Multi-Agent Path Finding for UAV Traffic Management

Robotics Track

Florence Ho
National Institute of Informatics
Tokyo, Japan
florence@nii.ac.jp

Ana Salta
INESC-ID
Lisbon, Portugal
anasalta@ist.utl.pt

Ruben Geraldes
National Institute of Informatics
Tokyo, Japan
rubengeraldes@gmail.com

Artur Goncalves
National Institute of Informatics
Tokyo, Japan
artur.alves.goncalves@gmail.com

Marc Cavazza
University of Greenwich
London, UK
m.cavazza@greenwich.ac.uk

Helmut Prendinger
National Institute of Informatics
Tokyo, Japan
helmut@nii.ac.jp

ABSTRACT

Unmanned aerial vehicles (UAVs) are expected to provide a wide range of services, whereby UAV fleets will be managed by several independent service providers in shared low-altitude airspace. One important element, or redundancy, for safe and efficient UAV operation is pre-flight Conflict Detection and Resolution (CDR) methods that generate conflict-free paths for UAVs before the actual flight. 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 paper, we extend CBS and ECBS 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 CBS and 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 and solution cost 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.

KEYWORDS

Unmanned Aircraft System Traffic Management; Pre-Flight Conflict Detection and Resolution; Multi-Agent Path Finding; Heterogeneous Agents

1 INTRODUCTION

In recent years, there has been an increasing focus on the use and deployment of UAVs in low-altitude airspace [4, 23]. Several independent operators will task multiple UAVs with limited capacities to visit specific locations. This situation results in an airspace populated by UAVs where no "conflict", i.e., possibility of collision, between UAVs is ever acceptable. Thus, there is a need to develop a Unmanned Aircraft Systems Traffic Management (UTM) system [15, 16] to ensure the safety of UAV operations. In the context of UTM, as in ATM (Air Traffic Management), 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" [17].

The main challenge in the UTM context is to design efficient Conflict Detection and Resolution (CDR) approaches [14]. Similar to the ATM concept [14, 17], 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 [1, 11, 13, 29, 36], pre-flight CDR methods remain mostly unexplored for this domain.

Thus, in this paper, 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 [20]. 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:

(1) UAV operations have different start times, hence newly computed flight paths must take into account the existing (already approved) flight paths;
(2) UAVs fly from their respective hubs to specific locations and then have to return to these hubs, which introduces a precedence relationship;
(3) UAVs are heterogeneous agents in terms of different sizes and speeds.

In this paper, 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) [25] or bounded suboptimal such as Enhanced CBS (ECBS) [3]. Differently, incremental techniques that plan agents sequentially in a predefined order, have also been proposed, such as Cooperative A* (CA*) [27] 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. We will compare CA* and CBS/ECBS approaches in terms of computation time and solution cost.

This paper makes 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.

The rest of the paper is structured as follows. Section 2 presents related works. Section 3 formulates our extension of the MAPF model. Section 4 describes the extensions made to these algorithms with spatio-temporal pruning, geometrical conflict detection and batch processing. Section 5 explains our simulations and its experimental results. Section 6 concludes the paper.

2 RELATED WORKS

Cooperative Multi-Agent Path Finding (MAPF) is known as NP-hard to solve optimally for the sum of individual costs (SIC) objective [37]. 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 [2] or taxation schemes [5] 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*) [10], Independence Detection and Operator Decomposition (OD-ID) [28], Increasing Cost Tree Search (ICTS) [26], M*[30] and Conflict Based Search (CBS) [6, 9, 25]. Other optimal approaches use Integer Linear Programming [37]. 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) [3] 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) [32] and ECBS+HWY [7].

We can also distinguish suboptimal search-based solvers that work in a prioritized way, but are incomplete and unbounded such as CA* [27], 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 [8, 18, 33]. However, all these approaches originally target "one-shot" scenarios, thus solved before all agents start to move. In this paper, 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 [31] where the ICTS algorithm is extended to the non-unit cost domain. [34, 35] 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, [21] 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.

