A Scheduling Algorithm Compatible with a Distributed Management of Arrivals in the National Airspace System

1st A. Sadovsky
Aviation Systems Division, Branch AFH
NASA Ames Research Center
Moffett Field, CA, U.S.A.
alexander.v.sadovsky@nasa.gov

2nd R. Windhorst
Aviation Systems Division, Branch AFH
NASA Ames Research Center
Moffett Field, CA, U.S.A.
robert.d.windhorst@nasa.gov

***Abstract*—The current system used by the FAA to schedule arrivals is the Traffic Based Flow Manager (TBFM). It is a centralized system that gives an operator (airline) no influence over scheduled times of arrival assigned to its flights. Future systems for managing arrival scheduling are proposed as distributed systems. Such a system is called upon to give operators influence to schedule and negotiate resources for their flights, and to resolve other technical challenges, such as eliminating a single point of failure. A distributed system for managing diverse air traffic will need the capability of computing a schedule for the given arriving flights in a way that complies with the operational constraints. This paper contributes an algorithm that computes such a schedule. Although developed as part of an effort toward a distributed system, the algorithm itself is neither inherently distributed nor inherently centralized and can be used in either type of system.**

***Index Terms*—arrival scheduling, distributed system, negotiation**

I. Introduction

Airport arrival scheduling assists air traffic controllers with keeping flights properly sequenced and spaced as they approach their destination and land. The current system used by the Federal Aviation Administration (FAA) for arrival scheduling is called Time Based Flow Management (TBFM) [2]. It works by assigning each inbound flight a scheduled time of arrival at its arrival meter fix, which is located in the vicinity of the airport and on the boundary of the *Center* and *TRACON* (Ref. [5]) airspaces. Each flight's scheduled time of arrival, along with a speed advisory, is displayed to the air traffic controllers responsible for the airspaces upstream of the arrival *meter fix* (Ref. [5], the Appendix). The controllers regulate the flights so that they cross their arrival fixes at times as close as possible to their scheduled times of arrival.

TBFM is a centralized closed system operated by the FAA. There are few, if any, ways for airline operators to influence the scheduled times of arrival that are assigned by TBFM to their flights. These features of the system entail the following risks for the TBFM and the FAA. A centralized design is always a potential single point of failure and can limit scalability if there is a need to add more users. Furthermore, the stakeholders (airline operators) are left with no way to influence the scheduled times of arrival.

To address these and other issues, future systems being proposed for managing air vehicles are *distributed systems* [8], offering operators influence to schedule and negotiate resources for their flights. One example of such a system is the *UAS Traffic Management (UTM)*, being developed at NASA for mission planning and management of *Unmanned Aerial Systems (UAS)*. The UTM is a distributed system and is to include a mechanism that guards against unending negotiations. A participant in the system, called a *UAS Service Supplier (USS)* (Ref. [7]), can join the system through a standardized process of authentication and communicates with the other parties via well-defined *Application Programming Interfaces (APIs)*. USS instances schedule and negotiate resources (extruded polygons in the airspace with a time window of proposed use) for UAS missions. This gives all participants direct influence over the negotiation process. Each USS instance is developed and operated by organizations other than the FAA, but they must be certified by the FAA to operate.

In the broader context, past literature contains also other uses of distributed systems in the broad research field of air traffic management. For example, Ref. [10] proposes a distributed system where the agents are the fixes, and Ref. [9] proposes a *swarm intelligence* system for *trajectory-based operations (TBO)*, where the agents are the pilots.

The key structural components of the arrival scheduling problem are the same as in UTM: different participants each operate their own mission, negotiate for resources, and get a service (scheduling). A distributed systems approach here, too, promises the merits outlined above: scalability, elimination of a single point of failure, and a way for each airline operator to develop its own agents to join the system (conforming to the accepted standard APIs) and to influence the negotiation directly.

