An Application of the Shortest-Path Problem to Routing Terminal Airspace Air Traffic

A. V. Sadovsky

NASA Ames Research Center, Moffett Field, CA 94035-0001 *

The flight navigation procedures envisioned under the Next Generation Transportation System will require a specification of the expected route of each flight. Once these specifications are provided, the flights must proceed along their routes with such speed profiles that every pair of aircraft complies with the minimal separation requirement imposed by the Federal Aviation Administration. The task of separation assurance is most challenging in terminal airspace, with many routes merging and crossing. This paper contributes a mathematical model and an algorithmic approach for routing flights strategically, with a foresight that potentially helps the subsequent computation of speed profiles compliant with the separation requirements and with each aircraft's feasible speed range. The approach consists of reducing a general routing problem to the Shortest-Path Problem.

Nomenclature

A	the number of aircraft in a given traffic sample
$\mathcal{A}$	a finite set of A aircraft: $\mathcal{A} = \{1, 2, \ldots, A\}$
α	the index of an aircraft in the given sample: $\alpha \in \mathcal{A}$
$G = (V, E)$	a directed graph with vertex set V and edge set E
p	a mapping that assigns to each aircraft in $\mathcal{A}$ a walk in G
$v^{ORIG;\alpha}$	the vertex in G that serves as the origin of aircraft α
$v^{DEST;\alpha}$	the vertex in G that serves as the destination of aircraft α
v_k^α	the k-th vertex in the walk of aircraft α ($v_k^\alpha \in V$)
$\mathbf{G}^{\mathcal{A}}$	the *depth-coordinated product* of the A copies of G (section II.1)
$\mathbf{u}, \mathbf{v}$	vertices of $\mathbf{G}^{\mathcal{A}}$

I. Introduction

Under the Next Generation Air Transportation System (NextGen), flights will follow procedures called *Area Navigation (RNAV)* and *Required Navigation Performance (RNP)*. RNAV and RNP procedures require a precise definition of the expected route of flight; see Ref.[1] for a detailed discussion. Thus, a necessary part of Air Traffic Management (ATM) under NextGen will be the *routing* of the flights, e.g. a specification of a path from origin to destination for each flight. If the airspace route network is modeled as a *directed graph* or *digraph* [2, section A.2], whose vertices are the waypoints and the runways, routing becomes a subclass of the problem of coordinating multiple moving agents subject to separation requirements.[3] Once a routing is provided, the flights must proceed along their routes with speed profiles that assure separation at all times[3] and comply with the feasible speed ranges of the aircraft. The term *speed profile* refers throughout this paper to a function that gives a time parameterization $x^\alpha(t)$ of the path of a flight α, where x^α is the arc length distance from the aircraft to some reference point along its path. Aircraft separation and compliance with the aircraft's feasible range of airspeed are among a relatively large number of operational requirements imposed on speed profiles in ATM; see Ref.[1] for a comprehensive survey. If a feasible solution

*Aerospace Engineer, Aviation Systems Division, NASA Ames Research Center, Mail Stop 210-6, Moffett Field, CA 94035-0001. Email: alexander.v.sadovsky@nasa.gov. AIAA Member.

cannot be found for a given routing, a new routing must be determined that does allow for compliance with all constraints[a].

In today's Air Traffic Operations (ATO), the task of separation assurance falls mostly on the human personnel of Air Traffic Control (ATC). This task is most challenging in a terminal airspace: while standard published routes have been specified in the en-route airspace, air traffic procedures used in terminal airspace seldom specify a continuous route of flight to the aircraft destination or to the exit from terminal airspace. From Top of Descent (i.e., the completion of the cruise stage of flight) to the start of the final approach to landing, arriving flights today navigate by procedures rather than by a fixed set of routes. Arrival and departure procedures may include constraints on speed and altitude, but usually do not specify a continuous route from Top of Descent to the destination airport (or from the origin airport to cruise altitude for departing aircraft). These procedures and practices require the Controller to make determinations of, and changes to, paths and speeds tactically, i.e. on relatively short notice. The necessity for tactical ATC adds to the Controller's workload and to such operational costs as fuel consumption and throughput loss. It is therefore worthwhile to research ways of routing aircraft with a foresight that will help separation assurance. One direction of such research, envisioned in NextGen, is development of automated decision support tools for Precision Air Traffic Operations (PATO). The purpose of such tools is to enable the existing airspace infrastructure to accommodate safely the forecast rapid increase in air traffic demand. Recent work on the problem of automating speed profile advisories in terminal airspace can be found in Refs.[5,6] In both of the latter papers, a speed profile is sought simultaneously *for the entire set of flights*, i.e. as a vector

$$\mathbf{s}(t) = \left(s^1(t), s^2(t), \ldots, s^A(t)\right), \tag{1}$$

where A is the number of flights and $s^k(t)$ is the speed of the k-th flight along its path at time t. For this reason, such a speed profile will be henceforth called *a collective speed profile*.

Automation of speed profile advisories under operational constraints falls under the general problem of multi-agent coordination, with time-continuous motion in a fixed route network. While this general problem has been researched, the focus has been mainly on applications outside of ATM. One such application is *multi-robot coordination*; see, e.g.,[7,8] and references therein. In the latter two references, however, a routing is assumed, and only the speed profiles are sought. In the past few decades of ATM research, the questions of routing and of designing a route network have received considerably more attention in air Traffic Flow Management (TFM),[9–14] concerned mainly with en route airspace, than in terminal airspace. Problems in TFM have been studied using graph theory, network flows, mathematical programming, and machine learning. A more recent publication on ATM in en route space is Ref.,[15] where a hierarchical framework for en route traffic planning and a decentralized traffic planning algorithm are proposed using a *traffic regulation function*. The ATO are modeled using discrete dynamics and decomposing the entire problem into hierarchical layers to achieve tractability. Aircraft separation is left to the human ATC. The discrete dynamical model formulated at the end of section II.C in Ref.[15] indicates that sending an aircraft into *holding* is allowed. This allowance is a key distinction between ATM problems in an en route airspace as opposed to a terminal airspace: while the former has holding capacity and is concerned mainly with traffic flows, the latter lacks such capacity and is concerned first and foremost with aircraft separation.

Work on ATM in terminal airspace has appeared in Refs.,[13,16–22] which use inherently discrete models and treat separation as a constraint. In the context of ATM in terminal airspace, however, such models have a number of shortcomings, summarized in Ref.[5] as follows (the references in the quote are re-numbered to match the present paper's bibliography):

> In such a model, aircraft are scheduled only at specific points (the vertices of the graph), with no time stamps maintained or imputed of the aircraft's position or speed away from such a point. One weakness of such inherently discrete models is their principal inability to assure, without adding artificial constraints, pairwise separation between the aircraft continuously in time, as is operationally required.[3] Another weakness, specific to the models that lead to a *mixed-integer program (MIP)*,[23] is the difficulty of finding an algorithm that performs sufficiently better than the NP-hard worst case of a general *mixed-integer linear program (MILP)*,[24] although a number of the algorithms developed for solving a MILP modeling air traffic have been shown to perform

[a]The feasible speed range of an aircraft is specified as an interval with endpoints the minimal and maximal *airspeeds*.[1] Consequently, this range depends on the wind conditions (*winds aloft*). Wind conditions can therefore affect the space of feasible solutions for a given routing and, in particular, can require another routing. A mathematical concept of air traffic schedule robustness to unexpected wind conditions and to other stochastic perturbations is being developed in.[4]

with a running time polynomial in the number of aircraft (see, for example, Refs.[19,25]). However, while sufficient constraints can be imposed in a discrete model to assure separation at all times, the authors are unaware of published discrete models that assure such separation and show computational performance suitable for aircraft traffic in a terminal airspace.

For further discussion of the problems and a survey of past research, see Ref.[1] and references therein.

The challenge and operational requirement of providing aircraft separation at all times in terminal space ATM problems call for treating the motion of the aircraft as a continuous function of time, as is done in Refs.[7,8] for abstract moving agents. The last three years have seen research efforts to address this need: models where the positions and velocities of the moving agents are continuous functions of time, have been used in Refs.[5,6] to capture the separation constraint continuously in time by computing a collective speed profile (1).

Such a speed profile has no meaning until one has specified for each aircraft a spatial route to follow: the speed profile describes the motion of the aircraft along the given route. This specification of routes (called more briefly, *routing*) is a problem in its own right. If the route network is modeled as a directed graph (described in more detail below), then the routing problem is the discrete problem of finding paths in a graph, which must be solved before the collective speed profile is sought. Navigation of a given set of aircraft in an airspace can, therefore, be thought of as a two-part problem:

$$\left.\begin{array}{l}\text{Part I: Routing (a discrete problem).}\\ \text{Part II: Computation of (1) (a continuous problem).}\end{array}\right\} \tag{2}$$

The feasibility of a collective speed profile corresponding to a given routing depends on a number of factors, including those that can change as the profile is being executed by the aircraft. Examples of such changes include those in the wind conditions and in the availability of the various parts of the airport (e.g., a runway may become non-functional because of unexpected debris). It is therefore desirable not only that the speed profile be feasible, but also that its feasibility be as robust as possible to such random perturbations. If an aircraft is forbidden to deviate from a given route network, then such robustness translates into having sufficient time to assure aircraft separation for the rest of the operation. An attempt at quantifying this sufficiency is made by introducing the *urgency* metric in section III, below.

