Safety Verification for Urban Air Mobility Scheduling [★]

Qinshuang Wei [*] **Gustav Nilsson** [**] **Samuel Coogan** [*]

[*] *Georgia Institute of Technology, Atlanta, GA 30332, USA (e-mail: {qinshuang,sam.coogan}@gatech.edu).*
[**] *École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, CH-1015 (e-mail: gustav.nilsson@epfl.ch)*

Abstract: In Urban Air Mobility (UAM) networks, takeoff and landing sites, called vertiports, are likely to experience intermittent closures due to, e.g., adverse weather. For safety, it will be required that all in-transit Urban Air Vehicles (UAVs) in a UAM network have alternative landing sites in the event of a vertiport closure. In this paper, we propose analytical conditions for developing an efficient algorithm that, given a proposed UAM schedule, verifies whether all UAVs are able to safely reach a back-up landing site in the event of a vertiport closure without violating the limited landing capacity of each vertiport in the network. If safety verification is not possible, the algorithm returns a counterexample demonstrating the violation. Our solution allows for uncertain travel time between UAM vertiports and scales quadratically with the number of scheduled UAVs. We demonstrate our algorithm on a UAM network with up to 1,000 UAVs.

Keywords: Urban Air Mobility, Safety Verification, Transportation Network

1. INTRODUCTION

Urban air mobility (UAM) refers to using urban airspace for transportation of people and goods in cities and surrounding regions and is being explored by both commercial mobility-on-demand operators, Holden and Goel (2016), and government-sponsored research institutes such as NASA, Thipphavong et al. (2018). Studies such as Balakrishnan et al. (2018); The MITRE Corporation (2014); Lascara et al. (2018); INRIX (2019); Al Haddad (2018); Ancel et al. (2017) propose various approaches to allow urban air vehicles (UAVs) to travel safely and efficiently through cities. These proposed ideas cover a wide range of possibilities such as allowing UAVs to land at *vertistops* or *vertiports* installed on roofs of existing buildings or within cloverleaf exchanges on freeways. Meanwhile, several simulation tools, e.g., Bosson and Lauderdale (2018); Xue et al. (2018); Aiello et al. (2019), have also been developed to study large-scale interactions of UAVs.

For any UAM solution, unforeseen disruptions such as intermittent closure of landing sites due to, e.g., extreme weather conditions as mentioned in Balakrishnan et al. (2018), needs to be considered. For the sake of safety, it must be ensured that, once the network is disrupted, there will be enough landing spots available for all UAVs conforming to the emergency rerouting rules. In this paper, we consider schedule disruptions within the model proposed in Wei et al. (2021), which accounts for uncertain travel time and limited landing capacity, and we develop a safety verification algorithm for a given UAM schedule. In Wei et al. (2021), the schedules are obtained so that the trip demands, i.e., flights, must travel through designated routes and meet their corresponding arrival deadlines at their destinations, while satisfying the landing-spot restrictions at the destination and intermediate nodes upon arrival. In this paper, we further consider the problem of verifying the safety of a given schedule upon the closure of a node in the network. The main contributions are as follows: First, we add the disruption model, which considers the closure of vertiports or vertistops, to the existing UAM network model in Wei et al. (2021). We then provide necessary and sufficient conditions for a given UAM schedule to be guaranteed as safe in the disrupted scenario under worst-case travel time realization, which leads to our proposed safety verification algorithm. We also provide necessary and sufficient conditions for guaranteed safety under best-case travel time realization. Lastly, we demonstrate our verification algorithm and its computation efficiency on a UAM network with up to 1,000 UAVs.

Safety of UAV scheduling has been explored in several papers. In Ancel et al. (2017), a risk assessment framework is developed to provide real-time safety evaluation and tracking capability for the Unmanned Aircraft System Traffic Management. This paper assesses the risk of off-nominal conditions to people on the ground by calculating the potential impact area and the effects of the impact instead of safely rerouting the UAV itself. The simulation tool AutoResolver has the ability of continuously ensuring safe separation between UAVs given both spatial and temporal constraints in Bosson and Lauderdale (2018), where the UAVs are expected to arrive at the destination as early as possible. In contrast, under the setting considered in this paper, we impose the separation constraint between UAVs

[★] This work was partially supported by the NASA University Leadership Initiative (ULI) under grant number 80NSSC20M0161 and by the National Science Foundation under grant number 1749357.

from another aspect, i.e., the limited landing capacities at the vertistops or vertiports along the routes.

The more general problem of verifying safety of schedules is considered in Wang et al. (2019); Liu and Joseph (1999); Yasmeen et al. (2012); Cimatti et al. (2000); Chen et al. (2017), which provide symbolic reliability checking for safety-critical systems including flight control. However, these approaches provide general verification for real-time scheduling protocols with interruption, and do not provide an approach for the network to deal with the specific interruption of node closures that reduce capacity, as is the case here.

