Minimum-Violation Traffic Management for Urban Air Mobility

Suda Bharadwaj[1], Tichakorn Wongpiromsarn[2], Natasha Neogi[3], Joseph Muffoletto[1], and Ufuk Topcu[1]

[1] The University of Texas at Austin, Austin TX 78712, USA
{suda.b,stevencarr,utopcu}@utexas.edu
[2] Iowa State University, Ames, IA 50011
nok@iastate.edu
[3] NASA-Langley Research Center, Hampton, VA
natasha.a.neogi@nasa.gov

Abstract. Urban air mobility (UAM) refers to air transportation services in and over an urban area and has the potential to revolutionize mobility solutions. However, due to the projected scale of operations, current air traffic management (ATM) techniques are not viable. Increasingly autonomous systems are a pathway to accelerate the realization of UAM operations, but must be fielded safely and efficiently. The heavily regulated, safety critical nature of aviation may lead to multiple, competing safety constraints that can be traded off based on the operational context. In this paper, we design a framework which allows for the scalable planning of a UAM ATM system. We formalize safety oriented constraints derived from FAA regulations by encoding them as temporal logic formulae. We then propose a method for UAM ATM that is both scalable and minimally violates the temporal logic constraints. Numerical results show that the runtime for our proposed algorithm is suitable for very large problems and is backed by theoretical guarantees of correctness with respect to given temporal logic constraints.

1 Introduction

1.1 Problem significance

Recent years have seen increased urbanization, economic expansion, underinvestment in infrastructure, and the rise of ride hailing services and e-commerce. These changes have led to an increase in transportation delays, vehicle congestion during peak times, and environmental impacts resulting in escalating mobility challenges in urban areas. The emerging Urban Air Mobility (UAM) aviation market is being catalyzed by advances in increasingly autonomous systems, electric propulsion, and novel business models such as on-demand, aerial ride sharing, thereby helping to address congestion issues in urban areas [5].

UAM has the potential to be a safe, functional solution to the air transportation problem for passengers and cargo in and around a densely populated urban area. An air traffic management system that governs a large number of these novel UAM operations over a small geographical area in a safe and efficient fashion is key to the realization and deployment of the UAM vision.

1.2 Scalable and Verifiable Safety for UAM

The scale and density of projected UAM operations will far exceed the safe workload capacity of human controllers, necessitating the deployment of increasingly autonomous solutions for functions like aircraft management (e.g., managing flightpath and altitude requests, managing airborne and ground based holding times, etc.) and aircraft separation.

Currently, there is no established infrastructure for air traffic management of a scalable UAM concept of operations. Under current aviation paradigms, air traffic management is carried out in a centralized fashion by air traffic controllers. The U.S. National Airspace System (NAS) is comprised of 5.3 million square miles of domestic airspace and 24 million square miles of oceanic airspace. There are approximately 5,000 flights airborne at any given moment. Over 14,000 air traffic controllers manage these aircraft and perform multiple safety-critical functions, such as air traffic separation which guarantees that a minimum spacing between aircraft is maintained [7]. In contrast, for UAM operations to deploy at scale for profit, it will be necessary to have hundreds (or even thousands) of UAM aircraft aloft over an urban airspace under 500 square miles [9]. The sheer number of vehicles, along with the necessary reduced separation criteria between them in order to achieve the required densities, will require the development of increasingly autonomous capabilities for aircraft clearance, separation, and flow management in the UAM ecosystem.

Deploying increasingly autonomous systems in the US airspace is a challenge. Commercial aviation is among one of the most safety-critical systems in the world and has stringent standards for the design, deployment, and operation of aircraft and air traffic control systems. These regulations are detailed in chapter 14 of the Code of Federal Regulations (14 CFR). The ability to assure increasingly autonomous systems to aviation grade standards is thus crucial for their acceptance. Safety-critical functions such as aircraft separation must provide strict guarantees on their behavior and the correctness of their outcomes. Thus, increasingly autonomous air traffic management systems will have to tackle the dual issues of scalability and verifiable safety in order to be deployed in the NAS.

1.3 Setting

In this paper, we employ a hierarchical decomposition of the UAM operations space motivated by the physical and geographical infrastructure required to field the system. Such an architecture has been studied in [3, 2] and allows for scalable air traffic management. UAM vehicles take off and land from a landing pad, called a vertipad, which includes the final approach and takeoff (FATO) area. A vertiport is comprised of several vertipads, the respective vertipad FATOs, and charging and maintenance facilities. A vertihub is comprised of several vertiports. Vertihubs provide air traffic control services (i.e., real time control of aircraft movement) between vertiports under their control and air traffic management services (i.e., strategic and long term planning of aircraft movement and flows) for vehicles transiting between adjacent vertihubs. Figure 1 provides a visual representation of vertiports and vertihubs. We focus on a synthesis strategy for vertihubs, each of which is responsible for assuring the safety of all vehicles in its airspace.

Fig. 1. Vertihub and vertiport depiction

In general, it is not always feasible to guarantee the satisfaction of all safety constraints under all conditions. Such scenarios are explicitly accounted for in 14 CFR §107.21, which allows for emergency deviation from regulations if required as long as the deviation is reported. For example, a vehicle may have to make an emergency landing if it is about to run out of fuel, even if violates separation requirements. Thus, there is a need for an air traffic management approach that allows for *formally justifiable* violation of safety constraints if necessary. In this paper, we study the synthesis of control strategies for automated air traffic management for UAM while obeying safety regulations. In particular, we focus on the case where all safety regulations cannot be feasibly satisfied, as is allowed for small, cargo-carrying UAS under emergency operation. We present a decentralized, scalable synthesis approach that provably *minimally violates* the given safety requirements. Note that if it is possible to satisfy all safety properties for the duration of the flight, the approach yields a solution with zero violations (i.e., all safety requirements are guaranteed for the duration of the flight).