The focus of this paper is Part I of problem (2) (formulated in detail as Problem I.1, below). The central motivation is to approach this part in a way that would facilitate a solution of the second part. In detail, the focus here is on routing a finite set

$$\mathcal{A} = \{1, 2, \ldots, A\}$$

of A flights in a route network, each flight $\alpha \in \mathcal{A}$ to go from its origin to its destination, both specified as an input to the problem. The route network is modeled as a *directed graph*, or *digraph* [2, section A.2], $G = (V, E)$. The vertices $v \in V$ are points in a Euclidean space of dimension 2 or 3, which models the physical airspace, and correspond to *waypoints*[1] and runways in the airspace. To each edge $e = (u, v) \in E$ corresponds a *rectifiable curve*[26] which, therefore, has a well-defined arc length. Henceforth, an edge $e \in E$ will be identified with the corresponding curve. A graph $G = (V, E)$ with this additional geometric setting will be called *an airspace graph*.

The construct of airspace graph allows to model the route of a flight as a *walk*[2] in G (i.e., a finite sequence of vertices, not necessarily distinct, with each consecutive pair an edge in G) and to furnish the route with an arc length coordinate. This coordinate will be used to specify unambiguously the current location of the aircraft conducting the flight. Thus, a routing of $\mathcal{A}$ in $G = (V, E)$ is, essentially, a mapping p that for each flight $\alpha \in \mathcal{A}$ specifies a walk

$$p(\alpha) \ : \ v_1^\alpha, v_2^\alpha, \ldots, v_{N^\alpha}^\alpha, \tag{3}$$

of some length N^α, where v_1^α is the origin, and $v_{N^\alpha}^\alpha$ the destination, of α. For clarity, we note that:

- a routing of $\mathcal{A}$ in G involves no temporal component, i.e. says nothing about the speeds with which the aircraft are to proceed along their routes, and
- collective speed profiles (which, as indicated, are functions of time) are not explicitly computed in this paper.

The first bullet implies, in particular, that even if the routes assigned to two aircraft overlap, this does not necessarily indicate a loss of separation, although certainly introducing the potential for such a loss. Separation is assured only if the collective speed profile, computed after a routing is specified, satisfies the appropriate set of constraints.[5,6] In a number of publications, such a collective speed profile is computed as a control strategy (classical or hybrid) to minimize a cost functional that may include such terms as fuel consumption, airport throughput, and excess separation; see, for example, Refs.,[6,15,22,27] and references therein. Furthermore, the continuous parameterization of each walk (3), as a rectifiable curve, by its arc length coordinate, allows to enforce the separation constraint for every pair of points that come from two different curves, whether or not these points are vertices of the graph.

An essential requirement of the problem will be that a sought and found routing p make it "as easy as possible" to find, subsequently, separation-compliant collective speed profiles that fit into the feasible speed ranges of the aircraft. This "ease" will be quantified in section III and used to construct a weighting on a digraph whose shortest paths (a *path*[2] is a walk with all vertices distinct) will give a routing. Assuming a rigorous definition of this quantification is available, the problem that motivates this paper can be formulated as follows:

Problem I.1 *Given an airspace graph $G = (V, E)$, a set $\mathcal{A} = \{1, 2, \ldots, A\}$ to be routed in G, and the specification of the origin $v^{ORIG;\alpha} \in V$ and the destination $v^{DEST;\alpha} \in V$, reachable from the origin, of each flight $\alpha \in \mathcal{A}$, construct a routing p of $\mathcal{A}$ in G such that*

- *for each flight $\alpha \in A$, the route $p(\alpha)$ starts with $v^{ORIG;\alpha} \in V$ and ends with $v^{DEST;\alpha} \in V$, and*
- *the routing makes the construction of separation-compliant collection-compliant and feasible speed profile as easy as possible.*

The approach taken herein to Problem I.1 consists of three steps, described in the following three paragraphs.

The first step is to use G to construct a certain new digraph $\mathbf{G}^{\mathcal{A}} = (\mathbf{V}^{\mathcal{A}}, \mathbf{E}^{\mathcal{A}})$, with the vertex set $\mathbf{V}^{\mathcal{A}}$ a subset of the *Cartesian product* [26, section 12.7-1] of A copies of V:

$$\underbrace{V \times V \times \ldots \times V}_{V \text{ occurs } A \text{ times}}, \tag{4}$$

and with the edge set $\mathbf{E}^{\mathcal{A}}$, analogously, a subset of $E \times E \times \ldots \times E$ (E occurs A times). Each vertex $\mathbf{v} \in \mathbf{V}^{\mathcal{A}}$ will, therefore, be an ordered A-tuple $\mathbf{v} = (v^1, \ldots, v^A)$ of vertices from V. It will be more convenient, however, to use the column vector-like notation

$$\mathbf{v} = \begin{bmatrix} v^1 \\ \vdots \\ v^A \end{bmatrix}, \tag{5}$$

whose advantage will be made clear in remark II.2. The definition and essential properties of the graph $\mathbf{G}^{\mathcal{A}}$ will be given in section II.

The second step is to furnish each edge in $\mathbf{E}^{\mathcal{A}}$ with a scalar *weight* (see *weighted graph* in Ref.[2]). One functional form for these weights, suitable for either arrivals only or departures only, is given in section III.

The third step is to solve the resulting *shortest-path problem*[2] on $\mathbf{G}^{\mathcal{A}}$. Here, one seeks shortest paths from

$$\mathbf{v}^{\mathcal{A};ORIG} = \begin{bmatrix} v^{ORIG;1} \\ \vdots \\ v^{ORIG;A} \end{bmatrix}$$

to

$$\mathbf{v}^{\mathcal{A};DEST} = \begin{bmatrix} v^{DEST;1} \\ \vdots \\ v^{DEST;A} \end{bmatrix}.$$

The contribution of this paper is, therefore, a model that reduces Problem I.1 to the well-known shortest-path problem on $\mathbf{G}^{\mathcal{A}}$. This reduction comes at the expense of forming a data set of a size that in the worst

case grows exponentially with A; this potential computational cost is addressed in appendix B. The central object of research in this approach is the construction of a weighting function on $\mathbf{G}^{\mathcal{A}}$ to reflect the difficulty of separation assurance and, it is hoped, to give routings (computed as shortest paths in the sense of this weighting, e.g. using Dijkstra's algorithm–see section IV) that make finding a separation-compliant and speed range-compliant collective speed profile as easy as possible.

II. A construction and analysis of $\mathbf{G}^{\mathcal{A}}$

A. Depth-coordinated products of rooted digraphs: an abstract graph-theoretic concept

This section is an introduction to an abstract graph-theoretic concept–*the depth-coordinated product of rooted digraphs*–that will be used in a later section to construct and analyze $\mathbf{G}^{\mathcal{A}}$. The ATM context is temporarily abandoned here (and in appendix B) and is re-entered in the next section.

A *rooted digraph* is a digraph with one of the vertices chosen to be distinguished from all the others and called the *root* of the graph. In this paper, the notation for a rooted digraph will be an ordered triple of the form

$$(\text{the vertex set}, \text{the edge set}, \text{the root}). \tag{6}$$

Definition II.1 *Given two rooted digraphs,*

$$G^{(1)} = \left(V^{(1)}, E^{(1)}, v^1_{root}\right), \quad G^{(2)} = \left(V^{(2)}, E^{(2)}, v^2_{root}\right), \tag{7}$$

with no cycles that have two or more edges, their depth-coordinated product *is defined as the graph whose vertex set, $V^{DCP} \subset V^{(1)} \times V^{(2)}$, and edge set, $E^{DCP} \subset E^{(1)} \times E^{(2)}$, meet <u>all</u> of the following conditions:*

(a)

$$\begin{bmatrix} v^1_{root} \\ v^2_{root} \end{bmatrix} \in V^{DCP} \tag{8}$$

(b) $\begin{bmatrix} u^1 \\ u^2 \end{bmatrix} \in V^{DCP}$ *if and only if u^1 and u^2 are, each in its respective graph, reachable from the root by walks with the same number of edge occurrences.*

(c) $\mathbf{e} = \left(\begin{bmatrix} u^1 \\ u^2 \end{bmatrix}, \begin{bmatrix} v^1 \\ v^2 \end{bmatrix} \right) \in E^{DCP}$ *if and only if <u>all</u> of the following conditions are met:*

$$\begin{bmatrix} u^1 \\ u^2 \end{bmatrix} \in V^{DCP} \quad \textit{and} \quad \begin{bmatrix} v^1 \\ v^2 \end{bmatrix} \in V^{DCP},$$

$$\left(u^1, v^1\right) \in E^{(1)} \quad \textit{and} \quad \left(u^2, v^2\right) \in E^{(2)},$$

$\mathbf{e}$ *is not a* self-loop[b].

Remark II.1 *(a) It can be proved that the depth-coordinated product of two rooted digraphs is thus defined uniquely.*

(b) While each of the constituent rooted digraphs, $G^{(k)}$, is allowed to have self-loops (an allowance that will turn out instrumental in the routing application; see section B), their depth-coordinated product is allowed no self-loops. An example of two rooted digraphs and their depth-coordinated product is shown in figure 1 and, in particular, illustrates this point: while the graph in panel (A) of the figure has the self-loop $(3, 3)$, and the graph in panel (B) has the self-loop $(11, 11)$, the depth-coordinated product shown in panel (C) does not have the self-loop

$$\mathbf{e} = \left(\begin{bmatrix} 3 \\ 11 \end{bmatrix}, \begin{bmatrix} 3 \\ 11 \end{bmatrix} \right).$$

This restriction is imposed in anticipation of furnishing the depth-coordinated product with a weighting and computing shortest paths: absence of self-loops can help obtain a weighted graph which is acyclic, a property that facilitates a search for shortest paths.

