Throughput/Complexity Tradeoffs for Routing Traffic in the Presence of Dynamic Weather

Jimmy Krozel
Metron Aviation, Inc.
Dulles, VA 20166, USA
krozel@metronaviation.com

Joseph S.B. Mitchell
Applied Math and Statistics Department
Stony Brook University
Stony Brook, NY 11794, USA
jsbm@ams.sunysb.edu

Anne Pääkkö and Valentin Polishchuk
Computer Science Department
University of Helsinki
Helsinki, FI-00014, Finland
{apaakko,polishch}@cs.helsinki.fi

Abstract—We present efficient algorithms for finding trajectories for routing multiple aircraft avoiding a set of static or dynamic obstacles (e.g., hazardous weather cells). We present results of an implementation of our algorithms, comparing the throughput and traffic complexity across three routing paradigms:

Static airlanes: A set of lanes for air traffic is established. The aircraft move in trail along each lane, altogether forming a highly coherent traffic pattern. The drawback is that the lanes, being static, do not stay clear of hazardous weather as the weather cells move and block the airline.

FreeFlight: Each aircraft determines its own trajectory in space-time, avoiding moving weather cells and other aircraft. This strategy can result in highly complex traffic patterns that are not amenable to human controller oversight.

Flexible Flow Corridors: This model combines the advantages of the static airlanes and the FreeFlight solution. The aircraft are routed along a set of lanes that slowly change as the weather cells move. This results in a correlated traffic flow amidst moving weather.

Our routing algorithms are guided by a hexagonal packing of disks in free space, and a uniform discretization of time. The discretization allows us to take into account additional requirements relevant for Air Traffic Management (ATM).

I. INTRODUCTION

A fundamental task in Air Traffic Management (ATM) is to plan a large number of flight trajectories through a weather-impacted airspace. The capacity of the airspace for a given period of time (planning horizon) is defined as the maximum number of aircraft that can be routed through the airspace during the period, while avoiding hazardous weather cells and respecting the separation standards. This paper investigates the capacity of the airspace and the associated traffic complexity under three routing paradigms: Static Airlanes, Free Flight, and Flexible Flow Corridors. When humans oversee ATM traffic flows, complexity as well as weather hazards will together be the limiting factor in throughput, motivating our study to focus on both throughput and complexity as a function of the amount of weather constraints.

Static airlanes

The simplest solution to the routing problem is to establish a set of airlanes along which the aircraft travel while respecting the miles-in-trail (MIT) requirements (Fig. 1, left).

Drawback – lanes are static: The airlanes solution is good only in the case of static obstacles, as is the case, e.g., with static Special Use Airspace (SUA) constraints. At the same time, the major impact on the capacity of the airspace comes from the weather [1], [2]. Convective weather cells are not static, and as they move they may intersect (and hence – make infeasible) some of the airlanes. Thus, the static airlanes can only serve as a solution on a clear day or during certain periods of time when weather is not overlapping the route structure.

FreeFlight

The aircraft do not have to use a predefined set of airlanes; instead, each aircraft is cleared to fly on its own trajectory (Fig. 1, middle), usually selected in order to take advantage of winds and optimize fuel consumption. The trajectories never intersect among themselves or with the obstacles. Because the planning is done in space-time, the trajectories are designed to avoid moving obstacles (provided the obstacles’ motion is well predicted from the weather forecast).

Drawback – no well-defined airlanes: Prescribing an individual trajectory for each aircraft is far beyond the currently available systems functionality, which are not yet ready for full automation. This is due, in particular, to the presence of humans-in-the-loop – air traffic controllers are generally not able to control more than 12-15 trajectories simultaneously crossing one controller’s airspace (sector). Thus, a FreeFlight solution, in which the trajectories of different aircraft can in
principle be totally uncorrelated, may turn into an “air traffic controller (ATC) nightmare” – a complicated set of paths tangled in space-time. A controller would prefer the majority of aircraft to follow each other in an orderly fashion, in trail, along a set of (more or less) stable routes – much like it happens in the static-airlanes solution.

