such as ECBS [6]; (2) Each UASSP agent communicates with other UASSPs to deconflict operations between them.

needs to communicate the subset of operations $\Omega_k^{shared} \subseteq \Omega_k$ that may be in potential conflicts with the other UASSPs based on their overlapping service areas as depicted in Fig. 4.1. For this purpose, each UASSP performs an initial step where operations that are crossing an overlapping area are filtered with a simple geometrical verification of each flight path segments, based on the radius of each service area.

In other words, we draw a distinction between conflicting paths belonging to a same UASSP ('internal conflicts') and those between different UASSPs ('external conflicts'). Without this step, new internal conflicts might be detected and created after solving external conflicts, which might be impractical, as this would increase the amount of communications. Hence, we always start with conflict-free paths from each UASSP side.

4.5 Decentralized Approaches

We assume that each UASSP agent uses a given MAPF solver to solve their 'internal conflicts', such as the bounded suboptimal ECBS algorithm [6] (here, we refer to the optimality in terms of total deviation as mentioned in SECT. 4.3.5). So, each UASSP agent processes its own MAPF instance in a bounded suboptimal way with the use of the ECBS algorithm. This bounded suboptimality guarantee is relative to the consideration of other UASSPs agents' operations as spatio-temporal obstacles. Then, each UASSP agent starts the de-confliction with other UASSP agents, based on its conflict-free set of operations.

In this section, we introduce two distinct decentralized algorithms to solve

the Pre-Flight CDR problem between UASSPs agents.

- A *prioritization* approach where UASSPs determine an order of replanning between themselves. Here each UASSP will replan their respective operations considering the trajectories of operations of higher placed UASSPs as fixed spatio-temporal obstacles.
- A *pairwise negotiation* approach where UASSPs make concessions on the operations that they both will replan. This allows UASSPs to share the costs between them by determining which operations should be replanned and which not.

Let us first consider the prioritization approach as a baseline approach. Then we will describe the pairwise negotiation resolution approach, and then generalize to the resolution with more than two agents.

4.5.1 Prioritization Approach

Similar to the Cooperative A* approach [77] used to solve MAPF instances for a centralized setting, we hereby propose a decentralized resolution with prioritization between UASSPs. In Cooperative A*, each agent is initially associated to a unique and distinct priority, and when two agents are in conflict, the agent with lower priority must replan its path considering the agent with higher priority as obstacle.

Differently, since we consider UASSP agents with given UAV operations, the determined priority order for each UASSP agent applies to all of its operations, thus several paths at once. However, randomly attributing priorities to UASSP agents might be unfair, as it would distribute costs among UASSP agents without a rationale, and thus may lead to arbitrarily high individual costs for UASSP agents.

Therefore, we propose an informed "cost-based" strategy that relies on the computed costs of each UASSP agent, so as to determine the priority ordering among UASSP agents (see Algorithm 5). Here, UASSP agents will mutually determine their pairwise order by communicating their incurred costs with Eq. 4.3.

Each UASSP agent computes its incurred cost as if it was to resolve all conflicts alone, i.e. by determining conflict-free paths while considering all the paths of the other given UASSP as spatio-temporal obstacles.

Then, the UASSP which would have incurred a higher cost is placed higher, as "winner", and does not have to replan its paths, while the UASSP with lower cost, as "loser", has to replan all its paths in conflict with those of the "winner". The rationale here is that the "loser" has to re-plan because it results in less total costs.

Note that each UASSP agent carries out the same method of conflict detection computation to determine which conflicts need to be solved with the other UASSP agent. This ensures that if a conflict is detected by one UASSP agent, it will also be detected by the other UASSP agent.

Algorithm 5: Pairwise Prioritization

Data: $(A_i; A_j)$ two UASSP agents with respective associated set of operations Ω_i^{shared} and Ω_j^{shared}

Result: Conflict-free paths for all UASSPs

```
/* Pseudocode for agent A_i                                          */
/* Cost evaluation for each agent operations                         */
```

$\Omega_i^{replanned} \leftarrow SolveMAPF(\Omega_i^{shared}, \Omega_j^{shared})$;

Send $C_i^v(\Omega_i^{replanned})$ to A_j;

Receive $C_j^v(\Omega_j^{replanned})$ from A_j;

```
/* Determine if ``winner'' or ``loser''                              */
```

if $C_i^v(\Omega_i^{replanned}) \leq C_j^v(\Omega_j^{replanned})$ **then**

```
    /* A_i is the ``loser'' and A_j's paths become obstacles for A_i */
```

    $\Omega_i^{shared} \leftarrow \Omega_i^{replanned}$;

```
/* Else A_i is the ``winner'' and does not replan its paths          */
```

Regarding the computation of conflict resolution, one UASSP agent is not repeating the computation that has been performed by the other one, in the sense that each UASSP agent will compute and propose a different conflict-free solution to the other UASSP agent. Such creation of a conflict-free solution is done by the UASSP agent's re-planning of its own UAV operations, while considering the other UASSP agent's UAV operations as spatio-temporal obstacles.

4.5.2 Negotiation Approach

The prioritization approach presents a simple decentralized MAPF solver that incorporates individual costs of each UASSP agent and provides a resolution in a systematic manner. However, this method might be considered "unfair", as some UASSP agents may have to sustain a large amount of costs by replanning all their operations, only because they are ranked lower than other UASSP agents.

Therefore, we propose a simple yet practical resolution method based on negotiation by incremental concession making. This allows UASSPs to bargain and effectively share their replanning costs by mutually determining which of their operations to replan or not to replan. Thus, there is no "loser" who has to bear all costs or "winner" who does not replan at all.

Operators submit operations some time ahead of their desired start time, e.g., at least 30 minutes before their UAV desired takeoff time. Typically, the chosen time window is large enough to handle a significant amount of operations from different UASSPs. Note that the actual negotiation time is a few minutes.

Bargaining	**Pre-Flight CDR**
Feasibility set $F \subset \mathbb{R}^2$	All attainable costs for each UASSP
Disagreement point $d \in F$	Cost for each UASSP resolving all conflicts alone
Agreement point $s \in F$	Conflict-free solution with better costs than d for each UASSP

Table 4.2: Reduction of the Pre-Flight CDR problem to the bargaining problem.

Bargaining problem

This approach can be assimilated as solving a bargaining problem, which is a game theory concept that aims to find an equilibrium when conflicts of interest arise between players with separate and conflicting objectives [60].

We propose a mapping of the Pre-Flight CDR problem to the bargaining problem as presented in Table 4.2. Here, conflict resolution is framed as a two-player game where each UASSP agent would prefer the other agent to bear the costs, i.e. resolve the given conflicts by replanning their own operations.

Theoretical Properties

Solving the bargaining problem means finding an agreement $s \in F$ for both UASSP agents A_i and A_j according to their respective cost function C_i^v and C_j^v. The agreement point s hereby represents the costs computed by each UASSP agent for a conflict-free solution where certain operations are replanned between both UASSP agents, and s is on the Pareto frontier [60]. In particular, if the Pareto frontier of the costs to all agents is known, then the agreement is the Kalai-Smorodinsky solution [46]. The properties of this solution are based on the axioms of game theory: Pareto optimality, symmetry, invariance and monotonicity [46]. However, here the Pareto frontier is not known, as the cost of each proposal (conflict-free solution) can only be valued by the UASSP agent that will replan its operations. Thus, the process converges to a solution that is as close as possible to the optimal "fair" solution with its costs on the Pareto frontier.

Our Approach

A pair of agents communicate with each other to negotiate which paths to replan within a fixed maximum number of rounds N_{Max} (see Algorithm 6).

For this purpose, at each round, they follow three main steps as follows:

1. Each UASSP agent A_i makes a *proposal* $\Omega_i^{proposal}$ to the other agent. A *proposal* contains all operations of the agent whereby the agent proposes to solve certain conflicts by replanning certain of its operations, while assuming the other agent will solve the remaining conflicts.

2. Then, each UASSP agent A_i evaluates the proposal received from the other agent by computing a *response* $\Omega_i^{response}$. A *response* is a conflict-free solution computed by the agent which replans its operations considering the paths proposed by the other agent as spatio-temporal obstacles.

3. Each agent determines the costs from their proposal made $C_i^v(\Omega_i^{proposal})$ and their response to the proposal of the other agent $C_i^v(\Omega_i^{response})$. An *agreement* is reached if the cost of the response is lower than the cost of the proposal: $C_i^v(\Omega_i^{response}) \leq C_i^v(\Omega_i^{proposal})$.

Otherwise, the bargaining process iterates again for another round through these steps while incrementing the costs for the respective new proposals until an agreement is reached. An agreement represents conflict-free solutions that effectively distribute costs between the two UASSPs.

The key step in this approach is the determination of a proposal by each agent in each round. A proposal is a concession on which operations to replan whose associated cost increases at each round to ensure convergence to an agreement.

To generate a proposal, an agent must identify which of its operations are in conflict with the other agent and determine which operations it will replan to resolve some of the identified conflicts. The proposal is updated at each round by additionally replanning some operations according to the criterion of their incurred costs.

For this purpose, both agents must communicate an initial proposal which is a "zero-cost" solution where they do not replan any of their paths, i.e., the replanning cost is zero, and assume that the other agent will resolve all of the existing conflicts. Here, for each agent the computed response is the same computed solution as in Section 4.5.1, where each agent computes the conflict-free paths by considering all the paths of the other agent as obstacles.

This first step allows both agents to identify the existing conflicts between them, so that each agent A_i can determine the set G_i of groups of operations involved in a conflict. A group of operations $g \in G_i$ is a subset of operations of the same agent which are impacted by the same conflict with an operation of the other agent.

There may be several distinct conflicts between agents, thus several groups of operations. Then, a set contains the groups of operations to be proposed for replanning at each round of negotiation. Each group $g \in G_i$ is associated to a cost value $C_i(g)$ computed with Eq. 4.3, which represents the individual incurred cost to replan the operations of the group. G_i is sorted by ascending order of costs for each group so that the groups with lower costs are selected first.

The agents then determine the value δ_i, which is the number of groups of operations from the sorted set G_i that they will need to re-plan to update their current proposal at each round, so that the negotiation process converges in less than N_{Max} rounds. In the case that both agents agree on their respective proposals, the proposal with the lowest variance in costs between the two agents is then chosen.

Convergence Analysis

This approach will always terminate in less than N_{Max} rounds since each agent will progressively have to deviate their own proposal closer to the "zero-cost" solution of the opposite UASSP agent.

The N_{Max} value is the maximum number of rounds to drive the costs of the proposed solutions for the given pair of UASSP agents from their "zero-cost" proposal to their "maximum cost" proposal. The "maximum cost" proposal represents the response solution to the "zero-cost" proposal of the opposite UASSP agent, whereby the UASSP agent will solve all conflicts by itself.