(c) The number of edge occurrences mentioned in condition (b) of definition II.1 is the "depth" used to "coordinate" the product of the two graphs. This use of the term "depth" is consistent with that in "the depth of a tree." In particular, if the rooted digraphs (7) are trees of respective depths $D^{(1)}$ and $D^{(2)}$, then their depth-coordinated product is a tree of depth $\min\{D^{(1)}, D^{(2)}\}$. The question arises whether the ATM application central to this paper is best served by coordinating the product by depth or by some other quantity, possibly with non-integral values allowed, or, more generally, by an equivalence relation[26] *on $V^{(1)} \cup V^{(2)}$ (generalizing the criterion "at the same depth" to "in the same equivalence class"). The author is aware of no prior research on this question.*

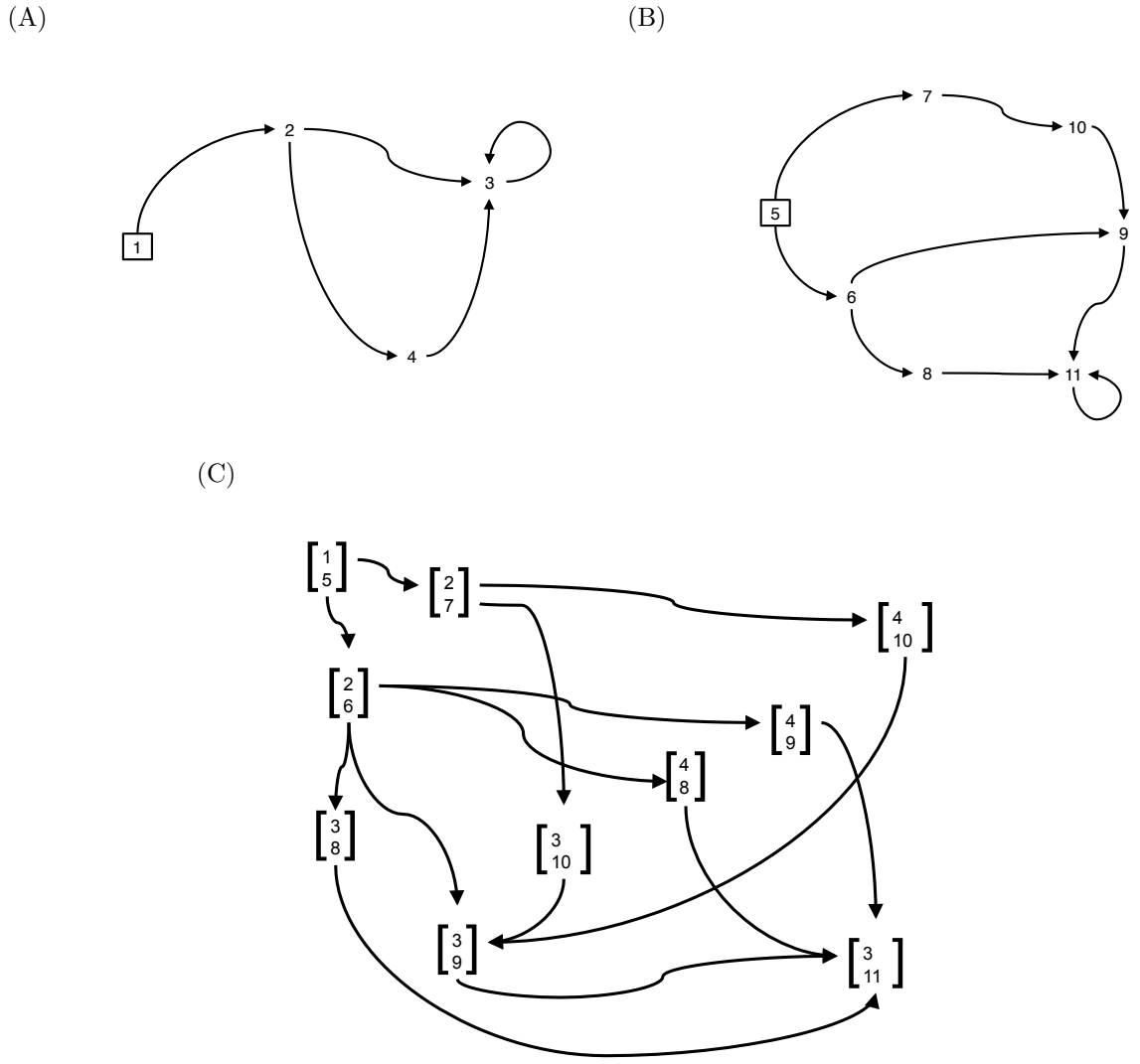

Figure 1. (A), (B) An example of two rooted digraphs (the roots are shown in boxes), and (C) their depth-coordinated product.

One analogously defines the depth-coordinated product of any finite (or even countably infinite) collection of rooted digraphs. The task and potential costs of computing depth-coordinated products are addressed in appendix B.

Remark II.2 *A walk in the depth-coordinated product of K rooted digraphs $G^{(k)}$, $k = 1, \ldots, K$, can thus be written*

$$\mathbf{u}_1 = \begin{bmatrix} u_1^1 \\ u_1^2 \\ \vdots \\ u_1^k \end{bmatrix}, \quad \mathbf{u}_2 = \begin{bmatrix} u_2^1 \\ u_2^2 \\ \vdots \\ u_2^K \end{bmatrix}, \quad \ldots, \quad \mathbf{u}_M = \begin{bmatrix} u_M^1 \\ u_M^2 \\ \vdots \\ u_M^K \end{bmatrix}. \tag{9}$$

However, it will be also convenient to regard the walk (9) as the "column vectors" of a matrix, and to write the same walk in the form

$$\begin{bmatrix} u_1^1 & u_2^1 & \ldots & u_M^1 \\ u_1^2 & u_2^2 & \ldots & u_M^2 \\ & & \vdots & \\ u_1^K & u_2^K & \ldots & u_M^K \end{bmatrix}. \tag{10}$$

The k-th row of the matrix is a walk in $G^{(k)}$.

B. An assumption about the airspace graph $G = (V, E)$

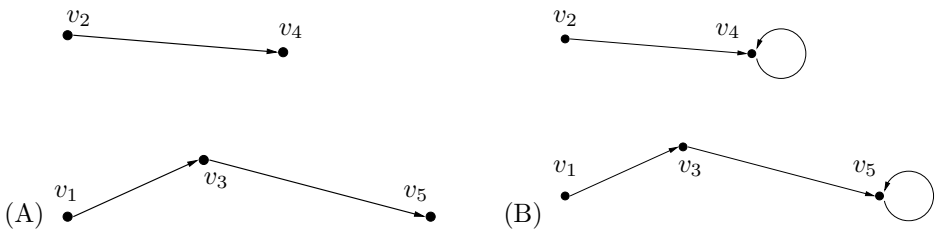

Figure 2. (A) A (fictitious) airspace graph for two aircraft with respective origins v_1, v_2 and respective destinations v_5, v_4. The only routing is the path v_1, v_3, v_5 for the first aircraft and the path v_2, v_4 for the second. The two paths have different lengths, hence the routing for the two aircraft cannot be computed as a path in $\mathbf{G}^{\mathcal{A}}$. (B) Inserting artificially the self-loops (v_4, v_4) and (v_5, v_5) allows a routing as the path $\begin{bmatrix} v_1 & v_3 & v_5 \\ v_2 & v_4 & v_4 \end{bmatrix}$ in $\mathbf{G}^{\mathcal{A}}$.

If a routing of $\mathcal{A}$ in $G = (V, E)$ is sought as a path (10) in the depth-coordinated product of $K = A$ rooted copies of G, where the α-th row of the matrix is the individual walk assigned to flight α, then such a routing must assign to all flights walks of the same length. There are airspace graphs, however, that do not allow this, because the maximum number of edges that one flight must traverse from origin to destination can be smaller than the minimum number of edges another flight must traverse from origin to destination. An example of such a situation is shown in figure 2(A) for a set of 2 aircraft with origins at v_1 and v_2. Figure 2(B), on the other hand, illustrates that this obstacle can be overcome by inserting artificially a self-loop at each flight's destination. These considerations lead to the following assumption about the given airspace graph $G = (V, E)$, made throughout the rest of the paper.

Assumption II.1 *(a) For every flight $\alpha \in \mathcal{A}$, the destination is reachable from the origin; i.e., there exists a walk in G from $v^{ORIG;\alpha}$ to $v^{DEST;\alpha}$.*

(b) If a vertex $v \in V$ is the destination of a flight $\alpha \in \mathcal{A}$, then E contains the self-loop (v, v).

(c) There are no other self-loops in G.

C. Definition and essential properties of $\mathbf{G}^{\mathcal{A}}$

By associating to each flight α a "copy" of the airspace graph $G = (V, E)$ and designating $v^{ORIG;\alpha}$ as the root, one obtains the rooted digraphs

$$\left(V, E, v^{ORIG;\alpha}\right), \quad \alpha \in \mathcal{A}.$$

Their depth-coordinated product will be taken as the definition of $\mathbf{G}^{\mathcal{A}}$.

The intuitive content of the following theorem is that, by finding all paths in $\mathbf{G}^{\mathcal{A}}$ from $\mathbf{v}^{ORIG}$ to $\mathbf{v}^{DEST}$, one finds all possible routings (3).