**Flexible Flow Corridors**

Flexible Flow Corridors combine the best of both worlds: they feature organized flows and low-complexity traffic amidst moving obstacles. The corridors are thick paths that morph slowly as the obstacles move (Fig. 1, right). Every flow corridor, while morphing, maintains its “threading” through the obstacles. That is, the obstacles that are below the path never “jump” above it, and vice versa. This allows a controller to issue pilot instructions like “Stay north of obstacle 1 but south of obstacle 2”; the instructions remain valid for the whole planning horizon time period.

**Motivation:** One of the themes of SESAR’s [3] long-term and innovative research (Work Package E [4]) is to explore the possibilities of shifting towards full automation in ATM. In line with this task, we estimate the capacity ignoring the possible presence of human-in-the-loop (which corresponds to the full Automation Level 10 [5]). On the other hand, a central role for the human, widely supported by advanced tools to work safely and without undue pressure, is stated as a key feature of the SESAR operational concept. Caring about human factors by bounding the complexity of traffic which a controller will have to monitor, served as our motivation for studying the flexible flow corridors.

A. Related work

Capacity estimation was traditionally done via empirical analysis of controllers’ practices [6], [7], [8], [9], [10]. When weather hazards interfere with static routes, for instance, route blockage or route availability may be determined by analyzing the pilot deviations that are allowed by ATC relative to convective weather cells that overlap jet routes [11]. That is, the capacity estimation was human-centered. In contrast, we analyze the capacity independently of workload considerations and existing jet routes. We first compute the flow rates ignoring the controllers’ workload, and only then measure the complexity of the obtained traffic patterns to see if the complexity is within the controller’s workload limits.

Other related work includes [12], where machine learning techniques were applied to identify routes that survive given the inaccuracy of weather forecasts. Sohier, Bui, and Duong [13] used a packing similar to ours. A grid-based approach to en route dynamic weather avoidance was also studied in [14].

**B. Our contributions**

This paper is a continuation and extension of our prior research on airspace capacity estimation [15], [16], [17], [18]. The novelty of our investigation in comparison with the prior work is four-fold:

- **We consider dynamic weather hazards;** in [15] the weather hazards were static.
- **For static airlanes,** in earlier work only algorithms for computing the maximum number of airlanes were implemented; here, we report on algorithms that also produce the lanes themselves. Moreover, we find a set of shortest lanes. In addition, we produce a set of “conforming” routes, taking into account the geometry of the airspace.
- **To find the paths for FreeFlight,** previously, the Flow-Based Route Planner (FBRP) [19] was used; here, we implement the algorithm of [18] which allows us to compute trajectories in 3D space-time (i.e., \((x, y, t)\)) whose projections onto 2D space \((x, y)\) may intersect each other. (Similarly, FBRP computes paths in space-time, but does so iteratively, potentially trying different insertion orders; however, it does not have the same theoretical guarantees as the algorithm of [18] that we implemented for our study.) In addition, we show how to use our FreeFlight algorithm to execute holding/airborne delay.
- **We study Flexible Flow Corridors;** in [15] only static airlanes and FreeFlight were considered. A minor technical difference is that we use a triangular grid (as opposed to square grid) to discretize the domain.
C. Paper outline

The next section discusses the preliminaries. In Sections III, IV and V we report on the experiments with the static airplanes, FreeFlight, and flexible flow corridors respectively. Section VI compares the throughput and complexity under the three routing paradigms. In Section VII we present extensions of our algorithms, addressing further specifics of motion planning for ATM: we show how to compute routes conforming to sector geometry, and how to use our algorithms to execute holding.

II. MODELING

A. Airspace boundaries

Our focus is on en-route airspace at a constant flight level. In the basic setting, the airspace is modeled as a 300nmi-by-210nmi rectangle; later, we also consider general-shaped airspace representing a sector, Center, Flow Evaluation Area (FEA), or Flow Constrained Area (FCA). The traffic enters the airspace through the West (left) side of the rectangle and must exit through the East (right) side. This assumption of mostly unidirectional traffic is justified by the “Alternating Altitude Rule” according to which the flow with West-to-East heading is altitude-separated from the East-to-West traffic.

