Safe Schedule Verification for Urban Air Mobility Networks with Node Closures

Qinshuang Wei, *Student Member, IEEE*, Gustav Nilsson *Member, IEEE*, and Samuel Coogan, *Member, IEEE*

***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. To ensure safety, all in-flight Urban Air Vehicles (UAVs) in a UAM network must therefore have alternative landing sites with sufficient landing capacity in the event of a vertiport closure. In this paper, we study the problem of safety verification of UAM schedules in the face of vertiport closures. We first provide necessary and sufficient conditions for a given UAM schedule to be safe in the sense that, if a vertiport closure occurs, then all UAVs will be able to safely land at a backup landing site. We then extend these results to the scenario of multiple vertiport closures. Next, we convert these conditions to an efficient algorithm for verifying the safety of a UAM schedule via a linear program by using properties of totally unimodular matrices. Our algorithm 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.**

***Index Terms*—Safety Verification, Transportation Network, Urban Air Mobility,**

I. Introduction

Urban airspace is promising for transporting people and goods in cities and surrounding regions to avoid ground transportation congestion. Both commercial mobility-on-demand operators [2] and government-sponsored research institutes such as NASA [3] are actively involved in developing such urban air mobility (UAM) solutions. Safety and efficiency of the urban air vehicles (UAVs) are major concerns in all UAM solutions [4]–[8]. The work [8] observes that safety is one of the key factors affecting the adoption of UAM, while [4]–[6] provide guidelines for safely integrating the UAVs into the existing airspace. The paper [7] provides insight into the improvement of commute efficiency with the usage of urban airspace compared to ground transportation. Proposed UAM solutions 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. In addition, a growing number of simulation tools have been developed to study large-scale interactions of UAVs [9]–[11].

Unforeseen disruptions such as intermittent closure of landing sites due to, e.g., extreme weather conditions must be considered for any UAM solution [4]. UAVs have limited energy storage and therefore limited ability to remain in a hover or holding pattern. Therefore, a key safety constraint is to ensure that a backup landing spot is available for all in-flight UAVs. In this paper, we model a UAM network as a graph with nodes that are finite-capacity vertiports and links that are transportation links between vertiports. A key feature of our model is the allowance of uncertain travel time between vertiports represented as an interval of possible travel times. Flights depart from origin nodes at a scheduled departure time and visit one or more vertiports along a route through the UAM graph. When a vehicle arrives at a vertiport, it occupies one of a finite number of landing spots for a fixed ground service time to, e.g., offload and load passengers, refuel or recharge the battery, etc. In this framework, the defining feature of safety is that a landing spot must always be available when the UAV arrives at the vertiport. The fact that travel times are uncertain adds to the complexity of the safety problem. In [12], we considered the problem of scheduling flight departures to ensure arrival at final destinations before prescribed deadlines while ensuring safety with respect to landing capacity throughout the network, but did not consider any vertiport closures which is the focus here.

In this paper, we assume given a schedule that is *a priori* nominally safe—that is, a schedule that allows all flights to land at their intermediate nodes along the routes without landing-spot conflict despite uncertain travel times—obtained via, e.g., the methodology proposed in [12]. Given such a schedule, the goal is to ensure that it remains safe even if a vertiport closes and in-flight UAVs must be rerouted. We assume that each link in the UAM network possesses a set of backup nodes such that any flight on that link that is inbound for a closed vertiport must be safely rerouted to one of those nodes at the moment of closure with the restriction that landing capacity is not exceeded for any node within the network.

Our main contributions are as follows. First, we present necessary and sufficient conditions for ensuring safety in the event of a vertiport closure, i.e., for ensuring that all in-flight UAVs are able to land at a backup vertiport without exceeding landing spot capacity constraints. These conditions ensure safety for any realization of the link travel times, which are uncertain and only assumed to lie between known lower and upper bounds. We therefore refer to these conditions as worst-case safety guarantees. Second, we present an efficient

Qinshuang Wei and Samuel Coogan are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, 30332, USA. {qinshuang, sam.coogan}@gatech.edu. S. Coogan is also with the School of Civil and Environmental Engineering, Georgia Institute of Technology.

Gustav Nilsson is with the School of Architecture, Civil and Environmental Engineering, École Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland. gustav.nilsson@epfl.ch

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.

Some preliminary results presented in this paper have been presented at NecSys 22 [1].

algorithm for checking whether a schedule satisfies the theoretical necessary and sufficient conditions for worst-case safety. This algorithm leverages the theory of totally unimodular matrices to losslessly convert a mixed integer program into a linear program, enabling scalability to schedules with large numbers of UAVs. In particular, the proposed algorithm scales quadratically with the number of scheduled flights. Third, we present necessary and sufficient conditions for safety under some realization of the travel times. We refer to this as best-case safety. These conditions, for example, could help a UAM operator determine if a schedule could be rendered safe by reducing travel time uncertainty. Fourth, we extend our results to the scenario of multiple vertiports being disabled simultaneously. This extension requires an additional mild assumption on the assignment of backup nodes. We demonstrate our results with several examples. This paper extends our prior work in [1] which only allowed for one backup node for each link in the network. Extending to multiple backup nodes is a significant generalization requiring the theory of totally unimodular matrices for an efficient algorithm that allows for checking a much larger class of safe schedules.

Safety of UAM scheduling has been explored in prior work such as [13], which presents a risk assessment framework to provide real-time safety evaluation where the risk of off-nominal conditions in a UAV is assessed by calculating the potential impact area and the effects of the impact to people on the ground.

In ground transportation settings, most of the disruptions in the network can be modeled as capacity reductions, where totally disabled roads have zero capacity. The challenge is then to reroute the vehicle flows to ensure resilient operation of the network, where the flows are often assumed to be continuous quantities in the network [14], [15].

In this regard, our analysis is closer to classical airspace operation, where disruptions have previously been modeled and investigated to enable efficient recovery plans after the perturbations. Much of the existing literature focuses on generating a new recovery schedule [16]–[22], rerouting aircrafts [23]–[28], or are integrated with recovering crew schedules [29]–[34] while minimizing a cost related to deviation to original schedules, available resources, and other system constraints. Other literature considers airport closures as disruptions [21], [22], [28]. However, these works do not consider the capacity constraints of the airports, as needed here for the vertiports. Moreover, the present paper views the scheduling problem as a hard safety constraint rather than from the perspective of efficient operation.

The remainder of the paper is organized as follows: In Section II, we first define the UAM network model followed by the disruption model that reduces the capacity of the network. We then establish safety criteria and develop necessary and sufficient conditions for a schedule to be safe under disruptions in Section III. We then develop an efficient algorithm to check that a schedule satisfies these conditions using the theory of totally unimodular matrices. In Section V, we demonstrate our safety verification algorithm on a UAM network. The paper is concluded with some ideas for future work.

II. Problem Formulation

A. 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 $C = \{C_v\}_{v\in\mathcal{V}}$.

For any link $e = (v_1, v_2) \in \mathcal{E}$, we denote $\sigma_e = v_2$ (resp., $\tau_e = v_1$) as the head (resp., tail) of e. Let $S \subseteq \mathcal{V}$ (resp., $T \subseteq \mathcal{V}$) be the set of source (resp., terminal) nodes that are not the head (resp., tail) of any link, $S = \{v \in \mathcal{V} \mid \sigma_e \neq v \; \forall e \in \mathcal{E}\}$ 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 path in the graph $\mathcal{G}$ from a source node to a terminal node, i.e., $R = (e_1, e_2, \ldots, e_{k_R})$ is a sequence of links such that there exists a sequence of vertices $(v_0, v_2, \ldots, v_{k_R})$ with $e_\ell = (v_{\ell-1}, v_\ell)$ for all $\ell \in \{1, \ldots, k_R\}$, $v_0 \in S$, and $v_{k_R} \in T$ where k_R is the number of links in route R. We denote the set of nodes that R travels through as $V(R)$, i.e., $V(R) = \{v_0, v_2, \ldots, v_{k_R}\}$. We also sometimes consider R as a set, e.g., $e \in R$ means e is some link in R.

To simplify subsequent notation and indexing, when the particular route R is clear, we reference link $e_\ell \in \mathcal{E}$ simply as link ℓ and, likewise, vertex v_ℓ simply as vertex ℓ. We therefore use ℓ to denote both a link and its head node along a route, i.e., $\ell = \sigma_\ell$ for all $\ell \in \{1, \ldots, k_R\}$; the intended meaning will always be clear from the context. To emphasize which route we are indexing, we occasionally use a superscript, e.g., 0^R is understood as the source node of route R. We let $\mathcal{R}$ denote the set of allowed routes through the UAV network.

Since, in reality, the travel time depends on external factors such as weather conditions or a vehicle's operational capability, 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 and minimum travel time, respectively, for the link, and let $\underline{x} = \{\underline{x}_e\}_{e\in\mathcal{E}}$ and $\overline{x} = \{\overline{x}_e\}_{e\in\mathcal{E}}$ 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 $w \geq 0$. 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, minimum and maximum link travel times, and ground service time 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 set $\mathcal{S} = \{(R_j, \delta_j)\}_{j\in\mathcal{J}}$ where $\mathcal{J}$ is a finite index set of *flights* and for each $j \in \mathcal{J}$, R_j is the route of flight j and $\delta_j \geq 0$ is the appointed departure time for the flight from the first node along the route.