The remainder of the paper is organized as follows: In Section 2, we first recall the UAM network model in Wei et al. (2021), and then introduce the disruption model that reduces capacity of the network. In Section 3, we establish safety criteria and develop necessary and sufficient conditions for the schedule to be safe under disruptions. We then demonstrate how these constraints can be used for safety verification on a UAM network in Section 4 and compare the verification time for different sizes of schedules.

2. PROBLEM FORMULATION

2.1 Network Model and Nominal Scheduling

We model an urban air mobility (UAM) network with a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of links for the network. Nodes are physical landing sites for the UAVs, sometimes called *vertistops* or *vertiports*. Links are corridors of airspace connecting nodes. Each node $v \in \mathcal{V}$ has capacity $C_v \in \mathbb{N}_0$, that is, there are C_v *landing spots* at node v where each landing spot allows at most one UAV to stay at any time. We denote the vector of capacities $C = \{C_v\}_{v \in \mathcal{V}}$.

We define $\tau : \mathcal{E} \to \mathcal{V}$ and $\sigma : \mathcal{E} \to \mathcal{V}$ so that for all $e = (v_1, v_2) \in \mathcal{E}$ where $v_1, v_2 \in \mathcal{V}$, $\tau(e) = v_1$ is the tail of edge e and $\sigma(e) = v_2$ is the head of edge e. Let $S \subseteq \mathcal{V}$ (resp., $T \subseteq \mathcal{V}$) be the set of nodes that are not the head (resp., tail) of any edge, $S = \{v \in \mathcal{V} \mid \sigma(e) \neq v \ \forall e \in \mathcal{E}\}$ and $T = \{v \in \mathcal{V} \mid \tau(e) \neq v \ \forall e \in \mathcal{E}\}$. We assume $S \cap T = \emptyset$.

A *route* R is a sequence of connected links. Denote the number of links in route R by k_R and enumerate the links in the route $1^R, 2^R, \ldots, k_R^R$ and the nodes in the route $0^R, 1^R, \ldots, k_R^R$. To avoid cumbersome notation, we use ℓ^R to denote both a link and its head node along a route, i.e., $\ell^R = \sigma(\ell^R)$ for all $\ell \in \{1, \ldots, k_R\}$; the intended meaning will always be clear from context. Thus the route links and nodes are enumerated so that $0^R = \tau(1^R)$ is the origin node, k_R^R is the destination node, and $\sigma(\ell^R) = \tau((\ell+1)^R)$ for all $\ell \in \{1, \ldots, k_R\}$ ensures the sequence is connected. Further, when the route R is clear from context, we drop the superscript-R notation. We denote the set of nodes that R travels through as $V(R)$.

We assume that, due to operational reasons, the UAVs are only allowed to travel along a set of routes $\mathcal{R}$.

Since, in reality, the travel time depends on external factors such as weather conditions, we assume that the travel time for each link is not exact, but rather bounded by a time interval. For each link $e \in \mathcal{E}$, let $\overline{x}_e$ and $\underline{x}_e$ with $\overline{x}_e \geq \underline{x}_e > 0$ denote the maximum travel time and minimum travel time, respectively, for the link, and let $\underline{x} \in \mathbb{R}_+^{\mathcal{E}}$ and $\overline{x} \in \mathbb{R}_+^{\mathcal{E}}$ be the corresponding aggregated vectors. Once a UAV has landed at any node, it is assumed to block a landing spot for a fixed ground service time $w \in \mathbb{R}_+$. For ease of notation, we assume the ground service time is uniform at all nodes, but this assumption is straightforward to relax.

Definition 1. (UAM Network). A UAM network $\mathcal{N}$ is a tuple $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \overline{x}, w)$ where $\mathcal{G}, C, \mathcal{R}, \underline{x}, \overline{x}, w$ are the network graph, node capacities, routes, and minimum and maximum link travel times as defined above.

To model the schedule of UAV flights in a UAM network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \overline{x}, w)$, we assume that every flight is associated to a route $R \in \mathcal{R}$ and stops at intermediate nodes along the route. Therefore, a *schedule* is a pair (R, δ) where $R \in \mathcal{R}$ and $\delta \in \mathbb{R}_+$ is the appointed departure time from the first node along the route. A *schedule profile* for a UAM network is a set $\mathcal{S} = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$ where $\mathcal{J}$ is a finite index set of *flights*, and hence, $\mathcal{S}$, is assumed finite.

For safety reasons, it is assumed that a UAV must be able to land immediately upon arrival at any node along its route. We let $\ell \geq 1$, the latest arrival time at the node $\sigma(\ell)$ along the route is denoted a_ℓ^j and given by

$$a_\ell^j = \delta_j + \sum_{k=1}^{\ell} \overline{x}_k + (\ell - 1)w\,, \tag{1}$$

i.e., a_ℓ^j is the departure time from node 0 plus the upper bound of the time interval it takes to travel through the links $\{1, 2, \ldots, \ell\}$ with the time spent at each intermediate node. Further, the time interval that the flight will potentially block a landing spot at node ℓ is given by

$$\mathcal{M}_\ell^j = \left[\delta_j + \sum_{k=1}^{\ell} \underline{x}_k + (\ell - 1)w,\ a_\ell^j + w\right]. \tag{2}$$

We let $\mathcal{M}_v^j = \mathcal{M}_\ell^j$ and $a_v^j = a_\ell^j$ if $v \in V(R_j)$ and $v = \ell^{R_j}$.