1.4 Contribution and Innovation

We present the first decentralized approach to minimum violation planning. We use temporal logic for a formal representation of safety regulations and guarantee that these regulations are minimally violated across the global system. Furthermore, we establish a *framework* in this paper for minimum-violation for the small UAS cargo-carrying application over urban areas. Our framework has the following properties:

- *Scalability* - UAM operations are envisioned to occur at a scale well beyond the capabilities of current air traffic management approaches, as current day approaches are typically labor-intensive. It is crucial to safely guarantee operations of increasingly autonomous vehicles at scale in order for UAM to be commercially viable.
- *Decentralization* - The environment is likely to encompass multiple service providers and stakeholders. Each stakeholder will have potentially competing priorities and requirements. Consequently, the full state of the entire system is unlikely to be controlled or even observed by a single entity.
- *Transparency* - With companies ranging from startups to corporations developing UAM vehicles, services, and capabilities, there is a disconnect between regulation and the pace of technological development. Bridging the gap between regulation and real-world implementation practices is necessary for a viable path to deployment.
- *Flexibility* - The technological and regulatory landscape of UAM is rapidly changing. Any proposed framework that cannot efficiently incorporate a change in regulation or emerging capabilities is not a viable solution.
- *Auditability* - All violations of regulations must be formally accounted for and reported. Our proposed method implicitly allows for such an analysis, as it not only synthesizes a control strategy for air traffic management but also the associated violation.

1.5 Path to deployment

Assessing the safety of an increasingly autonomous system relies on being able to bound the behavior and interactions of the components of the system as well as its interfaces with its operational environment. Performance-based regulation is employed in aviation to specify explicit properties that must be evinced by a component or element of the system (and/or operational environment) in order for the system safety claims to be met. For example, 14 CFR §107.49 (d) states that: “If the small unmanned aircraft is powered, ensure that there is enough available power for the small unmanned aircraft system to operate for the intended operational time”. This fuel requirement forms a temporal logic constraint on the vehicle during its flight. The controller synthesis method for the vertihubs must adhere to this constraint, in order to demonstrate compliance to 14 CFR §107.49. The minimum violation guarantees provided by the presented synthesis process help to demonstrate that this regulation will be satisfied as much as possible throughout the flight process. Thus, the guarantees provided by the synthesis method presented in this paper may serve as a partial means of compliance to the regulation—supplemented with the generation of test and design analysis artifacts as well as operational procedures.

The presented synthesis framework provides a path to deployment of these increasingly autonomous systems in safety-critical contexts. Air traffic management services, such as aircraft separation, may then be offered in a UAS Traffic

Management (UTM) inspired framework, which interfaces with today's traditional Air Traffic Management framework [17]. In this framework, authority may be delegated by the FAA to provide select air traffic management services such as low-altitude weather information, congestion management, terrain avoidance, route planning, re-rerouting, separation management, and contingency management [13, 15]. One of the main attributes of the UTM system is that it does not require human operators to monitor every vehicle continuously, as in the traditional ATM system, thereby enabling increasingly autonomous realizations of specified air traffic management functions. We believe that integration of the approach in this paper into a UTM-like construct provides a path to deployment for UAM operations, as the provided safety guarantees will greatly enhance the assurance case for higher-risk operations currently not supported by UTM (e.g., operations in dense urban environments with UAS exceeding 55 lbs).

1.6 Related work

Some preliminary work is being done in cooperative ATM for next generation air traffic management [16], but this work considers a scheduled approach for large passenger aircraft and cannot handle management for on-demand flights. Similarly, there is work done on distributed control for ATM of small unmanned aerial systems (UAS) [8], but this work relies on cloud based architectures that do not currently satisfy strict aviation safety requirements. Hybrid control approaches have been applied [19], however scalability proves to be an issue. To the best of our knowledge, this is the first approach implementing minimally violating controller synthesis for large-scale UAM ATM operations. Formally verified tools such as DAIDALUS [14] provide safety guarantees at lower levels of operations, however, it does not handle the fleet-level operations. Another approach called *runtime enforcement* [6, 18] aims to guarantee a specified property by detecting and altering the behavior of the system at runtime. An existing approach called shielding [4, 12] uses reactive synthesis and assumes that the shield has full knowledge and control of the whole system — in this case the entire UAM system and the vehicles it handles. A technique for synthesizing quantitative shields for multi-agent systems in a fully centralized manner was presented in [1]. However, all these approaches are only applicable if a feasible solution exists. If it is not possible to satisfy all safety requirements, no solution is possible. In contrast, in the approach presented in this work, if no feasible solution exists, we can still synthesize a controller that minimally violates safety.

Our approach is based on minimum-violation planning [20, 21] for systems that are subject to potentially infeasible safety requirements. Given a prioritized safety specification, which specifies the priority and weight of each safety requirement, minimum-violation planning computes a plan that minimally violates the requirements. In particular, we propose a novel *decentralized approach* to minimum-violation planning in order to handle large-scale systems that cannot be handled by current state-of-the-art approaches.

2 Notation and Setting

In this section, we present the relevant technical notation for the minimum violation synthesis problem in the UAM setting.

2.1 Operating environment

Recall that the environment is divided into smaller regions called vertihubs. Each vertihub is governed by a *vertihub controller* that has radio line of sight to all vehicles in the region denoted as $\mathcal{T}_i$ as shown in Figure 2. A vertihub controller is responsible for managing *requests* by UAM vehicles (henceforth referred to as vehicles) to either *land at* or *take off from* a desired vertiport in its region or *pass through* to a neighboring region.