Due to the limited energy storage capacity of UAVs, it is assumed that a UAV must be able to land immediately upon arrival at any node along its route. For flight j with schedule (R_j, δ_j), for any link ℓ along route R_j, the earliest (resp., latest) arrival time at the ℓ-th node along the route is denoted $\underline{a}^j_\ell$ (resp., $\overline{a}^j_\ell$) and given by

$$\underline{a}^j_\ell = \delta_j + \sum_{m \in R_j : m \le \ell} \underline{x}_m + (\ell - 1)w\,, \tag{1}$$

$$\overline{a}^j_\ell = \delta_j + \sum_{m \in R_j : m \le \ell} \overline{x}_m + (\ell - 1)w\,, \tag{2}$$

i.e., $\underline{a}^j_\ell$ (resp., $\overline{a}^j_\ell$) is the departure time from source node 0 of the route R_j plus the lower (resp., upper) bound of the time interval 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 $\left[\underline{a}^j_\ell,\, \overline{a}^j_\ell + w\right]$. If $v \in V(R_j)$ is the ℓ-th node along route R_j, we also use the notation $\underline{a}^j_v := \underline{a}^j_\ell$, and $\overline{a}^j_v := \overline{a}^j_\ell$.

Definition 2 (Feasible Schedule)**.** *A schedule* $\mathcal{S} = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$ *where* $\delta_j \ge 0$ *for all* $j \in \mathcal{J}$ *is a* feasible schedule *if the number of vehicles at a node never exceeds capacity, i.e., for all* $v \in \mathcal{V}$ *and all* $t \ge 0$*,*

$$\sum_{j : v \in V(R_j)} \mathbf{1}\left(t; \left[\underline{a}^j_v,\, \overline{a}^j_v + w\right]\right) \le C_v \tag{3}$$

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.*

Since the time interval $\left[\underline{a}^j_\ell,\, \overline{a}^j_\ell + w\right]$ considers lower and upper bounds on the uncertain travel time, the definition of feasibility accommodates all possible travel times satisfying these lower and upper bounds, motivating the next definition.

Definition 3 (Realization)**.** *A* realization *of a scheduled flight is a realization of the travel times such that the flight departs at the given departure time and has a fixed travel time along each link that falls within the given time interval for the link. While each realization of the same flight has the same departure time, different realizations generally have different travel times on at least one link due to uncertain travel times.*

A feasible schedule ensures that node capacity is not exceeded for any realization of scheduled flights. Every feasible schedule will by definition ensure proper operation of the UAM network under normal circumstances. Our goal in this paper is to check whether the schedule is further resilient to interruptions in the network.

B. Disruption Model

In actual operation, it is expected that unforeseen disruptions that disable a node, such as adverse weather conditions, will be common. Flights affected by the disabled node must have a rerouting plan that ensures the 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 a backup node depending on the link they are traveling through.

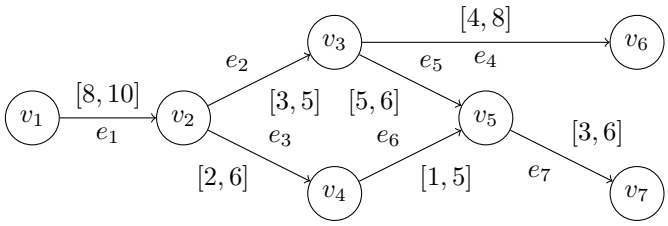

Fig. 1. A graph with 7 nodes and 7 links is used to illustrate the network in Example 1 and the case study.

In this subsection, we introduce the assignment of the backup nodes and the operating mechanism once a node is disabled. For clarity, we first consider that only one node may be disabled at a time. We extend our results to the case where multiple nodes are disabled in Section III-C. In order to guarantee that each disrupted flight will be able to be assigned to a node after the disruption, we assign a set of backup nodes $\mathcal{B}_e$ to each link e in the network. The assignment of backup nodes can be based on some rules, e.g., distances between nodes and available reserve energy of UAVs. Notice that we are not optimizing the assignment of backup nodes, however, we make a natural assumption that the set of backup nodes for any link includes its tail node and head node, i.e., $\tau_e, \sigma_e \in \mathcal{B}_e$ for any $e \in \mathcal{E}$. Then, a flight traveling on some link e whose route is potentially blocked by a node closure will continue to the head node σ_e on its route if that node is functioning, or reroute to one of its backup nodes if the head node is disabled. A similar strategy applies to the case that multiple links with the same direction exist between two nodes, and hence a UAV heading toward a destination from an origin may travel with different routes and thus with different sets of backup nodes.

A realization of 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. The realization of the j-th flight is *not affected* when node v_c is disabled at time t_c otherwise. The j'th flight is *possibly affected* by disabling node v_c at time t_c if $v_c \in V(R_j)$ and $t_c < \overline{a}^j_{v_c} + w$, i.e., the flight may have to travel through the disabled node later than t_c and hence is affected for some realization of travel times.

Below is a set of natural rules that all flights are assumed 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 $\delta_j \le t_c$ traveling on a link $e \in \mathcal{E}$ with $\sigma_e \ne v_c$ will continue to the head node σ_e and stop there indefinitely (block the landing spot indefinitely);
4) an affected flight j with $\delta_j \le t_c$ that is temporarily stopped at a node at time t_c will remain there indefinitely;
5) an affected flight j with $\delta_j \le t_c$ traveling on a link $e \in \mathcal{E}$ with $\sigma_e = v_c$ will be rerouted to one of the other backup nodes of the current link in $\mathcal{B}_e \backslash v_c$ and stop there indefinitely.

Note that we do not consider the problem of recovering a new

TABLE I
BACKUP NODES FOR EACH LINK IN EXAMPLE 1

Link ($e \in \mathcal{E}$)	$\mathcal{B}_e$	Possible backup nodes when v_5 is disabled
$e_1 = (v_1, v_2)$	$\{v_1, v_2\}$	v_2
$e_2 = (v_2, v_3)$	$\{v_2, v_3, v_4\}$	v_3
$e_3 = (v_2, v_4)$	$\{v_2, v_3, v_4\}$	v_4
$e_4 = (v_3, v_6)$	$\{v_3, v_5, v_6\}$	Link not affected
$e_5 = (v_3, v_5)$	$\{v_3, v_4, v_5\}$	v_3, v_4
$e_6 = (v_4, v_5)$	$\{v_3, v_4, v_5, v_7\}$	v_3, v_4, v_7
$e_7 = (v_5, v_7)$	$\{v_4, v_5, v_7\}$	Link not affected

schedule after a disabled node becomes operational again, as our focus is on safety. Further, we postulate the above rules as to provide a well-defined problem formulation; alternative rules might be also plausible.

Example 1. *Consider Fig. 1 with 7 nodes and 7 links,* $\mathcal{V} = \{v_1, v_2, \ldots, v_7\}$ *and* $\mathcal{E} = \{e_1 = (v_1, v_2), e_2 = (v_2, v_3), e_3 = (v_2, v_4), e_4 = (v_3, v_6), e_5 = (v_3, v_5), e_6 = (v_4, v_5), e_7 = (v_5, v_7)\}$. *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 a capacity constraint, while* $C_{v_2} = 8$, $C_{v_3} = 6$, $C_{v_4} = 4$, $C_{v_5} = 5$, $C_{v_6} = 3$ *and* $C_{v_7} = 5$. *The links are indicated in the figure and the corresponding travel time intervals are labeled beside the links, e.g., the interval* $[8, 10]$ *above the link* e_1 *means that the shortest (resp., longest) possible time for traveling through the link is* 8 *(resp.,* 10*) time units. We consider three routes* $\mathcal{R} = \{R^1, R^2, R^3\}$ *with* $R^1 = \{e_1, e_2, e_4\}$, $R^2 = \{e_1, e_2, e_5, e_7\}$ *and* $R^3 = \{e_1, e_3, e_6, e_7\}$. *Each UAV remains at the vertistops along its path for* $w = 1$ *time unit after landing.*

Table I shows an example of the assignment of backup nodes for each link $e \in \mathcal{E}$ *in the second column and the node (or nodes) that the UAV on the link can be rerouted to if node* v_5 *is disabled according to our rules in the third column. Notice that a link* e *is affected by the closure if there exists a route that traverses* e *before reaching* v_c. *For example, although link* e_1 *is not directly affected by the closure of node* v_5, *some flights using this link have a route that passes through* v_5 *and will therefore land at the head node* v_2 *and remain there due to the closure of* v_5. *A similar explanation holds for rows 2 and 3 of the table. Flights traveling on links* e_4 *and* (v_5, v_7) *are not affected if* v_5 *fails, hence the corresponding entries in the third column are empty. Lastly, flights traveling on links* e_5 *and* e_6 *must instead route to one of the backup nodes as indicated.*

III. NECESSARY AND SUFFICIENT CONDITIONS FOR SAFE SCHEDULES

In this section, we formally define safety and present sufficient and necessary conditions for verification of safety under different criteria.

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 $\mathcal{S} = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$, the closure of a node can affect the schedule in different ways. In particular, the schedule is:

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.

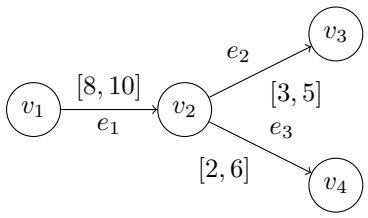

Fig. 2. A graph consisting of 4 nodes and 3 links is used to illustrate the simple network in Example 2.

Example 2. *We illustrate the safety criteria through the simple network shown in Fig. 2 with* 4 *nodes and* 3 *links. For this example, the set of nodes* $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$ *and the set of links* $\mathcal{E} = \{e_1, e_2, e_3\}$. *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. We consider two possible routes* $R^1 = \{e_1, e_2\}$ *and* $R^2 = \{e_1, e_3\}$. *Each flight remains at the intermediate nodes or destination along its path for* $w = 1$ *time unit after landing. The backup nodes for each link are* $\mathcal{B}_{e_1} = \{v_1, v_2\}$, $\mathcal{B}_{e_2} = \{v_2, v_3, v_4\}$ *and* $\mathcal{B}_{e_3} = \{v_2, v_3, v_4\}$. *Consider 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 possibilities for* δ_3. *Assume* v_4 *is disabled at time* $t_c = 15$. *Based on the rerouting rules for the flights, then at time* t_c, *a flight traveling on link* e_1 *will be rerouted to node* v_2, *while a flight traveling on link* e_3 *can be rerouted to either* v_2 *or* v_3. *Though it is possible that flight* S_1 *has already completed its journey by time* $t_c = 15$, *in the worst case where we consider any link that it may be traveling on, it is possible for flight* S_1 *to be traveling on* e_3 *and needs to be rerouted to* v_2 *or* v_3, *so that we have to reserve a landing spot at node* v_2 *or* v_3 *for* S_1; *flight* S_2 *must be traveling on* e_1 *and needs to stay at* v_2 *upon arrival. If* $\delta_3 = 0$, *then flight* S_3 *is not affected and should continue its journey; however, if* $\delta_3 = 10$, *then either* v_2 *or* v_3 *will have insufficient landing spots, since the flight* S_1 *must have been rerouted to either* v_2 *or* v_3 *upon the arrival of* S_3. *Therefore,* $\mathcal{S}$ *is worst-case time-node conditionally safe for node* v_4 *at time* 15 *if and only if* $\delta_3 \leq 4$. *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.