Theorem II.1 *Suppose there exists a routing (3) such that*

$$v_1^{\alpha} = v^{ORIG;\alpha} \quad \textit{and} \quad v_{N^{\alpha}}^{\alpha} = v^{DEST;\alpha} \textit{ for each } \alpha \in \mathcal{A}.$$

Then there exists a walk

$$\mathbf{u}_1 = \begin{bmatrix} u_1^1 \\ \vdots \\ u_1^A \end{bmatrix}, \quad \mathbf{u}_2 = \begin{bmatrix} u_2^1 \\ \vdots \\ u_2^A \end{bmatrix}, \quad \ldots, \quad \mathbf{u}_M = \begin{bmatrix} u_M^1 \\ \vdots \\ u_M^A \end{bmatrix}, \quad M = \max_{\alpha \in \mathcal{A}} N^{\alpha}, \tag{11}$$

in $\mathbf{G}^{\mathcal{A}} = (\mathbf{V}^{\mathcal{A}}, \mathbf{E}^{\mathcal{A}})$ *such that for each* $\alpha \in \mathcal{A}$ *the following conditions are met:*

- $u_k^{\alpha} = v_k^{\alpha}$ *for* $k = 1, 2, \ldots, N^{\alpha}$,
- $u_k^{\alpha} = v_{N^{\alpha}}^{\alpha}$ *for* $k = N^{\alpha}, \ldots, M$, *and*
- $u_M^{\alpha} = v^{DEST;\alpha}$.

Theorem II.1, proved in appendix A, implies that the flights $\alpha \in \mathcal{A}$ can be routed all within the same algorithm by seeking paths in $\mathbf{G}^{\mathcal{A}}$ from $\mathbf{v}^{\mathcal{A};ORIG}$ to $\mathbf{v}^{\mathcal{A};DEST}$. A weighting on $\mathbf{G}^{\mathcal{A}}$ will be constructed in section III in such a way that the shorter of the paths from $\mathbf{v}^{\mathcal{A};ORIG}$ to $\mathbf{v}^{\mathcal{A};DEST}$ are the ones that make subsequent computation of separation-compliant speed profiles easier.

III. A weighting function on $\mathbf{G}^{\mathcal{A}}$ suitable for routing arrivals only or departures only

The main source of difficulty in enforcing separation between two aircraft in an airspace, represented by an airspace graph $G = (V, E)$, is the increasing proximity of their routes, i.e. those portions of the routing that allow a simultaneous positioning of the aircraft in which at least two of the safety envelopes overlap. In particular, if the two aircrafts' routes merge or intersect at a point[28c], then the two aircraft are sure to constrain each other's motions spatially in the vicinity of the point.

A point where two or more routes merge, or a *merge point*, would correspond to a vertex $v \in V$ of *indegree* ≥ 2.[2] If the positions of two aircraft, $\alpha = 1$ and $\alpha = 2$, are specified by the arc length coordinates $s^{(1)}$ and $s^{(2)}$ along their respective routes, then the set of all pairs $\left(s^{(1)}, s^{(2)}\right)$ corresponding to a loss of separation between the two aircraft can be calculated explicitly from the minimal separation requirements for the two aircraft types and from the shape of the route segments that merge. For the theoretically special (but operationally frequent) case when the two merging segments are rectilinear, this calculation is carried out in detail in[5,6] and confirms the following qualitative considerations:

- The closer the merge point to the two aircraft, the smaller the space of separation-compliant collective speed profiles.
- The smaller the angle between the two merging route segments, the smaller the space of separation-compliant collective speed profiles.

} (12)

Throughout this section, let $d(v, v')$ denote the Euclidean distance between vertices v and v' of G. If these vertices constitute an edge, $e = (v, v') \in E$, then notation $d(e)$ will also be sometimes used. In what follows, we adopt the following assumption.

[c]The term *critical point* as defined in Ref.[28] is not used in the present work and occurs in other literature with a different definition; e.g., see Ref.[26]

Assumption III.1 *(a) All edges in E are rectilinear segments. (The main shortcoming of this part of the assumption is the difficulty of using a polygonal curve to approximate RNAV routes that will have curvilinear parts corresponding to turns.[29] The requirement[29] that such routes accommodate all types of aircraft to fly that route and all wind conditions affects the turn radius designed into the route, thereby affecting the number of polygonal segments required for a good approximation.)*

(b) An intersection of two routes is either a vertex in G or set of edges in G.

(c) No two routes share a pair of adjacent vertices with opposite edge orientations; i.e., there do not exist vertices u, v such that one route contains the edge (u, v), and another route, the edge (v, u). (Restrictive as this part of the assumption may seem, most current airport operations satisfy it by separating arrivals from departures by altitude. This fact, however, does not principally rule out the necessity to address the case of two mutually opposite edges, which is therefore addressed in remark III.1.)

Suppose the vertices

$$\mathbf{u}^{\mathcal{A}} = \begin{bmatrix} u^1 \\ \vdots \\ u^A \end{bmatrix}, \quad \mathbf{v}^{\mathcal{A}} = \begin{bmatrix} v^1 \\ \vdots \\ v^A \end{bmatrix}$$

constitute an edge $\mathbf{e}^{\mathcal{A}} = \left(\mathbf{u}^{\mathcal{A}}, \mathbf{v}^{\mathcal{A}}\right) \in \mathbf{E}^{\mathcal{A}}$ (this is true if and only if $e^{\alpha} = (u^{\alpha}, v^{\alpha}) \in E$ for every $\alpha \in \mathcal{A}$). For a pair α_1, α_2 of flight indices, the quantity

$$d^{\alpha_1,\alpha_2}\left(\mathbf{e}^{\mathcal{A}}\right) = \frac{1}{2}\left(d\left(u^{\alpha_1}, v^{\alpha_1}\right) + d\left(u^{\alpha_2}, v^{\alpha_2}\right)\right) \tag{13}$$

is small if the edges $(u^{\alpha_1}, v^{\alpha_1})$ and $(u^{\alpha_2}, v^{\alpha_2})$ are both short. If, furthermore, the edges merge ($d(v^{\alpha_1}, v^{\alpha_2}) = 0$), this merge hampers collective speed profiles that will keep the two aircraft on these edges properly separated. Finally, if at least one of the edges $(u^{\alpha_1}, v^{\alpha_1})$, $(u^{\alpha_2}, v^{\alpha_2})$ is a loop, then the weight of the edge pair is zero, because self-loops are introduced artificially, to allow repetition of vertices in an aircraft's walk that is shorter than another's (section B).

Two distinct issues thus emerge: the shortage of time available for (or the *urgency* of) separation assurance, and the difficulty of separation assurance arising from the proximity of the two edges. The first issue will be addressed here by observing that quantity (13) measures the amount of room (and, therefore, the amount of time) available for the two aircraft to assure separation. The aforementioned urgency, then, will be defined as the reciprocal of (13). The second issue will be addressed by defining the proximity of the two edges as the quantity

$$\frac{1}{c_1 + d\left(u^{\alpha_1}, u^{\alpha_2}\right)} + \frac{c_3 d\left(u^{\alpha_1}, u^{\alpha_2}\right)}{c_2 + d\left(v^{\alpha_1}, v^{\alpha_2}\right)}, \quad \text{the } c\text{'s positive constants.} \tag{14}$$

Quantity (14) increases for the following types of configurations (shown in figure 3), in that order: the two edges have no overlap, the two edges share the initial vertex only, the two edges coincide, the two edges share the final vertex only.

Remark III.1 *Part (c) of assumption III.1, while it makes the weighting function developed herein unsuitable for routing traffic that contains a mixture of arrivals and departures, is inessential to the approach proposed herein and is adapted mostly for computational convenience. One straightforward way to include a penalty for the case when the two edges $(u^{\alpha_1}, v^{\alpha_1})$, $(u^{\alpha_2}, v^{\alpha_2})$ that are mutually opposite, i.e. satisfy*

$$u^{\alpha_1} = v^{\alpha_2} \quad \textit{and} \quad u^{\alpha_2} = v^{\alpha_1},$$

is as follows. Denote by μ_{α_k} the midpoint of $(v^{\alpha_k}, v^{\alpha_k}), k = 1, 2$, thinking of it as an artificially inserted vertex. Instead of assessing a separation-maintenance penalty for the edge pair $(u^{\alpha_1}, v^{\alpha_1})$, $(u^{\alpha_2}, v^{\alpha_2})$, assess and add such penalties for the edge pairs $(u^{\alpha_1}, \mu^{\alpha_1})$, $(u^{\alpha_2}, \mu^{\alpha_2})$ and $(\mu^{\alpha_1}, v^{\alpha_1})$, $(\mu^{\alpha_2}, v^{\alpha_2})$. This results in replacing (14) by the more cumbersome expression

$$\left[\frac{1}{c_1 + d\left(u^{\alpha_1}, u^{\alpha_2}\right)} + \frac{c_3 d\left(u^{\alpha_1}, u^{\alpha_2}\right)}{c_2 + d\left(\mu^{\alpha_1}, \mu^{\alpha_2}\right)}\right] + \left[\frac{1}{c_1 + d\left(\mu^{\alpha_1}, \mu^{\alpha_2}\right)} + \frac{c_3 d\left(\mu^{\alpha_1}, \mu^{\alpha_2}\right)}{c_2 + d\left(v^{\alpha_1}, v^{\alpha_2}\right)}\right]$$

A potential shortcoming of the latter functional form is that it may penalize a pair of mutually opposite edges on the same scale as a pair of merging edges, hence perhaps insufficiently.