Such a distributed system for airport arrival scheduling must include a complete and precise formulation of the constraints imposed on the schedules, as well as the ability for each flight that needs a schedule to formulate their objective. For each flight, some of the constraints result from the schedules computed previously for other flights. This leads to the need to negotiate. The distributed system, whatever architecture ends up being chosen for it, must enable such negotiations.

This paper contributes an algorithm for solving such a problem. For clarity, we note that this algorithm, although developed as part of a project aimed at a distributed system, is itself neither inherently distributed nor inherently centralized. It can be used in either type of system.

Before outlining the contribution, we give a brief overview of the challenges of designing such a system, so as to give the broad context for this work.

This system would possibly replace or drive upgrades to the current TBFM system for a futuristic traffic management system for the year 2045 or later. The UTM solution cannot be copied as is because of the following differences between UAS and conventional jet traffic. UAS traffic density is sparser and less directed than conventional jet traffic in the airspace close to large airports. Jet traffic in the vicinity of a large airport follows predefined arrival routes and procedures. The flow of traffic along the routes can be packed to the point where there is no longer space to safely fit flights. The traffic density and the need to land on a common runway will require more negotiation with fewer alternatives than those needed in UTM. Furthermore, jets cannot be stopped in mid-air, meaning that arriving at an agreed solution is time-critical. An arbiter is needed to step in and drive a solution in time critical situations. Finally, the constraints that must be satisfied in arrival scheduling are more complex than checking for airspace box intersections in time and space. In arrival scheduling, there are spacing constraints at the runway and merge points after the arrival meter fix, and there are arrival rate constraints that must be followed. These constraints occur at different points in both time and space and require network flow analysis to work out.

The complete set of arrival scheduling constraints is given in section III. The interdependence of the constraints is explained in section IV, with a high-level description of the algorithmic procedure (appendix A) . The algorithm is formulated in detail in section V. The design of the algorithm is independent of whether it will be run on a centralized or distributed system. Computed solutions to some sample, artificially generated problems are presented in section VI. A discussion on how the algorithm would fit into a distributed system and aid operators in influencing their flight schedules is given in section VII.

II. Problem input and output

A. Input data

Flight routes and the graph they make up: Assume we are given a set of flights, indexed by f: $f = 1, 2, \ldots, F$, and each flight is assigned a route, which is a sequence of waypoints. Generally, a flight is not scheduled at every waypoint on its route, but, rather, on some of those waypoints. These waypoints will be referred to as *nodes*, for consistency with the terminology generally used in scheduling and graph theory, and written in the following notation:

$$n_f^1,\ n_f^2,\ \ldots,\ n_f^{N_f}. \tag{1}$$

We now form the smallest possible undirected graph G that includes all the nodes and all the links involved in the routes (1). More precisely, the set of nodes of G will be

$$\{\ n\ :\ \text{at least one of the routes (1) has } n \text{ as a waypoint}\ \}$$

and the set of its links will be:

$$\{\ \{n_1, n_2\}\quad \begin{array}{l}\text{the waypoints } n_1, n_2 \text{ occur consecutively,}\\ \text{in any order, in at least one of the routes (1) }\}.\end{array}$$

Note: Although each route has its waypoints in a specific order, the graph G is constructed as *undirected*, for convenience of analysis and exposition. (In particular, in this paper G will be restricted to being a tree, and undirected trees allow simpler verbiage than do directed ones.)

Estimated Times of Arrival: Each node n_f^i in the route (1) of each flight f is supplied with an Estimated Time of Arrival ($\mathtt{ETA}$): $\mathtt{ETA}_f^{n_f^i}$, which is the nominal time.