In our negotiation process, at each round, the number of conflicts to be solved by each UASSP agent to generate its new proposal is determined by adding a constant increment to δ_i, which is determined as a function of the maximum number of conflicts and N_{Max} (see Algorithm 6).

So, if the process reaches the N_{Max}^{th} round, each UASSP will propose to solve all existing conflicts by itself, which is the same solution as proposed in the prioritization approach. Hence, the negotiation process will terminate in at most N_{Max} rounds.

In practice, an agreement between UASSPs can be found well before reaching this last N_{Max}^{th} round, and thus the actual number of rounds is much lower than N_{Max}. The value of N_{Max} is determined empirically based on the average number of conflicts between UASSPs given their number of operations and the practical computational costs.

The larger the value of N_{Max} is, the more precise the increment on the proposals at each round will be. Thus, the final solution will be closer to the optimal Pareto solution. However, if this value is too large, then this would require a heavier load of communications between UASSPs before reaching an agreement.

4.5.3 Generalization to Multilateral Case

We have presented two different decentralized approaches to resolve conflicts between pairs of UASSP agents. For more than two UASSP agents, each agent would need to communicate with all of the other agents and wait to collect all of their responses and vice versa for each involved agent, until an agreement is reached that satisfies all agents simultaneously.

However, such process would be complex and impractical to deploy since the amount of communications required to converge to an agreement and synchronization issues might be large, especially when many agents are simultaneously involved in conflicts between their operations.

Therefore, we propose to address the $K \geq 2$ UASSP agents case with a *sequential bilateral* approach as shown in Algorithm 7.

In this approach, agents sequentially deconflict between each other in pairs using the previously presented approaches. The sequence of pairwise processing is determined with a predefined criterion that all agents first communicate to determine which pair should first deconflict between each other. We hereby use

Algorithm 6: Pairwise Negotiation

Data: $(A_i; A_j)$ two UASSP agents with respective associated set of operations Ω_i^{shared} and Ω_j^{shared}; N_{Max}
Result: Conflict-free paths for all UASSPs

```
/* Pseudocode for agent A_i                                          */
/* Initial proposal and response                                     */
```

$\Omega_i^{proposal} \leftarrow \Omega_i^{shared}$;
Send $\Omega_i^{proposal}$ to A_j;
Receive $\Omega_j^{proposal}$ from A_j;
$\Omega_i^{response} \leftarrow SolveMAPF(\Omega_i^{shared}, \Omega_j^{proposal})$;

```
/* Determine the groups of operations g ∈ G_i involved in external
   conflicts, and sort by increasing value of individual costs
   C_i^v(g)                                                          */
```

$G_i \leftarrow GenerateGroupsOperations(\Omega_i^{response})$;
$\delta_i \leftarrow \lceil |G_i|/N_{Max} \rceil$;
while *true* **do**

```
    /* Generate new updated proposal with incremented cost           */
```

    $g_{\delta_i} \leftarrow getFirstGroups(G_i, \delta_i)$; ▷ Select δ groups with lowest costs to re-plan
    $\Omega_i^{proposal} \leftarrow Update(g_{\delta_i}, \Omega_i^{proposal})$;
    Send $\Omega_i^{proposal}$ to A_j;
    Receive $\Omega_j^{proposal}$ from A_j;

```
    /* Compute response to proposal of other agent                   */
```

    $\Omega_i^{response} \leftarrow SolveMAPF(\Omega_i^{shared}, \Omega_j^{proposal})$;

```
    /* Evaluate response and compare to proposal                     */
```

    if $C_i^v(\Omega_i^{response}) \leq C_i^v(\Omega_i^{proposal})$ **then**
        break; ▷ Agreement reached
    else
        $\delta_i \leftarrow \delta_i + \lceil |G_i|/N_{Max} \rceil$; ▷ Update δ_i

the number of conflicts detected between agents as a criterion to determine the sequence of processing for all agents.

Our approach is a more practical resolution method for several UASSP agents as the used criterion allows us to reduce the amount of communications needed to converge to the final conflict-free solution for all UASSP agents. This might lead to a lower overall solution quality because of the local, rather than global, aspect of this approach.

Note that in the prioritization approach, the paths of the "winner" of pairwise resolution are declared as obstacles for the other agents, which have not yet been deconflicted between each other. In the negotiation approach, the paths of the UASSP agent with the lowest remaining number of conflicts are declared

Algorithm 7: General approach with $K \geq 2$ agents

Data: $(A_k)_{k\in[1;K]}$ UASSP agents with associated set of operations Ω_k^{shared}

Result: Conflict-free paths for all UASSPs

```
/* Pseudo-code for agent A_i                                          */
```

$OtherBlockedPaths = \emptyset$ ▷ List of paths from other agents considered as obstacles

$Is\_Processed \leftarrow false$;

while *true* **do**

```
    /* Determine maximum number of conflicts with other agents        */
```

$(A_j; N_{ij}^{Conflicts}) = MaxConflictsPair((A_k)_{k\in[1;K];k\neq i})$ ▷ Send/Receive update message to/from other UASSPs

```
    /* If A_i has the highest number of conflicts with agent A_j      */
```

if $(A_i, A_j) == \arg\max_{(A_k,A_l)_{k\neq l\in[1;K]}} N_{kl}^{Conflicts}$ **then**

```
        /* Apply resolution process with A_j                          */
```

$\Omega_i^{shared} \leftarrow Pairwise\_Resolution(A_i, A_j)$;

$Is\_Processed \leftarrow true$; ▷ Send update message to other UASSPs

else if $Is\_Processed$ **then**

```
        /* A_i's paths become obstacles for the other agents if not
           re-selected next                                           */
```

$OtherBlockedPaths \leftarrow OtherBlockedPaths \cup \Omega_i^{shared}$;

Send($OtherBlockedPaths$); ▷ Send update message to other UASSPs

break;

as obstacles. Those are the ones that are less likely to have an impact on the other agents' paths.

Moreover, our proposed pairwise negotiation approach aims at maximizing the fairness objective as defined in Section 4.3.5. Thus, we aim at obtaining a solution that is as close as possible to the optimal solution in terms of fairness.

Note that the termination guarantee mentioned in Sect. 4.5.2 still holds here, since the multilateral case is a sequence of bilateral processes.

4.6 Experiments

In this section, we first describe the properties of the model case used for our simulations. Then, we present and analyze the results obtained in our experiments.

4.6.1 Model Case Scenario

This section describes the model case used in our experiments. The model case is based on a study that aims to project UAV service demand in the Sendai region in Japan in 2030. The study was conducted by a consulting company, as part of

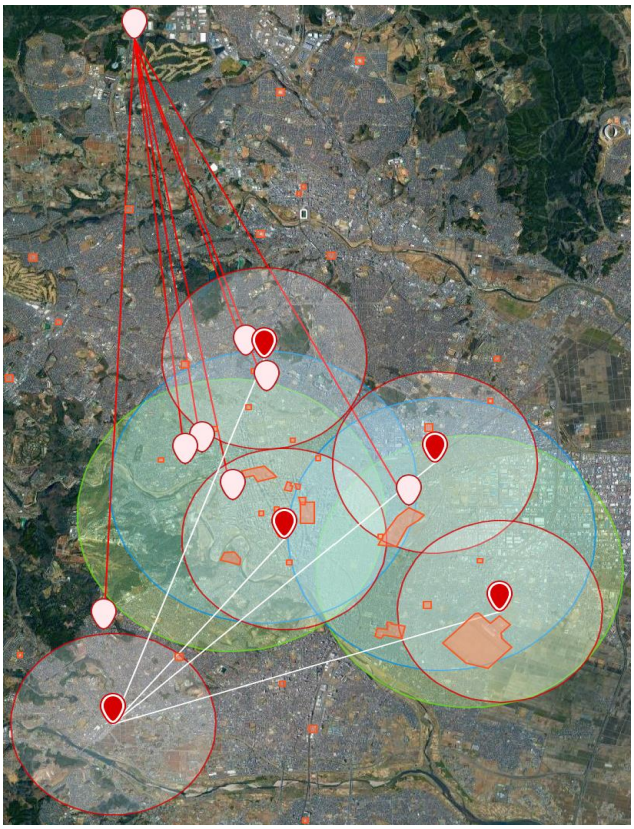


Figure 4.2: Map of Sendai 2030 scenario area, including hub locations for all UASSPs and service areas (circles).

a large-scale governmental project on designing, simulating and specifying the UTM system.

The dimensions of the considered area in the given region are 14.35 km×17.10 km. The representation of our space is a 3D grid map composed of cells with edges of 30 meters long. Thus, according to the dimensions of the model case, we consider a 3D grid map of dimensions 478×570 cells, which is delimited in altitude by a two cells range (60 m range) relative to the elevation. The positions of static obstacles, i.e., blocked cells, is fixed according to the given elevation map of the region. Further, there are 41 distinct no-fly zones identified by the study.

The study considers three major logistics companies (hereby anonymized as A, B and C) that provide deliveries of goods such as mail and package delivery, as well as a Red Cross blood center anonymized as D that collects blood samples between medical centers. The different hub locations are set by the study considering the expected demand in the area in 2030. The assignment of the service locations is assumed to be done independently by UAS operators

within a given service radius around a hub. Figure 4.2 depicts positions of hubs and their associated service area with a different color for each company. Inside these areas, we fix the minimum flight path length to 300 m.

We distinguish two types of deliveries: *Hub-to-Home*, that are deliveries performed from hubs to service locations (homes) located within a given service radius, and *Hub-to-Hub*, that are deliveries performed between given pairs of hubs only. The study indicates a given amount of *Hub-to-Home* and *Hub-to-Hub* deliveries for each of the companies.

In the study, one day of delivery service represents 13 hours, from 8 am to 9 pm. According to customers' demand in the region, expected weight of deliveries and expected capabilities of UAVs, the study estimates that there is a total demand of up to 13,910 operations per day in normal season and up to 21,235 operations per day in busy season, considering the given companies altogether. We consider quadcopter UAVs that have holonomic motion, and can move in any direction at any time, or hover. Each UAV has an attributed speed between 15m/s and 18m/s and a given radius r between 15m and 30m, as defined in Section 4.3.

In this chapter, we assume that each of the given companies uses its own UASSP, thus four UASSPs are considered in our experiments, and we consider fixed ratios of submitted operations for each UASSP according to the study.

4.6.2 Experimental Results

All approaches are implemented in Java within the JADE framework to simulate communications between UASSP agents, and are run on a 3.2GHz Intel Core i7-8700 desktop with 16 GB RAM. The starting times of operations are uniformly distributed within a 1-hour time window. In all experiments, the total number of submitted operations varies from 200 to 1000 operations in total, and the operations are distributed among the given UASSPs with fixed proportions according to the study.