The parameters c_1, c_2, c_3 in (14) should generally be chosen to take into account the following factors:

$$\left.\begin{array}{ll} \text{(a)} & \text{minimal separation requirements for the flight pair } (\alpha_1, \alpha_2) \\ \text{(b)} & \text{the feasible speed ranges of the two aircraft} \\ \text{(c)} & \text{any available information about stochastic phenomena that may hamper the execution of a speed profile (e.g., winds)} \end{array}\right\} \quad (15)$$

Of these, only (a) is addressed in this paper. The other two factors are discussed briefly in section VI.

One functional form of a weighting for $\mathbf{e}^{\mathcal{A}}$ that captures these considerations is

$$W\left(\mathbf{e}^{\mathcal{A}}\right) = \begin{cases} 0 \text{ if } \left(u^{\alpha_1} = v^{\alpha_1} \text{ and } u^{\alpha_2} = v^{\alpha_2}\right) \\ \sum_{\alpha_1 \neq \alpha_2} \underbrace{\left(\frac{1}{d^{\alpha_1,\alpha_2}\left(\mathbf{e}^{\mathcal{A}}\right)}\right)}_{\text{urgency}} \underbrace{\left(\frac{1}{c_1 + d\left(u^{\alpha_1}, u^{\alpha_2}\right)} + \frac{c_3 d\left(u^{\alpha_1}, u^{\alpha_2}\right)}{c_2 + d\left(v^{\alpha_1}, v^{\alpha_2}\right)}\right)}_{\text{difficulty arising from edge proximity}} & \text{otherwise} \end{cases} \quad (16)$$

The quantities c_1, c_2, and $1/c_3$ are in the units of distance, so W is in the units of distance squared, reflecting the fact that costs are assessed for each pair of aircraft. The heaviest contributions to (16) come from the summands corresponding to a pair of edges both short and coming near each other. Figure 3 shows four benchmark configurations of an edge pair $(u^{\alpha_1}, v^{\alpha_1})$ and $(u^{\alpha_2}, v^{\alpha_2})$ with the corresponding values of (16) computed with $c_1 = 0.05, c_2 = 0.01, c_3 = 1.00$.

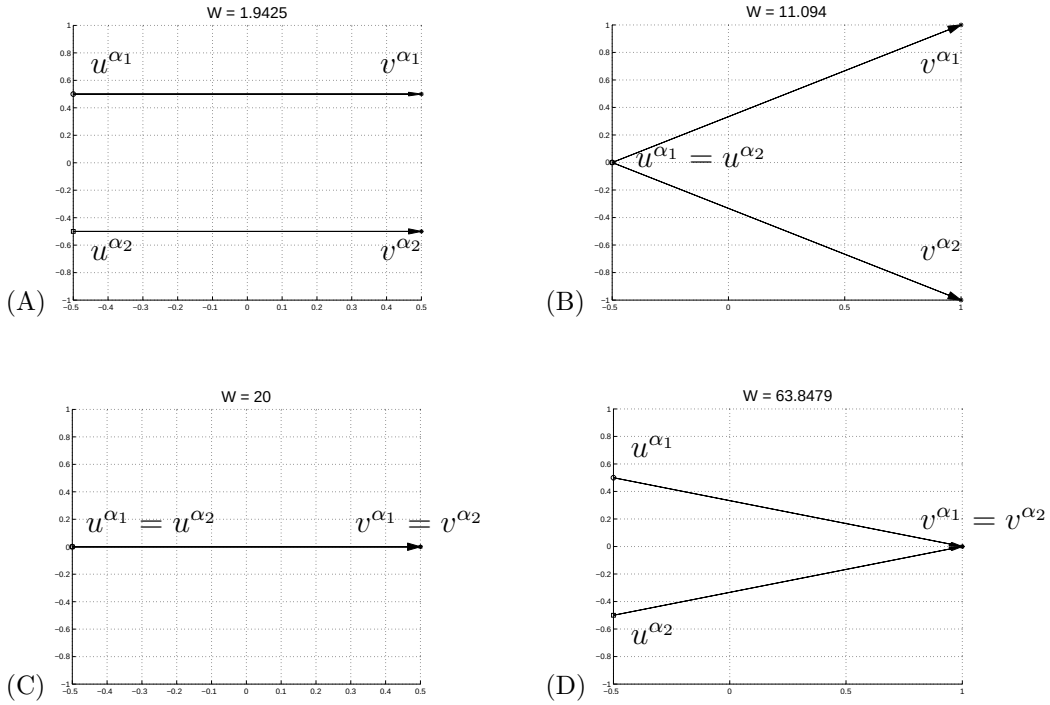

Figure 3. The values of the weighting function (16) for four benchmark configurations of the edge pair $(u^{\alpha_1}, v^{\alpha_1})$ and $(u^{\alpha_2}, v^{\alpha_2})$. Parameter values used: $c_1 = 0.05, c_2 = 0.01, c_3 = 1.00$.

IV. Sample numerical results

With the graph $\mathbf{G}^{\mathcal{A}}$ constructed and the weighting (16) defined, one can seek shortest paths using any of the known suitable algorithms; e.g., Dijkstra's.[30] The following examples were solved using the Matlab software.[31] In each example, the Shortest-Path Problem was solved using Dijkstra's algorithm in

the implementation,[32] using the weighting function (16), and the result recorded in the form (10) and accompanied by the physical run time of the entire computation (of $\mathbf{G}^{\mathcal{A}}$, the weighting function, and a Dijkstra shortest path), carried out on a Mac OS X 10.6.8. Each solution is briefly discussed from the point of view of the qualitative considerations (12).

The algorithms used here for computing $\mathbf{G}^{\mathcal{A}}$, the weighting on it, and shortest paths, are only a prototype, aimed at proof of concept. An automated tool that computes the same objects to aid Air Traffic Control in real operations would require that all the algorithms underlying all computations made in real time are scalable. Appendix B offers a detailed analysis of the complexity of computing $\mathbf{G}^{\mathcal{A}}$, which grows exponentially with A, and of some possible ways to either expedite this computation (e.g., using such a parallelization paradigm as *map-reduce*[33]) or to carry out as much of it as is possible off-line (i.e., to precompute a sufficient collection of the $\mathbf{G}^{\mathcal{A}}$'s).

Remark IV.1 *All examples below except the last are "abstract," in the sense that no particular scale of airspace is specified for them. Thus, the units of length and physical time are left unspecified. One can, however, obtain a realistic spatial scale of an airspace in the examples of sections C-E by taking the unit of length to equal* 3 *nmi. The last example (section F) is for the LAX arrival airspace, and the unit of length is* 1 *nmi.*

The number of aircraft in each example is small, compared to air traffic demand observed in real operations, to keep the computational load within the capabilities of Matlab.

A. Four vertices, 2-D airspace, two aircraft

The airspace graph used in this example has four vertices, labeled $1, 2, 3, 4$. The vertex positions and edge structure of the airspace graph $G = (V, E)$ (excluding the artificially inserted self-loops) are shown in figure 4.

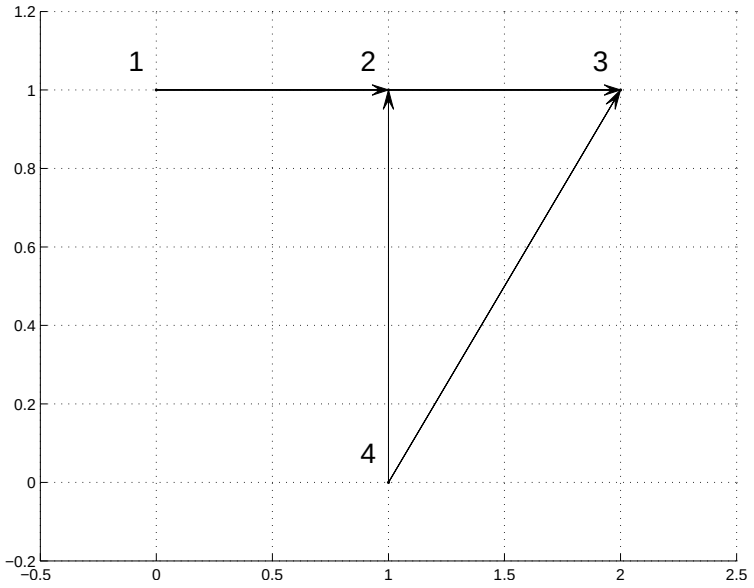

Figure 4. The airspace graph for the numerical example of section A.

Two aircraft are routed in the airspace. The matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 2 & 3 \end{bmatrix}$$

(comp. run time: 0.15s)

specifies the origins (leftmost column), destinations (rightmost column), and the computed routing (rows). The weighting function (16) has favored a routing with a larger angle of merge (see (12)).

B. Six vertices, 2-D airspace, two aircraft

The airspace graph used in this example has six vertices, labeled $1, 2, \ldots, 6$. The vertex positions and edge structure of the airspace graph $G = (V, E)$ (excluding the artificially inserted self-loops) are shown in figure 5.

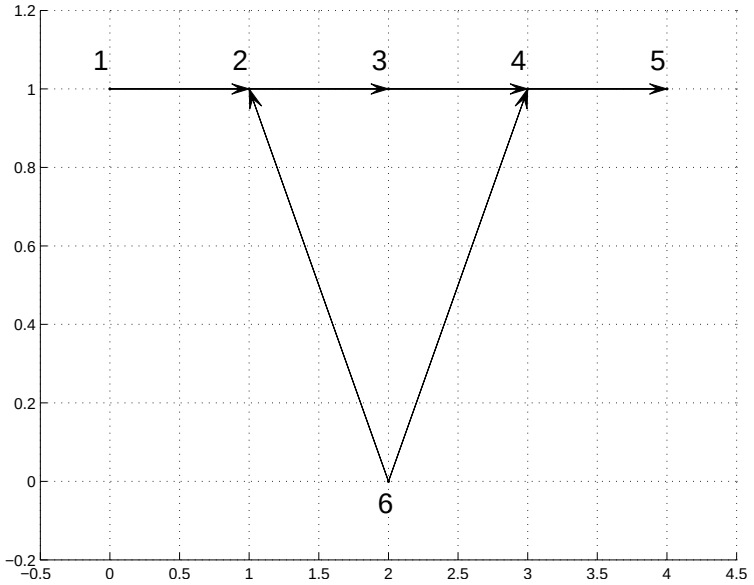

Figure 5. The airspace graph for the numerical example of section B.