Definition 2. (Realization). A *realization* of a schedule (R, δ) refers to a set of realizations for the travel times on each link along route R that obeys the minimum and maximum travel time limits. We do not assume any probability distribution on realizations, but we will consider certain conditions that hold in the worst-case, i.e., for all realizations, or in the best-case, i.e., there exists some realization satisfying the condition.

Definition 3. (Feasible Schedule Profile). A schedule profile $\mathcal{S} = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$ where $\delta_j \in \mathbb{R}_+$ for all $j \in \mathcal{J}$ is a *feasible schedule profile* if, for all realizations (that is, in worst-case), the number of vehicles at a node never exceeds capacity, i.e., for all $v \in \mathcal{V}$ and all $t \geq 0$,

$$\sum_{j : v \in V(R_j)} \mathbf{1}\left(t; \mathcal{M}_v^j\right) \leq C_v \tag{3}$$

where the notation $\mathbf{1}(\cdot;\cdot)$ is an indicator such that $\mathbf{1}(t; [a,b]) = 1$ if $t \in [a,b]$ and $\mathbf{1}(t; [a,b]) = 0$ otherwise.

While every feasible schedule will by definition ensure proper operation of the UAM network under normal circumstances, our objective is to ensure the schedule is resilient to interruptions in the network.

2.2 Disruption Model

In actual operation, it is expected that unforeseen disruptions that disable one or more nodes, such as adverse weather conditions, will be common. The arrangement for the flights affected by the disabled nodes needs to be considered in advance to ensure safety, that is, when a node is disabled, each flight passing through the disabled node must have a rerouting plan that ensures availability of a landing spot. In this paper, we postulate the existence of a set of *backup nodes* for the network so that when any node is disabled, the flights can be redirected to the backup nodes. To which backup node a flight will be directed to will depend upon which like the flight is currently travelling on.

In this subsection, we introduce the assignment of the backup nodes and the operating mechanism once a node is disabled. We assume that only one node may be disabled in the rest of the paper. In order to guarantee that each disrupted flight will be able to be assigned to a node after the disruption, we assign a backup node to each link in the network. Naturally, we let the backup node of the link be its head if the head node is not disabled, and let the backup node be the tail if the head node is disabled. Under this assignment of backup node, a flight whose route is potentially being blocked and is traveling on a link e will continue to the head node $\sigma(e)$ on its route if it is functioning, or return to the previous node $\tau(e)$ if the head node is disabled.

We say the j'th flight is *affected* by some disabled node $v_c \in \mathcal{V}$ at time t_c if $v_c \in V(R_j)$, i.e., the route of the flight travels through node v_c, and the flight has not yet reached v_c by time t_c in real time. The flight is *not affected* when the node v_c is disabled at time t_c otherwise.

Below is a set of rules that all flights need to follow once a node v_c is disabled at time t_c:

(1) flights not affected will continue normal operation;
(2) any affected flight that has not yet departed ($\delta_j > t_c$) will be canceled (no longer depart);
(3) an affected flight j with $\delta_j \leq t_c$ traveling on a link $e \in \mathcal{E}$ with $\sigma(e) \neq v_c$ will continue to the head node $\sigma(e)$ and stop there indefinitely (block a landing spot indefinitely);
(4) an affected flight j with $\delta_j \leq t_c$ that is landed at a node at time t_c will remain there indefinitely;
(5) an affected flight j with $\delta_j \leq t_c$ traveling on a link $e \in \mathcal{E}$ with $\sigma(e) = v_c$ will return to the tail of the link $\tau(e)$ and stop there indefinitely.

Note that we do not consider the problem of recovering a new schedule after a disabled node becomes operational again, as our focus is on safety. Further, we postulate the above rules to provide a well-defined problem formulation; alternative rules might be also plausible.

3. NECESSARY AND SUFFICIENT CONDITIONS FOR SAFE SCHEDULE

In this section, we formally define safety and present necessary and sufficient conditions for safety. Given a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \overline{x}, w)$ where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a feasible schedule $\mathcal{S} = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$, we regard the j'th flight as *possibly affected* by disabling node v_c at time t_c if $v_c \in V(R_j)$ and $t_c < \sup \mathcal{M}^j_{v_c}$, i.e., the flight may have to travel through the disabled node later than t_c, depending on realized travel times. The closure of a node can affect the set of schedules in different ways. In particular, the set of schedules is:

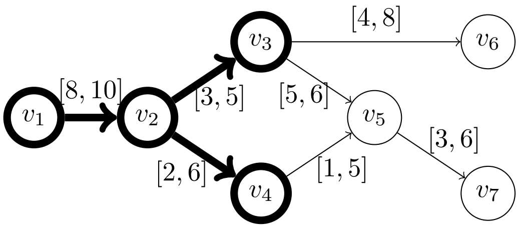

Fig. 1. (Bold partial graph) The sub-graph consisted of the bold lined 4 nodes and 3 links is used to illustrate the simple network in Example 1.
(Entire graph) The entire graph is used to illustrate the network with 7 nodes and 7 links in the case study.