We evaluate the performances of our approaches with two main metrics:

- Total accumulated delay, i.e. the sum of all delays of all operations of an UASSP agent;
- Amount of operations rejected of an UASSP agent

We also compute the standard deviation in total delays to quantify the difference in the distribution of delays among the UASSPs. 30 instances were generated for each experiment.

Comparison of Prioritization Approach with Randomized Ordering and Cost-Based Ordering

First, we compare the difference in terms of total accumulated delays between cost-based ordering as presented in Section 4.5.1 and a randomized ordering approach, where agents determine their priorities randomly and replan accordingly. For the randomized strategy, we generated all 24 possible combinations of

		Randomized		Cost-based		Negotiation	
#Ops	**UASSP**	delay (in min)	% ops rejected	delay (in min)	% ops rejected	delay (in min)	% ops rejected
200	A	1,4	0,3	2,15	0,25	1,25	0
	B	2,4	0,2	0,56	0	1,54	0
	C	0,95	0,3	0,44	0,3	0,58	0,2
	D	0,3	0	0,21	0	0,21	0
400	A	9,3	1,5	7,39	0,33	5,47	0,25
	B	5,7	0,7	4,14	0,3	4,5	0,3
	C	4,55	0,8	4,27	0,42	4,45	0,4
	D	0,47	2,1	0,44	1,9	0,51	0
600	A	22,8	1,4	16,12	0,52	13,1	0,4
	B	14,1	2,7	9,56	2	11,47	1,4
	C	10,2	1,3	8,3	1,1	9,52	0,83
	D	1,8	3,5	1,15	2,8	1,3	2,6
800	A	32,3	0,9	27,4	0,6	22,27	0,5
	B	23,8	2,5	15	1,7	17,56	1,56
	C	15,6	1,7	14,17	1,28	13,7	1,28
	D	0,6	3,8	0,5	3,3	0,5	2,5
1000	A	39,2	1,8	34,3	0,97	29,1	0,22
	B	26,5	4,3	21,51	3,5	23,4	1,92
	C	16,7	2,6	14	2,2	15	1,9
	D	1,5	6,8	1	6,25	1	1,7

Table 4.3: Comparison of the total delays (in minutes) and % of operations rejected among UASSPs between the negotiation approach and the prioritization approach with randomized strategy and with cost-based strategy.

orders among all four UASSPs. Table 4.3 reports the average delays obtained for each UASSP over all the combinations. As expected, the randomized strategy leads to distinctly larger delays in average for each agent than the cost-based strategy. Fig. 4.4 shows a higher standard deviation in total delays between the UASSPs.

Comparison of Prioritization Approach and Negotiation Approach

We compare the performance of the prioritization with cost-based ordering approach to the negotiation approach. Overall, the impacts of the methods can be observed mainly on UASSPs A and B, which have the highest number of operations and larger overlapping service areas, so most conflicts happen between them. The costs for UASSPs C and D do not significantly change due to their hubs configurations as shown in Fig. 4.2. UASSP C has less conflicts with the other UASSPs due to the smaller service areas of its hubs, and UASSP D has fewer operations to process.

We observe that the total delays are more uniformly distributed between UASSPs A and B with the negotiation approach than with the prioritization approach as shown in Fig. 4.3. More precisely, with the use of resolution with negotiation, the standard deviation in total delays becomes lower than with

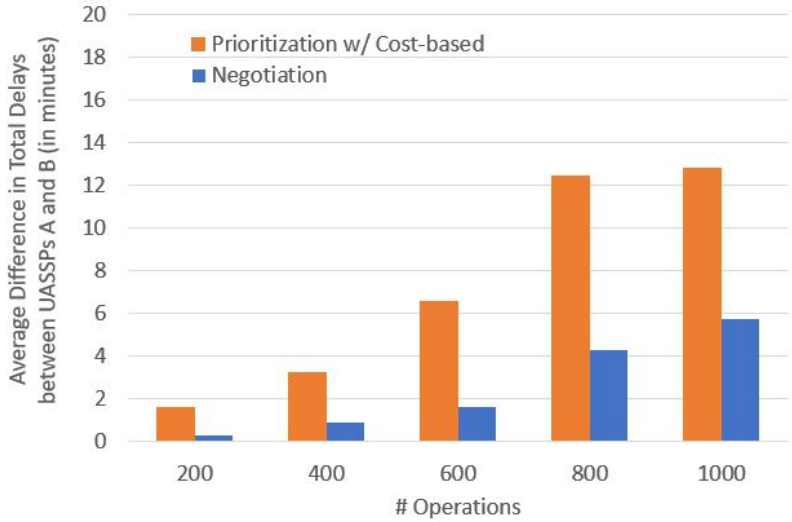


Figure 4.3: Comparison of the average difference in total delays among UASSPs A and B between the negotiation approach and the prioritization approach with cost-based strategy.

prioritization with cost-based ordering, in particular from 600 operations (see Fig. 4.4). This indicates a reduction in the difference in costs between UASSP agents when solving conflicts between their operations, as the bargaining process allows one to share the replanning costs among UASSP agents.

The solution obtained with the negotiation approach aims at minimizing the gaps in total delays between pairs of UASSP agents. That is, the negotiation approach tries to maximize "fairness" and determine a solution close to the Pareto optimal solution, whereby the gap in costs (e.g. in terms of total delays) between all agents is minimal. As observed in our experimental simulations, the average difference in *total* delays between pairs of UASSP agents is around 6 minutes for UASSPs A and B that have the most conflicts on average between their operations (see Fig. 4.3).

The amount of rejected operations is also slightly reduced with the negotiation approach, shown in Table 4.3, as rejecting an operation has more impact than delaying one operation. Yet, since our considered space is not cluttered but rather an open space, this number remains low. We observe that operations were mostly rejected for the following reasons: either (1) a conflict occurs near the start location of an operation, where the other operation is considered as an obstacle and thus cannot be replanned, or (2) a conflict occurs near a start location because of added delay on some operation.

The trade-off between these two approaches is in terms of processing time and solution quality. While resolution with prioritization generates higher degradation in costs than the resolution with negotiation, its processing is faster than

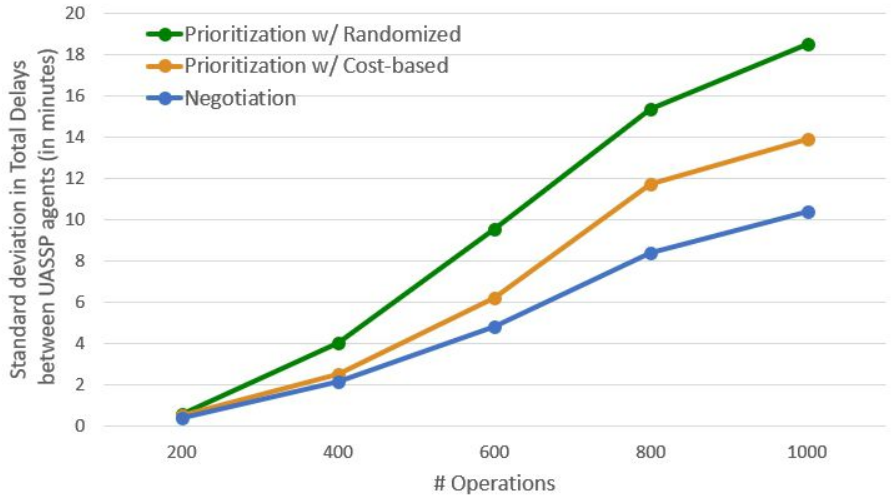


Figure 4.4: Comparison of the average standard deviation in total delays among all UASSPs between the negotiation approach and the prioritization approach with (1) randomized and with (2) cost-based strategy.

the negotiation approach.

The processing time includes the communication time and the computation time. The communication time is negligible; it is less than a few seconds on average. On the other hand, the computation time represents the major part of the processing time, as it involves the generation of a response to a proposal for each UASSP agent at each negotiation round. This requires replanning paths with the given MAPF solver, and this time increases with the number of operations as more conflicts occur. However, with a scalable MAPF solver like our extended ECBS method introduced in Chapter 2, computation time remains low even for a thousand operations.

Both approaches are able to find conflict-free solutions for all UASSP agents in less than 5 minutes on average, which suggests good scalability. Note that Pre-Flight CDR does not have strict deadlines as opposed to In-Flight CDR or Collision Avoidance, so communication and processing are not time critical.

We followed the Sendai 2030 model case, because it is a realistic and representative study of predicted demand of UAV delivery operations provided by a major consulting company. We expect that our methods and results are applicable in similar settings, e.g., a small number of UASSPs, with a large number of operations.

Also, we demonstrated that the negotiation approach always generates higher "fairness" than the prioritization approach, because it allows a more balanced distribution of total delay costs between UASSP agents.

4.7 Conclusions

The development of a UTM (Unmanned Aircraft System Traffic Management) system is required to safely integrate UAVs in low altitude airspace. UAS operators will task UAVs to perform flight operations for different applications, such as surveillance, delivery, and so on. Thereby, they will rely on UAS Service Providers (UASSPs) to provide conflict-free paths for their UAVs.

In this chapter, we extend the MAPF (Multi-Agent Path Finding) framework to model the Pre-Flight CDR problem. Since the standard, *centralized* MAPF formulation does not make any distinction between UASSP agents, we extend the MAPF formulation by supporting an individual cost function for each UASSP agent. The objective for each UASSP agent is thus to determine a conflict-free solution for their operations while minimizing their individual cost. These cost functions aggregate delays generated for each operation in conflict and the amount of operations rejected.

In response to the requirements of the UTM community [70], we present a novel *decentralized* resolution method for MAPF instances between UASSP agents. This method which aims at ensuring "fairness" in the distribution of costs, mainly in terms of total delays among all UASSP agents. Specifically, we introduce two efficient decentralized algorithms to address this problem: (1) resolution with prioritization as a baseline method and (2) resolution with negotiation as a practical method.

We experimentally compare the proposed approaches in a realistic model case based on a real world study of UAV delivery service in the Sendai region in Japan projected for 2030. Resolution with prioritization with a cost-based strategy for ordering among UASSPs allows one to generate solutions with on average lower costs for all UASSPs, as compared to a randomized ordering. Then, our results suggest that resolution with negotiation generates a better distribution in costs among all UASSP agents, in particular in total delays, than resolution based on prioritization with a cost-based strategy.

Thus, unlike the standard MAPF objective, our negotiation approach allows each UASSP agent to reduce its individual costs, and also improves "fairness" among UASSP agents.

Since Pre-Flight CDR methods are not time critical, the negotiation process can start ahead as early as possible, as long as the UASSP agents are able to converge to a solution before the actual start of their operations. In particular, in our case study in Sendai, we assumed that there are up to 1600 operations in total per hour, and in our experiments we also considered a 1-hour time window for the scheduled start times of the submitted operations. Assuming that operations are uniformly submitted in time, we thus consider that an average of 800 operations are submitted in 30 minutes. Therefore, a sufficient and convenient time interval to start the negotiation process between UASSP agents could happen every 30 minutes, so that a significant number of submitted operations can be processed.