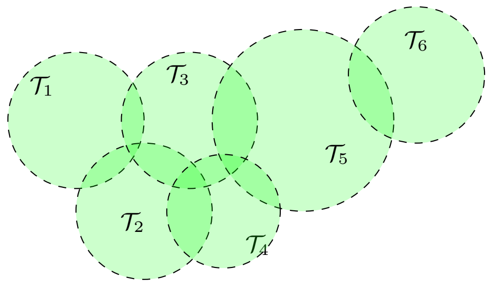

Fig. 2. Example UAM operating environment. Green circles correspond to the regions of vertihub controllers.

Requests We define a request as a tuple $O = (r, T, c)$ where

- $r \in \mathcal{R}$ is the request class from a predefined set of classes $\mathcal{R}$. Request classes include *landing at a particular vertiport in the region*, *pass-through region*, *take-off from vertiport in the region*. Depending on the nature of the problem, $\mathcal{R}$ can be defined to capture all types of desired outcomes for vehicles.
- $T \in \mathbb{N}$ is the amount of time left before the request *must* be granted. t can function as an analogue for fuel reserves as vehicles requesting to land cannot hover indefinitely.
- $c \in C$ is the class of UAM vehicle from a predefined set of vehicles C.

At any given timestep t the vertihub is managing requests from multiple vehicles. We define the initial set of requests as the *request allocation* and denote it as $\mathcal{O}^{\text{init}}$. We model the *vertihub controller* that is responsible for managing these requests as a *labeled weighted finite transition system*.

Example 1. Tower $\mathcal{T}_1$ is given a set of N requests to handle $\mathcal{O}^{\text{init}} = \{O_1^{\mathcal{T}_1} \dots O_N^{\mathcal{T}_1}\}$. For example, $O_1^{\mathcal{T}_1} = (port\,A, 5, passenger)$ corresponds to a request by a *passenger vehicle* trying to land at port A in *at most* 5 time steps.

Transition system model for vertihub controller A vertihub controller is modeled as a tuple referred to as a labeled weighted finite transition system $\mathcal{T}_i = (S_i, s_{\text{init}_i}, \varDelta_i, \text{AP}_i, L_i)$ where

- S_i is a finite state space. It is the set of all currently unapproved requests. Formally, if there are currently m unapproved requests, we write $S_i = \{O_{i,1} \dots O_{i,m}\}$. We note that the state space can also include additional features of interest such as number of vehicles in the airspace, their landing/take-off statuses, and others. However, for notational simplicity we do not include these in the definition presented in this paper. In practice, these features, alongside the relevant modifications to the transition function, are straightforward to include.
- $s_{\text{init}_i} \in S_i$ is the initial set of requests $\mathcal{O}_i^{\text{init}}$.
- $\varDelta_i \subseteq S_i \times S_i$ is the deterministic transition function that governs how requests evolve. We have $\left(S_i^t, S_i^{t+1}\right) \in \varDelta_i$ if:
  - all *approved* requests at timestep t, denoted $\mathcal{O}_{i,a}^t \subseteq S_i^t$, are not present in S_i^{t+1}, i.e., if $O_{i,j} \in \mathcal{O}_{i,a}^t$ then $O_{i,j} \notin S_i^{t+1}$, and
  - all *unapproved* requests at timestep t, denoted $S_i^t \setminus \mathcal{O}_{i,a}^t$, have their time remaining decremented, i.e., for all $O_{i,j} = (r, T, c)$ and $O_{i,j} \in S_i^t \setminus \mathcal{O}_{i,a}^t$, we will have $O_{i,j} \in S_i^{t+1}$ and $O_{i,j} = (r, \max(T-1, 0), c)$. Informally, if request is not approved $\varDelta_i$ decrements the remaining timer on the request by 1.
- AP is a set of atomic propositions.
- $L : S \to 2^{\text{AP}}$ is the labeling function.

At every timestep, the decision problem for the controller is to choose a set $O_{i,a}^t \subseteq S_i^t$ of requests to approve. The controller will also have the option to reroute the request to neighboring controllers. We discuss this process formally in Section 3.

Example 2. Consider vertihub controller $\mathcal{T}_1$ with two pending requests $O_1^{\mathcal{T}_1} = (\mathit{port}\,A, 5, \mathit{passenger})$ and $O_2^{\mathcal{T}_1} = (\mathit{port}\,A, 3, \mathit{passenger})$. In this case, at time step t we have $S_1^t = \{O_{1,1}, O_{1,2}\}$ where $O_{1,1} = O_1^{\mathcal{T}_1}$ and $O_{1,2} = O_2^{\mathcal{T}_1}$. Let us assume at time step t that the controller approves request $O_{1,2}$. We denote this as $\mathcal{O}_{1,a}^t = \{O_{1,2}\}$ and we will have $S_1^{t+1} = \{O_{1,2}\}$ where $O_{1,1} = (\mathit{port}\,A, 4, \mathit{passenger})$.

A finite trace of $\mathcal{T}_i$ is a finite sequence of states $\tau_i = s_{i,0} s_{i,1} \dots s_{i,n}$ such that $s_{i,0} = s_{init_i}$ and $(s_{i,j}, s_{i,j+1}) \in \varDelta_i$, for all $j \in \mathbb{N}_{\leq n-1}$. A finite trace $\tau = s_0 s_1 \dots s_n$ produces a finite word $w(\tau) = L(s_0) L(s_1) \dots L(s_n)$. For any $s \in S$, we let $\text{Traces}(\mathcal{T}, s)$ represents the set of all finite traces of $\mathcal{T}$ that ends with s.

The aim of the transition system $\mathcal{T}_i$ is to reach a *goal state* denoted $s_{\text{final}} \in S_i$. Since the hub controller needs to eventually grant all requests, s_{final} corresponds to the state with no more pending requests. Formally, we have $s_{\text{final}} = \emptyset$.

The goal of this approach is to synthesize a trace for each vertihub such that requests are accepted in a manner that satisfies all regulations. However, if this is not possible, it must approve requests in a way that minimally violates regulations. We employ linear temporal logic due to its ability to formally express a wide array or requirements.