(1) *worst-case (resp., best-case) time-node conditionally safe* for node v_c and time t_c if, supposing that v_c is disabled at time t_c, then all possibly affected flights are able to land at their designated backup nodes while not interfering with any unaffected flights, for all (resp., for some) realization of link travel times.
(2) *worst-case (resp. best-case) node conditionally safe* for node v_c if it is worst-case (resp. best-case) time-node conditionally safe for node v_c for all time $t_c \geq 0$.
(3) *worst-case (resp. best-case) 1-closure safe* if it is worst-case (resp. best-case) node conditionally safe for any node $v_c \in \mathcal{V}$.

Note that worst-case safety implies best-case safety.

Example 1. We illustrate the model and the safe criteria through the simple network shown in Fig. 1 with 4 nodes and 3 links (the bold lined sub-graph). For this example, the set of nodes $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$ and $\mathcal{E} = \{(v_1, v_2), (v_2, v_3), (v_2, v_4)\}$. We assume that the origin v_1 does not have a capacity constraint, while $C_{v_2} = 2$, $C_{v_3} = 1$ and $C_{v_4} = 1$. The links are indicated in the figure and the corresponding travel time intervals are labeled beside the links, e.g., the interval $[8, 10]$ above the link (v_1, v_2) means that the shortest (resp., longest) possible time for traveling through the link is 8 min (resp., 10 min). We consider two possible routes $R^1 = \{(v_1, v_2), (v_2, v_3)\}$ and $R^2 = \{(v_1, v_2), (v_2, v_4)\}$. Each flight remains at the intermediate nodes or destination along its path for $w = 1$ time unit after landing. Given a feasible schedule $\mathcal{S} = \{S_1, S_2, S_3\}$, where $S_1 = (R^2, 1)$, $S_2 = (R^2, 8)$ and $S_3 = (R^1, \delta_3)$, where we consider several δ_3. Assume v_4 is disabled at time $t_c = 15$ min. Under worst-case travel time realizations, flight S_1 will be traveling on (v_2, v_4) at $t_c = 15$ and needs to be rerouted to v_2; Flight S_2 will be traveling on (v_1, v_2) and needs to stay at v_2 upon arrival. If $\delta_3 = 0$, flight S_3 is not affected, and should continue its journey; however, if $\delta_3 = 10$, then v_2 will be short of landing spots upon the arrival of S_3. Therefore, $\mathcal{S}$ is worst-case time-node conditionally safe for node v_4 at time 15 if $\delta_3 \leq 4$ and is not if $\delta_3 > 4$. In contrast, it is always best-case time-node conditionally safe for node v_4 at time 15 regardless of the choice of δ_3. Meanwhile, suppose $C_{v_2} = 3$, and we let $\delta_3 = 10$, then obviously, the network will be able to handle the flights after closure no matter when the node v_4 is closed. Therefore, we see that $\mathcal{S}$ is worst-case node conditionally safe for node v_4 in this case. We further check that this is true for all nodes in the network, and thus $\mathcal{S}$ is also worst-case 1-closure safe. □

To obtain conditions for 1-closure safety, we start by observing that a feasible schedule is trivially node conditionally safe for node $v \in S$, where we recall S the set of source nodes that are not the head of any link.

We next explore safety of a disabled node not in S. Before we study the sufficient and necessary conditions for 1-closure safety when disabling a node $v_c \in \mathcal{V}\backslash S$, we first define several special sets that are used in the conditions. If $v_c = \ell^{R_j}$, we let $\ell_{v_c,R_j} = \ell$, and $\ell_{v_c,R_j} = 0$ if $v_c \notin V(R_j)$.