Now, suppose $\delta_3 = 10$ *and* $C_{v_2} = 3$. *Then there is sufficient capacity so that the network will be able to accommodate all rerouted flights after closure no matter when 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. In contrast, suppose $C_{v_3} = 2$ while $C_{v_2} = 2$, then $\mathcal{S}$ is worst-case time-node conditionally safe for node v_4 and $t_c = 15$ with any choice of δ_3, but is worst-case node-conditionally safe for v_4 if and only if $\delta_3 \geq 4$. □

To obtain constraints for 1-closure safety, we start by observing that a feasible schedule is trivially node conditionally safe for any node $v \in S$, where we recall S the set of source nodes that are not the head of any link. Whenever a node $v \in S$ is disabled, there will not be any UAV traveling toward node v while no future journey will depart from v.

We next explore the safety of a disabled node that is not a source node. There are several special sets we now define before presenting conditions for 1-closure safety when disabling a node $v_c \in \mathcal{V}\backslash S$. 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 we let $\mathcal{B}'_e$ be the set of nodes that any flight traveling on link e can be rerouted to if v_c is disabled:

$$\mathcal{B}'_e = \begin{cases} \{\sigma_e\} & \text{if } \sigma_e \neq v_c, \\ \mathcal{B}_e \backslash v_c & \text{if } \sigma_e = v_c. \end{cases} \tag{5}$$

$\mathcal{B}'_e$ is the set of *possible backup nodes* for link e when v_c is disabled. We then denote b_e as the node that a flight traveling on link e will be rerouted to if v_c is disabled, so that $b_e \in \mathcal{B}'_e$.

We denote the set of links on which flights will possibly be rerouted to node v when v_c is disabled as $\mathcal{B}_v$, which includes the links with the head node as v when $v \neq v_c$ and the links with the head node as v_c whose backup nodes include v, i.e.,

$$\mathcal{B}_v := \{e \in \mathcal{E} \mid v \in \mathcal{B}'_e\}. \tag{6}$$

If node v_c is along the route R_j of flight j, we define the set $\mathcal{E}^j_{v_c}$ as the links along the route of flight j whose head node is one of the backup nodes of link ℓ_0 where $\ell_0 \in \{1, \ldots, k_{R_j}\}$ is the link along route R_j satisfying $\sigma_{\ell_0} = v_c$, i.e.,

$$\mathcal{E}^j_{v_c} = \{\ell \mid \ell < \ell_0, \sigma_{\ell_0} = v_c, \sigma_\ell \in \mathcal{B}'_{\ell_0}\}. \tag{7}$$

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

$$\mathcal{J}_v := \{j \in \mathcal{J} \mid v \in V(R_j)\}, \tag{8}$$

and we further define the index set of the possibly affected flights when node v_c is closed at time t_c as

$$\mathcal{J}^* = \mathcal{J}^*(v_c, t_c) := \{j \in \mathcal{J}_{v_c} \mid \overline{a}^j_{v_c} + w > t_c\}, \tag{9}$$

where we drop arguments (e.g., v_c and t_c), when they are clear. Therefore, the index set for the flights passing through node v that are not possibly affected when node v_c is closed at t_c is $\mathcal{J}_p(v) := \mathcal{J}_v \backslash \mathcal{J}^*$.

Similarly, we use $\mathcal{J}^c$ to represent the set of indices for canceled journeys with departure time greater than the node-disabling time t_c:

$$\mathcal{J}^c = \mathcal{J}^c(v_c, t_c) := \{j \in \mathcal{J}_{v_c} \mid \delta_j > t_c\}. \tag{10}$$

We then define the index set of rerouting flights $\mathcal{J}^{*\backslash c}$ as the possibly affected flights not canceled when node v_c is disabled at time t_c, i.e., $\mathcal{J}^{*\backslash c} := \mathcal{J}^* \backslash \mathcal{J}^c$. We let $N_R(v)$ be the maximal number of not possibly affected flights that may land at node v at the same time once node v_c is disabled at time t_c, which can be computed as

$$N_R(v) = \sup_{t \geq t_c} \sum_{j \in \mathcal{J}_p(v)} \mathbf{1}\left(t; \left[\underline{a}^j_v, \overline{a}^j_v + w\right]\right). \tag{11}$$

All of the above components (4)–(11) are easily computed from a given feasible schedule.

A. Necessary and Sufficient Condition for Worst-Case Safe Schedules

Our first main result provides necessary and sufficient conditions for worst-case safe schedules in the form of a Mixed Integer Linear Program (MILP). MILPs are generally sensitive to scale and can be time-consuming once the size of the schedule under verification grows. In Section IV, we will losslessly recast this MILP as a linear program (LP), leading to an efficient safety-verification algorithm.

Theorem 1. *Consider a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \overline{x}, w)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and given backup nodes assignment $\mathcal{B}_e$ for all $e \in \mathcal{E}$. Assume given a feasible schedule $\mathcal{S} = \{(R_j, \delta_j)\}_{j \in \mathcal{J}}$.*

The schedule $\mathcal{S}$ is worst-case time-node conditionally safe for node v_c and time t_c if and only if there exists an integer set $\{N_{e,v}\}_{e \in \mathcal{E}, v \in \mathcal{V}}$ that satisfies the following constraints for all $v \in \mathcal{V}$ and $e \in \mathcal{E}$:

$$\sum_{e \in \mathcal{B}_v} N_{e,v} \leq C_v - N_R(v), \quad \forall v \in \mathcal{V}, \tag{12}$$

$$\sum_{v \in \mathcal{B}'_e} N_{e,v} = \sum_{j \in \mathcal{J}^{*\backslash c}} \mathbf{1}\Big(t_c; [L^j_e, U^j_e] \backslash \{\cup_{\ell \in \mathcal{E}^j_{v_c}} [L^j_{\ell^{R_j}}, U^j_{\ell^{R_j}}]\}\Big), \quad \forall e \in \mathcal{E}_{v_c}, \tag{13}$$

$$N_{e,\sigma_e} = \sum_{j \in \mathcal{J}^{*\backslash c}} \mathbf{1}(t_c; [L^j_e, U^j_e]), \quad \forall e \notin \mathcal{E}_{v_c}, \tag{14}$$

$$N_{e,v} = 0, \quad \forall e \in \mathcal{E}_{v_c}, v \notin \mathcal{B}'_e, \tag{15}$$

$$N_{e,v} \geq 0, \quad \forall e \in \mathcal{E}_{v_c}, v \in \mathcal{B}'_e, \tag{16}$$

where for all $j \in \mathcal{J}$, the lower and upper bounds of the time interval are defined as

$$L^j_e = \begin{cases} \underline{a}^j_{\tau_e} + w & \text{if } \tau_e \neq 0^{R_j} \\ \delta_j & \text{if } \tau_e = 0^{R_j}, \end{cases} \tag{17}$$

and

$$U^j_e = \begin{cases} \overline{a}^j_{\sigma_e} + w & \text{if } \sigma_e \neq v_c \\ \overline{a}^j_{\sigma_e} & \text{if } \sigma_e = v_c. \end{cases} \tag{18}$$

Further, $\mathcal{S}$ is worst-case node-conditionally safe for node v_c if and only if such a set $\{N_{e,v}\}_{e \in \mathcal{E}, v \in \mathcal{V}}$ satisfying (12)–(16) exists for the finite number of times t_c where the values of $N_R(v)$ and the time-varying index sets $\mathcal{J}^*$, $\mathcal{J}^c$ possibly change, i.e., at both endpoints of the interval $\mathcal{M}^j_v$ for all $v \in \mathcal{V}$, at times δ_j for all $j \in \mathcal{J}$, and at times L^j_e, U^j_e for all $j \in \mathcal{J}$ and $e \in \mathcal{E}$.

The second part of Theorem 1 states that, while the definition for a schedule to be worst-case node-conditionally safe requires checking safety for all times $t_c \geq 0$, such conditions in fact only need to be checked at a finite number of times.

Proof. The schedule is worst-case time-node conditionally safe for node v_c and time t_c if and only if, for any possibly affected flight that is not canceled and may be rerouted to some node in $\mathcal{V}$ at time t_c, an available landing spot needs to be reserved. Hence 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 in $\mathcal{V}$.

We let the set $\{N_{e,v}\}_{e\in\mathcal{E},v\in\mathcal{V}}$ be the set of variables that denote the number of possibly affected flights that may proceed to node $v \in \mathcal{V}$ when traveling on the link $e \in \mathcal{E}$. Hence, for all $v \in \mathcal{V}, e \in \mathcal{E}$, $N_{e,v}$ is required to be a non-negative integer. The interval defined as $[L_e^j, U_e^j]$ is the time interval during which flight j will possibly be rerouted to b_e if v_c is closed, where the lower bound L_e^j is the earliest time that the flight may leave the previous node τ_e, and if σ_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 σ_e is disabled, the upper bound U_e^j for the time interval that the flight may be rerouted to the backup node τ_e will be the latest time that the corresponding flight may arrive at node v_c, since otherwise it will continue its normal operation without rerouting. For any $e \in \mathcal{E}$, if $\sigma_e \neq v_c$, then the possibly affected flights traveling on the link will land at its head node σ_e, and thus the number of possibly affected flights rerouting to node σ_e from link e, N_{e,σ_e} is deterministic, which can be simply counted as in (14). The constraint (15) prevents flights from proceeding to any node v not in the set of possible backup nodes for link e when node v_c is disabled, $\mathcal{B}'_e$.