3 PROBLEM FORMULATION

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 paper, 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 paper does not include the allocation of tasks to the agents, unlike in [12, 19]. The role of the UTM System is to provide conflict-free paths for all submitted flight requests from all UAV operators.

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 [24].
- 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 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 can either be the total distance or 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 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.

4 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.

4.1 Background

We hereby briefly describe CBS [25] and its bounded suboptimal variant ECBS [3]. CBS is a two-level search algorithm, and is complete and optimal with respect to the sum of individual costs.

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 the path costs of all agents. The root node of the CT contains an empty set of constraints. 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. Otherwise, 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. In the high and low level of ECBS, Focal search [22] is used, it is a suboptimal variant of A* where a FOCAL list

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 1: Extending the MAPF framework.

is maintained alongside the OPEN list with a second inadmissible heuristic that estimates the number of conflicts. 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 FOCAL are within a constant factor w of the best cost in OPEN.

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

4.2 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 1. Then conflict detection (described below) is performed only within the reduced set of identified operations and common paths segments.

Algorithm 1: 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: $\forall O_i \in O, O_i^{Conflict} \subseteq O$
for $O_i \in O$ **do**
  for $O_j \in O$ **do**
    /* Determine if temporal overlap between O_i and O_j */
    if $T_i \cap T_j \neq \emptyset$ **then**
      /* Determine if spatial overlap during common time interval */
      if $ShortestDistance(O_i, O_j, T_i \cap T_j) \leq r_i + r_j$; **then**
        Determine if the potential conflict is on the outbound and/or return path of each operation;
        Add O_i in $O_j^{Conflict}$ and O_j in $O_i^{Conflict}$;

4.3 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 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 straighforward 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.

4.3.1 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 in Fig. 1. 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. 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

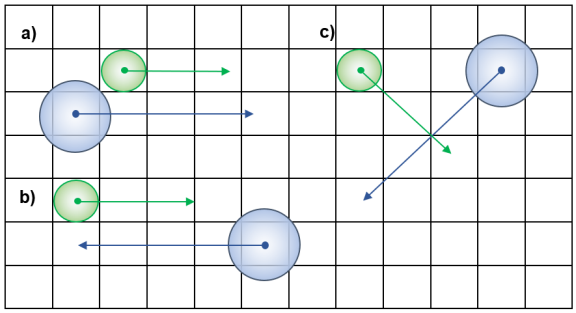

Figure 1: Types of conflicts between two agents of different sizes and speeds. a) Pursuit conflict, where the agents move closely towards the same direction ; b) Head-on conflict, where the agents move into opposite directions ; c) Intersection conflict, where the agents cross paths.

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:

$$\begin{aligned} \|p_i(t_C) - p_j(t_C)\|^2 &= (r_i + r_j)^2 \\ i.e. \quad (p_i(t_a) + v_i \cdot t_C - (p_j(t_a) + v_j \cdot t_C))^2 &= (r_i + r_j)^2 \end{aligned} \tag{1}$$

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 of the separation distance between two agents occurs,

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

.

4.3.2 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 $\mathcal{V}(a_i, I_C)$ the space occupied by an agent during the conflict interval. With our geometrical considerations, it is a "capsule" volume shown in Fig. 2 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 $(a_i, \mathcal{V}(a_j, I_C), I_C)$ and $(a_j, \mathcal{V}(a_i, I_C), I_C)$ 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 2, 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. 1. If a conflict is detected, then the move is not considered in the generation of the new path