B. Airspace constraints

The term obstacle is used for any region through which flying is not permitted; we do not differentiate between the obstacles induced by the no-fly zones and the constraints induced by hazardous weather cells (possibly, with added safety margins). The Weather Severity Index (WSI) of an airspace is defined as the fraction of its area that is occupied by the obstacles. We experiment with two types of obstacles’ organization: Popcorn Convection (PC) and Squall Line (SL). In the first one (PC), thunderstorms form on a scattered basis, e.g., in the afternoon in response to diurnal heating. The second type (SL) is a solid or nearly solid line or band of active thunderstorms.

To create an instance of the routing problem, we populate the airspace with obstacles until reaching the desired WSI. The obstacles are generated using a common distribution across all severity levels; this way the structure of the obstacles can be expected to be similar even as the severity is varied. Each obstacle is a random polygon with 4 to 6 vertices generated in a 40nmi-by-40nmi square (Fig. 2). After the obstacle has been generated, we place it uniformly at random in the airspace: for PC, the obstacle is placed anywhere within the airspace, for SL – only within a vertical band of width 66nmi in the middle of the airspace. Finally, for each obstacle we choose uniformly at random the direction of motion and the speed of motion (this results in a more complicated obstacles motion than is expected in a real-world scenario where the direction of motion of different obstacles are correlated due, e.g., to the wind).

Overall, we experimented with 100 instances for each WSI; for PC we used WSI = 10%, 20%, 30%, 40%, 50%, 60%, 70%, for SL — WSI = 5%, 10%, 15%, 20%, 25%, 30%, 35% (larger WSIs are not possible for SL in our model).

Although we used synthetic weather constraints in the experiments, our algorithms are applicable to real weather data as well. Given a snapshot of real weather, the user may lasso the regions that must be treated as obstacles, or threshold the weather forecast data at a user-specified level based on the observed severity of the weather cells. This is in line with current practices for some airlines, where dispatchers may draw their own boundaries around regions of hazardous weather after viewing convective weather forecasts or turbulence forecasts. The reason for such practices is that there is no straightforward objective criteria to understand which weather cells serve as obstacles for the traffic. In fact, determining the boundaries of hazardous weather is an active research area in ATM; deciding the areas that should be avoided by aircraft is influenced by a multitude of factors – weather cells shape and structure, accuracy of prediction, pilot preference and experience, airline policies, the altitude of radar return echo tops in severe storms, etc. [20], [21], [22].

C. Aircraft trajectories

In our experiments all aircraft travel at a constant speed of 420 kn. The Required Navigation Performance (RNP) requirement for every aircraft is 5nmi (that is, each aircraft can deviate by 5nmi from the route centerline). We assume 10nmi MIT separation. This means that at peak throughput a single airplane can carry 42 aircraft/h past any point in space. Naturally, packing the aircraft “head to tail” is not practical nor realistic. We use this tight packing because our focus is on determining the maximum theoretically possible throughput rates and associated traffic complexity.

Assuming RNP = 5nmi, MIT = 10nmi allows us to model each aircraft as a disk of 10nmi in diameter; the disk represents the protected airspace zone (PAZ) around the aircraft. During the motion, the disks have to stay disjoint from each other and the obstacles. Note that we allow the disks to come arbitrarily close to the obstacles (but not to penetrate them); this is because we assume that the hazardous weather regions have safety margins added to them by the user.

It is natural to require that the flight trajectories be monotone in the direction from origin to destination (Fig. 3, left). A flight trajectory will generally never head in the reverse direction
The complexity in the square variance, calculated as $\sum p, t C(p,t) = 0.36\text{Var}(p,t) + 2|A(p,t)|$ where $\text{Var}(p,t)$ is the “scaled-contribution” velocity variance, calculated as $\text{Var}(p,t) = \sum_{a \in A(p,t)} s_a ||v_a - V_{avg}||^2$. Here, $v_a$ is the velocity of the aircraft $a$, $s_a = 1 - |ap_c|/R$ is the scaling factor for $a$ ($p_c$ is the center of $p$), and $V_{avg} = \frac{\sum_{a \in A(p,t)} s_a v_a}{\sum_{a \in A(p,t)} s_a}$ is the (scaled) “local average velocity”. The final expression for the traffic complexity is

