Misfortunes Never Come Alone: Balancing Occupancy of UAM Alternate Landing Sites

Hardy Hasan
Lidingö Stad
Stockholm, Sweden

Ryota Mori
Graduate School of Maritime Sciences
Kobe University, Japan

Tatiana Polishchuk Valentin Polishchuk Leonid Sedov
Flight Logistics Unit
Communications and Transport Systems
Linköping University, Sweden

***Abstract*—We study routing of Urban Air Mobility (UAM) flights while ensuring robustness to large-scale disruptions during which many aircraft may need to initiate contingency maneuvers at the same time. We propose algorithmic solutions for routing drones so that they always stay sufficiently close to potential alternate landing locations, while preventing the situations when too many drones rely on a single landing site at once. The performance of our algorithms is evaluated on simulated urban scenarios.**

I. Introduction

EASA's Special Condition for small-category Vertical Take-Off and Landing (VTOL) aircraft [1] suggests that every drone's flightplan (called *U-plan* by SESAR's CORUS-XUAM project [2]) specifies potential alternate landing sites (ALSs) for the whole duration of the flight. Such sites, where drones in distress can safely land, may become a part of the emergency management plans which U-space service providers (USSPs) shall have and of the contingency procedures which UAS (Unmanned Aircraft Systems) operators should provide to USSPs according to EU U-space regulation 664 [3, Articles 6.8 and 15.2].

ALSs, however, may be a scarce resource, especially in metropolitan regions which usually do not have a lot of space to spare. At the same time, densely populated urban areas may generate higher demand for alternate landings, simply due to the larger drone traffic volumes. This may lead to the situation when several flights *simultaneously* rely on the same ALS, should the emergency affect multiple UAM vehicles (e.g., sudden bad weather or network connectivity outage).

This paper focuses on safety-oriented route planning for UAM, taking into account provision of alternate landing locations along the whole route in such a way that there are no ALSs on which too many drones rely simultaneously. We propose an Integer Program optimizing the efficiency of the operations and compare it with a simple "First Come First Served" (FCFS) approach routing the drones sequentially.

Note that some ALSs may be situated in places specifically designated for contingency/emergency landings only. At the same time, due to the cost of land, construction, and maintenance, it is possible that instead of reserving specific locations for ALSs, already existing vertiports will be used for alternate landing (similarly to the ATM practice where ordinary airports are used for emergency landing). Therefore, in this paper we will use the words "ALS" and "vertiport" interchangeably; we also do not differentiate between contingency and emergency situations, and assume that in both off-nominal cases the UAM vehicles will need to land at an alternate location.

Related work

EASA's requirements of the continued safe flight and landing (CSFL) for the Enhanced category aircraft (Fig. 1) and the controlled emergency landing (CEL) for the Basic category VTOLs [1] may be viewed as UTM counterparts of ICAO's Extended Diversion Time Operations (EDTO) requirements [5] (formerly known as ETOPS: Extended-range Twin-engine Operations Performance Standards) which stipulate (Fig. 2) that at any point during the flight, the aircraft has an alternate airport reachable within certain time (called Maximum Diversion Time). For a single UAM flight, routing close to ALSs was considered in [6] which suggested that U-plans "include contingency or emergency response plans" and "be subject to a constraint that the UAS must remain in range of an adequate landing site at any time through the operation." We keep the same constraint on the UAS flights when considering large-scale UAM operations over a metropolitan area, and extend the work in [6] from 1 to multiple drone flights. The important new aspect of such an extension is that we have to make sure that no ALS is ever overbooked, i.e., that any ALS is assigned to at most as many drones as the "capacity" of the site is.

Vertiport localization is an active research area: an interactive tool for identifying ALSs compliant with EASA's vertiport geometry [4] was recently reported in [7], and a system for evaluation and optimization of vertiport siting was presented in [8]. In general, risk management in drone routing gained significant traction in both civil and military applications: the

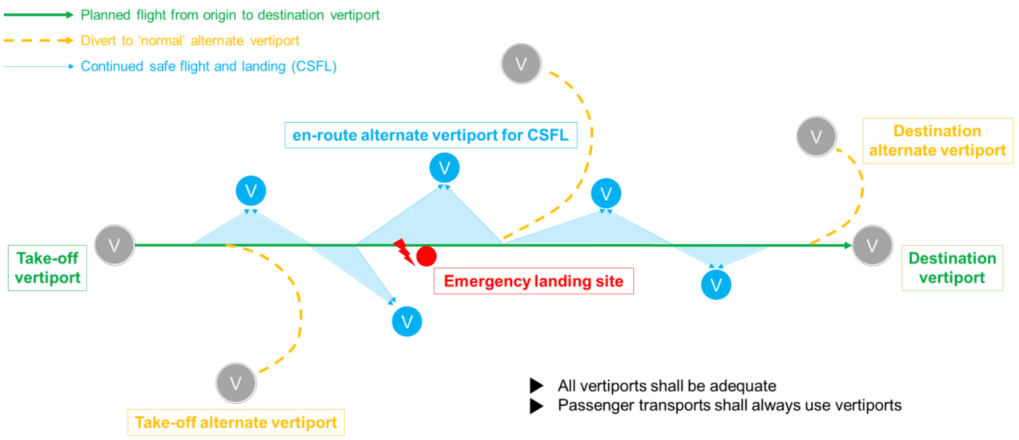

Figure F-1. En-route alternate vertiport for CSFL

Fig. 1. From EASA VTOL PTS [4, Figure F-1]: At any moment during the flight, the vehicle must have a location for safe landing

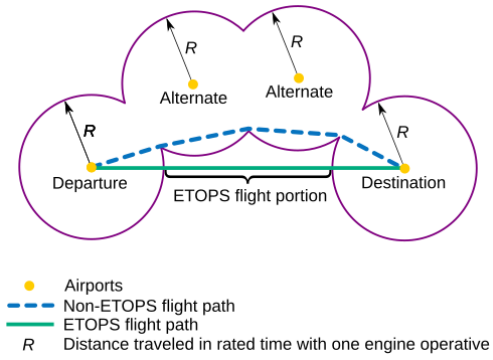

Fig. 2. From https://en.wikipedia.org/wiki/ETOPS: The standard (non-ETOPS) path must stay within union of radius-R disks centered on airports

former mitigates 3rd-party risks to people on the ground and in other aircraft [9]–[15], while the military ensures the drone itself survives the mission in a hostile environment (avoiding radars, anti-aircraft defense systems, etc.) [16]–[19].