As a safety requirement, when v_c is disabled at time t_c, any possibly affected flight needs to be rerouted to a node. Consider a fixed $e \in \mathcal{E}_{v_c}$, a flight whose possibly traveling on this link at time t_c is obviously a possibly affected flight when node v_c is disabled at time t_c and needs to be rerouted to one of its backup nodes. Therefore, (13) is the link safety constraint depicting that all flights possibly traveling on e at t_c need to be rerouted to one of the possible backup nodes for link e when v_c is disabled. Notice that, supposing the backup nodes of the link e include a node $v' \leq v_c$ that is along the route of the flight, and the flight is also possibly traveling on a link whose head node is v' at t_c, then this means a landing spot at node v' has to be reserved, and we do not need to prepare another one. This situation is reflected through $\{\cup_{\ell\in\mathcal{E}^j_{v_c}}[L^j_{\ell^{R_j}}, U^j_{\ell^{R_j}}]\}$ in (13). Finally, $N_R(v)$ is the maximum number of flights not possibly affected that may park at node v at any time once v_c is disabled at t_c, and the summation $\sum_{e\in\mathcal{B}_v} N_{e,v}$ is the total number of possibly affected flights rerouting to node v. Therefore, (12) is a necessary and sufficient condition to avoid the capacity conflict between the rerouted flights and those surely not affected by all realization of link travel times. □

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 worst-case node conditionally safe for all $v_c \in \mathcal{V}$, we can conclude the 1-closure safety.

In the following subsection, we explore the safety constraints for a given UAM schedule in the best-case scenario.

B. Necessary and Sufficient Condition for Best-Case Safe Schedules

Theorem 1 provides a set of constraints that serve as a necessary and sufficient condition for a feasible schedule to be worst-case time-node conditionally, node conditionally, or 1-closure safe. In this subsection, we provide constraints for a feasible schedule to be best-case safe. In the best-case scenario, we consider the realization with the least number of rerouting flights and the most flexible rerouting plan needed among all possible realizations. Therefore, we assume that all flights possible to have arrived at or passed through the closed node v_c have already arrived or left by the time of node failure.

We denote the index set of the *definitely affected* flights as

$$\mathcal{J}^m = \{j \in \mathcal{J}^* \backslash \mathcal{J}^c \mid \underline{a}^j_{v_c} \geq t_c\}\,. \tag{19}$$

The definitely affected flights are the flights that must be rerouted under any possible realization.

Theorem 2. *Consider a network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \overline{x}, w)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and given backup nodes assignment $\mathcal{B}_e$ for all $e \in \mathcal{E}$. A given 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 if and only if there exists a non-negative integer set $\{N_{j,v}\}_{j\in\mathcal{J}^m,v\in\mathcal{V}}$ that satisfies the following constraints for all $j \in \mathcal{J}^m$ and $v \in \mathcal{V}$:*

$$C_v - \sum_{j\in\mathcal{J}^m} N_{j,v} \geq N_R(v)\,, \quad \forall v \in \mathcal{V}\,, \tag{20}$$

$$\sum_{v\in\mathcal{V}} N_{j,v} = 1\,, \quad \forall j \in \mathcal{J}^m\,, \tag{21}$$

$$0 \leq N_{j,v} \leq \max_{e\in R_j} \mathbf{1}(t_c; [L^j_e, \hat{U}^j_e]) \cdot \mathbf{1}(v; \mathcal{B}'_e)\,, \quad \forall v \in \mathcal{V}\,, j \in \mathcal{J}^m\,, \tag{22}$$

where L_e^j is defined in (17) *and*

$$\hat{U}^j_e = \begin{cases} \overline{a}^j_{\sigma_e} + w & \text{if } \sigma_e \neq v_c\,, \\ \underline{a}^j_{\sigma_e} & \text{if } \sigma_e = v_c\,. \end{cases} \tag{23}$$

Further, $\mathcal{S}$ is best node-conditionally safe for node v_c if and only if the set $\{N_{j,v}\}_{j\in\mathcal{J}^m,v\in\mathcal{V}}$ that satisfies (20)–(22) *exists and the conditions holds for the finite number of times t_c where the values of $N_R(v)$ and the time-varying index set $\mathcal{J}^m$ possibly change, i.e., at both endpoints of the interval $\left[\underline{a}^j_v, \overline{a}^j_v + w\right]$ for all $v \in \mathcal{V}$, δ_j for all $j \in \mathcal{J}$ and at times L^j_e, $\hat{U}^j_e$ for all $j \in \mathcal{J}$ and $e \in \mathcal{E}$.*

Proof. The proof of Theorem 2 applies the similar logic as in Theorem 1 to the best-case scenario, while from the perspective of flights instead of the links. First of all, we can focus only on the definitely affected flights, since any flight that is possibly affected but not definitely affected is either canceled or has at least a realization of travel time such that the flight has already passed through or landed at node v_c and does not need to be rerouted.

We regard $N_{j,v}$ as the indicator of j'th flight to be rerouted to node v if node v_c is disabled at time t_c, for all $j \in \mathcal{J}$ and $v \in \mathcal{V}$. As a result, the non-negative variable $N_{j,v}$ is actually binary. We enforce this binary condition in (22), where $N_{j,v} \leq 1$ if there exists $e \in R_j$ such that v is one of its possibly backup nodes when v_c is closed and the flight is definitely affected and possibly traveling on the link e at time t_c and $N_{j,v} = 0$ otherwise. Notice that the upper-bound of the time interval for the flight to travel through link e and its head node and be rerouted to one of its possible backup nodes, $\hat{U}_e^j$, is adjusted comparing to U_e^j defined in (18) to include only the definitely affected flights. Once node v_c is disabled at time t_c, a flight will actually be rerouted to exactly one node, which is depicted in (21). Moreover, (20) is the capacity constraint, where $\sum_{j\in\mathcal{J}^m} N_{j,v}$ is the number of definitely affected flights rerouted to node v.

If we are not able to find a set of non-negative integers $\{N_{j,v}\}_{j\in\mathcal{J}^m, v\in\mathcal{V}}$ that satisfies (20)–(22), then there must exist a conflict of occupation at one or more nodes once v_c is closed at time t_c, and hence (20)–(22) are sufficient and 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 sufficient and 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.

C. Extension to Multiple Node Closures

In this subsection, we consider the possibility that multiple vertiports are disabled simultaneously. Our main result is to show that, with slight modifications and a mild additional assumption, the conclusions of Theorem 1 and 2 remain true.

We first revise the definitions of some sets to suit the multiple-closure case. As before, consider a UAM network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \overline{x}, w)$ where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, given backup node assignment $\mathcal{B}_e$ for all $e \in \mathcal{E}$, and schedule $\mathcal{S} = \{(R_j, \delta_j)\}_{j\in\mathcal{J}}$. Now, consider a set of nodes $V_c \subseteq \mathcal{V}$ disabled simultaneously. Paralleling the prior notation, define

$$\mathcal{E}_{V_c} := \{e \in \mathcal{E} \mid \sigma_e \in V_c\}, \tag{24}$$

$$\mathcal{B}'_e := \begin{cases} \{\sigma_e\} & \text{if } \sigma_e \notin V_c \\ \mathcal{B}_e \backslash V_c & \text{if } \sigma_e \in V_c\,, \end{cases} \tag{25}$$

$$\mathcal{J}_{V_c} := \{j \in \mathcal{J} \mid V_c \cap V(R_j) \neq \emptyset\}\,. \tag{26}$$

For any flight $j \in \mathcal{J}$ and for all $v_c \in V_c$, let $\mathcal{E}^j_{v_c} = \emptyset$ if $v_c \notin V(R_j)$, i.e., v_c is not along the route of flight j, and otherwise, let

$$\begin{aligned}\mathcal{E}^j_{v_c} :=& \{\ell \mid \ell < \ell_0 \text{ and } \mathcal{B}'_\ell \subseteq \mathcal{B}'_{\ell_0}\} \\ &\cup \{\ell \mid \ell_0 < \ell < \overline{\ell}, \ell \notin V_c \text{ and } \sigma_\ell \in \mathcal{B}'_{\ell_0}\}\end{aligned} \tag{27}$$

where ℓ_0 is the index of the link along route R_j with head node v_c, i.e., $\sigma_{\ell_0} = v_c$, and $\overline{\ell}$ is the largest index of the link along route R_j whose head node is disabled, i.e., $\sigma_{\overline{\ell}} = \max\{\ell \in R_j \mid \sigma_\ell \in V_c\}$. All the other related sets and variables—such as $\mathcal{J}^*$, $\mathcal{J}^c$, and $N_R(v)$—then change subsequently.

We then analogously define a schedule as

1) *worst-case time-node conditionally safe* for the set of nodes $V_c \subseteq \mathcal{V}$ and time t_c if, supposing that the nodes in V_c are disabled at time t_c, then all possibly affected flights $\mathcal{J}_{V_c}$ are able to land at their designated backup nodes while not interfering with any unaffected flights for all realization of link travel times.
2) *worst-case node conditionally safe* for V_c if it is worst-case time-node conditionally safe for the set of nodes V_c for all time $t_c > 0$.

We make Assumption 1 below about the network topology and travel-time uncertainty before extending Theorem 1 to be compatible with multiple simultaneous closures of vertiports. Assumption 1 below states that if, due to uncertain travel times, there is a time when a flight may be traveling on two different links both destined for possibly closed vertiports V_c, then the backup set of one link is either a subset of or is disjoint with the other's. This avoids the ambiguity of rerouting an affected flight when the link it is traveling on is not definite at the time of vertiports closure.