$$\text{complexity} = \frac{1}{30} \sum_{t=1}^{30} \sum_{p} C(p,t)$$

III. STATIC AIRLANES

The basic routing problem asks for a maximum number of disjoint obstacle-avoiding “thick” paths connecting the source and the sink edges (see Fig. 1, left). Formally a thick path is the Minkowski sum of a usual (thin) path and the disk centered at the origin; the radius of the disk is equal to the RNP. The paths serve as airlanes for the traffic flow.

**Theoretical solution from prior work:** In [18], a continuous-Dijkstra algorithm for computing the maximum number of thick paths in a polygonal domain was suggested. The algorithm runs as follows. The source and the sink edge split the boundary of the outer polygon of the domain into two parts: the top $T$ and the bottom $B$ (Fig. 3, right). The paths are routed in a topmost fashion: the first path runs “as close as possible to $T$”, the second – as close to the first as possible, and so on (see Fig. 3, right).

While routing uppermost (or bottommost) paths guarantees that a maximum number of paths will be found, no bound on the length of the paths is possible. Another issue is that the algorithm can be hard to implement since the efficient implementation of the continuous Dijkstra method involves wavefront tracking, clipping, intersection, etc.

**Our solution:** We address both of the above issues by discretizing the domain. We start with a hexagonal packing of congruent disks and remove disks intersected by the obstacles. Next, a graph $G$ is formed whose nodes are identified with the disks, and whose edges connect adjacent disks; the graph is a triangular subgrid (Fig. 4).

We connect the disks along the source and the sink to a supersource node and a supersink node and compute a maximum supersource-supersink flow. By the Flow Decomposition Theorem [29], the flow decomposes into a maximum number of disjoint paths in the graph. Moreover, by computing the minimum-cost flow, we find a set of shortest paths.

Figure 10, left, presents a sample output of our algorithm.

A. Throughput and Complexity

Using the above algorithm, for each problem instance we compute the maximum number of airlanes routable through the airspace at time 0. We then check how many airlanes stay clear of the obstacles during the planning horizon. The throughput is calculated as the number of open airlanes multiplied by the rate of the flow along one lane (42 aircraft/hour) multiplied by the planning horizon length (.5 hours).

The complexity is calculated as described in Section II-D. Table I presents the results.

IV. FREEFLIGHT

We implemented the algorithm from [18], which finds the maximum number of aircraft trajectories that can be routed through the airspace during the planning horizon by computing a maximum flow in the “motion graph” laid out in the $(x,y,t)$-space (Fig. 5, top). Videos with the output of the implementation can be viewed at our webpage tinyurl.com/ATMexamples; Fig. 5, bottom shows a screenshot.

1tinyurl.com/ATMexamples is a shorter alias for http://www.cs.helsinki.fi/group/compgeom/examples.html
Formally, let $\Pi = (\Pi_1, \ldots, \Pi_K)$ be the uppermost paths. Start with the path $\Pi_K$, and fix its threading by bridging the obstacles may move and grow/shrink but do not appear or disappear, so the threading is well defined.

Videos with the algorithm’s output can be viewed at tinyurl.com/ATMexamples. Figure 6 compares the FreeFlight solution with the Flexible flow corridors on the same instance.