Our work is related to finding disjoint paths in graphs [20]–[23] and geometric domains [24]: we compute routes for drones, making sure that at any time, no location is assigned as an ALS to too many drones. The Non-Stop Disjoint Trajectories Problem (NSDTP) [25] deals with routing a set of vehicles in a graph while ensuring that they do not intersect in time and space. We do not concern ourselves with keeping aircraft separated during the flight (assuming that tactical conflict detection and resolutions (CD&R) is available to them); we only care about avoiding conflicts at ALSs due to too many drones potentially landing at one site.

Due to the temporal component, our setting is related to vehicle routing [26]–[28] with time windows [29] where customers have bounds on when they can get the deliveries. We employ the standard way to ensure temporal separation of multiple vehicles – time-expanded network [30] (used also in ATM [31]–[33]). Flying from vertiport to vertiport is reminiscent of beacon routing in robotics, i.e., sending robots towards predetermined locations (beacons) – an extensively studied area [34]–[40], considered also for UTM [41].

Roadmap: The remainder of the paper is organized as follows. In Section II we present the problem, cast it into precise mathematical framework, and propose solution approaches. Section IV describes implementation of the proposed solutions and presents results of experiments in simulated urban environments. We conclude in Section V by discussing the results and outlining directions for future research.

II. Modeling and Problem Formulation

This section defines the input, output, constraints and the objective function for our optimization problem. We are given a region of interest (ROI) with a set V of vertiports between which the drones fly (each vertiport $v \in V$ is a point in the ROI). Recall that we do not differentiate between the nominal-operation vertiports and ALSs, and use the terms "vertiport" and "ALS" interchangeably to indicate a place where a drone can safely land. The ROI may contain restricted areas (geofences, or obstacles) flying through which is forbidden.

Let L be the maximum distance from the nearest vertiport that drones are required to always maintain during the flight. For simplicity we assume that all drones are the same in the sense that the set V and the distance L are the same for all drones; our algorithms may be extended to handle a diverse fleet as well. Our focus is on vertiports, and we acknowledge their diversity: some would allow only one drone to land at a time, while others may have larger capacities. We thus let c_v denote the capacity of vertiport $v \in V$; the capacity is a number that indicates how many drones may simultaneously indicate v as their ALS. We assume that c_vs are known to the routing algorithm, for all $v \in V$ (determining the vertiport capacities is an interesting research direction by itself).

The next item in the input is the time uncertainty U which represents the maximum time by which drones can deviate from their assigned timing along the route. This is inline

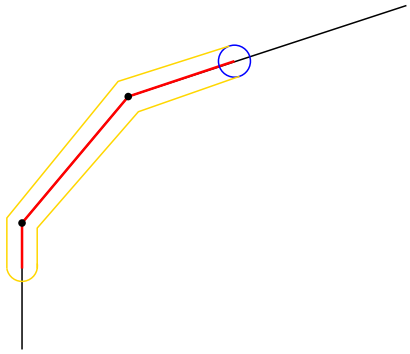

Fig. 3. The time uncertainty may be thought of as a "snake", where the planned position $S(\tau)$ of a drone d is the snake's "head": black is d's route, and the blue circle is the planned location $S(\tau)$ of d at time τ. Due to the time uncertainty U, the drone can be delayed and be anywhere on the red portion (also represented by the yellow buffer), or, formally, anywhere on $[S(\tau - U), S(\tau)]$. In particular, vertiports (black dots) on the red are all reserved by d at τ

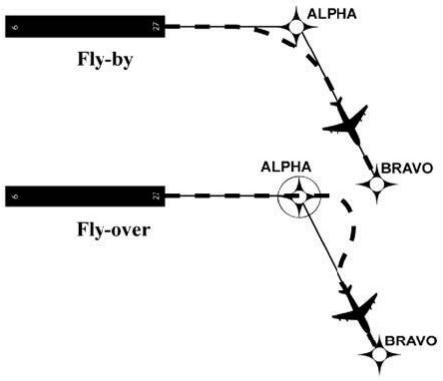

Fig. 4. Figure ENR 1.16-2 from FAA's PBN&RNAV AIP ENR [43]: The aircraft does not fly directly over a Fly-by waypoint

with EASA's Acceptable Means of Compliance and Guidance Material (AMC/GM) to EU's U-space Regulation 664 [3], according to which a U-plan is "a series of one or more 4D volumes" in which "Each dimension includes the uncertainty of the flight, considering the UAS operational performance, and the assumptions on the operator proficiency and weather conditions."; we assume that the drones remain within their 4D volume 100% of the time, as per AMC/GM's suggestion that "The UAS operator submits this series of volumes, committing to remain within them". For simplicity, we assume that this time uncertainty U is the same for all drones, noting again that our algorithms may be extended to diverse drones, having different Us. Also for ease of presentation we assume that drones only can fly slower than planned, but not faster, so this time uncertainty U works only one way (Fig. 3); in particular, when routing a drone, any vertiport on its way should be reserved for the additional time U. Again, this assumption is without loss of generality, and can easily be lifted by setting U to be the sum of both forward- and backward-looking time uncertainties and adjusting all times accordingly.