We employ finite linear temporal logic (FLTL) to precisely describe the safety-oriented regulations. We note this FLTL has been used in the context of autonomous driving to formally represent road safety laws for planning [20, 21].

Finite linear temporal logic An FLTL formula is built up from (a) a set of atomic propositions; (b) the logic connectives: negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$), and material implication ($\implies$); and (c) the temporal operators: next ($\bigcirc$), always ($\Box$), eventually ($\Diamond$), and until ($\mathcal{U}$). We refer the reader to [11] for full FLTL semantics.

An FLTL formula ψ over a set AP of atomic propositions is interpreted over a finite word $w = l_0 l_1 \dots l_n \in (2^{\mathrm{AP}})^{n+1}$, and we write $w \models \psi$ if w satisfies ψ. In particular, consider $p \in \mathrm{AP}$. Then, $w \models p$ if and only if $p \in l_0$. Also, $w \models \Box p$ if and only if $p \in l_i$ for all $i \in \{0, \dots, n\}$.

Example 3. 14 CFR §107.49 (d) requires the vehicle to have enough power for its operations and hence, a vertihub cannot force a vehicle to loiter for too long. We can capture this as a specification $\psi = \Box\{\neg fuel\_too\_low_i\}$ where $fuel\_too\_low_i$ is an atomic proposition that is true when request $O_i = (r, T, c)$ has $T = 0$.

Prioritized safety specification A *prioritized safety specification* is a tuple $\mathcal{P} = (\mathrm{AP}, \Omega, \Psi, \varpi)$ where AP is a set of atomic propositions, Ω is a set of FLTL formulas over AP, $\Psi = (\Psi_1, \Psi_2, \dots, \Psi_N)$, $\Psi_i \subseteq \Omega$ for all $i \in \{1, \dots, N\}$, and $\varpi : \Omega \to \mathbb{N}$ is the priority function that assigns the weight to each $\psi \in \Omega$.

Example 4. A potential prioritized safety specification $\Psi = \{\Psi_1, \Psi_2\}$ for a vertihub to satisfy is $\Psi_1 = \{\psi_{1,1}\}, \Psi_2 = \{\psi_{2,1}, \psi_{2,2}, \psi_{2,3}\}$ where

- $\psi_{1,1}$ = Never allow the timer on a request to expire.
- $\psi_{2,1}$ = Do not land vehicles past a vertiport's capacity.
- $\psi_{2,2}$ = Do not allow more than M vehicles in the vertihub's airspace at a time.
- $\psi_{2,3}$ = Do not land a vehicle at a vertiport it did not request.

Note that the requirements at level i are strictly more important than those at level i+1, i.e., the system first attempts to minimize the amount of violation of level-1 requirements. Then among all the policies that minimize the violation of level-1 requirements, it attempts to minimize the amount of violation of level-2 requirements, and so on. In Example 4, the first specification is the highest priority and the remaining specifications are all of equal priority. We use this example in the case study detailed in the Experimental Results section.

Lack of safety Consider an FLTL formula ψ over AP and a finite word $w = l_0 l_1 \dots l_n \in (2^{\mathrm{AP}})^{n+1}$. The *lack of safety* of w with respect to ψ is defined as

$$\lambda(w, \psi) = \min_{I \subseteq \mathbb{N}_{\leq n} | \mathsf{vanish}(w, I) \models \psi} |I|, \tag{1}$$

where for any given finite sequence $w = l_0 l_1 \dots l_n$ and a set $I \subseteq \mathbb{N}$, $\mathsf{vanish}(w, I)$ is defined as a subsequence of w obtained by removing all l_i, $i \in I$.

Let $\mathcal{P} = (\mathrm{AP}, \Omega, \Psi, \varpi)$ be a prioritized safety specification where $\Psi = (\Psi_1, \Psi_2, \dots, \Psi_N)$. We define the *lack of safety* of w with respect to $\mathcal{P}$ as

$$\lambda(w, \mathcal{P}) = (\lambda(w, \Psi_1), \dots, \lambda(w, \Psi_N)) \in \mathbb{N}^N, \tag{2}$$

where for each $i \in \{1, \dots, N\}$,

$$\lambda(w, \Psi_i) = \sum_{\psi \in \Psi_i} \varpi(\psi) \lambda(w, \psi). \tag{3}$$

Note that there are two mechanisms to address the unequal importance of safety specifications. Namely, the prioritization of specifications by Ψ and the weighting function $\varpi(\psi)$. The weights ϖ indicates the importance among different requirements within the same level.

The lack of safety of a trace τ of a finite transition system with respect to $\mathcal{P}$ is defined based on its produced word, i.e., $\lambda(\tau, \mathcal{P}) = \lambda(w(\tau), \mathcal{P})$. The standard lexicographical order is used to compare the lack of safety between different traces.

Remark 1. There are two mechanisms to specify the unequal importance of different specifications, the hierarchy $(\Psi_1, \Psi_2, \dots, \Psi_N)$ and the weights captured by ϖ. As the standard lexicographical order is used to compare the lack of safety between different traces, the algorithm first minimizes the lack of safety with respect to the specifications in Ψ_1. Then, among all the traces that minimizes the lack of safety with respect to Ψ_1, it minimizes the lack of safety with respect to the specifications in Ψ_2, and so on. The weights ϖ only matter for the specifications within the same level of hierarchy.

3 Problem Formulation

Global system We define the global system as the composition of the vertihub controllers. Assume we have N vertihub controllers $\mathcal{T}_1 \dots \mathcal{T}_N$. We define a connectivity graph $G_{\mathcal{T}}$ as a directed graph with each vertex corresponding to a controller. We say two controllers are *connected* if they share an edge in the graph. Let *connect*$(\mathcal{T}_i)$ be the set of hub controllers $\mathcal{T}_j$ where $i \neq j$, that share an edge with $\mathcal{T}_i$. For example, in Figure 2, overlapping hub regions share an edge in the corresponding directed graph in Figure 3, and therefore the corresponding controllers are connected.