A flight's minimal and maximal times for traversing a link: For each flight f and each link $n_f^i n_f^{i+1}$, we are given the minimal and maximal traversal times

$$tt_f^{n_f^i n_f^{i+1}}, \quad TT_f^{n_f^i n_f^{i+1}}. \tag{2}$$

B. Output data: the format of a "schedule"

A schedule will be defined here as a mapping that, for each flight f, provides every node on its route with a *Scheduled Time of Arrival* ($\mathtt{STA}$) for that flight. In other words, a schedule is an assignment, to each flight f, of a table, denoted by $S(f)$, of the format (refer to (1) for some of the notation):

$$S(f)\ :\ \begin{array}{c|c|c|c|c} \text{node} & n_f^1 & n_f^2 & \ldots & n_f^{N_f} \\ \hline \mathtt{STA} & \mathtt{STA}_f^{n_f^1} & \mathtt{STA}_f^{n_f^2} & \ldots & \mathtt{STA}_f^{n_f^{N_f}} \end{array}. \tag{3}$$

III. Scheduling constraints

A. $\mathtt{ETA}$*s, schedule-frozen nodes, no-passing edges*

For each flight f routed through a node n, the structure of air traffic operations suggests using two types of time stamps: (i) the *estimated time of arrival* $\mathtt{ETA}_f^n$, which is the estimated time when flight f can reach n or, if n is a departure airport, to leave n, and (ii) the *scheduled time of arrival*[1] $\mathtt{STA}_f^n$, which is the time, computed on the basis of the $\mathtt{ETA}$s, at which flight f will be expected at n in actuality.

At some of the schedule-furnished nodes, some flights routed through that node may have pre-assigned and fixed $\mathtt{STA}$s. In real operations, this can happen at a node that is within a flight's *freeze horizon* [5]. Such nodes will be called *schedule-frozen.*

On some of the edges, passing of one flight by another will be forbidden. One operational reason for such a constraint is

[1]If the node is a departure node, this time is actually the schedule time of departure, STD, but for cleanliness of exposition we retain the notation $\mathtt{STA}$ even in this case.

that an airspace may be too narrow to allow safe passing. Such an edge will be called a *no-passing edge*. Also, in real operations, flights are not allowed to pass each other on Standard Arrival Routes and Departure Procedures.

The STAs are subject to constraints that can be separated into two categories, node constraints and edge constraints. We now list the constraints in each category. Every occurrence of the letter c with superscripts and subscripts is a constant parameter, specified as part of the problem statement.

B. Node constraints

The constraints on the STAs of given flights at a given node considered here are of three types. We list these types, labeling each by the letter that matches the corresponding part of formula (4):

(a) The flight cannot leave its first node, n_f^1, earlier than at the corresponding ETA.
(b) Two flights traversing a given node must traverse it at STAs that are no closer together than a required minimal time duration. In other words, the difference between the two STAs must be $\geq$ the pre-specified minimal time separation.
(c) For some nodes along the routes of some flights, the flight (a schedule-frozen flight) must traverse the node at exactly the ETA, which effectively requires the STA to equal the ETA.

The formulas for these constraints are given in the corresponding parts of (4).

$$\left.\begin{array}{lll}
\mathtt{STA}_f^{n_f^1} \geq \mathtt{ETA}_f^{n_f^1} & \left.\begin{array}{l}\text{at the first node}\\ n_f^1 \text{ of flight } f\end{array}\right\} & \text{(a)}\\[2ex]
\mathtt{STA}_{f_2}^{n} - \mathtt{STA}_{f_1}^{n} \geq c_{f_1,f_2}^{n} & \left.\begin{array}{l}\text{if flights } f_1, f_2\\ \text{reach } n\\ \text{consecutively,}\\ \text{in that order}\end{array}\right\} & \text{(b)}\\[2ex]
\mathtt{STA}_f^{n} = \mathtt{ETA}_f^{n} & \left.\begin{array}{l}\text{for all } f \text{ routed}\\ \text{through node } n\\ \text{if } n \text{ has been}\\ \text{schedule-frozen}\end{array}\right\} & \text{(c)}
\end{array}\right\} \quad (4)$$