The last but not the least part of the input are the origins and destinations of the drones flights, along with their desired takeoff times and maximum speeds. Each drone d is thus represented by a 4-tuple $v_{orig}(d), v_{dest}(d), t_{departure}(d), s(d)$.

We assume a simple model where drones fly between vertiports in straight segments (this is similar to the ATM practice of routing aircraft by specifying waypoints; in particular, RNAV operations [42] allow several waypoints types which may be reused also in UTM – see Fig. 4): this way, if at any point in time, an emergency happens, the drone needs to just continue flying straight ahead until it reaches the vertiport where it can safely land. This means that we assume that during the flight, the drone always relies on the next vertiport for alternate landing, not on the previous. The assumption is without loss of generality: allowing a drone to also U-turn to go to the previous vertiport can be handled by setting $L \to 2L$ (if the distance between 2 consecutive vertiports $i.j$ on the drone's path is at most $2L$, then anywhere on the segment ij the drone is at distance $\leq L$ from one of the two vertiports).

Our goal is to compute a route for every drone d, from $v_{orig}(d)$ to $v_{dest}(d)$, possibly via intermediate vertiports on the way. The drone may start flying later than the desired start time $t_{departure}(d)$ – we assume that we cannot schedule drones earlier than they desire (our algorithms can easily handle the case when each drone has a window of time when it may start). The constraint is that at any time during the flight, the next vertiport is at distance at most L from the drone – this vertiport is the ALS for d, should the need to land arise. That is, a vertiport v is assigned as an ALS for d during the whole time while d flies from to v from the preceding vertiport, plus the time U (the uncertainty). On the vertiports side, we have the requirement is that at any time, any vertiport v is the ALS for at most c_v drones – handling this requirement is the **main novelty** of this paper. Note that our model ignores keeping the aircraft separated, assuming that tactical CD&R is available to them (e.g., assigning conflicting drones to different airspace layers [44], [45]); instead, we focus on the vertiport capacities, i.e,, drones reserving a vertiport until they reach it and proceed to fly to the next vertiport in their routes.

In order to quantify the quality of our solutions, we first introduce μ_d — the ideal time when drone d would arrive at the destination vertiport, assuming it were the only drone in the system; μ_d is computed assuming d starts at $t_{departure}(d)$ and follows the shortest path from $v_{orig}(d)$ to $v_{dest}(d)$ with speed $s(d)$. The objective function which we seek to minimize in our optimization problem is the total travel extension (or delay) $\sum_d (\text{assigned_arrival_time_for_}d - \mu_d)$.

To put our model in context of other problems, note that if all vertiports have unlimited capacity ($\forall v \in V, c_v = \infty$), then our problem equals finding shortest path for each drone, such that consecutive vertiports on the path are at distance at most L from each other. On the other hand, if we assume that each drone blocks every vertiport forever ($U = \infty$), each vertiport can accommodate only one aircraft ($\forall v \in V, c_v = 1$) and all drones fly with the same speed (without loss of

generality, $\forall d, s(d) = 1$), then our problem is the Disjoint Shortest Path problem, i.e., finding internally vertex-disjoint paths of minimum total length between the drones origins and destinations – an NP-hard problem [23].

III. Algorithmic solutions

We present two approaches to our problem: a greedy algorithm and and Integer Program (IP). A common ingredient in both approaches is that we discretize the time into equal-duration intervals Δt and define the planning horizon T as a sufficiently large time (e.g., the sum of flight times for all drones along shortest paths from their origins to their destinations) within which all the flights may reach their destinations. For every drone d we round up its desired departure time $t_{departure}(d)$ to the closest integer multiple of Δt larger than $t_{departure}(d)$); we also assume that T is an integer multiple of Δt. The other common ingredient to our solutions is the graph $G = (V, E, w)$ which is the weighted graph whose vertices V are the vertiports, and two vertices $i.j \in V$ are connected by an edge $ij \in E$ iff a drone can reach j from i by flying in a straight line (i.e., if the segment ij does not go through a restricted area) and the distance $w(i, j)$ between i and j is at most L.

A. Greedy routing

Order the drones by the submission time of their operational intents (any other priority order may be used for the greedy algorithm), and identify each drone with its position in the order (i.e., drone 1 is the one whose intent submitted first, etc.). We find the drone trajectories one by one: drone 1 is routed along a shortest $v_{orig}(1)$-$v_{dest}(1)$ path in G, drone 2 is routed along a shortest path that does not conflict with drone 1, etc.: drone d is routed along the shortest path not conflicting with the trajectories of drones $1, 2, \ldots, d-1$. The shortest paths for the drones are found in the so-called *time-expanded network* (TEN) [30] whose edges are modified after routing each drone. TEN can be visualized as a sequence of "layers" in a 3D space where the z-axis represents time and each layer (or timestep) represents an integer multiple of Δt in $[0, T]$ (Fig. 5). We put a copy of the vertiports set V into every layer; the crux of the method is defining the edges of TEN before routing every drone d – as described next.

First, we take all previously scheduled drones $1, \ldots, d-1$ and reduce capacities of all vertiports "reserved" by them during the appropriate time periods (i.e., in the corresponding "layers"): if some drone $d^* < d$ was routed from vertiport v_1 to vertiport v_2 during the time interval $[\tau_1, \tau_2]$, then capacity of v_2 is reduced by 1 in all layers between $\tau_1 - U$ (to account for the time uncertainty U of the currently planned drone d) and $\tau_2 + U$ (to account for the uncertainty of d^*). Next, we connect (copies of) vertiports in the TEN by adding upward edges between vertiports that have positive capacity: if vertiport v_i at layer t has a positive capacity and a vertiport v_j at layer $t' > t$ has a positive capacity, then we connect the vertiports by a directed edge from iff the distance $w(v_i, v_j) \leq L$ (i.e., $v_i v_j$ was an edge in G) and the drone d can reach v_j from v_i within time $t' - t$, i.e., if $w(v_i, v_j) \leq s(d)(t' - t)$; the weight of the edge is $t' - t$. This way, TEN edges represent possible connections for d.