We let the set of links with head $v \in \mathcal{V}$ be

$$\mathcal{E}_v := \{e \in \mathcal{E} \mid \sigma(e) = v\}\,, \tag{4}$$

and let b_{e,v_c} be the node that any flight traveling on link e will proceed to if v_c is disabled:

$$b_{e,v_c} = \begin{cases} \sigma(e) & \text{if } \sigma(e) \neq v_c\,, \\ \tau(e) & \text{if } \sigma(e) = v_c\,. \end{cases} \tag{5}$$

We denote the set of links on which flights will be rerouted to node v when v_c is disabled as

$$\mathcal{B}_{v,v_c} := \{e \in \mathcal{E} \mid b_{e,v_c} = v\}\,. \tag{6}$$

We define the set $\mathcal{J}_v$ as the index set of the flights with routes passing through the node v,

$$\mathcal{J}_v := \{j \in \mathcal{J} \mid v \in V(R_j)\}\,, \tag{7}$$

and we further define the index set of the flights that might possibly be landed at v after time t as

$$\mathcal{J}_v^*(t) := \{j \in \mathcal{J}_v \mid a_v^j + w > t\}\,. \tag{8}$$

Therefore, the index set of the possibly affected flights when node v_c is closed at time t_c is $\mathcal{J}_{v_c}^*(t_c)$, while the index set for the flights passing through the node v that are not possibly affected when the node v_c is closed at t_c is $\mathcal{J}_p(v,t_c,v_c) = \mathcal{J}_v\backslash\mathcal{J}_{v_c}^*(t_c)$.

We use $\mathcal{J}_c(t_c)$ to represent the set of indices for canceled flights with departure time greater than the closing time t_c:

$$\mathcal{J}_{c,v_c}(t_c) := \{j \in \mathcal{J}_{v_c} \mid \delta_j > t_c\}\,. \tag{9}$$

We let $N_R(v,t_c,v_c)$ be the maximum number of flights that might possibly be landed at node v at the same time once the node v_c is closed at time t_c, which can be computed as

$$N_R(v,t_c,v_c) = \sup_{t \geq t_c} \sum_{j \in \mathcal{J}_p(v,t_c,v_c)} \mathbf{1}(t; \mathcal{M}_v^j)\,. \tag{10}$$

In the rest of the paper, we sometimes drop the notations in the parentheses, t, t_c, v, v_c, when they are clear from the context.

Theorem 1. Consider a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \overline{x}, w)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Assume given a feasible schedule $\mathcal{S} = \{(R_j, \delta_j)\}_{j\in\mathcal{J}}$. Define $N_e(t_c, v_c)$ as the number of flights on link e that stay at b_{e,v_c} if node v_c is disabled at time t_c for all $e \in \mathcal{E}$, i.e.,

$$N_e(t_c,v_c) = \sum_{j \in \mathcal{J}_{v_c}^{*\setminus c}(t_c)} \mathbf{1}(t_c; [L_e^j, U_e^j]\backslash\mathcal{O}) \tag{11}$$

where

$$\mathcal{O} = \begin{cases} \emptyset & \text{if } e \notin \mathcal{E}_{v_c} \\ [L^j_{(\ell_{v_c,R_j}-1)^{R_j}}, U^j_{(\ell_{v_c,R_j}-1)^{R_j}}] & \text{if } e \in \mathcal{E}_{v_c} \end{cases} \tag{12}$$

and we define the lower and upper bounds of the time interval as

$$L_e^j = \begin{cases} \inf\{\mathcal{M}^j_{\tau(e)}\} + w & \text{if } \tau(e) \neq 0^{R_j} \\ \delta_j & \text{if } \tau(e) = 0^{R_j} \end{cases} \tag{13}$$

and

$$U_e^j = \begin{cases} \sup\{\mathcal{M}^j_{\sigma(e)}\} & \text{if } \sigma(e) \neq v_c \\ \sup\{\mathcal{M}^j_{\sigma(e)}\} - w & \text{if } \sigma(e) = v_c. \end{cases} \tag{14}$$

Further define $N_v(t_c, v_c)$ as the number of possibly affected flights that may block the node v indefinitely, i.e., $N_v(t_c,v_c) = \sum_{e\in\mathcal{B}_{v,v_c}} N_e(t_c,v_c)$.

Then $\mathcal{S}$ is worst-case time-node conditionally safe for node v_c and time t_c if and only if, for all $v \in \mathcal{V}$,

$$C_v - N_v(t_c,v_c) \geq N_R(v,t_c,v_c)\,. \tag{15}$$

Further, $\mathcal{S}$ is worst-case node conditionally safe for node v_c if and only if (15) holds for the finite number of times t_c where the values of $N_v(t_c,v_c)$ and $N_R(v,t_c,v_c)$ possibly change, i.e., at both endpoints of the interval $\mathcal{M}_v^j$ for all $v \in \mathcal{V}$ and at times L_e^j, U_e^j for all $e \in \mathcal{B}_{v,v_c}$.

Proof. To guarantee the schedule is time-node conditionally safe for node v_c and time t_c, for any possibly affected flight that is not canceled and may be rerouted to some node $v \in \mathcal{V}$ at time t_c, a landing spot needs to be reserved. Then the problem becomes to ensure the flights surely not affected will have no capacity conflict with any possibly rerouted flights. We then consider the maximum (worst-case) occupation of the node v.