Computing the corridors: To compute the corridors we first route uppermost paths at every time slice. As with static airlines, uppermost paths tend to be unduly long, so we shorten them iteratively and greedily in the “bottommost” fashion: the last (bottommost) path is “pulled taut” treating the next-to-last path as an obstacle, then the next-to-last path is pulled taut treating the path just above it as an obstacle, and so on. Formally, let $P = (\Pi_1, \ldots, \Pi_K)$ be the uppermost paths. Start with the path $\Pi_K$, and fix its threading by bridging the obstacles.
that are above (resp. below) $\Pi_K$ to $T$ (resp. $B$); refer to Fig. 7. Now replace $\Pi_K$ with the shortest path, $\Pi^*_K$, in the free space between $B$ and the path $\Pi_{K-1}$. (Because the uppermost paths are routed conservatively, in any collection of $K$ paths, the lowest, $K$th, path cannot intersect $\Pi_{K-1}$; thus, routing $\Pi^*_K$ as we do seems like a natural idea.) We proceed with the $(K-1)$st path: after fixing its threading, route the shortest path, $\Pi^*_{K-1}$, in the free space between $\Pi^*_K$ and $\Pi_{K-2}$. Continuing this way, we obtain a collection $P^* = (\Pi^*_1, \ldots, \Pi^*_K)$ of shorter paths.

Solution uniqueness: In our implementation, the shortest paths were routed by finding the shortest path in the underlying grid (dual to the disk packing). While in general working with the grid allowed us to overcome many of the difficulties present in continuous versions of the algorithms, shortest paths between two nodes in the grid tend not to be unique. In some instances, this led to the paths “jumping back and forth for no reason” between the neighboring time slices, undermining the very idea of slow morphing.

We thus had to enforce shortest paths uniqueness by perturbing the weights of the grid edges. Easiest to implement was random perturbation; this, however, resulted in the paths looking somewhat random, often making unnatural zigzags. A more consistent perturbation was obtained by introducing a “gravity field”: the edges were heavier towards the middle of the domain thus favoring the flow corridors to run through the middle. This way not only we have enforced the path uniqueness, but also took into account typical controller’s preferences for working with the flights well within the boundaries of their sectors and trying to avoid letting traffic flows get too close to the boundaries.

VI. CONCLUSIONS

Free Flight allowed us to achieve the highest capacity, followed by the flexible flow corridors, followed by the static airlanes (Table I). This is to be expected, since any static airline that remains open for the planning horizon, is also a feasible corridor (which actually does not morph at all). In turn, a trajectory along a flexible flow corridor is also a feasible FreeFlight trajectory.

As expected, the traffic complexity shows the trend opposite to the capacity (Table I): FreeFlight has highest complexity, followed by the flexible corridors, followed by the static airlanes.

In Fig. 8 the throughput is plotted against the capacity at different WSIIs. Overall, we conclude that the throughput and complexity of traffic following the flexible flow corridors is only slightly higher than that of the static airlanes; the throughput and complexity for the FreeFlight is in general considerably higher.

As a trade-off between throughput and complexity clearly exists, researchers, policy makers and operations personnel (controllers and pilots) need to weigh the compromises be-
between maximizing capacity (throughput) and the potential for complexity-driven safety mishaps. Our research helps to quantify the trade-offs, and allows one to make the decision on when the complexity is the limiting factor, as well as to estimate the consequences of limiting the capacity due to the complexity being too high.

VII. EXTENSIONS

We emphasize that our algorithmic solutions are applicable to airspaces of various shapes and purposes, without the simplifying assumptions inherent to our basic model. The airspace does not have to be rectangular, but may have an arbitrary geometry. In particular, in Section VII-A below, we consider the case when the airspace represents several adjacent sectors, each with a complicated geometry. We care about the fact that crossing the sector boundary involves a “hand-off” between the sectors controllers. To reduce the communication overhead, we plan flight paths that minimize the number of sector boundary crossings.

Treating the airspace as a 2D \((x, y)\) region makes computing permanent airlanes amidst static obstacles a two-dimensional problem. Taking into account obstacles’ motion translates the problem into 3D \((x, y, t)\). Our dynamic motion planning algorithms for this 3D problem (Section IV) can be extended to plan routes in the full 4D \((x, y, z, t)\) as well. In addition, the algorithms work in the case when the obstacles do not just move retaining their shape, but also grow and shrink.

Entering/ exiting the airspace only through its West/East side is relevant for en-route airspaces where traffic with different headings occupies different flight levels. We can also assume general source/sink areas, including those inside the domain. In particular, in Section VII-B below, we consider an instance where the aircraft are initially placed in the airspace interior (and have to exit through the East).