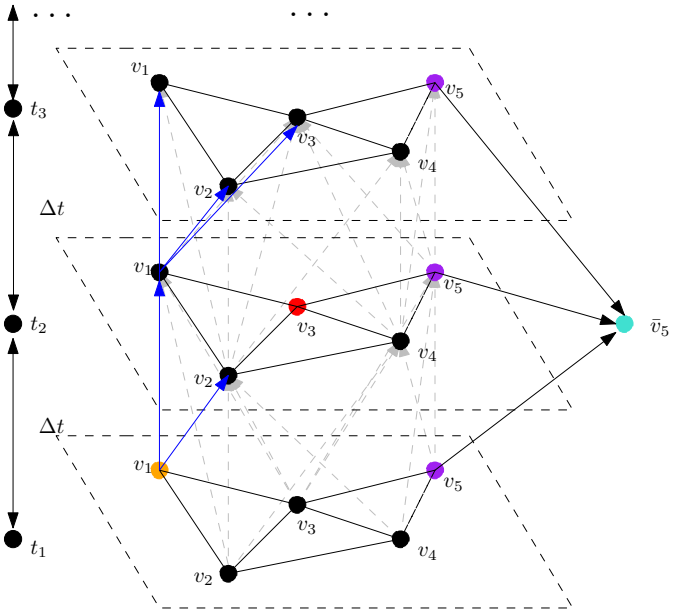

Fig. 5. TEN: a layer holds a copy of V at each timestep (edges within each layer are depicted only for demonstrative purposes). Blue are outgoing edges from vertiport v_1 to the next layer, or the next timestep. To reduce visual clutter, edges originating from other vertiports are dashed and light gray. Vertiport v_3 is full at timestep t_2 (the vertiport is an ALS for c_{v_3} earlier-scheduled operations), so there is no edge from v_1 at t_1 to v_3 at t_2; at t_3, the vertiport becomes available, so there is a corresponding edge from v_1 at t_2. The currently scheduled drone desires to depart from v_1 (orange) at t_1 and fly to v_5 (purple). The vertices which represent the destination vertiport v_5 in all layers are connected to $\bar{v}_5$ (turquoise).

In order to implement ground delay, the origin vertiport $v_{orig}(d)$ at the layer representing the time $t_{departure}(d)$ is connected to the same vertiport in all next layers $t_{departure}(d) + \Delta t, t_{departure}(d_j) + 2\Delta t, \ldots$ for the entire planning horizon (hovering above a vertiport can be implemented similarly). The weights of these edges are Δt (if ground delay is preferable to spending time in the air, the weights may be adjusted).

Finally, we add to TEN an additional vertex $\bar{v}_{dest}(d)$ to represent the destination. We add edges to $\bar{v}_{dest}(d)$ from all vertiports with positive capacity (in all layers) closer than L to $v_{dest}(d)$; the weight of such edge from a copy of vertiport v is $w(v, v_{dest}(d))/s(d)$ (irrespective of the layer in which the copy of v resides). We now find the route for d as the shortest path between the vertex that represents the origin vertiport $v_{orig}(d)$ at the layer representing the desired departure time $t_{departure}(d)$ and the destination vertex $\bar{v}_{dest}(d)$.

While the greedy algorithms always finds a feasible solution, it is not guaranteed to produce optimal routes minimizing the total travel time of all drones. In the remainder of this section we present an IP to find the optimum.

B. Integer Programming formulation

In this section we give an IP for minimizing the total delay of the drones. We start from introducing the notation. Let

$$\bar{w}(i,j,d) = \begin{cases} \Delta t & \text{if } i = j = v_{orig}(d) \\ \lceil (w(i,j)/s(d))/\Delta t \rceil \, \Delta t & \text{otherwise} \end{cases}$$

be the time, in multiples of Δt, that d spends on edge $ij \in E$ (recall that $w(i.j)$ is the distance between vertiports i and j, and $s(d)$ is the maximum speed of d): the drone spends one timestep if it waits at the origin, and we round up the time it takes to traverse ij. Our IP has the following binary variables:

- x_{ijdt} is 1 if d is scheduled to take edge ij of G at time t
- y_{ijdt} is 1 if drone d is scheduled to finish traveling along the edge ij at time t (i.e., arrive at j at time t)
- r_{idt} is 1 if vertiport i is reserved for drone d at time t

The IP is given on Figure 6. The objective function is the total delay (μ_ds may be removed from the objective function since they are constant; without them, the objective function will represent the total planned travel time of all operations). The constraint 1b is the flow conservation constraint borrowed from the IP for the shortest path [46]. Constraint 1c ensures that each vertiport is reserved for the entire durations of drone flights, taking into account the time uncertainties (M is just some large number; the RHS of the constraint is divided by M to ensure that r is a binary variable, i.e., that it does not have to be larger than 1). Constraint 1d ensures that vertiports' capacities are never exceeded. Constraint 1e enforces the earliest permitted takeoff time, restricting drones from taking off earlier, and constraint 1f ensures that every drone arrives to its destination. Finally, constraint 1g ensures consistency between the x and y variables.

IV. Simulation

We implemented both of our algorithmic solutions and ran them on simulated scenarios; this section presents the results.