C. Edge constraints

The constraints on the STAs of given flights at a given node considered here are of two types. We list these types, labeling each by the letter that matches the corresponding part of formula (5):

(a) A flight's travel time between two consecutive nodes–which equals the difference between the corresponding two STAs–must lie between the corresponding minimal and maximal travel times (2).
(b) If one flight passes another on an edge $\left(n^i, n^{i+1}\right)$, then these two flights traverse node n^{i+1} in the order opposite to that of traversing node n^i. Therefore, a no-passing constraint would imply that the two flights must traverse each of the nodes in the same order.[2]

$$\left.\begin{array}{ll}
\left.\begin{array}{l}\text{Bounds on travel time}\\ \text{along } \left(n_f^i, n_f^{i+1}\right):\\[1ex] tt_f^{n_f^i n_f^{i+1}} \leq \mathtt{STA}_f^{n_f^{i+1}} - \mathtt{STA}_f^{n_f^i} \leq TT_f^{n_f^i n_f^{i+1}}\end{array}\right\} & \text{(a)}\\[3ex]
\left.\begin{array}{l}\text{No passing on } \left(n^i, n^{i+1}\right):\\[1ex] \left(\begin{array}{l}\mathtt{STA}_{f_2}^{n^i} \geq \mathtt{STA}_{f_1}^{n^i}\\ \text{implies}\\ \mathtt{STA}_{f_2}^{n^{i+1}} \geq \mathtt{STA}_{f_1}^{n^{i+1}}\end{array}\right) \begin{array}{l}\text{if edge } \left(n^i, n^{i+1}\right)\\ \text{is shared by the}\\ \text{routes of } f_1, f_2\end{array}\end{array}\right\} & \text{(b)}
\end{array}\right\} \quad (5)$$

IV. The interdependence of the constraints

The given flights are scheduled one at a time. In the computation of a schedule, the first step (assumed to have been carried out before the execution of the algorithm in section V) is to impose the order in which the flights are to be scheduled. The STAs computed for the flights scheduled so far will constrain the scheduling options for the next flight to be scheduled through constraints (4.b), (5.b). This section is an explanation of how constraints arise at a node and propagate to other nodes.

Each node used by one or more of the flights is assigned its own "copy" of the time axis. Following the presentation in Ref. [4], this will be illustrated here (figures 1 and 2, below) as a plot where the horizontal axis has only discrete points, corresponding to the nodes of the given flight, and each node's copy of the time axis is drawn as a vertical axis, with upward being the direction of increasing time. The time variable is denoted by t, and its specific values sometimes by T with suitable superscripts. These copies are aligned so that a horizontal line crossing two or more copies crosses them at the same time instant.

For example, suppose the route of flight f includes three consecutive nodes

$$k-1,\ k,\ k+1,$$

and suppose these three nodes are to be used by a previously scheduled flight m, and the two nodes $k,\ k+1$ are to be used by a previously scheduled flight n. Furthermore, suppose node $k-1$ is the first node on the route of flight f. This situation is illustrated notionally in figure 1(A). In general, there can be other previously scheduled flights, using any or all of the three nodes. Following Ref. [4], the color red is used to depict the time windows unavailable to f because of

[2]This restriction on the order of the node traversal does not guarantee the absence of passing. For example, if multiple passings along the edge are possible, then an even number of passings would end up meeting constraint (5.b). But this is the strongest constraint we can impose toward preventing passing without involving the positions of the flights between the nodes and the speeds of the flights.