With the use of MAPF solvers such as the extended ECBS algorithm introduced in Chapter 2, which is scalable with the number of operations to process,

our decentralized Pre-Flight CDR approach using negotiation provides low processing times for each round.

Finally, we hope that this work on Pre-Flight Conflict Detection and Resolution can contribute to a candidate solution for UTM technology, with the inclusion of individual cost functions to reflect the economic aspects of each UAS Service Provider.

Chapter 5

Scheduling in Pre-Flight Conflict Detection and Resolution

In this chapter, we propose to explore a different extension of the Multi-Agent Path Finding (MAPF) problem for Pre-Flight Conflict Detection and Resolution (CDR). UAV flight paths requests with initial scheduled takeoff times from different service providers will be processed with a Pre-Flight CDR method. This may limit the performance of such algorithms especially in high density structured airspace. These assumptions can lead to large deviations to solve conflicts or even worse, to a flight path being rejected because no solution was found. Here, we extend the MAPF model by allowing agents start times and speeds to be changed to resolve conflicts. Hence, we propose a novel MAPF solver that incorporates temporal resolution to solve conflicts, that is "scheduling". Thus, aside from the standard method of replanning used in most MAPF solvers, we introduce two additional resolution techniques: (1) Takeoff scheduling whereby an agent start time is delayed, and (2) Speed scheduling whereby an agent speed is decreased over a segment of its path. Moreover, we introduce a distinction of types of conflicts that allows us to effectively combine the three different resolution techniques. We evaluate our proposed approaches on a realistic high density and structured airspace scenario in Tokyo, Japan. We show that our method that combines all techniques is able to reduce the number of flight paths rejected and the average amount of generated delay per flight path.

5.1 Introduction

In this chapter, we focus on the conception of Pre-Flight CDR methods. Here, the use of four dimensional (3D plus time) trajectory (4DT) executions is envisioned, whereby UAVs will follow given waypoints in space and time. Following

this concept, the Pre-Flight CDR problem can be formulated as a Multi-Agent Path Finding (MAPF) problem [37, 55], whereby the agents are UAVs, each with an associated flight path, i.e. operation submitted by a UAV Operator.

In the MAPF setting, agents located in a graph must follow a path from their given start locations to their given goal locations without colliding with each other. However, the standard MAPF formulation relies on simplified assumptions that restrict the resolution of instances, such as a fixed start time for each agent, and a fixed speed considered uniform through the path from start to goal. Usually, an A* based path planner is applied to replan the paths of the given agents in conflict, i.e. a "Replanning" resolution technique is used. These assumptions may lead to no conflict-free solution found for a given agent and thus the rejection of operations.

Here, a rejected operation may not be an acceptable and practical outcome for a UAV Operator who must then re-submit the operation at another time without any guarantee to be accepted next time.

In particular, in the UTM context, the development of UAV traffic management in a structured urban airspace [23, 40, 84] will lead to a high density low altitude airspace. Hence, the possibly more restricted space for conflict resolution requires the conception of efficient Pre-Flight CDR methods.

Therefore, in this chapter we propose to improve the performance of MAPF solvers for the Pre-Flight CDR problem with two main contributions:

- We introduce a novel extension of the MAPF model that incorporates temporal resolution, that is "scheduling". We propose two methods based on temporal resolution: (1) "Takeoff scheduling", whereby the start times of each UAV agent can be postponed, and (2) "Speed scheduling", whereby the speed of a UAV agent can be decreased on a segment of its given path.

- We propose a distinction of different conflict types that enables the combination of different resolution techniques, "Takeoff scheduling", "Speed scheduling" and the standard "Replanning", into the ECBS algorithm which is a state-of-the-art MAPF solver.

To the best of our knowledge, we are the first to introduce an informed conflict resolution strategy and incorporate a temporal resolution with start times delay and speed change into the MAPF framework. We compare several combinations of the three considered resolution techniques that are "Replanning", "Take-off scheduling", and "Speed scheduling", in terms of the amount of rejected operations and average delays from initial flight path.

We evaluate our approaches on scenarios based on a realistic estimated demand of UAV delivery operations in an urban high density airspace in Tokyo, Japan.

5.2 Related Works

5.2.1 Multi-Agent Path Finding

The Multi-Agent Path Finding (MAPF) problem has been extensively studied in many works proposing approaches to solve instances for 2D benchmarks [26,55]. The existing methods generally assume a global objective to minimize, such as the sum of costs or the makespan. To minimize these global objectives, optimal and suboptimal approaches have been proposed [6,75,76,81,82]. However, the MAPF standard formulation does not address the practical requirements of real world domains. Recently, some extensions of the MAPF formulation have been proposed to handle real-world scenarios, notably for ground robot path finding in Amazon Robotics warehouses [55]. These extensions [38,54,56] mainly consider teams of agents and focus on allocating tasks to each agent while providing conflict-free paths. Other recent works on MAPF also focus on the specific traffic topology in Amazon warehouses [18,52]. [7] proposed a scheduling-based approach for MAPF on weighted graphs, whereby the agents will wait a certain amount of time in some vertices of their path to resolve their conflicts. In [37], an extension of the MAPF problem to the Pre-Flight CDR problem in the UTM context was presented. However, as mentioned in 5.1 several extensions remain to be made to fully address all the requirements and specifics of the Pre-Flight CDR problem. Here, we will relax some assumptions of the MAPF formulation for Pre-Flight CDR by allowing start times to be rescheduled and speeds to be changed.

5.2.2 Traffic Scheduling

Scheduling is a widely studied problem in a variety of domains in operations research, such as the job shop scheduling problem and variants [12], in air traffic [100], in ground traffic [19,20] and so on. The base concept is to determine an order between agents by assigning a different start time for each one so that there is no conflict for a common resource usage. In particular, for ground traffic, the intersection scheduling problem is a well studied problem where scheduling resolution is needed. Here, a collision-free solution must be determined for given cars at a traffic intersection by computing an optimal sequence (schedule) that defines the order of each vehicle passing through the intersection. Typically, mixed integer programming techniques [1,19,20] are used to model an instance and find an optimal solution whereby the total generated delay for each car or the makespan will be minimized.

In contrast, in the UTM context, low altitude airspace is not as restricted as ground traffic. Cars must be driven on predetermined roads with unique directions and rules. Thus, frontal collisions, i.e. collisions between two cars going in opposite directions and that we will call "head-on" collisions in the following would never happen. In contrast, there are no fixed roads in airspace which is an open space where UAVs can fly in any directions. Thus, "head-on" collisions between two UAVs might happen, making the known scheduling

techniques inefficient, as one of the two agents in conflict would have to let the other go through the whole path segment.

5.3 Problem Formulation

In this section, we present a novel extension of the MAPF formulation which relaxes some assumptions of the standard formulation to allow more flexibility in the resolution of instances.

A MAPF problem can be extended to a Pre-Flight CDR problem as follows:

An instance of our problem is composed of N operations $O = \{O_1, \ldots, O_N\}$, which are performed by agents (UAVs), and a 3D undirected graph $G = (V, E)$. Agents can 'move' along an edge of G. Note that 'wait' moves on a vertex of G is not considered here as this would increase the computational complexity of our approach by significantly increasing the search space, since several possibilities would need to be considered as to where to hover exactly in the flight path. Furthermore, generally, some UAVs like fixed-wings might not be able to stop instantly mid-air, that is 'hover'.

An agent a_i assigned to perform O_i is characterized by:

- A *radius* r_i: each UAV is conceived as a sphere of "total" radius, which is composed of several layers that relate to operational risk such as navigation errors and uncertainties;
- A *speed* sp_i: the given maximum speed of a_i
- A *start* s_i and *goal* g_i location: an operation O_i is composed of a pair of paths, an outbound path and a return path. The outbound path leads from a hub location s_i to a delivery location g_i, and the return path is assumed to be symmetrical to the outbound path. We assume a fixed duration δ_i for the duration between when a UAV lands to deliver some good and start returning to its hub location;

  The flight path Π_i of a_i, which is composed of the outbound path and the return path, is thus a sequence of *segments* composed of pairs of consecutive waypoints with associated timesteps that a_i must follow. Each segment of the flight path has an associated speed that is initially set to sp_i and that a_i will apply when moving through this segment.

  When an agent returns to the hub location, we assume that they do not remain in the space since each UAV lands when reaching a given location.
- A *start time* $t_i^s > 0$: the time at which the operation must start, hence when the agent takes off.

Here, we propose a novel formulation of the MAPF model for Pre-Flight CDR which relaxes two assumptions of the standard formulation by making the start time and the speed of each agent flexible, as summarized in Table 5.1.

UTM Standard MAPF problem assumptions	UTM Extended MAPF problem assumptions
Fixed start time	Variable start time
Uniform speed on the whole flight path	Maximum speed that can be decreased on given path segments

Table 5.1: Comparison between Standard MAPF model assumptions and Extended MAPF model for Pre-Flight CDR assumptions.

In the UTM context, we define the altitude constraints for any given point in space of coordinates (x, y, z) and which belongs to the path of an operation as:

$$elev(x, y) + alt_{min} \leq z \leq elev(x, y) + alt_{max} \tag{5.1}$$

With $elev(x, y)$ the elevation value of the point of coordinates (x, y) in a path, and alt_{min} and alt_{max} the fixed altitude bounds. Each agent is represented by a sphere of given radius r_i, and a center position p_i. To avoid a conflict, we need to prevent any violation of the minimum separation distance between two agents a_i and a_j, i.e., the sum of their respective radii, $r_i + r_j$. Hence, we define the following constraint to ensure there is no conflict:

$$\forall t, dist(p_i(t), p_j(t)) > r_i + r_j \tag{5.2}$$

The objective we hereby adopt remains the same as in standard MAPF, i.e., to minimize the sum of individual costs: $\min \sum_{O_i \in O} T_i$, with T_i the total cost of the operation O_i, which is the total duration of an operation. A solution consists of conflict-free paths for all N operations such that no violation of minimum separation occurs.

Here, the precise integration of kinematics is not in the scope of this chapter. We only consider quadcopters which are holonomic agents, and thus can directly follow any given trajectory without particular motion constraints.

5.4 Approach

In this section, we describe the different baseline approaches for conflict resolution considered and their properties. We will incorporate these proposed approaches into the ECBS algorithm 2.4. Finally, we introduce a distinction of types of conflict and describe a combination mechanism of the resolution techniques that relies on the conflict type detected.

5.4.1 Baseline Approaches

In this section, we present three baseline approaches for conflict resolution whereby each approach applies one distinct resolution technique.

Technique	Abbreviation
Replanning	R
Takeoff Scheduling	T
Speed Scheduling	S

Table 5.2: Abbreviations used.