A. Simulation setup

As vertiports are not yet established anywhere on a large scale, we construct a simple artificial vertiport network, based on real-world data, to test our algorithms (Fig. 7). Following EASA [4], we took buildings with flat roofs as potential vertiport locations: this ensures that the vertiports are well elevated above the ground, so the Obstacle-Free Volumes (OFVs) [4, Ch. D, subpart 2], within which VTOL vehicles takeoff and land, are less likely to intersect other buildings (such assumption was also supported, e.g., in [47]). We obtained data for all buildings with flat roofs in Stockholm city from OpenStreetMap [48] and selected those roofs that physically can support at least one vertiport, i.e, those into which a disk of diameter D =20m fits – see [7, Section II] for a detailed account of how the diameter is determined from the vehicles D-value defined in EASA VTOL PTS [4, Figure 1] and FAA's Vertiport Design Guidance [49, Figure 1.1]. That is, we modeled final-approach and take-off areas (FATOs) as radius-10m disks, and for every suitable flat roof surface we counted how many such disks can be placed there – that number is the capacity of the vertiport. We then build the vertiport network G by connecting two vertiport vertices (located in the center of the corresponding roofs) by an edge iff the distance between them is at most $L = 2500$m: this ensures that a drone, routed in G, is always within L from the next vertiport (in accordance with our model from Section II). In order to make the network more realistic, we removed all edges that go in proximity of public and sensitive buildings such as, e.g., schools, kindergartens, police stations, and other.

To generate the demand, we chose random origin–destination pairs and assumed uniform distribution of the desired departure times in a 3-hour time window. We ran experiments on Tetralith supercomputers [50]. Since the runtime of the greedy algorithm was negligible compared to solving the IP (seconds vs hours/days), we used the greedy solutions as a "warm-start" for the IP solver (in our case — Gurobi [51]). The source code and input data are available at [52].

B. Results

We used $\Delta t = 120$ seconds. Even with such a (arguably large) value, the size of the IP for the entire vertiport network in Fig. 7 is very large to be solved to optimality in reasonable time. We therefore conducted two sets of experiments. In the first set we focus on the smaller part of the vertiport network (yellow vertices connected with red edges on Fig. 7) where we solve the IP to optimality — we dubbed this the partial-network scenario. In the second set of experiments we deal with the full vertiport network, but allow optimality gap and stop the IP solver after a timeout instead. We note that handling the same number of drones in the partial network is more challenging than in the full network because flights are routed on a much smaller network, therefore creating more demand for every vertiport.

1) Partial network scenario: Figure 8 shows the average and total travel time extension (i.e., the objective function) of the routed drones. While at lower traffic intensities the greedy approach performs on par with the IP, as the traffic intensity increases, IP becomes twice better. Figure 9 presents the percentage of drones whose travel time got extended by more than 20min. The longest extension among all drones is shown on Figure 10.

2) Full network: Figure 11 presents the travel time extension as a function of the traffic intensity. While the IP performs better than greedy, the difference is smaller than for the partial network. Figures 12 and 13 show the percentage of drones with extension >20min and the longest extension resp.

The results show that our IP can handle large instances, but does not always solve them to optimality.

V. Conclusion and Future work

We considered routing UAM vehicles with robust contingency plans and proposed two approaches for achieving the robustness. The greedy approach is relatively quick and allows working even with very high demand and big vertiport

$$\min \quad \sum_d \left(\left[\sum_{(i,j)\in E^*} \sum_t x_{ijdt} \bar{w}(i,j,d) \right] - \mu_d \right) \tag{1a}$$

$$\text{s.t.} \quad \sum_{(i,k)\in E} y_{ikdt} - \sum_{(k,j)\in E} x_{kjdt} \begin{cases} = -1 & k = v_{orig}(d), t = t_{departure}(d) \\ \leq 1 & k = v_{dest}(d) \\ = 0 & \text{otherwise} \end{cases} \quad \forall t, d, k \tag{1b}$$

$$r_{idt} \geq \sum_{k\in V} \sum_{\tau\in\{\max(0,t-U-\bar{w}(k,i,d)),\ldots,t\}} x_{kid\tau}/M \qquad \forall i, d, t \tag{1c}$$

$$\sum_d r_{idt} \leq c_i \qquad \forall i, t \tag{1d}$$

$$\sum_{(i,j)\in E} \sum_{t\in 1,\ldots,t_{departure}(d)} x_{v_{orig}(d)jdt} = 0 \quad \forall d \tag{1e}$$

$$\sum_{\forall (i,k)\in E} \sum_t x_{ikdt} = 1 \quad \forall d, k = v_{dest}(d) \tag{1f}$$

$$y_{ijdt} = x_{ikd(t-\bar{w}(i,j,d))} \quad \forall i, j, d, t \tag{1g}$$

Fig. 6. Integer programming formulation of our optimization problem

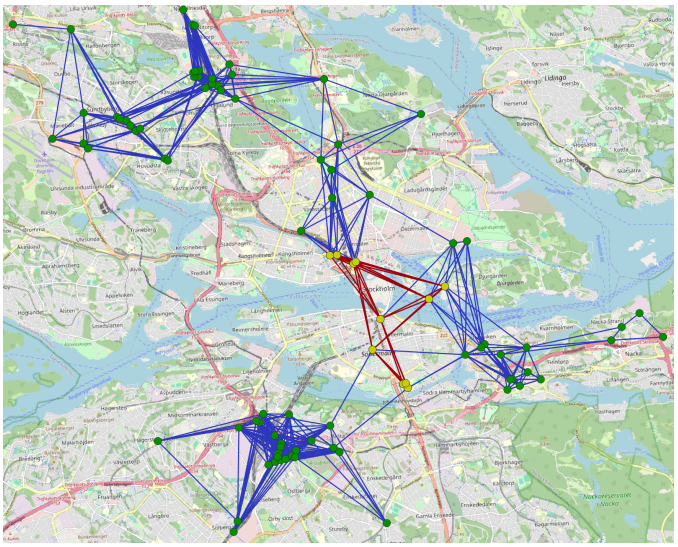

Fig. 7. A possible vertiport network in Stockholm: Yellow vertices connected by red edges is a smaller induced subgraph used for the partial-network scenario in Section IV-B1, while the entire graph containing all vertices was used in the full-network scenario (Section IV-B2)

networks; however it provides no guarantees on how close to the optimum the solution is. On the other hand, IP is capable of obtaining the optimal solution on moderate-size networks, but requires extensive calculation effort. Our solutions can be used together, with the greedy algorithm generating an initial solution, and IP improving upon it. On the practical frontier, our somewhat simplistic model lacks many fine details; in particular, we did not take into account interaction between drones flying along the same edge of the network, as well as physics of take-off and landing – future studies may consider adding these to the framework.