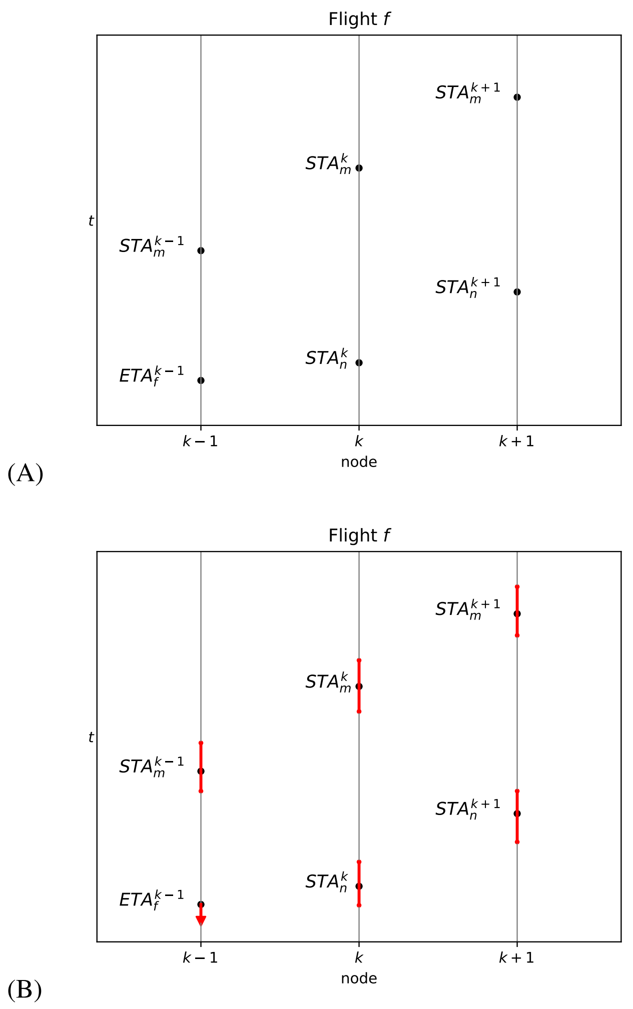
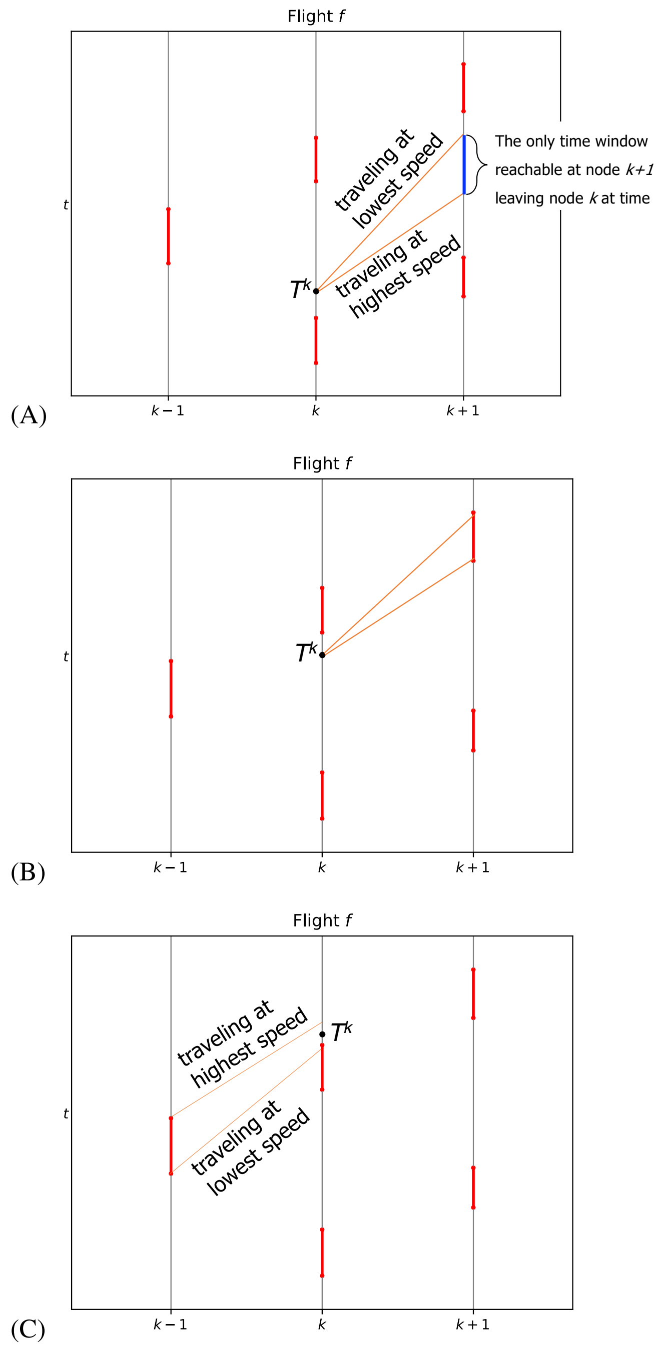