Replanning

Replanning refers to the standard conflict resolution technique used in ECBS with an A* based path planner. When a conflict is detected between two agents as in Eq. 5.2, a constraint is generated. This constraint corresponds to an interval in space and time that each agent must respectively avoid. With Replanning, the A* based path planner computes a new conflict-free path that avoids all the given constraints in space and time.

Takeoff Scheduling

When a conflict is detected between two agents as in Eq. 5.2, a conflict time interval $\Delta_{conflict}$ is computed which represents the time when the conflict starts and the time when it ends, hence until there is no violation of the minimum separation distance between the two agents. With Takeoff scheduling, a delay is respectively added to the start times of the agents to postpone their takeoff.

Speed Scheduling

With Speed scheduling, the computed conflict time interval $\Delta_{conflict}$ is again used, but unlike Takeoff scheduling, the delay is not added by postponing the takeoff time of the agent but on a segment of its flight path as defined in Section 5.3. We decrease the assigned speed on the segment of duration $\Delta_{segment}$ (the time taken by the UAV agent to go through this segment) that is situated before the segment where the conflict occurs to generate a delay equivalent to the conflict time interval $\Delta_{conflict}$. If the conflict occurs on the first segment of the flight path, then the delay is added to the start time of the agent as in Takeoff scheduling.

5.4.2 Combination Approaches

In the following, we will use abbreviations to refer to the introduced baseline approaches as shown in Table 5.2. We propose to consider Replanning (R), Takeoff scheduling (T), and Speed scheduling (S) resolution techniques into the ECBS algorithm when resolving a detected conflict at each node of the constraint tree.

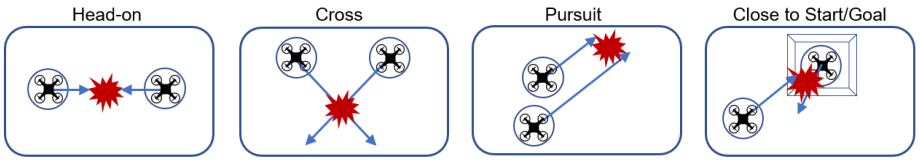


Figure 5.1: The different conflict types that occur in our instances.

Conflict Types

We propose a classification of the conflict types detected. We distinguish four distinct types as represented in Fig. 5.1:

- ***Head-on***: The two agents are going towards each other in opposite directions.
- ***Cross***: The directions of each agent are non-colinear.
- ***Pursuit***: If one of the two agents overpass the other. In the following, this conflict type is grouped with the *Cross* type, as *Pursuit* is a particular case of *Cross* whereby the agents go towards the same direction.
- ***Close to Start/Goal***: If the conflict occurs near a hub or service location while one of the agent is taking off or returning.

We enable the conflict type identification with the use of simple geometrical computations in the conflict detection step in ECBS. In this step, agents paths are compared pairwise to detect the earliest conflict between two agents. Since each agent's path is composed of a sequence of *segments*, we compare the directions taken by each agent on their respective *segments* during the common time interval. Depending on the colinearity of the two agents' directions when a conflict is detected, we can thus identify the corresponding conflict type. If the conflict is located near a hub or a service location within a fixed small time window, then it is considered as a *Close to Start/Goal* conflict.

Conflict Type	**R**	**T**	**S**	**T+S**	**R+S**	**R+T**	**R+T+S**
Head-on	✓	✓	✗	✓(T)	✓(R)	✓(R)	✓(R)
Cross / Pursuit	✓	✓	✓	✓(S)	✓(S)	✓(T)	✓(S)
Close to Start/Goal	✗	✓	✗	✓(T)	✗	✓(T)	✓(T)

Table 5.3: Conflict resolution techniques considered and solvability for each conflict type. The ✓ mark means that the technique can solve the given conflict type, the ✗ mark means that it cannot solve the given conflict type. For combined techniques, the associated baseline technique used to solve a particular conflict type is indicated in parenthesis.

With this classification of conflict type, we can distinguish several cases

where one resolution technique can or cannot solve the given conflict (see Table 5.3).

Replanning and Speed scheduling cannot solve *Close to Start/Goal* conflicts as a hub or service location is fixed. This conflict type can only be solved with Takeoff scheduling.

A *Head-on* conflict cannot be solved by Speed scheduling since it does not change the path.

A *Cross* or *Pursuit* conflict can be solved by any of the considered resolution techniques, however using Takeoff scheduling or Speed scheduling may provide more efficiency in the induced delays, as we only add the precise delay in time without changing the path unlike with Replanning.

Therefore we present a solver which combines the baseline approaches Replanning, Takeoff scheduling and Speed scheduling according to the conflict type detected (see Table 5.3). We consider every possible combinations: T+S, R+S, R+T and R+T+S.

Note that for R+S and R+T+S approaches, Replanning is used in case of the resolution of Head-on conflicts as indicated in Table 5.3 and this changes the spatial configuration of a flight path. These modifications might cancel the effects of the previous resolutions of conflicts with Speed scheduling on certain segments of the flight path. Thus, when a flight path is replanned to solve a given conflict, we introduce in the A* based search additional constraints on the speed to be considered when searching for a path. By default, Replanning always considers the maximum speed of the given agent. However, if the current node being explored by the path planner and its neighbour node being expanded belong to a segment that has been previously affected by Speed scheduling (has a lower speed associated), then a computation is performed to ensure that the maximum speed satisfies the previously solved conflicts or if it must apply a lower speed in this case.

Solution quality criteria

The introduction of temporal resolution techniques into MAPF solvers offers different features in terms of practical criteria on the obtained solutions. These criteria are the amount of delay induced, the additional battery consumption, the flight plan spatial modification, and rejected operations.

To understand the possible trade-offs of our proposed approaches, we compare the advantages and drawbacks for these properties between the different considered approaches in Table 5.4.

First, using the Replanning approach (R) presents three drawbacks:

- It induces additional battery consumption with the possibly longer path computed.
- It modifies the path which might not always be tolerated by some users.
- It can generate rejected operations due to not being able to solve *Close to Start/Goal* conflicts

Solution quality criteria	**R**	**T**	**S**	**T+S**	**R+S**	**R+T**	**R+T+S**
Delay	☺	☹	☺	☹	☺	☺	☺
Battery consumption	☹	☺	😐	😐	☹	😐	😐
Flight plan spatial modification	☹	☺	☺	☺	😐	😐	😐
Rejected operations	☹	☺	☹	☺	☹	☺	☺

Table 5.4: Criteria for solution quality analysis for the three considered baseline resolution techniques and their different combinations. ☺: Low to None effect = Good; 😐: Moderate effect = Medium; ☹: High effect = Bad.

In contrast, the Takeoff scheduling approach (T) induces a possibly high amount of generated delay because of *Head-on* conflicts, as one of the two agents would have to be delayed until the other has moved through the entire path segment where the *Head-on* conflict occurs.

However it does not require additional battery consumption since UAVs will takeoff later, nor any path spatial modification, and does not induce any rejected operations by allowing to solve *Close to Start/Goal* conflicts.

Then, while the Speed scheduling approach (S) does not modify a path spatially like Takeoff scheduling, slowing down induces more battery consumption, and it cannot solve *Close to Start/Goal* and *Head-on* conflicts, thus generating a possibly larger number of rejected operations.

The combinations approaches T+S and R+S both present drawbacks inherited from the baseline approaches used. In contrast, the combinations R+T and R+T+S both present no drawbacks for these criteria.

5.5 Experiments

In this section, we first describe the characteristics of the realistic model case used for our simulations based on Tokyo area. Then, we present and analyze the results obtained in our experiments.

5.5.1 Model Case Scenario

This section describes the model case used in our experiments that is based on the estimated UAV service demand in Tokyo area, in Japan.

The dimensions of the considered area in the given region are of 12.8 km×12.8 km. The representation of our space is a 3D octree, a hierarchical data structure that recursively divides a spatial volume into smaller subspaces. Here the octree is composed of cells with maximum granularity edges of 30 meters long.

The positions of static obstacles, i.e., blocked cells, is fixed according to the given elevation map of the region.

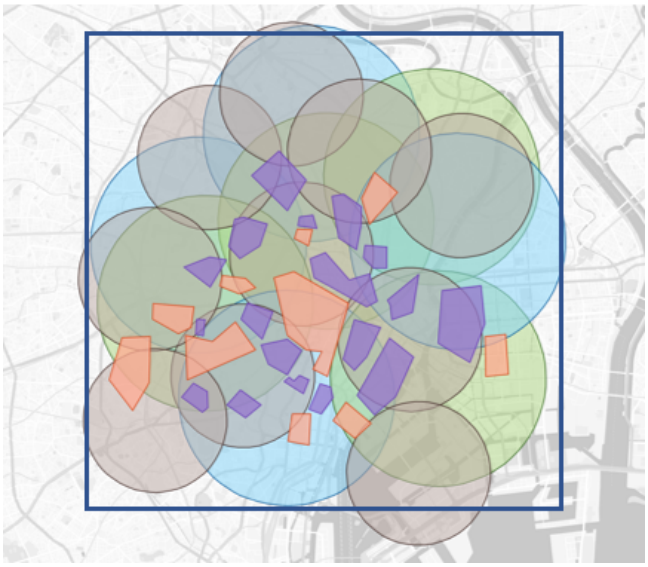


Figure 5.2: Tokyo 12.8km × 12.8km area scenario. Each defined hub has a service area (circle) represented with a different color corresponding to the associated company (A, B or C). No-Fly-Zones are represented in red and Restricted-Fly-Zones in purple.

We also include no-fly-zones whereby no UAVs are allowed to fly through, and restricted-fly-zones whereby some UAVs belonging to a designated class may be allowed to fly through, while others would have to avoid these zones based on areas with different ground risks. The study considers three major logistics companies (hereby anonymized as A, B and C) that provide deliveries of goods such as mail and package delivery. The different hub locations, including a fixed number of drone ports where UAVs takeoff/land for each hub, and their service area are set from the expected demand in the area as shown in Fig. 5.2. The assignment of the service locations is assumed to be done independently by UAS operators within a given service radius around each hub. Figure 5.2 depicts positions of hubs and their associated service area with a different color for each company. Inside these areas, we fix the minimum flight path length to 300 m.

We extrapolated data from an existing study on UAV delivery in another region in Japan predicted for 2030. One day of delivery service represents 13 hours, from 8 a.m to 9 p.m. According to customers' demand in the study, expected weight of deliveries, and expected capabilities of UAVs, we estimate that there is a total demand of up to 3000 operations per hour considering the given companies altogether. Each UAV has an attributed speed between 9m/s and 21m/s and a given radius r between 24m and 30m.