Another possible research direction is to develop a heuristic algorithm for routing. In particular, one may explore how the order in which drones are routed by the greedy algorithm affects its efficiency — touching upon the novel concept of Reasonable Time to Act (RTTA) introduced in the CORUS ConOps [2], [53]. In particular, RTTA is envisioned to improve fairness of the operations: our objective of minimizing the total delay may lead to a solution in which one flight is delayed a lot, while others have small delays; a more fair distribution could be achieved by minimum-regret optimization that minimizes the largest delay incurred by a flight.

Finally, we had a conservative safety model in which UAVs continue flying forward to the alternate landing location. Instead, higher navigational capabilities can be assumed in case of an emergency, allowing the drone to change the heading for flying to a pre-determined alternate landing location.

VI. Acknowledgements

We thank the anonymous reviewers for their helpful comments. This work was partially supported by the Swedish Transport Administration (Trafikverket), and by Norrköping Municipality's fund for Research and Development. Part of the work was done during research visit by LS to Kobe University, supported by Scandinavia–Japan Sasakawa Foundation. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at NSC Tetralith partially funded by the Swedish Research Council. We thank Billy Josefsson (LiU) for consultations.

References

[1] EASA, "Special Condition for small-category VTOL aircraft ," 2022.
[2] SESAR, "Concept of Operations for European U-space Services - Extension for Urban Air Mobility (CORUS-XUAM)," 2023.

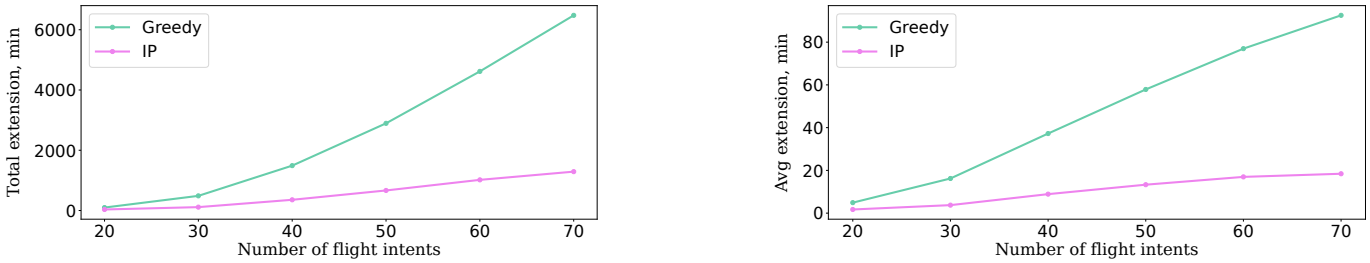

Fig. 8. Travel time extension in the partial network as function of traffic intensity. Left: total extension (the objective function). Right: mean extension

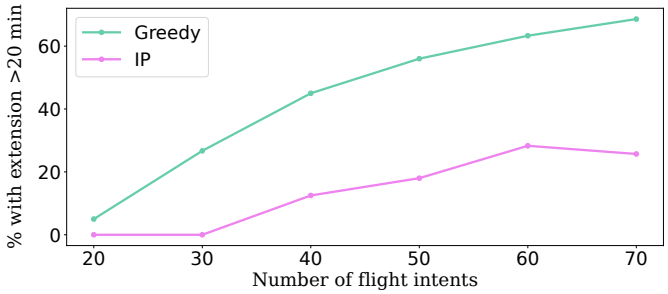

Fig. 9. Percentage of drones with travel time extension >20min in the partial network scenario as function of the traffic intensity

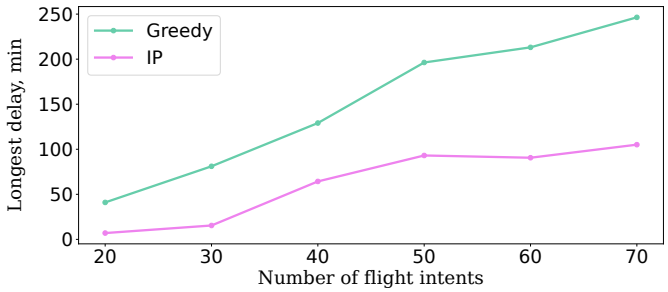

Fig. 10. The highest observed travel time extension in the partial network scenario as a function of traffic intensity

[3] EC, "Commission implementing regulation (eu) 2021/664 of 22 april 2021 on a regulatory framework for the u-space," 2021.

[4] EASA, "Prototype Technical Specifications for the Design of VFR Vertiports for Operation with Manned VTOL-Capable Aircraft Certified in the Enhanced Category," 2022.

[5] ICAO, *Extended Diversion Time Operations (EDTO) Manual*, 2017, document Number 10085. [Online]. Available: https://www.icao.int/MID/Documents/2020/EDTO%20Workshop/10085_cons_en.pdf

[6] Y. Tang, Y. Xu, G. Inalhan, and A. Tsourdos, "An integrated approach for dynamic capacity management service in u-space," in *14th USA/Europe ATM R&D Seminar*, 2021.