Assumption 1. *Let $R \in \mathcal{R}$ be any route passing through at least two nodes from the set of disabled nodes excluding the source node of R. That is, recalling that we enumerate the links in route R as $\{1, \ldots, k_R\} \subset \mathcal{E}$, there exists some pair $\ell_1, \ell_2 \in \{1, \ldots, k_R\}$ such that $\sigma_{\ell_1}, \sigma_{\ell_2} \in V_c \backslash 0^R$. Without loss of generality, take $\ell_1 < \ell_2$. For any such pair, compute*

$$L_{\ell_i} = \sum_{m\in R: m\leq \ell_i - 1} \underline{x}_m + (\ell_i - 1)w\,, \quad i \in \{1, 2\}\,, \tag{28}$$

$$U_{\ell_i} = \sum_{m\in R: m\leq \ell_i} \overline{x}_m + (\ell_i - 1)w\,, \quad i \in \{1, 2\}\,. \tag{29}$$

We assume that, whenever $[L_{\ell_1}, U_{\ell_1}] \cap [L_{\ell_2}, U_{\ell_2}] \neq \emptyset$, either $B'_{\ell_1} \subseteq \mathcal{B}'_{\ell_2}$ or $B'_{\ell_1} \cap \mathcal{B}'_{\ell_2} = \emptyset$.

We now extend Theorem 1 as follows:

Theorem 3. *Consider a UAM network $\mathcal{N} = (\mathcal{G}, C, \mathcal{R}, \underline{x}, \overline{x}, w)$ where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, given backup node assignment $\mathcal{B}_e$ for all $e \in \mathcal{E}$. Assume given a feasible schedule $\mathcal{S} = \{(R_j, \delta_j)\}_{j\in\mathcal{J}}$ and the set of disabled nodes V_c. We further assume that the network $\mathcal{N}$ satisfies the Assumption 1.*

The schedule $\mathcal{S}$ is worst-case time-node conditionally safe for the set of nodes V_c and time t_c if and only if there exists an integer set $\{N_{e,v}\}_{e\in\mathcal{E}, v\in\mathcal{V}}$ that satisfies the following constraints for all $v \in \mathcal{V}$ and $e \in \mathcal{E}$:

$$\sum_{e\in\mathcal{B}_v} N_{e,v} \leq C_v - N_R(v)\,, \quad \forall v \in \mathcal{V}\,, \tag{30}$$

$$\sum_{v\in\mathcal{B}'_e} N_{e,v} = \sum_{j\in\mathcal{J}^*\backslash c} \mathbf{1}\Big(t_c; [L^j_e, U^j_e] \backslash \{\cup_{\ell\in\mathcal{E}^j_{\sigma_e}} [L^j_{\ell^{R_j}}, U^j_{\ell^{R_j}}]\}\Big)\,, \quad \forall e \in \mathcal{E}_{V_c}\,, \tag{31}$$

$$N_{e,\sigma_e}(t_c) = \sum_{j\in\mathcal{J}^*\backslash c} \mathbf{1}(t_c; [L^j_e, U^j_e])\,, \forall e \notin \mathcal{E}_{V_c}\,, \tag{32}$$

$$N_{e,v} = 0\,, \quad \forall e \in \mathcal{E}_{V_c}, v \notin \mathcal{B}'_e\,, \tag{33}$$

$$N_{e,v} \geq 0\,, \quad \forall e \in \mathcal{E}_{V_c}, v \in \mathcal{B}'_e\,. \tag{34}$$

Further, $\mathcal{S}$ is worst-case node-conditionally safe for the set of nodes V_c if and only if such a set $\{N_{e,v}\}_{e\in\mathcal{E},v\in\mathcal{V}}$ satisfying (30)–(34) *exists for the finite number of times t_c where the values of $N_R(v)$ and the time-varying index sets $\mathcal{J}^*$, $\mathcal{J}^c$ possibly change, i.e., at both endpoints of the interval $\mathcal{M}_v^j$ for all $v\in\mathcal{V}$, at time δ_j for all $j\in\mathcal{J}$, and at times L_e^j, U_e^j for all $j\in\mathcal{J}$ and $e\in\mathcal{E}$.*

A similar extension of the conditions for best-case safety applies as well, and the proofs are essentially the same as the proofs of Theorem 1 and 2. We observe that Assumption 1 together with (31) guarantees that, if a flight j is possibly traveling on link e whose head node is in V_c and is disabled at time $t_c\in\cup_{\ell\in\mathcal{E}^j_{\sigma_e}}[L^j_{\ell^{R_j}},U^j_{\ell^{R_j}}]$, we do not double-count the number of landing sites needed at the backup nodes.

IV. Simplification for Verification

The necessary and sufficient conditions for safety derived in Section III involve integer constraints and therefore are inefficient for use in direct numerical implementation. In this section, we show that these conditions can in fact be translated to efficient linear programming (LP) constraints. We first establish a lemma explaining the mathematical foundation for our simplification of Theorem 1 and 2, followed by a theorem that turns the MILP problem in Theorem 1 and 2 into an LP problem. In particular, the following lemma shows that, for a special set of constraints on a set of variables, the existence of a solution over the real numbers induces the existence of a solution over the integers.

Lemma 1. *Given a set $L=\{(l_1,l_2)\in\mathbb{N}^2_{>0}\mid l_1\le N_1, l_2\le N_2\}$ for some positive integers N_1,N_2, and let L_{var} be a subset of L. Let α_{l_1} (resp., β_{l_2}) be non-negative integers for all $l_1=1,\dots,N_1$ (resp., $l_2=1,\dots,N_2$), and γ_{l_1,l_2} be integers for all $(l_1,l_2)\in L_{var}$. If there exists a set of real numbers $\{n_{l_1,l_2}\}_{(l_1,l_2)\in L}$ that satisfies*

$$\sum_{l_2=1}^{N_2} n_{l_1,l_2}=\alpha_{l_1}\,,\qquad \forall l_1=1,\dots,N_1\,, \tag{35}$$

$$\sum_{l_1=1}^{N_1} n_{l_1,l_2}\le\beta_{l_2}\,,\qquad \forall l_2=1,\dots,N_2\,, \tag{36}$$

$$n_{l_1,l_2}=\gamma_{l_1,l_2}\,,\qquad \forall (l_1,l_2)\in L_{var}\,, \tag{37}$$

$$n_{l_1,l_2}\ge 0\,,\qquad \forall (l_1,l_2)\in L\,, \tag{38}$$

then there exists a set of integers $\{n'_{l_1,l_2}\}_{(l_1,l_2)\in L}$ also satisfying (35)–(38).

The proof for Lemma 1 can be found in Appendix A. Further, the remark below can be shown with some trivial revisions to the proof.

Remark 1. *Lemma 1 holds if n_{l_1,l_2} is bounded from above by an integer, i.e.,* (38) *is changed to $0\le n_{l_1,l_2}\le U_{l_1,l_2}$ for all $(l_1,l_2)\in L$ for some integer number $U_{l_1,l_2}>0$.*

The simplified corollary below makes use of Lemma 1 above and provides an LP alternative to the MILP problem in Theorem 1.

Corollary 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 $\mathcal{S}=\{(R_j,\delta_j)\}_{j\in\mathcal{J}}$. There exists a set of real numbers $\{N_{e,v}\}_{e\in\mathcal{E},v\in\mathcal{V}}$ that satisfies the constraints* (12)–(16) *if and only if there exists a set of integers $\{N'_{e,v}\}_{e\in\mathcal{E},v\in\mathcal{V}}$ that satisfies the constraints.*

Proof. We first show that the conditions (12) and (13) conform to the form in Lemma 1. For the sake of convenience, we fix t_c and v_c and drop the notation from $N_{e,v}$ and $N_R(v)$, i.e., we write them as $N_{e,v}$ and $N_R(v)$ in this proof.

We can observe that, by the definition of $\mathcal{B}_v$ in (6), assume $e\notin\mathcal{B}_v$, if $\sigma_e=v_c$, then $v\notin\mathcal{B}_e\backslash v_c$, otherwise $\sigma_e\neq v_c$. We can therefore conclude from (14) and (15) that $N_{e,v}$ is a fixed number that can be computed if $e\notin\mathcal{B}_v$.

By adding the fixed terms of $N_{e,v}$ for $e\notin\mathcal{B}_{v,v_c}$ to both sides of (12) we can then obtain that, for all $v\in\mathcal{V}$,

$$\sum_{e\in\mathcal{B}_{v,v_c}} N_{e,v}\le C_v-N_R(v)$$
$$\sum_{e\in\mathcal{E}} N_{e,v}\le C_v-N_R(v)+\sum_{e\notin\mathcal{B}_{v,v_c}} N_{e,v}\,. \tag{39}$$

Similarly, if $v\notin\mathcal{B}'_e$, then $N_{e,v}$ is a fixed number. By adding $\sum_{v\notin\mathcal{B}'_e}N_{e,v}$ to both sides of (13), we have

$$\sum_{v\in\mathcal{V}} N_{e,v}=\sum_{v\notin\mathcal{B}'_e} N_{e,v}+$$
$$\sum_{j\in\mathcal{J}^{*\backslash c}}\mathbf{1}\Big(t_c;[L_e^j,U_e^j]\backslash\{\cup_{\ell\in\mathcal{E}^j_{v_c}}[L^j_{\ell^{R_j}},U^j_{\ell^{R_j}}]\}\Big)\,. \tag{40}$$

We then let $L=\{(e,v)\mid e\in\mathcal{E},v\in\mathcal{V}\}$, and $L_{var}=L\backslash\{(e,v)\mid\sigma_e=v_c,v\in\mathcal{B}'_e\}$. Then we can combine and rewrite (14)–(16) as

$$N_{e,v}=\gamma_{e,v}\,,\qquad \forall(e,v)\in L_{var}\,, \tag{41}$$

$$N_{e,v}\ge 0\,,\qquad \forall(e,v)\in L\,, \tag{42}$$

where $\gamma_{e,v}=\sum_{j\in\mathcal{J}^{*\backslash c}}\mathbf{1}(t_c;[L_e^j,U_e^j])$ if $\sigma_e=v\neq v_c$ and $\gamma_{e,v}=0$ if $\sigma_e\neq v$ and $v\notin\mathcal{B}'_e$.