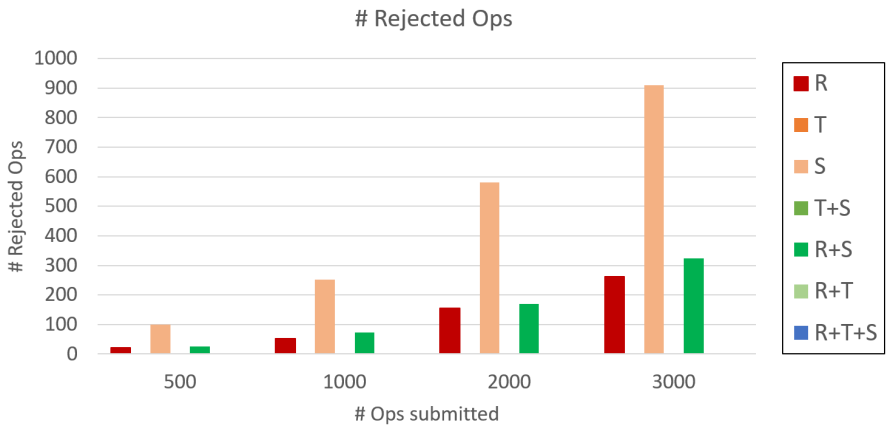


Figure 5.3: Comparison of average number of rejected operations for each approach.

5.5.2 Experimental Results

All approaches are implemented in Java and are run on a 3.2GHz Intel Core i7-8700 desktop with 16 GB RAM.

The starting times of operations are uniformly distributed within a 1-hour time window. In all experiments, the number of submitted operations varies from 500 to 3000 operations in total, and the operations are distributed among the given hubs with fixed proportions according to the study. 30 instances were generated for each experiment.

We evaluate the performances of our approaches with two main metrics:

- Average number of operations rejected
- Average delay per operation

Number of rejected operations

In Fig. 5.3, we observe that the number of rejected operations is reduced to zero with the use of Takeoff scheduling, that is, in the T, R+T, T+S, R+T+S approaches. Otherwise, this number is non negligible for R, S and R+S and is even higher for the S approach which can neither solve *Close to Start/Goal* conflicts nor *Head-on* conflicts.

We also observe that the rejected operations are mostly caused because of the traffic pattern induced by UAV delivery service whereby several *Close to Start/Goal* conflicts tend to occur since many UAVs will takeoff from fixed hubs locations.

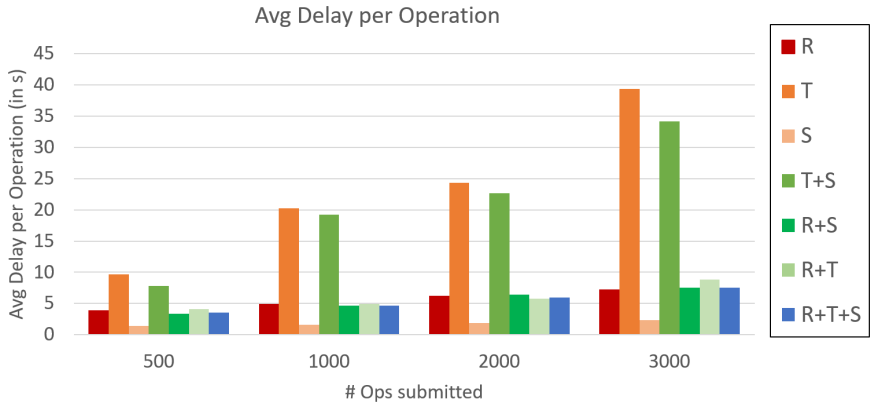


Figure 5.4: Comparison of average delay per operation for each approach.

Delays

As shown in Fig. 5.4, R, S and R+S approaches have the highest number of rejected operations, but they present a lower deviation in average. In contrast, T and T+S approaches present a higher generated delay due to the presence of *Head-on* conflicts but no rejected operations.

The S approach allows lower average delays compared to the T approach, because it only impacts a portion of the flight path when adding a delay, while T impacts the whole flight path since the delay is added at the start. So S is less likely to create new conflicts than T.

Finally, in Fig. 5.4, we observe that the R+T and R+T+S approaches as also showed in Table 5.4 have the best performances, as they allow to avoid rejected operations and maintain low delay at the same time. The R+T+S approach presents a slightly lower average delay with the use of the S technique than the R+T approach. This result is expected, as R+T and R+T+S approaches both combine Replanning and Takeoff scheduling resolution techniques.

5.6 Conclusions

In this chapter, we proposed a novel extension of the MAPF (Multi-Agent Path Finding) framework to the Pre-Flight CDR problem. The standard MAPF formulation provides limited assumptions whereby UAV agents' start times are fixed, and they apply a uniform speed on their path. Therefore, we hereby propose to relax those assumptions to improve the efficiency of MAPF solvers, in particular in high density structured airspace. We introduce two methods for conflict resolution based on temporal resolution, that are Takeoff scheduling and Speed scheduling. These techniques present different characteristics that can improve the solution quality in terms of practical properties, such as battery

consumption, spatial modifications and amount of rejected operations. We efficiently combine these techniques with the standard Replanning technique used in the ECBS algorithm that spatially modifies a path based on time-space A* to avoid conflicts obstacles. For this purpose, we introduced a conflict type distinction into the conflict detection step of ECBS to determine the most efficient resolution technique to be applied depending on the conflict type detected.

We experimentally compared the proposed approaches in a realistic model case of UAV delivery service in a a high density structured airspace predicted in the Tokyo area in Japan. Our results suggest that incorporating scheduling techniques into MAPF solvers allows to reduce significantly the number of rejected operations, and that the efficient combination of replanning and scheduling reduces the average deviation generated for all given UAV operations.

Future Works for Pre-Flight CDR

In our future work, we plan to address several points to improve our proposed approaches for Pre-Flight CDR:

- In Chapter 2, we used arbitrarily fixed sizes for each batch processing, so the efficiency of the mechanism could be improved with the choice of these batch sizes. A "smart" implementation of a Pre-Flight CDR method will estimate the expected demand (i.e., number of requests) and select the batch size accordingly, resulting in a dynamic size.

- In this thesis, we only consider quadcopters UAVs, thus that have holonomic motion. We want to also consider fixed-wing UAVs with precise kinematics to better represent the future airspace that might include passenger drones. Fixed-wings UAVs have different kinematics from quadcopters UAVs as they are non-holonomic and cannot hover. The consideration of wind dynamics and effects is another limitation here to provide more accurate description of the trajectories of UAVs in these conditions.

- We also plan to address other realistic scenarios distinct from the Sendai and the Tokyo model cases with different traffic patterns.

- Finally, as discussions on the conception of a UTM system are ongoing, we plan to explore more advanced concepts in high density structured airspace with the consideration of corridors and other possible requirements that challenge the efficiency of CDR methods.

Final Conclusions

As Unmanned Aerial Vehicles (UAVs) are expected to populate the skies, the safe integration of UAVs into shared low altitude airspace will require the deployment of a specific traffic management system in the near future, that is an Unmanned Aircraft System Traffic Management (UTM) system. Therefore the conception of a UTM system to ensure safety poses several challenges that we have proposed to tackle in this dissertation.

The main objective of UTM is to ensure that all existing conflicts, i.e. predicted collisions between UAVs, are resolved. This issue leads to the need for the conception of Conflict Detection and Resolution (CDR) methods.

Therefore, our research has focused on the conception and development of CDR methods and the exploration of different settings and concepts to provide meaningful recommendations emerging from our studies.

Following the defined concept of redundancy layers from ICAO [39] for UTM, we have addressed and contributed to the conception of the two main CDR layers: In-Flight CDR and Pre-Flight CDR. These layers are complementary and have distinct properties that require different methods for each.

- First, in Chapter 1 we have proposed a suitable and practical method for In-Flight CDR that allows UAVs to avoid collisions in real time while flying. This method was based on the state-of-the-art algorithm Optimal Reciprocal Collision Avoidance (ORCA) which was originally applied to ground robots or pedestrian simulations. We extended the application of this algorithm to the practical constraints of In-Flight CDR for UAVs.

- Then, in Chapter 2 we proposed a mapping of the Pre-Flight CDR problem to the Multi-Agent Path Finding (MAPF) problem. Although MAPF is a well-studied problem in the multi-agent systems (MAS) community, many realistic considerations remain to be addressed to be applicable to real world problems, in particular for the Pre-Flight CDR problem in the UTM domain. Therefore we addressed these limitations by extending the standard MAPF formulation to the properties of the Pre-Flight CDR problem such as heterogeneous UAV agents and ongoing processing of submitted operations. For these purposes, we incorporated efficient conflict detection mechanisms, and assessed different processing methods like first come first served and batch processing. We based our extension on the state-of-the-art MAPF solver, ECBS that is a bounded suboptimal variant of the CBS algorithm. We showed the efficiency of batch processing over first come first served in high demand situations.

- As UTM is a recent field with ongoing discussions between different stakeholders and that a concrete system has not yet been deployed, there is the need to study and assess the efficiency and applicability of the different possibilities. In Chapter 3, we proposed such assessments with simulations and adapted metrics, to provide recommendations for the practical requirements of UTM. In particular, we showed that 4DT (3D space +

time Trajectories) was a practical and effective concept applicable to the Pre-Flight phase to process UAV operations before their takeoff.

- Another main practical requirement considered in the design of a UTM system is the conception of a decentralized Pre-Flight CDR approach. In Chapter 4 we extended the MAPF framework to allow UAS Service Providers (UASSP) agents to process conflicts between them and to be able to introduce individual costs as economic entities. We introduced the notion of "fairness" as a new objective to comply with the requirement of Pre-Flight CDR between communicating UASSP agents to provide a balanced distribution of costs in terms of induced delays and rejected operations. To fulfill this objective, we proposed a practical negotiation approach.

- Finally, in Chapter 5, we explored another extension of the MAPF model for Pre-Flight CDR by relaxing some assumptions of MAPF. We introduce two methods for conflict resolution based on temporal resolution, that are takeoff scheduling and speed scheduling. We efficiently combine these techniques with the standard replanning technique used for MAPF solvers such as the ECBS algorithm which spatially modifies a path, based on space-time A* search to avoid spatio-temporal obstacles. For this purpose, we introduced a conflict type distinction into the conflict detection step of ECBS that allows to determine the most efficient technique to be applied depending on the conflict type detected. Incorporating scheduling techniques into MAPF solvers allows to reduce significantly the number of rejected operations, and that the efficient combination of replanning and scheduling reduces the average delays generated for all given UAV operations.

In this dissertation, we proposed practical methods for two phases in the UTM conception to ensure the safe integration of UAVs in low altitude airspace, Pre-Flight CDR and In-Flight CDR. These methods are scalable with the number of UAV agents considered.

For In-Flight CDR, our proposed method based on ORCA is shown to be scalable with the number of UAV agents. We further improved the scalability of the ORCA method by the incorporation of conflict detection filtering steps that allowed to reduce the amount of computations performed for each UAV agent.