[7] B. Josefsson, K. Huynh, A. Lemetti, V. Polishchuk, and L. Sedov, "Geometric vertiport localization: Uam procedural envelopes in a cityscape," in *Digital Avionics Systems Conference*, 2025.

[8] Y. Lu, W. Zeng, W. Wei, W. Wu, and H. Jiang, "Vertiport location selection and optimization for urban air mobility in complex urban scenes," *Aerospace*, vol. 12, no. 8, p. 709, 2025.

[9] A. la Cour-Harbo and H. Schiøler, "Probability of Low-Altitude Midair Collision Between General Aviation and Unmanned Aircraft," *Risk Analysis*, vol. 39, no. 11, pp. 2499–2513, 2019.

[10] X. Olive, P. Le Blaye, L. Sedov, and T. Krauth, "Quantitative air risk assessment for a drone inspection mission along fast-train lines," in *ATM Seminar*, Jun. 2023.

[11] A. la Cour-Harbo, "Quantifying risk of ground impact fatalities for small unmanned aircraft," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 1-2, pp. 367–384, 2019, iSBN: 0921-0296 Publisher: Springer.

[12] ——, "Ground impact probability distribution for small unmanned aircraft in ballistic descent," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 1442–1451.

[13] S. Primatesta, A. Rizzo, and A. la Cour-Harbo, "Ground Risk Map for Unmanned Aircraft in Urban Environments," *Journal of Intelligent & Robotic Systems*, vol. 97, no. 3, pp. 489–509, Mar. 2020.

[14] L. Sedov, V. Polishchuk, T. Maury, M. Ulloa, and D. Lykova, "Qualitative and quantitative risk assessment of urban airspace operations," in *SESAR Innovation Days (SID 2021)*, 2021.

[15] L. Sedov, V. Polishchuk, and V. Bulusu, "Ground risk vs. efficiency in urban drone operations," in *ATM Seminar*, 2021.

[16] P. Ogren and M. Winstrand, "Combining path planning and target assignment to minimize risk in sead missions," in *AIAA guidance, navigation, and control conference and exhibit*, 2005, p. 5865.

[17] T. Erlandsson and L. Niklasson, "Automatic evaluation of air mission routes with respect to combat survival," *Inf Fusion*, vol. 20, 2014.

[18] T. Erlandsson, "A combat survivability model for evaluating air routes in future decision support systems," Ph.D. dissertation, Örebro U, 2014.

[19] K. Buchin, V. Polishchuk, L. Sedov, and V. Roman, "Geometric secluded paths and planar satisfiability," in *Symp on CompGeom (SoCG)*, 2020.

[20] F. Iqbal and F. A. Kuipers, "Disjoint paths in networks," *Wiley Encyclopedia of Electrical and Electronics Engineering*, vol. 4, 2015.

[21] N. Robertson and P. D. Seymour, "Graph minors. xiii. the disjoint paths problem," *J comb theory*, vol. 63, no. 1, pp. 65–110, 1995.

[22] W. Lochet, "A polynomial time algorithm for the k-disjoint shortest paths problem," in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2021, pp. 169–178.

[23] T. Eilam-Tzoreff, "The disjoint shortest paths problem," *Discrete applied mathematics*, vol. 85, no. 2, pp. 113–138, 1998.

[24] S. Eriksson-Bique, J. Hershberger, V. Polishchuk, B. Speckmann, S. Suri, T. Talvitie, K. Verbeek, and H. Yıldız, "Geometric k shortest paths," in *ACM-SIAM Symp on Discr Algorithms*, 2014, pp. 1616–1625.

[25] B. Hoch, F. Liers, S. Neumann, and F. J. Z. Martínez, "The non-stop disjoint trajectories problem," *Preprint. url: http://www. optimizationonline. org/DB_HTML/2020/09/8015. html*, 2020.

[26] S.-Y. Tan and W.-C. Yeh, "The vehicle routing problem: State-of-the-art classification and review," *Appl Sci*, vol. 11, no. 21, p. 10295, 2021.

[27] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Computers & industrial engineering*, vol. 99, pp. 300–313, 2016.

[28] H. Zhang, H. Ge, J. Yang, and Y. Tong, "Review of vehicle routing problems: Models, classification and solving algorithms," *Archives of Computational Methods in Engineering*, pp. 1–27, 2021.

[29] B. Kallehauge, J. Larsen, O. B. Madsen, and M. M. Solomon, *Vehicle routing problem with time windows*. Springer, 2005.

[30] J. Jarvis and D. Ratliff, "Some Equivalent Objectives for Dynamic Network Flow Problems," *Mgmt sci*, vol. 28, no. 1, pp. 106–109, 1982.

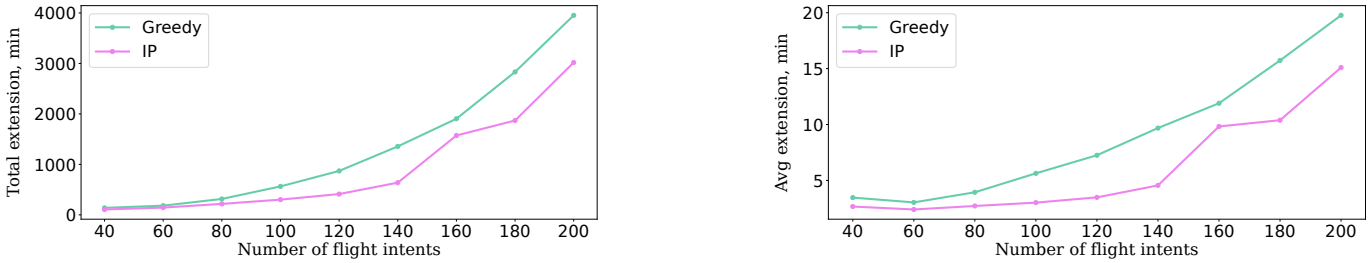

Fig. 11. Travel time extension in the full network as function of the intensity. Left: the total extension (the objective function). Right: the average extension