Formally, we define the global system as a tuple $(\mathcal{O}^{\mathrm{init}}, \Phi, \mathcal{T})$ where $\mathcal{O}^{\mathrm{init}} = \{O_1, \dots, O_M\}$ is the global set of requests across all vertihubs, $\Phi : \{\mathcal{T}_1, ..., \mathcal{T}_N\} \to$

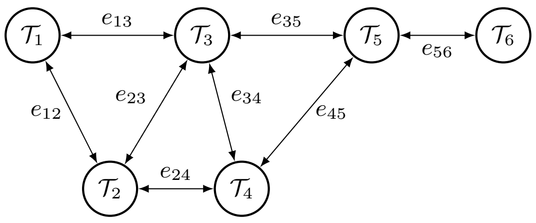

Fig. 3. The connectivity graph $G_{\mathcal{T}}$ of the UAM hub controllers $\mathcal{T}$ depicted in Figure 2. Each edge e_{ij} corresponds to $\mathcal{T}_i$ and $\mathcal{T}_j$ being connected, i.e., the outputs of $\mathcal{T}_i$ are inputs to $\mathcal{T}_j$ and vice versa.

$2^{|\mathcal{O}^{\text{init}}|}$ is the *request allocation function* such that $\Phi(\mathcal{T}_i)$ is the request set allocated to hub $\mathcal{T}_i$, and $\mathcal{T} = (S, s_{\text{init}}, \Delta, \text{AP}, L)$ is a networked composition of N hub controllers $\mathcal{T}_1, \ldots, \mathcal{T}_N$ such that:

- $S = (S_1, S_2, \ldots, S_N)$ where S_i is the set of all currently unapproved requests assigned to hub $\mathcal{T}_i$. Note, however, that due to possible reallocation of requests, it is not necessary that S_i only contains the requests in s_{init_i}. Instead, it contains requests in $\bigcup_i s_{\text{init}_i}$ such that each request is assigned to at most one hub, i.e., $S_i \cap S_j = \emptyset$ for all i, j.
- $s_{\text{init}} = (s_{\text{init}_1}, s_{\text{init}_2}, \ldots, s_{\text{init}_N})$
- $\Delta \subseteq S \times S$ such that $\left(S^t, S^{t+1}\right) \in \Delta$ if for each unapproved request $O_i = (r, T, c) \in S_i^t$ at each hub $\mathcal{T}_i$,
  - it remains unapproved with its time remaining decremented and either assigned to the same hub, i.e., $O_i = (r, T-1, c) \in S_i^{t+1}$, or a connected hub, i.e., $O_i = (r, T-1, c) \in S_j^{t+1}$ for some $\mathcal{T}_j \in \mathit{connect}(\mathcal{T}_i)$, or
  - it is approved by hub $\mathcal{T}_i$ or a connected hub $\mathcal{T}_j \in \mathit{connect}(\mathcal{T}_i)$, i.e., $O_i \in \mathcal{O}_{i,a}^t \cup \mathcal{O}_{j,a}^t$, in which case the request is not present in $\bigcup_k S_k^{t+1}$.
- $\text{AP} = \text{AP}_1 \cup \text{AP}_2 \cup \cdots \cup \text{AP}_N$
- $L : S \to 2^{\text{AP}}$ such that $L(s_1, \ldots, s_N) = \bigcup_i L_i(s_i)$.

Put simply, we construct the transition function Δ as the composition of *intra-hub* transitions and *inter-hub* transitions. Since vehicles can only move between neighbouring hubs, *inter-hub* transitions are limited to occurring only between those hubs that are connected in the graph $G_{\mathcal{T}}$. finish ex

Example 5. Consider request $S_1^0 = \{O_1^{\mathcal{T}_1} = (\mathit{port}\,A, 5, \mathit{passenger}), O_2^{\mathcal{T}_1} = (\mathit{port}\,A, 3, \mathit{passenger})\}$ corresponding to a request by a *passenger vehicle* trying to land at port A in *at most* 5 time steps.

Vertihub violation cost Each vertihub $\mathcal{T}_i$ is given a prioritized safety specification $\mathcal{P}_i$ and request allocation $\Phi(\mathcal{T}_i) = s_{\text{init}_i}$. Given a request allocation function Φ, the *violation cost* of vertihub $\mathcal{T}_i$ executing a finite trace $\tau_i \in \text{Traces}(\mathcal{T}_i, s_{\text{final}})$ is denoted $\lambda_{\Phi(\mathcal{T}_i)}(\tau_i, \mathcal{P}_i)$. We denote the *optimal* violation cost of $\mathcal{T}_i$ for a request allocation function Φ as $\lambda^*_{\Phi(\mathcal{T}_i)} = \min_{\tau_i \in \text{Traces}(\mathcal{T}_i, s_{\text{final}})} \lambda_{\Phi(\mathcal{T}_i)}(\tau_i, \mathcal{P}_i)$.