As $\mathcal{E}$ and $\mathcal{V}$ are both finite sets, while the right sides of the inequalities (39) and (40) are integers, the conditions (39)–(42) exactly follows the conditions (35)–(38) in Lemma 1. Therefore, by Lemma 1, there exists a integral set $\{N_{e,v}\}_{e\in\mathcal{E},v\in\mathcal{V}}$.

As a result, given the schedule $\mathcal{S}$, if there exists a set of real numbers $\{N_{e,v}\}_{e\in\mathcal{E},v\in\mathcal{V}}$ that satisfies the constraints (12)–(16), then there exist a set of integers $\{N'_{e,v}\}_{e\in\mathcal{E},v\in\mathcal{V}}$ that satisfies the constraints. The other direction of the corollary is immediate. □

Combining Theorem 1 and Corollary 1, we are then able to verify 1-closure safety of given feasible schedules by solving an LP. Similarly, we develop a corollary for simplification of best-case safety mirroring Corollary 1 given Remark 1.

Corollary 2. *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 $\mathcal{S}=\{(R_j,\delta_j)\}_{j\in\mathcal{J}}$. There exists a set of real numbers $\{N_{j,v}\}_{j\in\mathcal{J}^m_{v_c},v\in\mathcal{V}}$ that satisfies the constraints* (20)–(22) *if and only if there exists a set of non-negative integers $\{N'_{j,v}\}_{j\in\mathcal{J}^m_{v_c},v\in\mathcal{V}}$ that satisfies the constraints.*

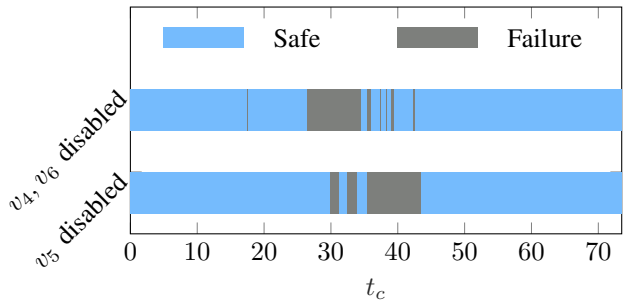

Fig. 3. Observation of whether the network is safe when node v_5 is disabled or when the set $\{v_4, v_6\}$ is disabled at any time $t_c > 0$.

The proof for Corollary 2 is immediate from Lemma 1 as we can easily convert the constraints (20)–(22) to the same form as in Lemma 1 and Remark 1.

Remark 2. *Theorem 3 in Section III-C for the multiple-closure scenario can also be simplified to an LP with the same method.*

V. Case Study

In the case study, we demonstrate the verification algorithm based on Theorem 1 and Corollary 1 on the UAM network in Example 1 with 20 scheduled flights. We also demonstrate the efficient scaling of the algorithm on examples with up to 1,000 UAVs.

To ensure the worst-case node conditional safety when v_c is closed, we check whether there exists a set of real numbers $\{N_{e,v}\}_{e\in\mathcal{E},v\in\mathcal{V}}$ that satisfies the constraints (12)–(16) over the time interval $t_c \in [0, +\infty)$ As stated in Theorem 1, we only need to solve the LP feasibility problem at each point of time that any value may change, i.e., L_e^j, U_e^j, both ends of $\mathcal{M}_v^j$, and δ_j for any counted flight $j \in \mathcal{J}$ and link $e \in \mathcal{E}$ for some fixed node v, since the system of linear inequalities (12)–(16) will not change between these points. We randomly generate a feasible schedule with 20 flights and consider the constraints (12)–(16) in Theorem 1 for time-node conditionally safe for node $v_c = v_5$ at any time $t_c > 0$. We also check the worst-case safety when the set of nodes $V_c = \{v_4, v_6\}$ is disabled at some time. The verifications are implemented in MATLAB[1].

We observe when the schedule is safe in Fig. 3 (first stripe) and the redistribution of flights on link e_5 and e_6 in Fig. 4 when node v_5 is disabled at any time t_c. The first stripe of Fig. 3 demonstrates whether disabling node v_5 at time t_c is worst-case time-node conditionally safe (the blue segments) or not (the grey segments). For example, if node v_5 is disabled at time $t_c = 40$, the network is not worst-case safe.

The redistribution of flights on link e_5 (resp., e_6) shown in the top (resp., bottom) graph of Fig. 4 when node v_5 is disabled at any time t_c provide a detailed partition of flights onto the nodes to which they are rerouted. Since the head of the link e_5 (resp., e_6), v_5, is disabled, the flights traveling on the link need to be rerouted to one of the possible backup nodes, v_3 or v_4 (resp., v_3, v_4, or v_7). We use orange and purple (resp., green, red, and blue) rectangles to represent N_{e_5,v_3} and N_{e_5,v_4} (resp., N_{e_6,v_3}, N_{e_6,v_4}, and N_{e_6,v_7}), i.e., the number of affected flights traveling on link e_5 (resp., e_6) rerouted to the backup nodes v_3 and v_5 (resp., v_3, v_4, and v_7). We use grey rectangles to indicate the failure of obtaining the solution to the LP problem (12)–(16). The height of the grey rectangles represents the total number of flights that need to be rerouted from link e_5 (resp., e_6) when node v_5 is disabled at time t_c, i.e., $\sum_{v\in\mathcal{B}'_{e_5}} N_{e_5,v}$. Notice that the solution to the LP problem $(N_{\cdot,\cdot})$ for $t_c > 0$, if it exists, is not unique, and hence Fig. 4 is only one possible rerouting arrangement. Thus, as an example, the schedule is not time-node conditionally safe for node v_5 at $t_c = 40$, as the grey rectangle indicates there does not exist a solution to the problem (12)–(16) at time t_c. Therefore, the network is not able to accommodate the failure of v_5 at time $t_c = 40$.

For the sake of comparison, we then increase the capacity of v_4 to $C_{v_4} = 8$ while the other parts of the network remain the same. We then verify the safety of the same schedule with the algorithm, and these results are shown in Fig. 5.

In Fig. 5 (top), the blue rectangles correspond to flights that are not affected and continue to v_3 if node v_5 is disabled at time t_c, which is $N_R(v_3)$ in (12); the pink rectangles correspond to flights rerouted to node v_3 from link $e_2 = (v_2, v_3)$ if v_5 is disabled at time t_c, which is N_{e_2,v_3}; the orange (resp., green) rectangles correspond to the number of flights rerouted to node v_3 from link $e_5 = (v_3, v_5)$ (resp., $e_6 = (v_4, v_5)$) if node v_5 is disabled at time t_c, which is N_{e_5,v_3} (resp., N_{e_6,v_3}). Notice that N_{e_2,v_3} is fixed and can be computed by (14), since any flight traveling on e_2 at time t_c has to land at v_3 if v_5 is disabled at that time; meanwhile, N_{e_5,v_3} (resp., N_{e_6,v_3}) is an optimization variable computed through the LP problem (12)–(16). As a reference, the capacity $C_{v_3} = 6$ is shown as the dotted, horizontal line so that the height of the entire bar (the sum of all rectangles) must not exceed the capacity for safety. As shown in the plots, after increasing the capacity of v_4, which is a backup node for both e_5 and e_6, the solution to the problem (12)–(16) exists all the time. To conclude if the schedule is node conditionally safe when v_5 is disabled, we would need to observe all affected nodes and links in the network in the same way.

Further, as shown in the second stripe of Fig. 3, we verify when the schedule is safe if the set of nodes $V_c = \{v_4, v_6\}$ is disabled. We observe that the schedule is not worst-case safe if the disabling time t_c falls within much of the window $[25, 35]$ and a few other periods such as $t_c = 17$, $37, 39$, and 42.

The computation time for $N_R(v)$ in (12) increases quadratically with the size of the schedule, and as indicated in [35], solving the LP problem (12)–(16) with a fixed number of variables can be computed within linear time with respect to the number of constraints, while the number of constraints in the LP problem and the number of times the LP needs to be solved both grow linearly with the size of schedule. We thus conclude that the verification process is completed in $O(n^2)$ time. This efficient scaling implies that we are able to verify worst-case safety with large schedules. As an example, consider increasing the capacity for each node of the network in Fig. 1 by 10 to produce feasible schedules more easily.

[1]The related MATLAB code can be found in https://github.com/gtfactslab/Wei_TCNS_ScheduleVerification.git.

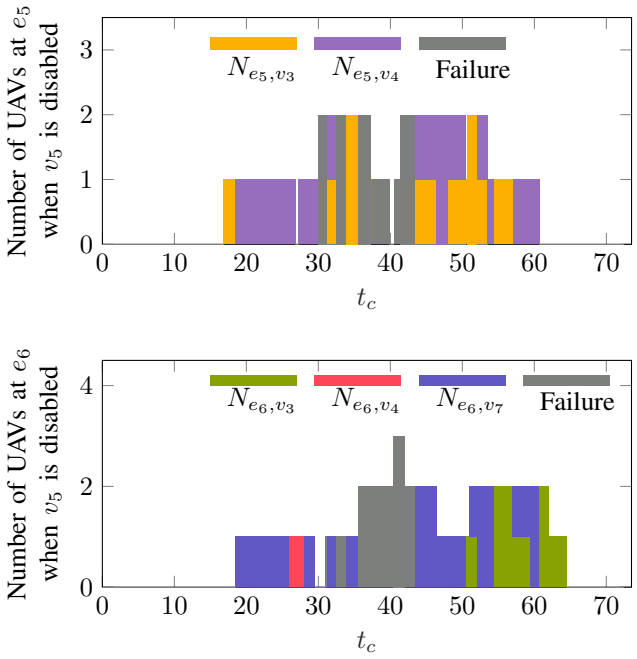

Fig. 4. Observation of the network when node v_5 is disabled at any time $t_c > 0$ when $C_{v_4} = 4$. (Top) Redistribution of flights on link e_5 to its backup nodes when node v_5 is disabled. (Bottom) Redistribution of flights on link e_6 to its backup nodes when node v_5 is disabled.