Three aircraft are routed in the airspace. The matrix

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 4 & 5 & 5 & 5 \end{bmatrix} \text{(comp. run time: 0.16s)}$$

specifies the origins (leftmost column), destinations (rightmost column), and the computed routing (rows). The weighting function (16) has favored a routing with a merge farther from the origins of the aircraft over a merge with a larger angle (see (12)).

C. Ten vertices, 2-D airspace, three aircraft

The airspace graph used in this example has ten vertices, labeled $1, 2, \ldots, 10$. The vertex positions and edge structure of the airspace graph $G = (V, E)$ (excluding the artificially inserted self-loops) are shown in figure 6.

Three aircraft are routed in the airspace. The matrix

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{bmatrix} \text{(comp. run time: 0.29s)}$$

specifies the origins (leftmost column), destinations (rightmost column), and the computed routing (rows). The weighting function (16) has favored a routing with a merge farther from the origins of the aircraft (see (12)).

D. Ten vertices, 3-D airspace, two aircraft

The airspace graph used in this example has ten vertices, labeled $1, 2, \ldots, 10$. The vertex positions and edge structure of the airspace graph $G = (V, E)$ (excluding the artificially inserted self-loops) are shown in figure 7.

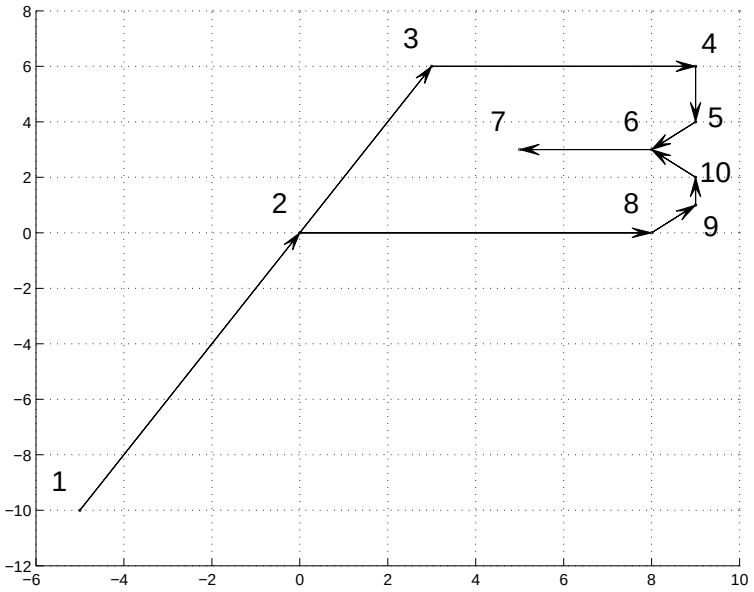

Figure 6. The airspace graph for the numerical example of section C. For a realistic interpretation of length units, see remark IV.1.

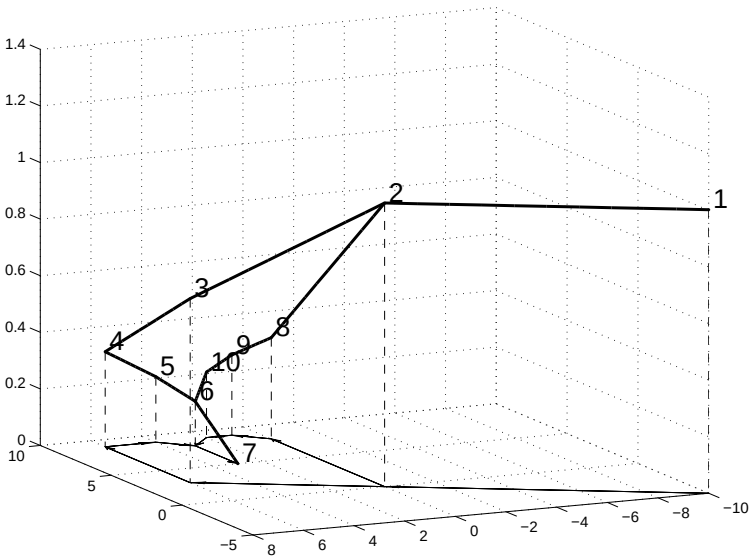

Figure 7. The airspace graph for the numerical example of section D. For a realistic interpretation of length units, see remark IV.1.

Two aircraft are routed in the airspace. The matrix

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{bmatrix}$$ (comp. run time: 0.2s)

specifies the origins (leftmost column), destinations (rightmost column), and the computed routing (rows). The weighting function (16) has favored that the four aircraft follow different routes toward the runway.

E. Seventeen vertices, 2-D airspace, four aircraft

The airspace graph used in this example has seventeen vertices, labeled $1, 2, \ldots, 17$. The topology and geometry of the graph models a hypothetical arrival airspace with multiple routing options. The flights approaching for arrival all pass vertex 17 and are headed to the runway threshold (vertex 14). The vertex positions and edge structure of the airspace graph $G = (V, E)$ (excluding the artificially inserted self-loops) are shown in figure 8.

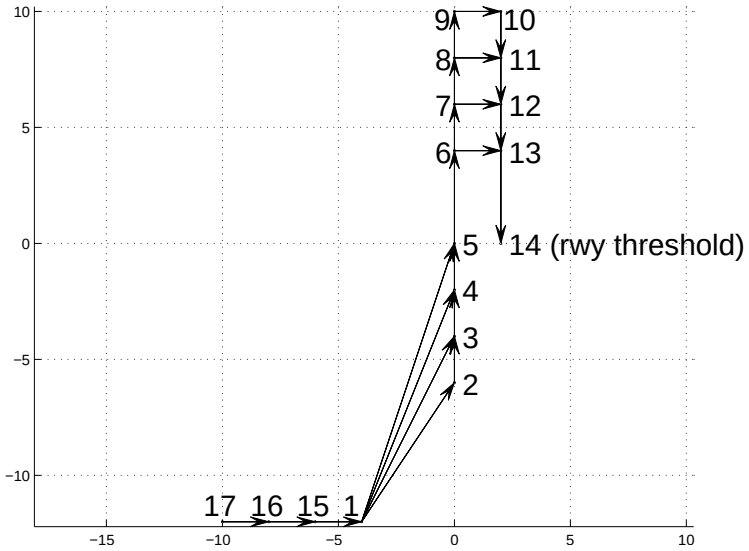

Figure 8. The airspace graph for the numerical example of section E, modeling a terminal airspace for arriving traffic. Vertex 14 is the runway threshold. For a realistic interpretation of length units, see remark IV.1.

Four aircraft are routed in the airspace. The matrix

$$\begin{bmatrix} 17 & 16 & 15 & 1 & 4 & 5 & 6 & 13 & 14 \\ 16 & 15 & 1 & 5 & 6 & 7 & 12 & 13 & 14 \\ 15 & 1 & 2 & 3 & 4 & 5 & 6 & 13 & 14 \\ 1 & 5 & 6 & 7 & 8 & 11 & 12 & 13 & 14 \end{bmatrix}$$ (comp. run time: 240.51s)

specifies the origins (leftmost column), destinations (rightmost column), and the computed routing (rows). The weighting function (16) has favored a routing with a merge farther from the origins of the aircraft (see (12)).

F. Fifty-one vertices, 2-D LAX airspace, five aircraft

The airspace graph used in this example has fifty-one vertices, labeled $1, 2, \ldots, 51$, and was constructed from live recordings of air traffic data in LAX using the method proposed in Ref.[34] The vertex positions and edge structure of the airspace graph $G = (V, E)$ (excluding the artificially inserted self-loops) are shown in figure 9.