For Pre-Flight CDR, our proposed method based on the ECBS algorithm scales with the number of UAV operations to process, that is with a high demand estimated in the Sendai area. Here, the performance of path planning techniques relies on the properties of the environment that are discretization, topology, and constraints. Our proposed Pre-Flight CDR method based on ECBS is shown to be scalable with the number of UAV agents, as we incorporated an efficient combination of spatio-temporal pruning, geometrical computations to efficiently handle heterogeneous UAV agents, grid discretization, and choice of heuristic. We showed the scalability in terms of the number of UAV operations for our

methods for Pre-Flight CDR by comparing FCFS processing to Batch processing in high demand instances.

Our proposed methods ensure that any pairs of UAV agents stay away from each other at the minimum separation distance at any time. Moreover, the redundancy provided by the complementary layers of Pre-Flight CDR and In-Flight CDR will allow to ensure safety and robustness in the deployment of a UTM system.

We built these methods by expanding upon existing models and algorithms from the AI research community. On the one hand, UTM is a recent domain which came to fruition with the increase in the use of UAVs in the last years and their predicted rise in the next following years. Large organizations in the world such as NASA (National Aeronautics and Space Administration) in the USA and JAXA (Japan Aerospace Exploration Agency) in Japan have started to promote concepts for such UTM system, but they are still in early stages of development. On the other hand, established research fields in the multi-agent systems (MAS) community such as MAPF, Negotiation, and Scheduling techniques have been well studied, although, mainly in theoretical settings thus lacking realistic and practical considerations.

We built practical and efficient solutions for CDR by extending theoretical methods to accommodate UTM specific requirements. We studied the various requirements in the UTM system design with key stakeholders discussions, and were able to assess them by performing simulations to provide meaningful recommendations for the conception of UTM.

Future research works will include studying and developing advanced and improved concepts for the deployment of a UTM system in the near future.

We hope that our work on CDR methods can contribute to effective candidate solutions for UTM technology, and ultimately to the efficient integration of UAVs in shared low altitude airspace.

Bibliography

[1] H. Ahn and D. Del Vecchio. Semi-autonomous intersection collision avoidance through job-shop scheduling. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 185–194, 2016.

[2] P. O. Alexander Alexopoulos, Amr Kandil and E. Badreddin. A comparative study of collision avoidance techniques for unmanned aerial vehicles. In *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1969–1974, 2013.

[3] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley. Collision avoidance for aerial vehicles in multi-agent scenarios. In *Autonomous Robots*, pages 101–121, 2015.

[4] O. Amir, G. Sharon, and R. Stern. Multi-agent pathfinding as a combinatorial auction. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, pages 2003–2009, 2015.

[5] L. K. Asante and F. J. S. Nieto. Complexity in the optimization of ATM performance metrics. In *Proceedings of the 2Nd International Conference on Application and Theory of Automation in Command and Control Systems*, pages 162–165, 2012.

[6] M. Barer, G. Sharon, R. Stern, and A. Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Seventh Annual Symposium on Combinatorial Search (SOCS)*, 2014.

[7] R. Barták, J. Švancara, M. Vlk, et al. A scheduling-based approach to multi-agent path finding with weighted and capacitated arcs. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 748–756. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[8] Z. Beck, L. Teacy, A. Rogers, and N. Jennings. Online planning for collaborative search and rescue by heterogeneous robot teams. In *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems*, pages 1024–1032, 2016.

[9] L. F. Bertuccelli, H.-L. Choi, P. Cho, and J. How. Real-time multi-UAV task assignment in dynamic and uncertain environments. In *Proceedings of American Institute of Aeronautics and Astronautics Guidance, Navigation, and Control Conference*, pages 1–16, 2009.

[10] J. Bezdek, R. Ehrlich, and W. Full. FCM: The fuzzy C-means clustering algorithm. *Computers & Geosciences*, 10:191–203, 1984.

[11] S. Bhattacharya. Distributed optimization with pairwise constraints and its application to multi-robot path planning. *Robotics: Science and Systems VI*, 177, 2011.

[12] J. Błażewicz, W. Domschke, and E. Pesch. The job shop scheduling problem: Conventional and new solution techniques. *European journal of operational research*, 93(1):1–33, 1996.

[13] Z. Bnaya, R. Stern, A. Felner, R. Zivan, and S. Okamoto. Multi-agent path finding for self interested agents. In *Symposium on Combinatorial Search (SOCS)*, 2013.

[14] E. Boyarski, A. Felner, R. Stern, G. Sharon, O. Betzalel, D. Tolpin, and E. Shimony. ICBS: the improved conflict-based search algorithm for multi-agent pathfinding. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, (IJCAI)*, pages 740–746, 2015.

[15] M. Cap, P. Novak, A. Kleiner, and M. Selecky. Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Trans. Automation Science and Engineering*, 12(3):835–849, 2015.

[16] M. Choi, A. Rubenecia, T. Shon, and H. H. Choi. Velocity obstacle based 3D collision avoidance scheme for low-cost micro UAVs. *Sustainability*, 9(7), 2017.

[17] R. Christien, A. Benkouar, T. Chaboud, and P. Loubieres. Air traffic complexity indicators ATC sectors classification. In *The 21st Digital Avionics Systems Conference*, volume 1, 2002.

[18] L. Cohen, T. Uras, and S. Koenig. Feasibility study: Using highways for bounded-suboptimal multi-agent path finding. In *Symposium on Combinatorial Search (SOCS)*, pages 2–8, 2015.

[19] A. Colombo and D. Del Vecchio. Efficient algorithms for collision avoidance at intersections. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pages 145–154, 2012.

[20] A. Colombo and D. Del Vecchio. Least restrictive supervisors for intersection collision avoidance: A scheduling approach. *IEEE Transactions on Automatic Control*, 60(6):1515–1527, 2014.

[21] K. Dalamagkidis, K. P. Valavanis, and L. A. Piegl. On integrating unmanned aircraft systems into the national airspace system. *International Series on Intelligent Systems, Control, and Automation: Science and Engineering*, 36, 2009.

[22] B. de Wilde, A. Mors, and C. Witteveen. Push and rotate: cooperative multi-agent path planning. In *International conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 87–94, 2013.

[23] DLR. Integrating uas into the future aviation system. Technical report, 2017.

[24] C. Ericson. *Real-time collision detection*. CRC Press, 2004.

[25] A. Felner, J. Li, E. Boyarski, H. Ma, L. Cohen, S. Kumar, and S. Koenig. Adding heuristics to conflict-based search for multi-agent path finding. In *International Conference on Automated Planning and Scheduling*, 2018.

[26] 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 *Proceedings of the Tenth International Symposium on Combinatorial Search (SOCS)*, pages 29–37, 2017.

[27] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760–772, 1998.

[28] A. G. Foina, C. Krainer, and R. Sengupta. An unmanned aerial traffic management solution for cities using an air parcel model. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1295–1300, 2015.

[29] A. Gilboa, A. Meisels, and A. Felner. Distributed navigation in an unknown physical environment. In *Proceedings of the 5th international joint conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 553–560, 2006.

[30] F. Glover and M. W. Laguna. *Tabu Search*. Springer, 1997.

[31] M. Goldenberg, A. Felner, , R. Stern, G. Sharon, N. Sturtevant, R. C. Holte, and J. Schaeffer. Enhanced partial expansion A*. In *Journal Of Artificial Intelligence Research*, volume 50, pages 141–187, 2014.

[32] R. Golding. Metrics to characterize dense airspace traffic. In *Altiscope Airbus Technical Report*, 2018.

[33] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey. Clearpath: Highly parallel collision avoidance for multi-agent simulation. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 177–187, 2009.

[34] F. Ho, R. Geraldes, A. Goncalves, M. Cavazza, and H. Prendinger. Simulating shared airspace for service UAVs with conflict resolution. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 2192–2194, 2018.

[35] F. Ho, R. Geraldes, A. Goncalves, M. Cavazza, and H. Prendinger. Improved conflict detection and resolution for service UAVs in shared airspace. *IEEE Transactions on Vehicular Technology*, 68(2):1231–1242, 2019.

[36] F. Ho, R. Geraldes, A. Goncalves, B. Rigault, A. Oosedo, M. Cavazza, and H. Prendinger. Pre-flight conflict detection and resolution for UAV integration in shared airspace: Sendai 2030 model case. *IEEE Access*, 7:170226–170237, 2019.

[37] F. Ho, A. Salta, R. Geraldes, A. Goncalves, M. Cavazza, and H. Prendinger. Multi-agent path finding for UAV traffic management. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 131–139, 2019.

[38] W. Hoenig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian. Conflict-based search with optimal task assignment. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 757–765, 2018.

[39] ICAO. Global air traffic management operational concept. In *Doc 9854, First Edition*, 2005.

[40] D.-S. Jang, C. A. Ippolito, S. Sankararaman, and V. Stepanyan. Concepts of airspace structures and system analysis for UAS traffic flows for urban areas. In *AIAA Information Systems-AIAA Infotech@ Aerospace*, page 0449. 2017.

[41] P. Janovsky, M. Cap, and J. Vokrinek. Finding coordinated paths for multiple holonomic agents in 2D polygonal environment. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multiagent Systems*, pages 1117–1124, 2014.

[42] Y. I. Jenie, E.-J. v. Kampen, C. C. de Visser, J. Ellerbroek, and J. M. Hoekstra. Selective velocity obstacle method for deconflicting maneuvers applied to unmanned aerial vehicles. *Journal of Guidance, Control, and Dynamics*, 38(6):1140–1146, 2015.

[43] Y. I. Jenie, E. van Kampen, J. Ellerbroek, and J. M. Hoekstra. Safety assessment of a UAV CDR system in high density airspace using Monte Carlo simulations. *IEEE Transactions on Intelligent Transportation Systems*, 19(8):2686–2695, 2018.

[44] Y. I. Jenie, E.-J. van Kampen, J. Ellerbroek, and J. M. Hoekstra. Taxonomy of conflict detection and resolution approaches for unmanned aerial

vehicle in an integrated airspace. *IEEE Transactions on Intelligent Transportation Systems*, 18(3):558–567, 2016.

[45] J. Jung, S. N. Souza, M. A. Johnson, A. K. Ishihara, H. C. Modi, B. Nikaido, and H. Hassee. Applying required navigation performance concept for traffic management of small unmanned aircraft systems. In *Proceedings of the International Council of the Aeronautical Sciences*, 2016.

[46] E. Kalai and M. Smorodinsky. Other solutions to Nash's bargaining problem. *Econometrica*, 43(3):513–518, 1975.

[47] S. Kiesel, E. Burns, C. Wilt, and W. Ruml. Integrating vehicle routing and motion planning. In *International Conference on Automated Planning and Scheduling*, pages 137–145, 2012.