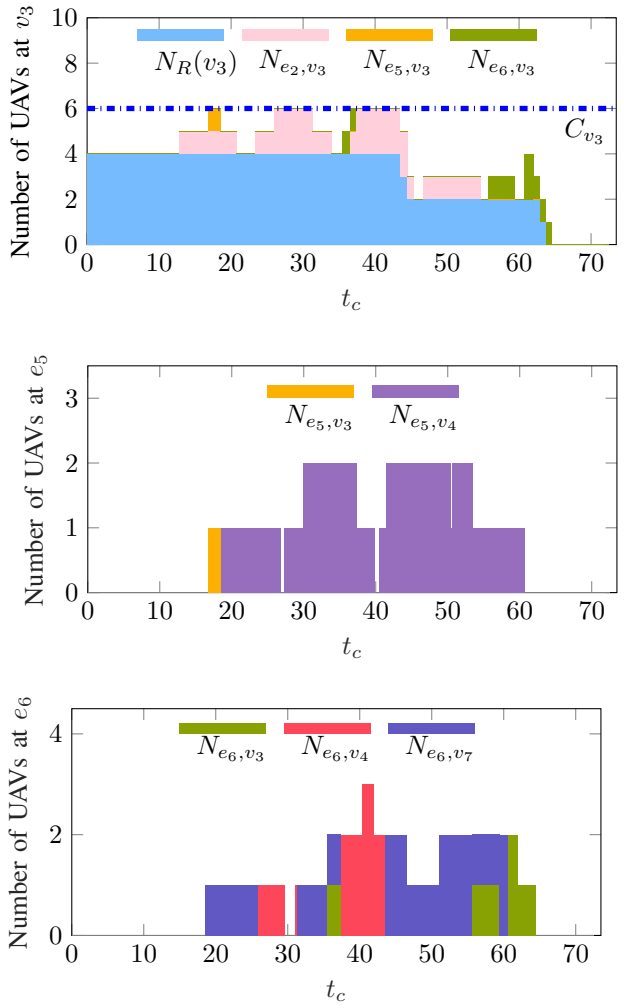

Fig. 5. Observation of the network when node v_5 is disabled at any time $t_c > 0$ when we adjust the capacity of node v_4 to $C_{v_4} = 8$. (Top) Expected landing-spot occupation at node v_3. (Middle) Redistribution of flights on link e_5 to its backup nodes. (Bottom) Redistribution of flights on link e_6 to its backup nodes.

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. 6 demonstrates the $O(n^2)$ computation complexity and shows that we are able to verify safety or demonstrate the safety failure for a schedule with 1,000 flights in under 50 seconds. As a baseline comparison, we also implement the verification algorithm with the naive MILP implied by Theorem 1 without the efficient simplification to an LP derived in Section IV. This implementation is solved using the Gurobi [36] solver through with the YALMIP MATLAB toolbox [37]. We test the same 20-flight schedule on this MILP algorithm, which takes 7.34 seconds to verify, while the algorithm we use with simplification to LP takes only 1.43 seconds. A 100-flight schedule takes around 40 seconds to verify with the naive MILP formulation, and 8 seconds with the LP algorithm.

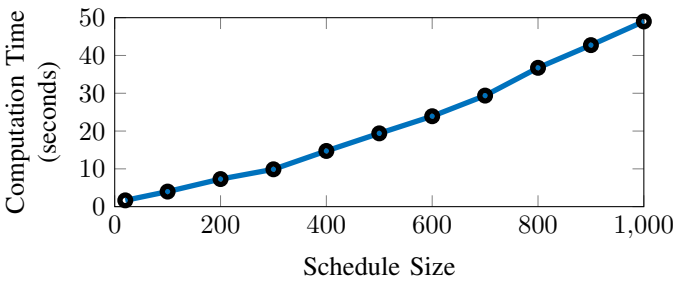

Fig. 6. The computation time for verifying the worst-case safety of schedules with different sizes. We test on 11 different sets of schedules with sizes from 20 to 1000. The data points demonstrate the $O(n^2)$ computational complexity.

VI. Conclusion

We studied the safety verification problem for Urban Air Mobility (UAM) schedules in the face of vertiport (i.e., landing site) closures. We adopt a UAM network model that considers a set of finite-capacity vertiports and links between vertiports with uncertain travel time. If a vertiport is closed at some time, then flights destined for the closed vertiport must be rerouted to one of a set of link-dependent backup nodes. A safety violation occurs if the finite landing capacity at any node is exceeded due to the rerouting. We first considered a single node closure and then extended these conditions to the possibility of multiple simultaneous node closures.

We consider the travel time uncertainty as a nondeterministic uncertainty, and therefore, we define appropriate notions of worst-case and best-case safety. We give necessary and sufficient conditions in both cases. As formulated, these conditions take the form of mixed integer linear programming (MILP) constraints. We then showed that these numerically inefficient MILP constraints are able to be converted into efficient linear

programming (LP) constraints using the theory of totally unimodular matrices (TUMs), resulting in an efficient algorithm for safety verification. We demonstrated our approach through several case studies.

There are several possibilities for extending the results of this paper. For example, we regard a disrupted node as completely malfunctioning, but a partial malfunctioning disruption model, where not all landing spots of the disrupted node are disabled, could also be investigated.

REFERENCES

[1] Q. Wei, G. Nilsson, and S. Coogan, "Safety verification for urban air mobility scheduling," *IFAC-PapersOnLine*, vol. 55, no. 13, pp. 306–311, 2022.

[2] J. Holden and N. Goel, "Fast-forwarding to a future of on-demand urban air transportation," 2016. [Online]. Available: https://www.uber.com/elevate.pdf

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

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

[5] The MITRE Corporation, "NextGen independent assessment recommendations," 2014. [Online]. Available: https://www.mitre.org/sites/default/files/publications/pr-14-3495-next-gen-independent-assessment.pdf

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

[7] INRIX, "Electric passenger drones could relieve housing costs and spread growth in nation's booming cities," 2019. [Online]. Available: https://inrix.com/campaigns/vtol-study/

[8] C. Al Haddad, E. Chaniotakis, A. Straubinger, K. Plötner, and C. Antoniou, "Factors affecting the adoption and use of urban air mobility," *Transportation research part A: policy and practice*, vol. 132, pp. 696–712, 2020.

[9] C. Bosson and T. A. Lauderdale, "Simulation evaluations of an autonomous urban air mobility network management and separation service," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3365.

[10] M. Xue, J. Rios, J. Silva, Z. Zhu, and A. K. Ishihara, "Fe3: An evaluation tool for low-altitude air traffic operations," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3848.

[11] M. A. Aiello, C. Dross, P. Rogers, L. Humphrey, and J. Hamil, "Practical application of SPARK to OpenUxAS," in *Formal Methods – The Next 30 Years*. Springer International Publishing, 2019, pp. 751–761.

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

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

[14] G. Como, K. Savla, D. Acemoglu, M. A. Dahleh, and E. Frazzoli, "Robust distributed routing in dynamical networks—part I: Locally responsive policies and weak resilience," *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 317–332, 2013.

[15] ——, "Robust distributed routing in dynamical networks–part II: Strong resilience, equilibrium selection and cascaded failures," *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 333–348, 2013.

[16] M. Løve, K. R. Sørensen, J. Larsen, and J. Clausen, "Disruption management for an airline—rescheduling of aircraft," in *Workshops on Applications of Evolutionary Computation*. Springer, 2002, pp. 315–324.

[17] G. Zhu, J. F. Bard, and G. Yu, "Disruption management for resource-constrained project scheduling," *Journal of the Operational Research Society*, vol. 56, no. 4, pp. 365–381, 2005.

[18] T. Andersson* and P. Värbrand, "The flight perturbation problem," *Transportation planning and technology*, vol. 27, no. 2, pp. 91–117, 2004.

[19] S. Bisaillon, J.-F. Cordeau, G. Laporte, and F. Pasin, "A large neighbourhood search heuristic for the aircraft and passenger recovery problem," *4OR*, vol. 9, no. 2, pp. 139–157, 2011.

[20] Z. Wu, Q. Cao, B. Li, C. Dang, and F. Hu, "A rapid solving method to large airline disruption problems caused by airports closure," *IEEE Access*, vol. 5, pp. 26 545–26 555, 2017.

[21] B. G. Thengvall, G. Yu, and J. F. Bard, "Multiple fleet aircraft schedule recovery following hub closures," *Transportation Research Part A: Policy and Practice*, vol. 35, no. 4, pp. 289–308, 2001.

[22] S. Yan and C.-G. Lin, "Airline scheduling for the temporary closure of airports," *Transportation Science*, vol. 31, no. 1, pp. 72–82, 1997.

[23] M. F. Argüello, J. F. Bard, and G. Yu, "A grasp for aircraft routing in response to groundings and delays," *Journal of Combinatorial Optimization*, vol. 1, no. 3, pp. 211–228, 1997.

[24] M. F. Arguello, *Framework for exact solutions and heuristics for approximate solutions to airlines' irregular operations control aircraft routing problem*. The University of Texas at Austin, 1997.

[25] J. M. Rosenberger, E. L. Johnson, and G. L. Nemhauser, "Rerouting aircraft for airline recovery," *Transportation Science*, vol. 37, no. 4, pp. 408–421, 2003.

[26] Z. Wu, B. Li, C. Dang, F. Hu, Q. Zhu, and B. Fu, "Solving long haul airline disruption problem caused by groundings using a distributed fixed-point computational approach to integer programming," *Neurocomputing*, vol. 269, pp. 232–255, 2017.