The cost of the global system $\mathcal{T}$ is dependent on the conjunction of the prioritized specifications $\mathcal{P} = \mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \cdots \wedge \mathcal{P}_N$, global requests $\mathcal{O}^{\text{init}}$, and request allocation function Φ. Formally, we define

$$\lambda_{\Phi} = \sum_{i=1}^{N} \lambda^*_{\Phi(\mathcal{T}_i)}. \tag{4}$$

3.1 Problem statement

Given a set of N hub controllers $\mathcal{T}_1 \dots \mathcal{T}_N$, a connectivity graph $G_{\mathcal{T}}$ and a prioritized safety specification for each controller $\mathcal{P}_1, \dots, \mathcal{P}_N$ with $\mathcal{P} = \mathcal{P}_1 \wedge \cdots \wedge \mathcal{P}_N$, construct a request allocation function Φ and corresponding trace $\tau^* = \{\tau_1, \dots, \tau_N\}$ where $\tau_i \in \text{Traces}(\mathcal{T}_i, s_{\text{final}})$ that minimizes the lack of safety for the entire system. Formally,

$$\tau^* = \underset{\tau \in \{\text{Traces}(\mathcal{T}, s_{\text{final}})\}}{\arg\min} \lambda(\tau, \mathcal{P}). \tag{5}$$

4 Solution Approach

4.1 Overview

Motivated by the cost structure in (4), we decompose the traffic management problem into two subproblems:

1. Compute an optimal request allocation Φ^* such that
$$\sum_{i=1}^{N} \lambda^*_{\Phi^*(\mathcal{T}_i)} \leq \sum_{i=1}^{N} \lambda^*_{\Phi(\mathcal{T}_i)},$$
for any request allocation Φ. Informally, the optimal request allocation Φ^* will have a lower or equal cost compared with any other allocation.
2. Given a request allocation Φ, compute an optimal trace τ_i^* for each vertiport $\mathcal{T}_i$ such that
$$\lambda_{\Phi(\mathcal{T}_i)}(\tau_i^*, \mathcal{P}_i) \leq \lambda_{\Phi(\mathcal{T}_i)}(\tau_i, \mathcal{P}_i),$$
for any trace $\tau_i \in \text{Traces}(\mathcal{T}_i, s_{\text{final}_i})$.

The second problem can be solved using minimum-violation planning as in [20, 21]. To solve the first problem, we need to find the globally optimal request allocation for all the vertihubs. This is a combinatorially hard problem. To solve the problem in a distributed manner, we propose an auction-based algorithm. In each round, each vertihub identifies potential requests to be reallocated and offers each of these requests to other connected vertihubs. The request with highest cost is then selected, and a connected vertihub accepts this request if it can accommodate the extra request with less cost than the original vertihub. Finally, the request will be reallocated to the vertihub that can accommodate the request with the lowest cost. This auction-based request allocation ensures that the overall cost decreases in each round and terminates when no more requests can be reallocated without extra cost.

Algorithm Given a request allocation Φ, we define the cost of vertihub $\mathcal{T}_i$ accommodating a request O as

$$C_i^{\Phi}(O) = \min_{\tau_i} \lambda_{\mathcal{O}_O^i}(\tau_i, \mathcal{P}_i) - \min_{\tau_i} \lambda_{\mathcal{O}_{\emptyset}^i}(\tau_i, \mathcal{P}_i), \tag{6}$$

where $\mathcal{O}_O^i = \Phi(\mathcal{T}_i) \cup \{O\}$ is the set of requests allocated to $\mathcal{T}_i$ together with the request O, and $\mathcal{O}_{\emptyset}^i = \Phi(\mathcal{T}_i) \setminus \{O\}$ is the set of requests allocated to $\mathcal{T}_i$ without the request O.

The algorithm initializes Φ based on the desired location associated with each request. Then, it updates Φ iteratively as follows.

(a) Initialize the set $\mathbb{O}$ of potential requests to be reallocated in this iteration as the empty set.
(b) Each vertihub $\mathcal{T}_i$ computes the cost $C_i^{\Phi}(O)$ for accommodating each request $O \in \Phi(\mathcal{T}_i)$. It then adds each request as well as its associated cost $(O, C_i^{\Phi}(O))$ to $\mathbb{O}$ for all requests O with $C_i^{\Phi}(O) > 0$.
(c) If $\mathbb{O}$ is empty, then the algorithm terminates and outputs Φ. Otherwise, we let O^* be the request with the highest cost in $\mathbb{O}$ and C^* be its associated cost.
(d) Each vertihub $\mathcal{T}_i$ computes the cost $C_i^{\Phi}(O^*)$ for accommodating O^*. Consider two possible cases.
   - $C_i^{\Phi}(O^*) \geq C^*$ for all $\mathcal{T}_i$, i.e., no other vertihub can better accommodate this request. Then, the request O^* is removed from $\mathbb{O}$ and the algorithm goes back to step (c) to attempt reallocating the next worst request.
   - $C_i^{\Phi}(O^*) < C^*$ for some $\mathcal{T}_i$. Then, Φ is updated so that the request O^* is allocated to $\mathcal{T}_{i^*}$ that minimizes the cost of accommodating O^*, i.e., $C_{i^*}^{\Phi}(O^*) \leq C_i^{\Phi}(O^*)$ for all $\mathcal{T}_i$ (see Example 5 for an illustrative example of request reallocation). This iteration finishes and the algorithm starts the new iteration with step (a). In the case where there are multiple vertihubs with equal lowest cost C^*, a tie breaker heuristic, e.g., based on tower id and priority, can be used.

As the number of requests is finite, the cost is non-negative and strictly decreases in every iteration except the last iteration. Thus, the algorithm is guaranteed to terminate and output a request allocation that is at least as good as the initial allocation. This is formally stated as follows.

Proposition 1. *Let Φ^{init} be the initial request allocation. Then, the algorithm terminates with request allocation Φ such that $\lambda_{\Phi} \leq \lambda_{\Phi^{init}}$.*

We remark that for ease of presentation the algorithm assumes the vertihub network is fully connected. In implementation, it is straightforward to modify the algorithm to directly incorporate the network constraints.