[48] P. Kopardekar, K. Bilimoria, and B. Sridhar. Initial concepts for dynamic airspace configuration. In *Proceedings of AIAA Aviation Technology, Integration, and Operations Conference*, 2007.

[49] P. Kopardekar, J. Rios, T. Prevot, M. Johnson, J. Jung, and J. E. Robinson. Unmanned Aircraft System Traffic Management (UTM) Concept of operations. In *Proceedings of AIAA Aviation Technology, Integration, and Operations Conference*, 2016.

[50] P. Kopardekar, A. Schwartz, S. Magyarits, and J. Rhodes. Airspace complexity measurement: An air traffic control simulation analysis. In *International Journal of Industrial Engineering*, pages 61–70, 2009.

[51] J. K. Kuchar and L. C. Yang. A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems*, 1(4):179–189, 2000.

[52] J. Li, G. Gange, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig. New techniques for pairwise symmetry breaking in multi-agent path finding. In *International Conference on Automated Planning and Scheduling, ICAPS*, 2020.

[53] R. Luna and K. E. Bekris. Push and swap: Fast cooperative path-finding with completeness guarantees. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 294–300, 2011.

[54] H. Ma and S. Koenig. Optimal target assignment and path finding for teams of agents. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1144–1152, 2016.

[55] H. Ma, S. Koenig, N. Ayanian, L. Cohen, W. Hoenig, T. K. S. Kumar, T. Uras, H. Xu, C. Tovey, and G. Sharon. Overview: Generalizations of multi-agent path finding to real-world scenarios. In *CoRR*, 2017.

[56] H. Ma, J. Li, T. K. S. Kumar, and S. Koenig. Lifelong multi-agent path finding for online pickup and delivery tasks. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 837–845, 2017.

[57] M. Machida. Polynomial-time multi-agent pathfinding with heterogeneous and self-interested agents. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 2105–2107, 2019.

[58] Y. Mohammad and S. Nakadai. Optimal value of information based elicitation during negotiation. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 242–250, 2019.

[59] A. Nash, K. Daniel, S. Koenig, and A. Felner. Theta*: Any-angle path planning on grids. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 1177–1183, 2007.

[60] J. F. Nash. The bargaining problem. *Econometrica: Journal of the Econometric Society*, pages 155–162, 1950.

[61] H. Y. Ong and M. J. Kochenderfer. Short-term conflict resolution for unmanned aircraft traffic management. In *Proceedings of the 34th IEEE/AIAA Digital Avionics Systems Conference*, 2015.

[62] J. Pearl and J. H. Kim. Studies in semi-admissible heuristics. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 392–399, 1982.

[63] P. Pianpak, T. C. Son, Z. O. Toups, and W. Yeoh. A distributed solver for multi-agent path finding problems. In *Proceedings of the First International Conference on Distributed Artificial Intelligence*, DAI19, pages 2:1–2:7, 2019.

[64] M. Prandini, L. Piroddi, S. Puechmorel, and S. Brazdilova. Toward air traffic complexity assessment in new generation air traffic management systems. *IEEE Transactions on Intelligent Transportation Systems*, 12:809 – 818, 2011.

[65] A. R. Pritchett and A. Genton. Negotiated decentralized aircraft conflict resolution. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):81–91, 2018.

[66] T. Radisic, D. Novak, and B. Juricic. Reduction of air traffic complexity using trajectory-based operations and validation of novel complexity indicators. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):3038–3048, 2017.

[67] S. D. Ramchurn, T. D. Huynh, Y. Ikuno, J. Flann, F. Wu, L. Moreau, N. R. Jennings, J. E. Fischer, W. Jiang, T. Rodden, E. Simpson, S. Reece, and S. J. Roberts. HAC-ER: A disaster response system based on human-agent collectives. In *Proceedings of the 2015 International Conference on Autonomous Agens and Multiagent Systems*, pages 533–541, 2015.

[68] L. Ren, M. Castillo, Effen, H. Yu, E. Johnson, T. Nakamura, Y. Yoon, and C. A.Ippolito. Small unmanned aircraft system (sUAS) trajectory modeling in support of UAS traffic management (UTM). In *AIAA Aviation Technology, Integration, and Operations Conference*, 2017.

[69] A. Richards, J. Bellingham, M. Tillerson, and J. How. Coordination and control of multiple UAVs. In *Proceedings of AIAA Guidance, Navigation, and Control Conference*, 2002.

[70] J. Rios. NASA UTM strategic deconfliction final report. Technical report, NASA Ames Research Center, 2018.

[71] J. L. Rios. UTM UAS service supplier specification. In *NASA Technical Report*, 2017.

[72] S. Roelofsen, D. Gillet, and A. Martinoli. Collision avoidance with limited field of view sensing: A velocity obstacle approach. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1922–1927, 2017.

[73] S. Roelofsen, A. Martinoli, and D. Gillet. Distributed deconfliction algorithm for unmanned aerial vehicles with limited range and field of view sensors. In *Proceedings of the American Control Conference*, pages 4356–4361, 2015.

[74] SESAR Joint Undertaking. *WP 4 En Route Operations*. SESAR, 2008.

[75] G. Sharon, R. Stern, A. Felner, and N. Sturtevant. Conflict-based search for optimal multi-agent path finding. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*, pages 563–569, 2012.

[76] G. Sharon, R. Stern, M. Goldenberg, and A. Felner. The increasing cost tree search for optimal multi-agent pathfinding. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 662–667, 2011.

[77] D. Silver. Cooperative pathfinding. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 117–122, 2005.

[78] D. Sislak, P.Volf, S. Kopriva, and M. Pechoucek. Agentfly: Scalable, high-fidelity framework for simulation, planning and collision avoidance of multiple UAVs. *Sense and Avoid in UAS: Research and Applications*, pages 235–264, 2012.

[79] D. Sislak, J. Samek, and M. Pechoucek. Decentralized algorithms for collision avoidance in airspace. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 543–550, 2008.

[80] D. Sislak, P. Volf, and M. Pechoucek. Agent-based cooperative decentralized airplane-collision avoidance. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):36–46, 2010.

[81] T. S. Standley. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

[82] P. Surynek, A. Felner, R. Stern, and E. Boyarski. Efficient SAT approach to multi-agent path finding under the sum of costs objective. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence*, pages 810–818, 2016.

[83] J. Svancara, M. Vlk, R. Stern, D. Atzmon, and R. Bartak. Online multi-agent pathfinding. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[84] D. P. Thipphavong, R. Apaza, B. Barmore, V. Battiste, B. Burian, Q. Dao, M. Feary, S. Go, K. H. Goodrich, J. Homola, et al. Urban air mobility airspace integration concepts and considerations. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3676, 2018.

[85] J. van den Berg, S. J. Guy, M. Lin, D. Manocha, C. Pradalier, R. Siegwart, and G. Hirzinger. Reciprocal n-body collision avoidance. *Robotics Research: The 14th International Symposium ISRR*, 70(1):3–19, 2011.

[86] S. Vera, J. A. Cobano, D. Alejo, G. Heredia, and A. Ollero. Optimal conflict resolution for multiple UAVs using pseudospectral collocation. In *23rd Mediterranean Conference on Control and Automation*, pages 28–35, 2015.

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

[88] 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)*, pages 534–540, 2018.

[89] S. Wandelt and X. Sun. Efficient compression of 4d-trajectory data in air traffic management. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):844–853, 2015.

[90] K.-H. C. Wang and A. Botea. Fast and memory-efficient multi-agent pathfinding. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS*, pages 380–387, 2008.

[91] K.-H. C. Wang and A. Botea. Mapp: a scalable multi-agent path planning algorithm with tractability and completeness guarantees. In *Journal Of Artificial Intelligence Research*, volume 42, pages 55–90, 2011.

[92] S. Wollkind, J. Valasek, and T. Ioerger. Automated conflict resolution for air traffic management using cooperative multiagent negotiation. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 4992, 2004.

[93] K. Yakovlev and A. Andreychuk. Resolving spatial-time conflicts in a set of any-angle or angle-constrained grid paths. In *CoRR abs*, 2016.

[94] K. Yakovlev and A. Andreychuk. Any-angle pathfinding for multiple agents based on SIPP algorithm. In *International Conference on Automated Planning and Scheduling*, page 586, 2017.

[95] J. Yang, D. Yin, Q. Cheng, and L. Shen. Two-layered mechanism of online unmanned aerial vehicles conflict detection and resolution. In *IEEE Transactions on Intelligent Transportation Systems*, pages 1–15, 2017.

[96] J. Yang, D. Yin, Y. Niu, and L. Zhu. Cooperative conflict detection and resolution of civil unmanned aerial vehicles in metropolis. In *Advances in Mechanical Engineering*, 2016.

[97] J. Yang, D. Yin, and L. Shen. Reciprocal geometric conflict resolution on unmanned aerial vehicles by heading control. In *Journal of Guidance, Control, and Dynamics*, pages 2511–2523, 2017.

[98] J. Yu and S. M. LaValle. Planning optimal paths for multiple robots on graphs. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3612–3617, 2013.

[99] Q. Zhang, Y. Song, R. R Routray, and W. Shi. Adaptive block and batch sizing for batched stream processing system. In *IEEE International Conference on Autonomic Computing (ICAC)*, pages 35–44, 07 2016.

[100] Y. Zhang, R. Su, Q. Li, C. G. Cassandras, and L. Xie. Distributed flight routing and scheduling for air traffic flow management. *IEEE Transactions on Intelligent Transportation Systems*, 18(10):2681–2692, 2017.

List of Publications

Journal Papers

[J1] Florence Ho, Ruben Geraldes, Artur Goncalves, Bastien Rigault, Benjamin Sportich, Daisuke Kubo, Marc Cavazza, and Helmut Prendinger. Decentralized multi-agent path finding for UAV traffic management (under review after revision). *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[J2] Florence Ho, Ruben Geraldes, Artur Goncalves, Bastien Rigault, Atsushi Oosedo, Marc Cavazza, and Helmut Prendinger. Pre-flight conflict detection and resolution for UAV integration in shared airspace: Sendai 2030 model case. *IEEE Access*, 7:170226–170237, 2019.

[J3] Florence Ho, Ruben Geraldes, Artur Goncalves, Marc Cavazza, and Helmut Prendinger. Improved conflict detection and resolution for service UAVs in shared airspace. *IEEE Transactions on Vehicular Technology*, 68(2):1231–1242, 2019.

Conference Papers

[C1] Florence Ho, Ana Salta, Ruben Geraldes, Artur Goncalves, Marc Cavazza, and Helmut Prendinger. Multi-agent path finding for UAV traffic management. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2019)*, pages 131–139, 2019.

[C2] Florence Ho, Ruben Geraldes, Artur Goncalves, Marc Cavazza, and Helmut Prendinger. Simulating shared airspace for service UAVs with conflict resolution. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018)*, pages 2192–2194, 2018.