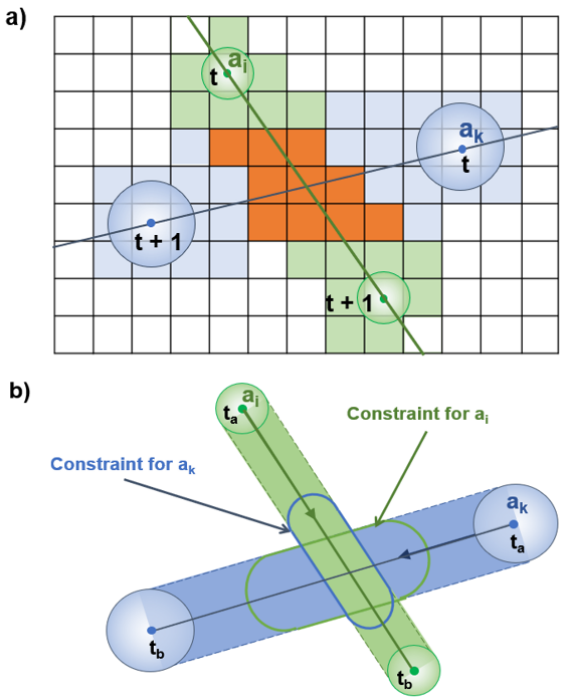

Figure 2: a) Conflict interval for the voxels intersection method (in red); b) Conflict interval for the computational geometry method (indicated as constraints).

for the agent. When planning the conflict-free paths for all operations, we also 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.

4.4 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

Algorithm 2: Reformulated Low Level Search in Extended CBS: Node expansion

Data: Node *current* to expand for agent i with $Constraints_i$
for *node* $n \in$ *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* $c \in Constraints_i$ **do**
      if $I_c \cap [t_{current}; t_n] \neq \emptyset$ **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 $Neighbors$;

are considered when replanning for conflicts within the same batch. For (2), as described in 4.1, (E)CBS considers conflicts between agents of the same batch and generates two child nodes for each agent of the conflict.

5 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.

5.1 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. 3 shows the runtimes for ECBS over the different number of operations

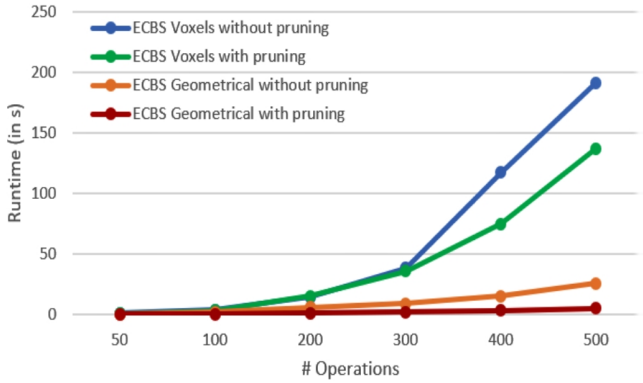

Figure 3: Average runtime for ECBS on $100 \times 100 \times 4$ grid map with 5% obstacle density.

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.

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

5.2.1 Experimental Setup. 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 × 17.10 km area. So we use a grid map of dimensions 478 × 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

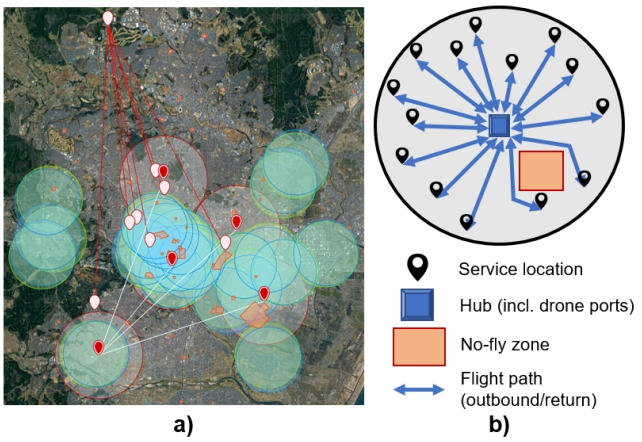

Figure 4: a) Overview of the area used in our simulations and of the considered vicinity areas for all hubs. The white and red icons indicate the hub locations for Hub-to-Hub operations. b) Example of path topology of UAV delivery use case for one hub.