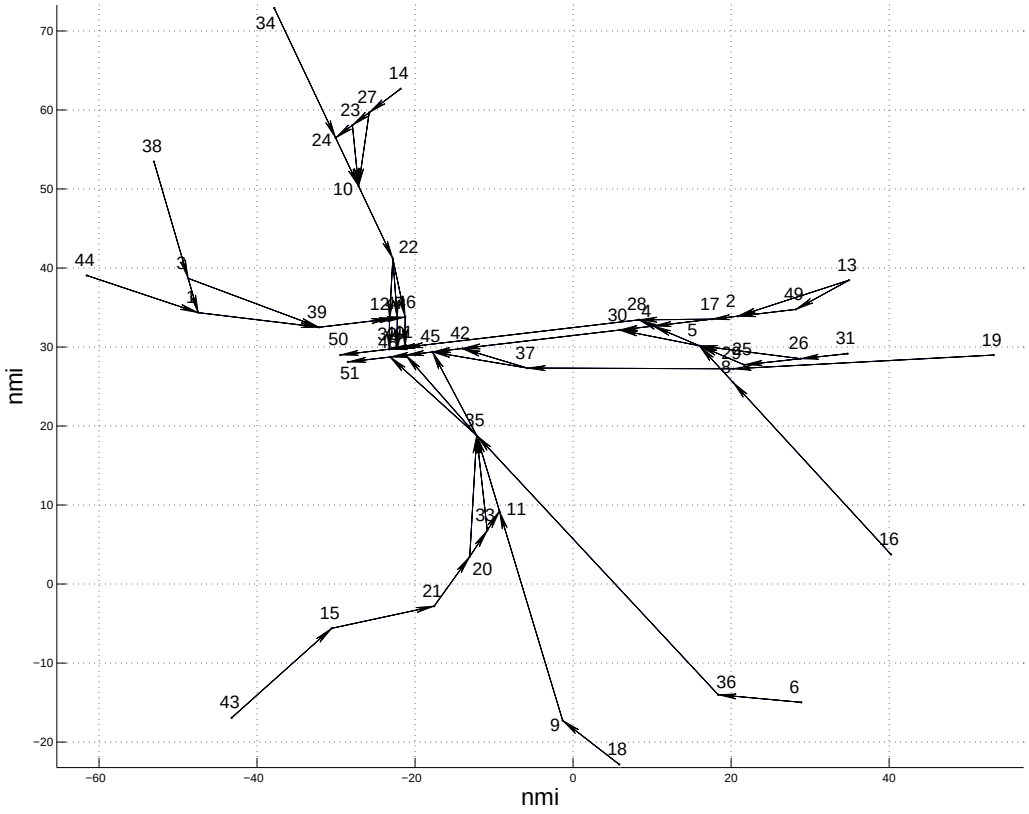

Figure 9. The airspace graph for the numerical example of section F, modeling the LAX terminal airspace for arriving traffic. Vertices 50 and 51 are the runway thresholds.

Five aircraft are routed in the airspace. The matrix

$$\begin{bmatrix} 14 & 27 & 10 & 22 & 12 & 32 & 50 \\ 27 & 23 & 10 & 22 & 12 & 32 & 50 \\ 43 & 15 & 21 & 20 & 35 & 48 & 51 \\ 15 & 21 & 20 & 33 & 35 & 48 & 51 \\ 21 & 20 & 33 & 11 & 35 & 48 & 51 \end{bmatrix} \text{(comp. run time: 1022.94s)}$$

specifies the origins (leftmost column), destinations (rightmost column), and the computed routing (rows).

V. Conclusions

The numerical evidence above suggests that the computation separation-compliant speed profiles can be facilitated by first selecting a routing. Namely, for the examples considered, the Shortest-Path model for routing the given set of aircraft has yielded those routings that leave the most time for assuring aircraft separation. The main advantages of such routings are two: ease of finding feasible speed profiles and robustness of these profiles to unexpected changes.

VI. Discussion

From the viewpoint of PATO safety, the key desired property of a weighting function on the graph $\mathbf{G}^{\mathcal{A}}$ is for the resulting routing to maximize the chances that a separation-compliant and speed range-compliant collective speed profile exists; refer to (15.a,b). In addition to this requirement, one must consider the phenomena, generally stochastic, that hamper the execution of a given collective speed profile (e.g., a tail wind that alters the aircraft's feasible range of ground speed); refer to (15.c).

Research into suitable forms of the weighting function can be approached using different methods. It appears generally true, however, that with a specific functional form of the weighting chosen, the goal is to fit the coefficients in the form so as to meet the requirements stated in the previous paragraph. Such a fitting problem has arisen in many fields and has been approached using various theoretical frameworks, such as statistics and machine learning. Given a collection of instances of problem I.1 and a computational method that, for a given routing, seeks collective speed profiles compliant with all constraints (separation, speed ranges) and, possibly, optimal to a given objective function (see, e.g.,[5,6]), one may pursue the statistical problem of finding *maximum-likelihood estimates*[26] of the desired coefficients that increase the probability that a feasible (i.e., compliant with the separation requirements and speed ranges) collective speed profile exists for the given routing. A weakness of this approach, however, can be seen as follows. The sample space for this approach would consist of pairs of the form (the initial collective position of the aircraft in $\mathcal{A}$, a speed range-compliant collective speed profile (1)). A suitable probability measure, required for the aforementioned maximum-likelihood estimates, is not readily defined on this sample space. Methods of resolving this issue are a topic for future research.

As an alternative to statistical methods, one may consider the following machine learning approach. Assuming a collective speed profile, for a given routing, can be quantitatively measured for *robustness* (research aimed at defining such a quantity rigorously is in progress[4]), pose the problem of classifying pairs of the form

$$(\text{routing},\ \text{collective speed profile})$$

as "robust" if their value of robustness is at or above the threshold, and as "non-robust" otherwise.

The form of the weighting function developed above does not include the capability to route a mixture or arrivals and departures; see remark III.1. A desired next step is a weighting function capable of serving such mixed traffic.

Acknowledgment

Thanks go to the following colleagues at the NASA Ames Research Center: J. Rios, D. Isaacson, and W. Malik for deep discussions that heavily influenced the paper; J. Love and D. Thipphavong for valuable editorial suggestions; S. Zelinski for providing the airspace data for the numerical example in section F. The

open-source Matlab subroutine `arrow.m` used in the generation of some of the graphics above was contributed by Dr. Erik A. Johnson.

A. Proof of theorem II.1.

For each α, extend the walk (3) by appending $(M - N^\alpha)$ occurrences of $v^\alpha_{N^\alpha}$ $\left(= v^{DEST;\alpha}\right)$ at the end, to obtain:

$$\underbrace{v^\alpha_1, v^\alpha_2, \ldots, v^\alpha_{N^\alpha}}_{\text{the walk (3)}}, \qquad \underbrace{v^\alpha_{N^\alpha}, \ldots, v^\alpha_{N^\alpha}}_{(M-N^\alpha) \text{ occurrences of } v^\alpha_{N^\alpha}} \tag{17}$$

Take the walk $u^\alpha_1, \ldots, u^\alpha_M$ to be (17). This construction results in the sequence (11).

It remains to be proved that, *(i)*, each

$$\mathbf{u}_k = \begin{bmatrix} u^1_k \\ \vdots \\ u^A_k \end{bmatrix}, \quad k = 1, \ldots, M,$$

thus constructed is a vertex in $\mathbf{V}^{\mathcal{A}}$, and that, *(ii)*, each consecutive pair $\mathbf{u}_k, \mathbf{u}_{k+1}$ constitutes an edge in $\mathbf{E}^{\mathcal{A}}$.

Both *(i)* and *(ii)* will now be proved by induction on k. The base case $k = 1$ follows immediately from the observation that

$$v^\alpha_1 = v^{DEST;\alpha} \quad \text{for all } \alpha \in \mathcal{A}.$$

For the inductive step, if $\mathbf{u}_{k-1}$ is in $\mathbf{V}^{\mathcal{A}}$, then part (c) of definition II.1 implies that

$$(\mathbf{u}_{k-1}, \mathbf{u}_k) \in \mathbf{E}^{\mathcal{A}},$$

since, for each α,

$$\left(u^\alpha_{k-1}, u^\alpha_k\right) \in E.$$

Consequently, $\mathbf{u}_k$ is a vertex in $\mathbf{V}^{\mathcal{A}}$. The proof is complete.

B. The graph $\mathbf{G}^{\mathcal{A}}$: complexity, sparsity, and an approach to pre-computing

Throughout this appendix, let $G^{DCP} = \left(V^{DCP}, E^{DCP}\right)$ be the depth-coordinated product of a finite collection of rooted digraphs $G^{(k)} = \left(V^{(k)}, E^{(k)}, v^k_{root}\right)$, $k = 1, 2, \ldots, K$.

Definition II.1 implies directly that if all $G^{(k)}$'s are trees, with $G^{(k)}$ having arity[d] $R^{(k)}$ and depth $N^{(k)}$, for $k = 1, \ldots, K$, then G^{DCP} is a tree with root (8), arity $R := \prod_k R^{(k)}$, and depth $N := \min_k N^{(k)}$. Thus, in the worst case (full tree), G^{DCP} has $O\left(R^N\right)$ vertices. The bound

$$R \leq \left(\max_k R^{(k)}\right)^A$$

shows that the size of this tree grows in the worst case exponentially with the number A of flights.

In general, some (possibly all) of the $G^{(k)}$'s may fail to be trees. If at least $G^{(k)}$ fails to be a tree, so does G^{DCP}. The following definition will be instrumental in estimating the size of G^{DCP} in this, general, case.

Definition B.1 *A path*

$$\begin{bmatrix} v^1_{root} & \cdots & v^1_M \\ v^2_{root} & \cdots & v^2_M \\ & \vdots & \end{bmatrix} \tag{18}$$

in G^{DCP} will be called destination-tight *if at least one of the walks*

$$v^k_{root}, \ \ldots, \ v^k_M$$

is a path (i.e., has all vertices distinct) in the corresponding graph $G^{(k)}$.

[d]The *arity* of a tree is the largest of the *outdegrees*[2] of its vertices.

Note that a destination-tight path by definition begins at the vertex in G^{DCP} that is the Cartesian tuple of the v^k_{root}'s. In what follows, the term *vertex length* of a walk will refer to the number of vertex occurrences in that walk.

Theorem B.1 *The vertex length of a destination-tight path in G^{DCP} cannot exceed the largest of the vertex lengths of the paths in the $G^{(k)}$'s that start with the root.*

Proof. For, suppose N is the largest vertex length of a path in any of the $G^{(k)}$'s that starts with the root. If (18) were destination-tight with $M > N$, then there is a k such that

$$v^k_{\text{root}}, \ldots, v^k_M$$

is a path. This path starts at the root in $G^{(k)}$, but has vertex length exceeding N. This contradiction completes the proof.