The interval defined as $[L_e^j, U_e^j]$ is the time interval during which flight j possibly be rerouted to b_{e,v_c} if v_c is closed since the lower bound L_e^j is the earliest time that the flight may leave the previous node $\tau(e)$, and, if $\sigma(e)$ is not disabled, the upper bound U_e^j is the latest time that the flight may leave the head node while, in the case that $\sigma(e)$ is disabled, the upper bound U_e^j for the time interval that the flight may be rerouted to the backup node $\tau(e)$ will be the latest time that the corresponding flight may arrive at the node v_c, since otherwise it will continue its journey without rerouting.

Therefore, $N_e(t_c,v_c)$ in (12) is the number of parking spots needs to be reserved at b_{e,v_c} for the possibly affected flights and will not be canceled. Notice that, if $e \in \mathcal{E}_{v_c}$, we only need to reserve an extra space at node $\tau(e)$ for a flight if it is not possibly traveling on the previous link along its route at time t_c. $N_v(t_c,v_c)$ is the number of landing spots needed to be reserved for the rerouted flights. Finally, $N_R(v,t_c,v_c)$ is the maximum number of flights not possibly affected that may park at the node v at any time once v_c is closed at t_c. Therefore, if (15) is satisfied if and only if capacity conflict does not exist for any realizations at any node in the network. Hence (15) is a necessary and sufficient condition for the time-node conditional safety. □

Theorem 1 provides a finite number of conditions to verify a schedule is worst-case node conditionally safe for node v_c. Furthermore, by checking that a schedule is node conditionally safe for all $v_c \in \mathcal{V}$, we can conclude the worst-case 1-closure safety. The same checking technique will be applied to the necessary condition below for best-case safety in Theorem 2, which is obtained by observing the best scenario case where we assume that all flights possible to have arrived at or passed the closed node v_c has already arrived or left by the time of failure. We denote the index set of the possibly affected flights that may have arrived at or passed the node v_c disabled at t_c as $\mathcal{J}''_{v_c}$, and

$$\mathcal{J}''_{v_c}(t_c) = \{j \in \mathcal{J}^*_{v_c} \mid \inf\{\mathcal{M}^j_{v_c}\} \le t_c\}\,. \tag{16}$$

Theorem 2. Consider a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \overline{x}, w)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. A feasible schedule $\mathcal{S} = \{(R_j, \delta_j)\}_{j\in\mathcal{J}}$ is best-case time-node conditionally safe for node v_c and time t_c only if there exists a set of non-negative integers $\{N_v\}_{v\in\mathcal{V}}$ that satisfies

$$\sum_{v\in\mathcal{V}} N_v = |\mathcal{J}^*_{v_c}(t_c)\backslash\{\mathcal{J}_{c,v_c}(t_c) \cup \mathcal{J}''_{v_c}(t_c)\}| \tag{17}$$

$$C_v - N_v \ge \sup_{t\ge t_c} \sum_{j\in\mathcal{J}_v\backslash\mathcal{J}^*_{v_c}(t_c)} \mathbf{1}\big(t; \mathcal{M}^j_v\big) \qquad \forall v \in \mathcal{V}\,. \tag{18}$$

Further, $\mathcal{S}$ is best-case node conditionally safe for node v_c only if, for the finite number of times t_c where the time-varying index sets $\mathcal{J}^*_{v_c}(t_c)$, $\mathcal{J}_c(t_c)$ and $\mathcal{J}''_{v_c}(t_c)$, and the endpoints of the interval $\mathcal{M}^j_v$ for all $v \in \mathcal{V}$ and $j \in \mathcal{J}$ possibly change, there exists a set of non-negative integers $\{N_v\}_{v\in\mathcal{V}}$ that satisfies the constraints (17)–(18).

Proof. The proof of Theorem 2 applies the similar logic as in Theorem 1 to the best-case scenario. First of all, the set of all rerouted flights has to be the same as the possibly affected flights $\mathcal{J}^*_{v_c}$ except for the canceled flights, $\mathcal{J}_c(t_c)$, or the flights that are possibly not affected, $\mathcal{J}''_{v_c}$, as depicted in (17). If we are not able to find a sequence of non-negative integers $\{N_v\}_{v\in\mathcal{V}}$ that satisfies (17)–(18), then there must exist a conflict of occupation at one or more nodes once v_c is closed at time t_c for some realizations, and hence (17) and (18) are necessary conditions for the set of schedules to be best-case time-node conditionally safe for node v_c and time t_c. □

Theorem 1 is both sufficient and necessary for worst-case 1-closure safety, while Theorem 2 is necessary for best-case 1-closure safety. Since worst-case safety implies best-case safety, satisfaction of the conditions in Theorem 1 implies satisfaction of the conditions in Theorem 2.

4. CASE STUDY

In the case study, we first demonstrate an algorithm for safety verification derived from the conditions in Theorem 1 on a UAM network with 7 nodes and 7 links as shown in Fig. 1 with a feasible schedule of size 20. Notice that the algorithm is also able to deal with other general directed graphs. We then show the relation between the verification time and the size of the schedules being examined through various sets of schedules with different sizes.