- 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. 4 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 [20], 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. 4.

5.2.2 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.

5.2.3 Analysis. In our experiments, we study runtime and solution costs of FCFS (hereby labeled as CA*) and batch processing (hereby labeled as CBS, ECBS). Note that the computational geometry and spatio-temporal pruning are hereby used in all approaches compared. Table 2 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. Accordingly, we represent the ongoing UAV traffic, i.e., UAVs in the air at any time, as "# Operations already accepted" in Table 2. Then, "# Operations newly submitted" refers to the operations that are not yet integrated to the airspace. We propose to compare the performances of CBS/ECBS with "batch" processing and CA*-based processing for FCFS.

Runtime. CBS with batch processing is the method with the longest runtime overall, which is expected for an optimal algorithm. 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 be processed. If requests drop in with low frequency, CA*-based is clearly more suitable, as ECBS would have to "wait" until the batch is filled.

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,

# Ops already accepted	# Ops newly submitted	Runtime (in s)			Total deviation (in m) from initial solution per operation			Rejection rate (in % of # Ops newly submitted)		
		CBS	ECBS	CA*	CBS	ECBS	CA*	CBS	ECBS	CA*
200	100	6	2	2	3	8	9	0	0	0
	300	47	8	11	4	11	19	0	0	0
	500	154	15	26	9	24	30	0	0	0.2
400	100	10	4	4	12	21	25	0	0	0
	300	278	17	37	12	25	33	0	0	0.2
	500	-	31	59	-	27	39	-	0.2	0.6
700	100	22	4	6	15	21	27	0.2	0.2	0.5
	300	-	27	48	-	36	45	-	0.5	1.5
	500	-	104	131	-	42	51	-	0.5	2

Table 2: Comparisons between CBS, ECBS with batch processing and CA*-based FCFS processing.

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.

6 CONCLUSIONS

We 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.

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.

CBS and ECBS are methods for cooperative MAPF. As a pre-flight CDR method, those methods globally optimize UAV traffic. In our study (see Table 2), we observe that most UAVs do not deviate from the initial flight path, whereby a small number of UAVs deviate by several hundreds of meters. This might be seen as undesirable. Therefore, in our future work, we want to study agent centered objectives, rather than a global objective, such as auction-based approaches [2]. We are also considering fixed-wing UAVs with precise kinematics to better represent the future airspace that might include passenger drones.

We hope that our work can provide a first realistic assessment into the research challenges of future UTM-based services and their management via a UTM system.

7 ACKNOWLEDGEMENTS

This paper is based on results that were obtained as part from the research conducted by the National Institute of Informatics (NII) in a project recommissioned by the Japan Aerospace Exploration Agency (JAXA), which is commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

REFERENCES

[1] 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.

[2] 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.

[3] M. Barer, G. Sharon, R. Stern, and A. Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *21st European Conference on Artificial Intelligence (ECAI)*, pages 961–962, 2014.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Y. I. Jenie, E.-J. van Kampen, 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, 2015.

[14] 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.

[15] P. Kopardekar, K. Bilimoria, and B. Sridhar. Initial concepts for dynamic airspace configuration. In *Proceedings of AIAA Aviation Technology, Integration, and Operations Conference*, 2007.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] J. Pearl and J. Kim. Studies in semi-admissible heuristics. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 392–399, 1982.

[23] 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 *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 533–541, 2015.

[24] 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.

[25] 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.

[26] 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.

[27] D. Silver. Cooperative pathfinding. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 117–122, 2005.

[28] T. S. Standley. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] K. Yakovlev and A. Andreychuk. Resolving spatial-time conflicts in a set of any-angle or angle-constrained grid paths. In *CoRR abs*, 2016.

[35] 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.

[36] 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.

[37] 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.