As one other possible application domain for our algorithms, consider terminal area operations. The transition airspace can be modeled by a portion of an annulus corresponding to one arrival metering fix (Fig. 9). The source/sink edges are then arcs of the outer/inner range rings – typically of radius about 150nm/40nm. While the vertical dimension is crucial here, the 2D \((x, y)\) projection of a full 3D \((x, y, z)\) trajectory may serve as a sufficient specification of the path when the plane follows a predefined descent profile or the altitude is specified along the way-points of the path [31], [32], [33]. In this scenario, in addition to the weather cells and no-fly-zones, the obstacles may represent human-defined “poke-throughs” for ascending/descending traffic following departure routes from an airport or from satellite airports.

A. Conforming flows

The discretization allows us to address additional, ATM-specific constraints. For instance, our airspace may consist of several air sectors, with each sector controlled by a separate controller. Crossing a boundary between adjacent sectors involves a “hand-off” between the controllers. Thus it is of interest to find paths that conform to the sectors’ geometry by not crossing the boundaries too often. This is easily achieved in our setting by assigning high costs to those edges of the graph \(G\) that cross the boundaries. Figure 10 presents an example.

Moreover, it is preferable that the paths cross the boundary at an angle close to 90°. This can be taken into account by having the cost of the crossing edges depend on the angle of the crossing.

B. Holding/Airborne delay

Finding a large number of aircraft trajectories, as our FreeFlight implementation does, is already not an easy task for a controller. It would be even harder for a human to “hold” a set of aircraft within the airspace for some time – such a need arises when the adjacent space downstream the flow is temporarily blocked (e.g., due to equipment failure, a security event, excessive congestion, or extreme growth of weather constraints). Interestingly, the algorithm of [18] allows the user to specify the entry (resp. exit) time interval for the aircraft; the disks can appear in (resp. leave) the airspace only during the entry (resp. exit) interval. Thus potentially, one may use the algorithm to add airborne delay to air traffic by simply making the exit time interval start later than the entry interval ends.

It appears though that in a straightforward implementation of the algorithm, the aircraft have the ability to “fool” the user: they all enter the airspace during the entry time interval, then each aircraft jiggles back and forth between two neighboring disks in the packing, and finally all aircraft exit during the exit interval (see Fig. 11). We remove the jiggling by forbidding the corresponding subpaths during the breath-first search (BFS) in Edmonds-Karp’s algorithm [34, p. 660] which we use for finding the maximum flow in the motion graph.

The output of our holding pattern implementations can be viewed at tinyurl.com/ATMexamples. In order to make the

![Fig. 9. A transition airspace.](image)

![Fig. 10. A sample output on an instance with PC, WSI=35% (the bold line is the sector boundary). Left: 5 paths of minimum total length, computed by the mincost flow. Right: 5 conforming paths minimize the total number of the sector boundary crossings.](image)
trajectories more interesting, we made the exit time interval very short; this enforced that before exiting the airspace, all aircraft align along the sink and leave almost simultaneously.

**Monotone trajectories:** When monotonicity of the flight paths is enforced, the jiggling is automatically eliminated. Unsurprisingly, monotone paths “eat up” time by zigzagging up and down (see, e.g., clip “ZigZagging” at tinyurl.com/ATMexamples). Nevertheless, there is a limit, imposed by the dimensions of the airspace, on how long the aircraft can zigzag, and for longer holding times the monotone paths do not exist at all.

**C. Real Weather**

The output of our implementation on a real-weather instance can be viewed at tinyurl.com/ATMexamples; more extensive experiments with real weather are left for the future work.

**ACKNOWLEDGMENTS**

This research is partially supported by Academy of Finland grant 118653 (ALGODAN). This work is being performed under NASA’s NextGen Airspace Super D ense Operations (ASDO) research focus area under contract NNA07BA84C, from NASA Ames Research Center. We appreciate the financial support of the sponsor of the research, NASA Ames Research Center and the NextGen Project Manager, Dr. Paramal Koprdekar. J. Mitchell is partially supported by the National Science Foundation (CCF-0528205, CCF-0729019).

**REFERENCES**