To ensure the safety for all realizations of link travel time, we check whether the condition of worst-case time-node conditional safety for node v_c and t_c is satisfied or not over the time interval $t_c \in [0, +\infty)$. As stated in Theorem 1, we only need to verify (15) at each point of time that any value may change, i.e., L^j_e, U^j_e and both ends of $\mathcal{M}^j_v$ for any counted flight $j \in \mathcal{J}$ and link $e \in \mathcal{E}$ for some fixed node v, since the inequality (15) will not change between these points.

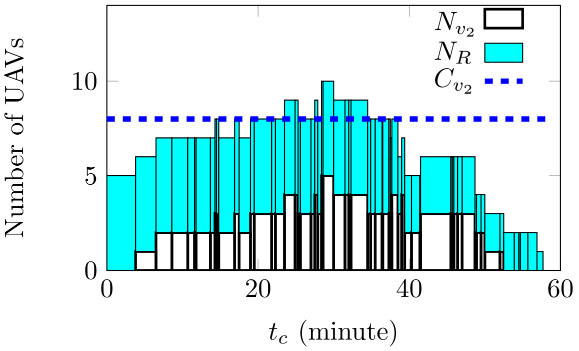

Fig. 2. The number of flights rerouted to v_2, $N_{v_2}(t_c, v_5)$ (simplified as N_{v_2}), represented by the upper blue rectangles and the maximum unaffected occupation at v_2 when the node v_5 is disabled at time t_c, $N_R(v_2, t_c, v_5)$ (simplified as N_R), represented by the lower white rectangles. The dotted line corresponds to the capacity at node v_2.

In the network in Fig. 1 (the entire graph), the set of all possible origins (resp., destinations) is $S = \{v_1\}$ (resp., $T = \{v_6, v_7\}$). We assume the origin v_1 does not have capacity constraint, while $C_{v_2} = 8$, $C_{v_3} = 4$, $C_{v_4} = 2$, $C_{v_5} = 4$, $C_{v_6} = 3$ and $C_{v_7} = 5$. The links are indicated in the figure and the corresponding travel time intervals are labeled beside the links. We consider three routes $\mathcal{R} = \{R^1, R^2, R^3\}$ with $R^1 = \{(v_1, v_2), (v_2, v_3), (v_3, v_6)\}$, $R^2 = \{(v_1, v_2), (v_2, v_3), (v_3, v_5), (v_5, v_7)\}$ and $R^3 = \{(v_1, v_2), (v_2, v_4), (v_4, v_5), (v_5, v_7)\}$. Each flight remains at the vertistops along its path for $w = 1$ time unit after landing. We randomly generate a particular feasible schedule profile with 20 flights and consider the inequality (15) in Theorem 1 for worst-case time-node conditionally safe for node $v_c = v_5$ and any time $t_c > 0$. The verification is implemented in MATLAB [1].

In Fig. 2, the white rectangles show the flights being rerouted to the node v_2 if the node v_5 is closed at time t_c, which is $N_{v_2}(t_c, v_5)$ in (15); the blue rectangles represent the number of flights not affected and continue to v_2 if the node v_5 is closed at time t_c, which is $N_R(v_2, t_c, v_5)$. As a reference, the capacity $C_{v_2} = 8$ is shown as the dotted, horizontal line so that if the height of the entire bar exceeds the capacity, then Theorem 1 is not satisfied at time t_c. For example, the schedule in this case study is not worst-case node conditionally safe for node v_5, as the node v_2 is not able to accommodate to the failure of v_5 in certain time intervals, e.g., $t_c \in [30, 35]$. Similarly, we can check if the other nodes are able to accommodate the failure of node v_5 for any t_c with the same technique. If for time $t_c > 0$, all the nodes are able to accommodate the failure of v_5, then we would conclude that the schedule profile is worst-case time-node conditionally safe for node v_5 and time t_c. We can also observe the performance of the network and the schedules with the failure of any other node at any time.

The computation for $N_R(v, t_c, v_c)$ in (15) implies $O(n^2)$ computational complexity of this verification process. Therefore, we are also able to efficiently verify worst-case safety large schedule profiles. As an example, consider

[1] The related MATLAB code and pseudo code of the algorithm can be found in https://github.com/gtfactslab/Wei_NecSys22.

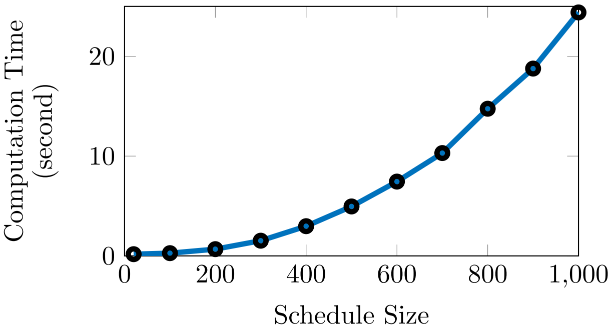

Fig. 3. The computation time for verifying the worst-case safety of schedules with different sizes. The data points constitute a parabola, which demonstrates the $O(n^2)$ computational complexity.

increasing the capacity for each node of the network in Fig. 1 by 10 (this does not change the verification time). We generate 10 more sets of random feasible schedules with sizes $100, 200, 300, \ldots, 1000$ and verify their safety using the same algorithm. Fig. 3 confirms the quadratic relation between the size of the set of schedules and the time it takes to check the safety and shows that we are able to verify safety or provide a counterexample for a schedule profile with 1,000 flights in under 30 seconds.