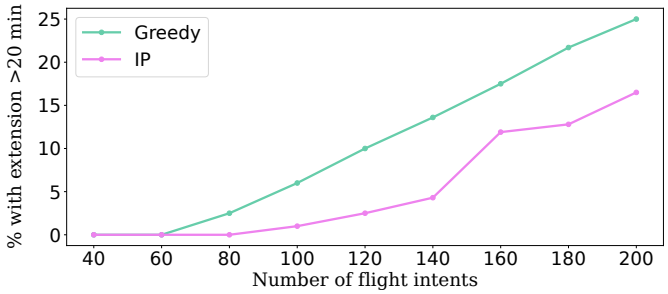

Fig. 12. Percentage of drones with travel time extension >20min in the full network scenario as function of the traffic intensity

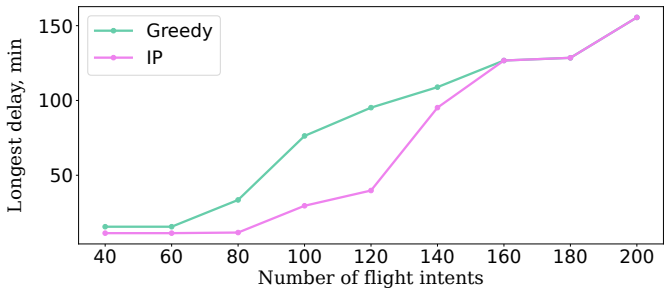

Fig. 13. The highest travel time extension in the full network scenario as a function of the traffic intensity

[31] J. Kim, J. S. B. Mitchell, V. Polishchuk, and A. Vihavainen, "Routing a maximum number of disks through a scene of moving obstacles," in *SoCG*, M. Teillaud, Ed. ACM, 2008, pp. 230–231.

[32] E. M. Arkin, J. S. B. Mitchell, and V. Polishchuk, "Maximum thick paths in static and dynamic environments," *Comput. Geom.*, vol. 43, no. 3, pp. 279–294, 2010.

[33] J. Krozel, J. S. Mitchell, A. Pääkkö, and V. Polishchuk, "Throughput/complexity tradeoffs for routing traffic in the presence of dynamic weather," in *ICRAT*, 2010.

[34] M. Biro, J. Gao, J. Iwerks, I. Kostitsyna, and J. S. Mitchell, "Beacon-based routing and coverage," in *21st Fall Workshop on Computational Geometry (FWCG 2011)*, 2011.

[35] S. W. Bae, C.-S. Shin, and A. Vigneron, "Tight bounds for beacon-based coverage in simple rectilinear polygons," *Computational Geometry*, vol. 80, pp. 40–52, 2019.

[36] M. Biro, J. Gao, J. Iwerks, I. Kostitsyna, and J. S. Mitchell, "Combinatorics of beacon-based routing and coverage," in *25th Canadian Conference on Computational Geometry (CCCG 2013)*, 2013, pp. 1–6.

[37] M. Biro, "Beacon-based routing and guarding," Ph.D. dissertation, State University of New York at Stony Brook, 2013.

[38] J. Cleve, W. Mulzer, and F. Hoffmann, "Combinatorics of beacon-based routing and guarding in three dimensions," *Master's thesis, Freie Universität Berlin*, 2017.

[39] J. Cleve and W. Mulzer, "Combinatorics of beacon-based routing in three dimensions," *Computational Geometry*, vol. 91, p. 101667, 2020.

[40] I. Aldana-Galván, J. L. Álvarez-Rebollar, J. C. Catana-Salazar, N. Marin-Nevárez, E. Solís-Villarreal, J. Urrutia, and C. Velarde, "Beacon coverage in orthogonal polyhedra." in *CCCG*, 2017, pp. 156–161.

[41] V. Angelakis, A. Efrat, E. Packer, V. Polishchuk, and L. Sedov, "Bbtm: New life for old atm paradigms," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE, 2016, pp. 1–10.

[42] ICAO, *PANS Doc 8168*, 2018, https://skybrary.aero/articles/waypoint.

[43] FAA. (2025) AIP ENR 1.16: PBN and RNAV.

[44] E. Sunil, J. Hoekstra, J. Ellerbroek, F. Bussink, D. Nieuwenhuisen, A. Vidosavljevic, and S. Kern, "Metropolis: Relating airspace structure and capacity for extreme traffic densities," in *ATM seminar 2015, 11th USA/EUROPE Air Traffic Management R&D Seminar*, 2015.

[45] L. Sedov and V. Polishchuk, "Centralized and distributed utm in layered airspace," in *ICRAT*, 2018, pp. 1–8.

[46] H. Joksch, "The shortest route problem with constraints," *J. of Mathematical Analysis and Applications*, vol. 14, no. 2, pp. 191–197, 1966.

[47] K. Wigler, "AiRMOUR D3.2 Ground risk and landing site management (prepared vs ad hoc)," 2022, https://airmour.eu/deliverables/.

[48] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org, 2017.

[49] FAA, "EB 105A, Vertiport Design, Supplemental Guidance to AC 150/5390-2D, Heliport Design," 2024.

[50] NAISS, "Tetralith server, NSC, Linköping University," https://www.nsc.liu.se/systems/tetralith/.

[51] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2025.

[52] H. Hasan and L. Sedov, 2022, https://github.com/undefiened/balance.

[53] SESAR, "ConOps for EuRopean UTM Systems - CORUS," 2019.