5 Experimental Results

In this section, we detail the results of a case study implementing the presented algorithm[4]. All experiments were run on an AMD Ryzen 5 3600x processor with 6 cores @ 4.3 Ghz and 16 GB RAM. For the purposes of this demonstration, we use the prioritized safety specifications given in Example 3. We use the toolbox TuLiP [22] to compute minimally violating traces. We randomly generate vehicle requests in a format compatible with the Mission Planner Algorithm [10] developed at NASA Langley. The data contains simulated, timestamped on-demand requests for origin-destination trips corresponding to vertiports in particular vertihubs. We then run our algorithm to minimally violate the regulations described in Example 3.

Scalability Figure 4 shows the runtime per iteration per vertihub for different numbers of total vertihubs as well as the number of iterations until the algorithm converges. It is clear that the average runtime scales efficiently as the number of

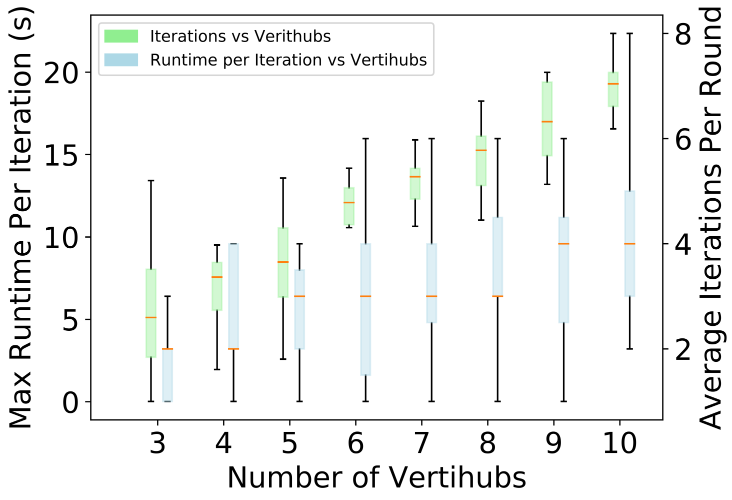

Fig. 4. Worst case runtime per iteration per vertihub (blue) and iterations until convergence (green).

vertihubs increases. In general, the total number of iterations before convergence also stays relatively constant albeit with a higher variance as the system size increases. However, since the overall violation of the system's safety decreases

[4] The code for the implementation can be found at https://github.com/JoeMuff999/Automata-Testing

with every iteration, in practice, we can always terminate the algorithm after a certain number of iterations and still have reduced the total violation cost. We note that even the smallest instance of the problem, i.e., 3 vertihubs with a maximum of 5 requests was unable to be solved in under 10 minutes using the centralized method in [20]. Furthermore, these results are a worst-case analysis as we assume the vertihubs are fully connected, i.e., all vertihubs can transfer requests to any other vertihub. In practice, the pool of vertihubs that can accept requests from other vertihubs will be smaller and this will limit the number of computations needed.

Violation cost To demonstrate the decreasing violation cost, we run our algorithm on the specific case of a 6 vertihub system with 10 requests. As shown in Figure 5, the initial request allocation has a cost of 11 for the highest priority regulation. After 4 iterations, the cost has decreased to 0 while the cost for the second priority regulation stays at 2.

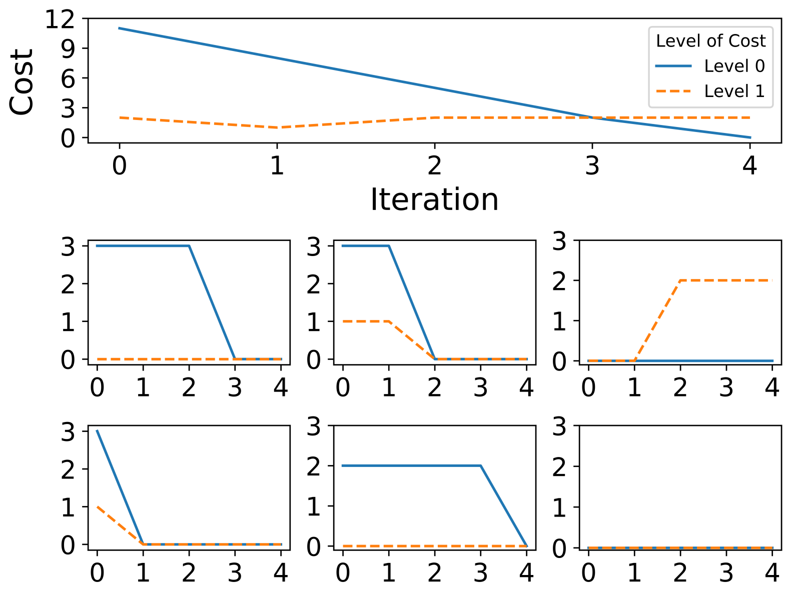

Fig. 5. Violation cost vs iteration for a 6 vertihub system. Total violation, i.e., sum of the violation of all the individual vertihubs, is shown on top with the individual vertihub violation costs shown below.

Batch processing The presented method in this paper relies on processing batches of requests at a time. However, in practice, requests arrive sequentially in real time. In most cases however, the requests are known in advance and hence can be planned for. In this result, we demonstrate the effect of different batch sizes on overall violation cost. The data used in the simulation was generated by NASA Langley in conjunction with partners performing UAM demand studies. We divide incoming requests into batches of 5,10,and 15 requests at a time. Each batch is then processed before moving on to the next batch. We then sum the total violation cost across all batches for the entire data set. We note that we only report the level 1 violation cost as the level 0 violation cost is 0 in all cases.

Batch Size	Violation Cost	Computation time (s)
5	71	6.6
10	63	32.5
15	56	161.9

Table 1. Level 1 violation cost and corresponding average synthesis time per batch per tower for different batch sizes.

As seen in Table 1, violation cost reduces as the batch size increases. This result is expected as the smaller batch sizes typically result in myopic plans that can cause violations down the road. However, it is not necessarily feasible to plan with large batch sizes as it requires a large look-ahead which may not be possible in practice.