5. CONCLUSION

In this paper, we explored the safety verification problem for a UAM schedule in a disruption scenario in which a node must close and inbound flights are immediately rerouted. We provide a reasonable link-related backup-node assignment for the UAM network and rules for the flights after a node is disabled. The main impediment to safety is that each flight needs to be provided an available landing spot upon arrival of any intermediate, destination, or backup node. We therefore derived sufficient and necessary conditions for worst-case and best-case safety of a given schedule. These theoretical results provide an efficient safety verification algorithm. We have also demonstrated through case study that the computational time for the algorithm grows quadratically with schedule size. As a result, the safety verification algorithm is applicable to large-scale UAM scheduling problems.

We focus on the case where only one node is disabled in this paper, while future work could consider the case where more than one node is disabled. This problem is more challenging because safety verification depends on whether the disabled nodes are adjacent or not and whether they are disabled concurrently or not. Another direction is to consider multiple possible backup nodes.

REFERENCES

Aiello, M.A., Dross, C., Rogers, P., Humphrey, L., and Hamil, J. (2019). Practical application of SPARK to OpenUxAS. In *Formal Methods – The Next 30 Years*, 751–761. Springer International Publishing.

Al Haddad, C. (2018). Identifying the factors affecting the use and adoption of urban air mobility. URL `https://mediatum.ub.tum.de/1482026`.

Ancel, E., Capristan, F.M., Foster, J.V., and Condotta, R.C. (2017). Real-time risk assessment framework for unmanned aircraft system (UAS) traffic management (UTM). In *17th AIAA Aviation Technology, Integration, and Operations Conference*, 3273.

Balakrishnan, K., Polastre, J., Mooberry, J., Golding, R., and Sachs, P. (2018). Blueprint for the sky. *The roadmap for the safe integration of autonomous aircraft. Airbus A*, 3.

Bosson, C. and Lauderdale, T.A. (2018). Simulation evaluations of an autonomous urban air mobility network management and separation service. In *2018 Aviation Technology, Integration, and Operations Conference*, 3365.

Chen, L., Jiao, J., Wei, Q., and Zhao, T. (2017). An improved formal failure analysis approach for safety-critical system based on mbsa. *Engineering Failure Analysis*, 82, 713–725.

Cimatti, A., Clarke, E., Giunchiglia, F., and Roveri, M. (2000). Nusmv: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4), 410–425.

Holden, J. and Goel, N. (2016). Fast-forwarding to a future of on-demand urban air transportation. URL `https://www.uber.com/elevate.pdf`.

INRIX (2019). Electric passenger drones could relieve housing costs and spread growth in nation's booming cities. URL `https://inrix.com/campaigns/vtol-study/`.

Lascara, B., Spencer, T., DeGarmo, M., Lacher, A., Maroney, D., and Guterres, M. (2018). Urban air mobility landscape report: Initial examination of a new air transportation system. *McLean, VA: The MITRE Corporation.*

Liu, Z. and Joseph, M. (1999). Specification and verification of fault-tolerance, timing, and scheduling. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 21(1), 46–89.

The MITRE Corporation (2014). Nextgen independent assessment recommendations. URL `https://www.mitre.org/sites/default/files/publications/pr-14-3495-next-gen-independent-assessment.pdf`.

Thipphavong, D.P., Apaza, R., Barmore, B., Battiste, V., Burian, B., Dao, Q., Feary, M., Go, S., Goodrich, K.H., Homola, J., et al. (2018). Urban air mobility airspace integration concepts and considerations. In *2018 Aviation Technology, Integration, and Operations Conference*, 3676.

Wang, M., Tian, C., Zhang, N., Duan, Z., and Du, H. (2019). Verifying a scheduling protocol of safety-critical systems. *Journal of Combinatorial Optimization*, 37(4), 1191–1215.

Wei, Q., Nilsson, G., and Coogan, S. (2021). Scheduling of urban air mobility services with limited landing capacity and uncertain travel times. In *2021 American Control Conference (ACC)*, 1681–1686. IEEE.

Xue, M., Rios, J., Silva, J., Zhu, Z., and Ishihara, A.K. (2018). Fe3: An evaluation tool for low-altitude air traffic operations. In *2018 Aviation Technology, Integration, and Operations Conference*, 3848.

Yasmeen, A., Feigh, K.M., Gelman, G., and Gunter, E.L. (2012). Formal analysis of safety-critical system simulations. In *ATACCS*, 71–81.