[27] Z. Wu, B. Li, and C. Dang, "Solving multiple fleet airline disruption problems using a distributed-computation approach to integer programming," *IEEE Access*, vol. 5, pp. 19 116–19 131, 2017.

[28] B. Li, C. Dang, and J. Zheng, "Solving the large airline disruption problems using a distributed computation approach to integer programming," in *2013 IEEE Third International Conference on Information Science and Technology (ICIST)*. IEEE, 2013, pp. 444–450.

[29] B. Aguiar, J. Torres, and A. J. Castro, "Operational problems recovery in airlines–a specialized methodologies approach," in *Portuguese Conference on Artificial Intelligence*. Springer, 2011, pp. 83–97.

[30] D. Zhang, H. H. Lau, and C. Yu, "A two stage heuristic algorithm for the integrated aircraft and crew schedule recovery problems," *Computers & Industrial Engineering*, vol. 87, pp. 436–453, 2015.

[31] B. Zhu, X. L. Cao, Y. Wang, and Q. Gao, "Constraint programming method for crew schedule recovery," in *Applied Mechanics and Materials*, vol. 496. Trans Tech Publ, 2014, pp. 1788–1791.

[32] R. Nissen and K. Haase, "Duty-period-based network model for crew rescheduling in European airlines," *Journal of Scheduling*, vol. 9, no. 3, pp. 255–278, 2006.

[33] J. Vink, B. Santos, W. Verhagen, I. Medeiros, and R. Filho, "Dynamic aircraft recovery problem - An operational decision support framework," *Computers & Operations Research*, vol. 117, p. 104892, 2020.

[34] C.-H. Chen, F.-I. Chou, and J.-H. Chou, "Multiobjective evolutionary scheduling and rescheduling of integrated aircraft routing and crew pairing problems," *IEEE Access*, vol. 8, pp. 35 018–35 030, 2020.

[35] N. Megiddo, "Linear programming in linear time when the dimension is fixed," *Journal of the ACM (JACM)*, vol. 31, no. 1, pp. 114–127, 1984.

[36] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2022. [Online]. Available: https://www.gurobi.com

[37] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[38] A. J. Hoffman and J. B. Kruskal, "Integral boundary points of convex polyhedra," in *50 Years of integer programming 1958-2008*. Springer, 2010, pp. 49–76.

[39] I. Heller and C. B. Tompkins, "An extension of a theorem of Dantzig's," *Linear inequalities and related systems*, vol. 38, pp. 247–254, 1956.

APPENDIX A
PROOF OF LEMMA 1

We prove in this appendix. The proof makes use of properties of *Totally Unimodular Matrices (TUMs)*.

Definition 4. *(Totally Unimodular Matrix) A matrix is* totally unimodular *if every square submatrix has determinant* $0, +1,$ *or* -1.

TUMs are widely used in the context of optimization problems. In particular, it can be shown that for a large class of linear programs defined via TUMs, the resulting optimal solution takes on integer values [38]. We use this property in the proof of Lemma 1 next.

Proof of Lemma 1. We first let the vector $\vec{n}$ be the vectorized sequence $\{n_{l_1,l_2}\}_{l_1=1,l_2=1}^{l_1=N_1,l_2=N_2}$, so that

$$\vec{n} = [n_{1,1}, n_{1,2}, \ldots, n_{1,N_2}, n_{2,1}, \ldots, n_{N_1,1}, \ldots, n_{N_1,N_2}]^T. \tag{43}$$

We can then simplify the constraint (35)–(36) as

$$A_1\vec{n} = \vec{\alpha}\,, A_2\vec{n} \leq \vec{\beta} \tag{44}$$

where $\vec{\alpha} = [\alpha_1, \ldots, \alpha_{N_1}]^T$ and $\vec{\beta} = [\beta_1, \ldots, \beta_{N_2}]^T$, and A_1 (resp., A_2) is a $N_1 \times N_1N_2$ (resp., $N_2 \times N_1N_2$) matrix that reflects the matrix form of the multiplication of the constraints. In particular, the row-i-column-j element of A_1 is $A_1(i,j) = 1$ if $(i-1)N_2 < j \leq iN_2$ and $A_1(i,j) = 0$ otherwise, and $A_2(i,j) = 1$ if $j = m \cdot N_2 + i$ for $m = 0, 1, \ldots, N_1 - 1$ and $A_2(i,j) = 0$ otherwise.

Since for $(l_1, l_2) \in L_{var}$, $n_{l_1,l_2} = \gamma_{l_1,l_2}$, we then subtract corresponding entries from both left sides of (44), and subtract the values from their right sides. We let A_3 (resp., A_4) be the resulting matrices, so that A_3 (resp., A_4) is a $N_2 \times N_1N_2$ (resp., $N_1 \times N_1N_2$) and the row-i-column-j element of A_3 is $A_3(i,j) = 0$ if $(i, j-(i-1)N_2) \in L_{var}$ and $A_3(i,j) = A_1(i,j)$ otherwise, and $A_4(i,j) = 0$ if $(i, (j-i)/N_2 + 1) \in L_{var}$ and $A_4(i,j) = A_2(i,j)$ otherwise. Let

$$\alpha'_{l_1} = \alpha_{l_1} - \sum_{l_2:(l_1,l_2)\in L_{var}} \gamma_{l_1,l_2} \quad \text{for all } l_1, \tag{45}$$

$$\beta'_{l_2} = \beta_{l_2} - \sum_{l_1:(l_1,l_2)\in L_{var}} \gamma_{l_1,l_2} \quad \text{for all } l_2, \tag{46}$$

$$\vec{\alpha'} = [\alpha'_1, \ldots, \alpha'_{N_1}]^T, \tag{47}$$

$$\vec{\beta'} = [\beta'_1, \ldots, \beta'_{N_2}]^T. \tag{48}$$

We can then reformulate (44) together with the constraints (37)–(38) as

$$\underbrace{\begin{bmatrix} A_3 \\ -A_3 \\ A_4 \\ -I_{N_1N_2} \end{bmatrix}}_{=:A} \vec{n} \leq \underbrace{\begin{bmatrix} \vec{\alpha'} \\ -\vec{\alpha'} \\ \vec{\beta'} \\ \vec{0}_{N_1N_2} \end{bmatrix}}_{=:\vec{b}}, \tag{49}$$

where $I_{N_1N_2}$ is the identity matrix with N_1N_2 rows and $\vec{0}_{N_1N_2}$ is the zero vector of length N_1N_2.

The first part of the lemma is then turned into the standard linear programming problem, which is finding the existence of $\vec{n}$ that satisfies $A\vec{n} \leq \vec{b}$. The next step is to prove that A is a totally unimodular matrix (TUM) as defined in Definition 4.

We first consider the matrix $\begin{bmatrix} A_3 \\ A_4 \end{bmatrix}$. Notice that for each column of A_3 and A_4, there exists at most one nonzero entry, 1, therefore, for each column of the matrix $\begin{bmatrix} A_3 \\ A_4 \end{bmatrix}$, there exists at most two nonzero entries, and for any column with two nonzero entries, both of them will be 1, and the row of one is in A_3 while the other is in A_4. According to Hoffman's sufficient conditions [39, Appendix], $\begin{bmatrix} A_3 \\ A_4 \end{bmatrix}$ is a TUM. By the general rule of TUM, $-\begin{bmatrix} A_3 \\ A_4 \end{bmatrix}$ is a TUM and thus $\begin{bmatrix} A_3 \\ A_4 \\ -A_3 \\ -A_4 \end{bmatrix}$ is also a TUM. According to the definition of TUM, deleting some rows from a TUM will produce a TUM, as any square non-singular submatrix of the new matrix will still be unimodular. As a result, $\begin{bmatrix} A_3 \\ A_4 \\ -A_3 \end{bmatrix}$ is TUM and so is $\begin{bmatrix} -A_3 \\ -A_4 \\ A_3 \\ I_{N_1N_2} \end{bmatrix}$ and it's additive inverse by the general rule of TUM. As switching rows does not affect the absolute value of the determinant of a matrix, then we conclude from above that A is a TUM.

Therefore, [38, Theorem 2] implies that if there exists a solution for the LP in (49), then there exists an integral solution for the same LP problem, which concludes the lemma. □

Qinshuang Wei received the B.S., M.S., and Ph.D. degrees in electrical engineering from Georgia Institute of Technology, Atlanta, Georgia, USA in 2017, 2018, and 2022, where she also received the B.S. degree in applied mathematics in 2017. She is currently a postdoctoral fellow in the University of Texas at Austin.

Her research interests include control systems, cyber-physical systems and optimization.

Gustav Nilsson received an M.Sc. degree in engineering physics and a Ph.D. degree in automatic control from Lund University, Lund, Sweden, in 2013 and 2019, respectively. He is currently a postdoctoral researcher at Urban Transport Systems Laboratory (LUTS), EPFL. Before his current position, he was a postdoctoral fellow at the School of Electrical and Computer Engineering (ECE), Georgia Institute of Technology. His primary research interests include non-linear control, network dynamics, and intelligent transportation systems.

Samuel Coogan received the B.S. degree in electrical engineering from the Georgia Institute of Technology, Atlanta, GA, USA in 2010 and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley, Berkeley, CA, USA in 2012 and 2015. He is currently an associate professor and the Demetrius T. Paris Junior Professor at the Georgia Institute of Technology, Atlanta, GA, USA in the School of Electrical and Computer Engineering and the School of Civil and Environmental Engineering. Prior to joining Georgia Tech in 2017, he was an assistant professor at the University of California, Los Angeles from 2015 to 2017. Prof. Coogan received a CAREER Award from the National Science Foundation in 2018, a Young Investigator Award from the Air Force Office of Scientific Research in 2019, and the Donald P. Eckman Award from the American Automatic Control Council in 2020. He is a member of SIAM and a senior member of IEEE.