In this paper, we focus on the decentralization of the minimum violation planning procedure. Looking forward, we plan to extend the work in this paper to a *real-time* planning framework that can resynthesize plans at runtime as batches of requests arrive in order to avoid violations resulting from myopic planning.

6 Conclusion

The work in this paper is the first to consider a decentralized minimum violation planning approach for UAM traffic management. The method is generalizable and flexible, as it is agnostic to the design of the underlying vehicles being controlled, and it can handle any changes in the safety constraints and still provide guarantees. Empirical results show the practical viability of our approach and is able to handle large numbers of connected vertihubs and requests. For future work, we aim to incorporate online re-planning in order to react to incoming vehicle requests in real-time and still satisfy regulations as much as possible.

References

1. Bharadwaj, S., Bloem, R., Dimitrova, R., Konighofer, B., Topcu, U.: Synthesis of minimum-cost shields for multi-agent systems. In: 2019 American Control Conference (ACC). pp. 1048–1055 (2019)
2. Bharadwaj, S., Carr, S., Neogi, N., Topcu, U.: Decentralized control synthesis for air traffic management in urban air mobility. UNDER REVIEW at IEEE Transactions on Control of Network Systems (2020)
3. Bharadwaj, S., Carr, S., Neogi, N., Poonawala, H., Chueca, A.B., Topcu, U.: Traffic management for urban air mobility. In: NASA Formal Methods - 11th International Symposium, NFM 2019, Houston, TX, USA, May 7-9, 2019, Proceedings. pp. 71–87 (2019)
4. Bloem, R., Könighofer, B., Könighofer, R., Wang, C.: Shield synthesis. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 533–548. Springer (2015)
5. EmbraerX: Flight plan 2030: An air traffic management concept for urban air mobility. EmbraerX (2019)
6. Falcone, Y.: You should better enforce than verify. In: Runtime Verification - First International Conference, RV 2010, St. Julians, Malta, November 1-4, 2010. Proceedings. pp. 89–105 (2010)
7. Federal Aviation Administration: Fact Sheet – Facts about the FAA and Air Traffic Control. https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=23315 (February 2020), (Accessed on 09/16/2020)
8. Foina, A.G., Sengupta, R., Lerchi, P., Liu, Z., Krainer, C.: Drones in smart cities: Overcoming barriers through air traffic control research. In: 2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS). pp. 351–359 (2015)
9. Goyal, R.: Urban air mobility (uam) market study (2018)
10. Guerreiro, N.M., Butler, R.W., Maddalon, J.M., Hagen, G.E.: Mission planner algorithm for urban air mobility–initial performance characterization. In: AIAA Aviation 2019 Forum. p. 3626 (2019)
11. Gunter, E., Peled, D.: Temporal debugging for concurrent systems. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 431–444. Springer (2002)
12. Könighofer, B., Alshiekh, M., Bloem, R., Humphrey, L., Könighofer, R., Topcu, U., Wang, C.: Shield synthesis. Formal Methods in System Design **51**(2), 332–361 (2017). https://doi.org/10.1007/s10703-017-0276-9
13. Moore, A., Balachandran, S., Young, S.D., Dill, E.T., Logan, M.J., Glaab, L.J., Munoz, C., Consiglio, M.: Testing enabling technologies for safe UAS urban operations. In: Proceedings of the 2018 Aviation, Technology, Integration, and Operations Conference. No. AIAA-2018-3200, Atlanta, Georgia (June 2018)
14. Muñoz, C., Narkawicz, A., Hagen, G., Upchurch, J., Dutle, A., Consiglio, M., Chamberlain, J.: Daidalus: Detect and avoid alerting logic for unmanned systems. In: 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC). pp. 5A1–1–5A1–12 (Sep 2015)
15. Neogi, N., Cuong, C., Dill, E.: A risk based assessment of a small UAS cargo delivery operation in proximity to urban areas. In: Proceedings of the 37th Digital Avionics Systems Conference (DASC). London, England, UK (September 2018)
16. Prevot, T., Callantine, T., Lee, P., Mercer, J., Battiste, V., Johnson, W., Palmer, E., Smith, N.: Co-operative air traffic management: a technology enabled concept

for the next generation air transportation system. In: 5th USA/Europe Air Traffic management Research and Development Seminar, Baltimore, MD (June 2005)

17. Prevot, T., Rios, J., Kopardekar, P., Robinson, J.E., Johnson, M., Jung, J.: UAS Traffic Management (UTM) concept of operations to safely enable low altitude flight operations. In: Proceedings of the 2018 Aviation, Technology, Integration, and Operations Conference. No. AIAA-2016-3292, Washington, DC (June 2016)
18. Schneider, F.B.: Enforceable security policies. ACM Trans. Inf. Syst. Secur. **3**(1), 30–50 (2000)
19. Tomlin, C., Pappas, G., Lygeros, J., Godbole, D., Sastry, S., Meyer, G.: Hybrid control in air traffic management systems. IFAC Proceedings Volumes **29**(1), 5512–5517 (1996)
20. Tumova, J., Castro, L.I.R., Karaman, S., Frazzoli, E., Rus, D.: Minimum-violation LTL planning with conflicting specifications. In: 2013 American Control Conference. pp. 200–205 (June 2013)
21. Tumova, J., Hall, G.C., Karaman, S., Frazzoli, E., Rus, D.: Least-violating control strategy synthesis with safety rules. In: Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control. pp. 1–10. HSCC '13, ACM, New York, NY, USA (2013). https://doi.org/10.1145/2461328.2461330, http://doi.acm.org/10.1145/2461328.2461330
22. Wongpiromsarn, T., Topcu, U., Ozay, N., Xu, H., Murray, R.M.: Tulip: a software toolbox for receding horizon temporal logic planning. In: Proceedings of the 14th international conference on Hybrid systems: computation and control. pp. 313–314 (2011)