Thus, a worst-case upper bound on the number of vertices in $G^{(k)}$ is

$$\sum_{n=0}^{N} \left(\max_k \max_{v \in G^{(k)}} \text{outdeg}(v) \right)^{nA}, \tag{19}$$

where N is as in the proof of theorem B.1, and "outdeg" denotes the *outdegree* [2, section A.2] of a vertex in a digraph. This bound, again, grows exponentially with NA.

These bounds suggest that the worst-case cost of computing $\mathbf{G}^{\mathcal{A}}$ is exponential in the number of flights. It is therefore desirable to avoid the necessity to compute $\mathbf{G}^{\mathcal{A}}$ in real time, e.g. to pre-compute. One obstacle to pre-computing is the requirement to know the number A, the origins $v^{ORIG;\alpha}$, and the destinations $v^{DEST;\alpha}$ of the flights in advance of the ATM operation. Since this knowledge is generally unavailable, one possible workaround is the following, analogous to reducing a problem of maximizing flow in a *network*[2] to one with a single *source*[2] and a single sink, by inserting artificially new vertices and edges. For each commonly observed value of A, carry out the following computations offline:

1. Insert into the airspace graph $G = (V, E)$ a new, artificial, vertex $v^{ORIG-ART}$, and insert artificial edges connecting it to all vertices in G with positive outdegree. After modifying $G = (V, E)$ thus, denote the obtained graph by $G' = (V', E')$.

2. Take the graph $\mathbf{G}^{\mathcal{A}}$ to be the depth-coordinated product of A copies of the rooted digraph $G' = \left(V', E', v^{ORIG-ART}\right)$.

By this construction, the vertex $v^{ORIG-ART}$ is adjacent to every possible origin of a flight. Consequently, every possible routing of A flights in G is contained in a path in $\mathbf{G}^{\mathcal{A}}$ starting from the vertex that is the Cartesian A-tuple of the $v^{ORIG-ART}$'s. A pair of artificially inserted edges would receive the weight of zero.

Two other ways of saving computational resources are as follows:

- Make use of the sparsity of $\mathbf{G}^{\mathcal{A}}$, which is a consequence of the sparsity of a typical $G = (V, E)$. For example, the densities (100% (number of edges) / (squared number of vertices)) of $\mathbf{G}^{\mathcal{A}}$ in the last two numerical examples were 0.04% and 0.08%, respectively.

- When computing the required Cartesian products, use parallelization, e.g. *map-reduce*.[33]

Lastly, a circumstance that may alleviate the computational demand of using $\mathbf{G}^{\mathcal{A}}$ is that the number A of aircraft does not grow indefinitely. Rather, an upper bound exists on the total number A of aircraft that may come through a terminal airspace within a time window that may reasonably allow the use of the approach contributed in this paper.

References

[1]Isaacson, D. R., Sadovsky, A. V., and Davis, D., "Scheduling for Precision Air Traffic Operations: Problem Definition and Review of Prior Research," *(in progress)*.

[2]Papadimitriou, C. H. and Steiglitz, K., *Combinatorial Optimization; Algorithms and Complexity*, Dover Publications, 1998.

[3]Administration, U. F. A., *Order JO 7110.65U, Air Traffic Control*, U.S. Dept. of Transportation, Washington, D.C., 2012.

[4]Isaacson, D. R. and Sadovsky, A. V., "Control robustness of Scheduling Precision Air Traffic Operations (PATO)," *(in progress)*.

[5]Sadovsky, A. V., Davis, D., and Isaacson, D. R., "Efficient Computation of Separation-Compliant Speed Advisories for Air Traffic Arriving in Terminal Airspace," Technical Memorandum NASA/TM-2012-216033, NASA, Ames Research Center, Moffett Field, CA 94035-0001, USA, 2012.

[6]Sadovsky, A. V., Davis, D., and Isaacson, D. R., "Optimal routing and control of multiple agents moving in a transportation network and subject to an arrival schedule and separation constraints," Technical Memorandum NASA/TM-2012-216032, NASA, Ames Research Center, Moffett Field, CA 94035-0001, USA, Oct. 2012.

[7]Ghrist, R., O'Kane, J. M., and LaValle, S. M., "Computing Pareto Optimal Coordinations on Roadmaps," *The International Journal of Robotics Research*, Vol. 24, No. 11, 2005, pp. 997–1010.

[8]Jung, J. B. and Ghrist, R., "Pareto optimal multi-robot coordination with acceleration constraints," *Robotics and Automation, ICRA. IEEE International Conference on*, may 2008, pp. 1942 –1947.

[9]Bertsimas, D. and Patterson, S. S., "The Traffic Flow Management Rerouting Problem in Air Traffic Control: A Dynamic Network Flow Approach," *Transportation Science*, 2000, pp. 239–255.

[10]Barnier, N. and Brisset, P., "Graph Coloring for Air Traffic Flow Management," *Annals of Operations Research*, Vol. 130, 2004, pp. 163–178, 10.1023/B:ANOR.0000032574.01332.98.

[11]Sun, D. and Bayen, A., "A multicommodity Eulerian-Lagrangian Large capacity cell transmission model for en route traffic," *AIAA Journal on Guidance, Control and Dynamics*, Vol. 31, 2008, pp. 616–628.

[12]Grabbe, S., Sridhar, B., and Cheng, N., "Central East Pacific Flight Routing," *AIAA Guidance, Navigation, and Control Conference and Exhibit, 21 - 24 August, Keystone, Colorado*, 2006.

[13]Bayen, A., Tomlin, C., Ye, Y., and Zhang, J., "An Approximation Algorithm for Scheduling Aircraft with Holding Time," *IEEE Conference on Decision and Control (CDC)*, 2003.

[14]Sun, D., Clinet, A., and Bayen, A., "A Dual Decomposition Method for Sector Capacity Constrained Traffic Flow Optimization," *Transportation Research Part B*, Vol. 45, 2011, pp. 880–902.

[15]0013, W. Z., Kamgarpour, M., Sun, D., and Tomlin, C. J., "A Hierarchical Flight Planning Framework for Air Traffic Management." *Proceedings of the IEEE*, Vol. 100, No. 1, 2012, pp. 179–194.

[16]Carr, G. C., Erzberger, H., and Neuman, F., "Delay Exchanges in Arrival Sequencing and Scheduling," *Journal of Aircraft*, Vol. 36, 1999, pp. 785–791.

[17]Roy, K., Bayen, A., and Tomlin, C., "Polynomial Time Algorithms for Scheduling of Arrival Aircraft," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005.

[18]Bianco, L., Dell'Olmo, P., and Giordani, S., "Scheduling models for air traffic control in terminal areas," *J. Scheduling*, Vol. 9, No. 3, 2006, pp. 223–253.

[19]Balakrishnan, H. and Chandran, B., "Scheduling Aircraft Landings under Constrained Position Shifting," *AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO*, 2006.

[20]Lee, H. and Balakrishnan, H., "A Study of Tradeoffs in Scheduling Terminal-Area Operations," *Proceedings of the IEEE*, Vol. 96, No. 12, dec. 2008, pp. 2081 –2095.

[21]Chipalkatty, R., Twu, P., Rahmani, A., and Egerstedt, M., "Distributed scheduling for air traffic throughput maximization during the terminal phase of flight," *Decision and Control (CDC), 2010 49th IEEE Conference on*, dec. 2010, pp. 1195 –1200.

[22]Kamgarpour, M., Zhang, W., and Tomlin, C. J., "Modeling and Optimization of Terminal Airspace and Aircraft Arrival Subject To Weather Uncertainties," *AIAA Guidance, Navigation, and Control Conference*, Portland, OR, Aug 2011.

[23]Bradley, S., Hax, A., and Magnanti, T., *Applied Mathematical Programming*, Addison Wesley, 1977.

[24]Vazirani, V. V., *Approximation Algorithms*, Springer, March 2004.

[25]Bayen, A. M., Zhang, J., Tomlin, C. J., and Ye, Y., "MILP formulation and polynomial time algorithm for an aircraft scheduling problem," 2003.

[26]Korn, G. A. and Korn, T. M., *Mathematical Handbook for Scientists and Engineers: Definitions, Theorems, and Formulas for Reference and Review*, McGraw-Hill, New York, 1961.

[27]Sadovsky, A., Swenson, H., Haskell, W., and Rakas, J., "Optimal Time Advance in Terminal Area Arrivals: Throughput vs. Fuel Savings," *IEEE 30th Digital Avionics Systems Conference (DASC), Seattle, WA*, 2011.

[28]Zelinski, S., "Defining Critical Points for Dynamic Airspace Configuration," *International Council for the Aeronautical Sciences (ICAS) Congress, Anchorage, AK*, 2008.

[29]Administration, U. F. A., *Order 8260.54A, The United States Standard for Area Navigation (RNAV)*, U.S. Dept. of Transportation, Washington, D.C., 2012.

[30]Cormen, T., Leiserson, C., and Rivest, R., *Introduction to Algorithms*, The MIT Press, Cambridge, MA, 1990.

[31]MATLAB, *Version 7.7.0.471 (R2008b)*, The MathWorks Inc., 2010.

[32]Gleich, D. F., *Models and Algorithms for PageRank Sensitivity*, Ph.D. thesis, Stanford University, September 2009, Chapter 7 on `gaimc`.

[33]Rajaraman, A. and Ullman, J., *Mining of Massive Datasets*, Cambridge University Press, 2011.

[34]Zelinski, S., "A Graph-Based Approach to Defining Nominal Terminal Routing," *31st Digital Avionics System Conference, Williamsburg, VA*, 2012.