Fig. 1: (A) The STAs of previously scheduled flights m, n and the ETA of flight f at node $k-1$ (here, for the purpose of illustrating semi-infinite unavailable intervals, assumed to be the first node on the route of flight f), each shown on the corresponding node's copy of the (vertical) time axis. (B) The resulting time windows unavailable to flight f because of the separation constraint (4.b) and the ETA constraint (4.a), which results in a semi-infinite unavailable interval (bottommost at $k-1$).

the separation requirements with the STAs of those previously scheduled flights that use any or all of these nodes.

On having computed all the time windows unavailable because of the previously scheduled flights' STAs at node k (figure 1(B)),–i.e., for reasons of time separation at k–and because of flight f's own ETA constraint at its first node, it is generally incorrect to conclude that all the other time instants at k are available to flight f. This is because the availability of node k to flight f is affected not only by the other flights' STAs at k, but also by flight f's speed range. A generic time instant T^k at node k, although separation-compliant, may yet be effectively unavailable for f's use at k



Flight f
traveling at lowest speed
The only time window reachable at node k+1 leaving node k at time
traveling at highest speed
T^k
t
$k-1$
k
$k+1$
(A)
Flight f
T^k
t
$k-1$
k
$k+1$
(B)
Flight f
traveling at highest speed
T^k
traveling at lowest speed
t
$k-1$
k
$k+1$
(C)

Fig. 2: (A) The only times (shown in blue) reachable at node $k+1$ by leaving node k at time instant T^k. (B) With a different choice of T^k, on leaving node k at time T^k, flight f can reach node $k+1$ only at times that are already unavailable to f at $k+1$. This makes node k effectively unavailable to flight f at T^k. (C) The only times of leaving node $k-1$ that allow the flight to reach node k at time T^k are the times unavailable to f at $k-1$.

for two other possible reasons:

- *Controllability:* By leaving node k at time T^k, flight f cannot reach node $k+1$ at any available time.
- *Reachability:* There is no time instant T^{k-1} available to flight f at $k-1$ from which the flight can reach node k at an available time instant at k.

We illustrate each of these reasons in the following two paragraphs, using also figure 2. For this figure, the assumption that $k-1$ is the first node for flight f no longer holds.

Controllability: In figure 2(A), we see that by leaving node k at T^k, flight f will be able to reach node $k+1$ only within a certain time window, shown in blue. Now consider a different T^k; see figure 2(B). The resulting new reachable time window at $k+1$ is contained entirely in an unavailable time window. This makes node k effectively unavailable to flight f at the latter choice of T^k.

Reachability: Figure 2(C) shows such a choice of T^k that flight f cannot reach node k at T^k except by passing through node $k-1$ at an unavailable time. This makes node k unreachable for (hence, effectively unavailable to) flight f at time T^k.

Controllability and reachability characterize the internodal influence of the time windows for f. The algorithm formulated in Ref. [4] is designed, and used here, to update a given list of unavailable time windows for one flight at each node on that flight's route.

V. Algorithm formulation

In a departure operation, one of the parameters that is generally specified by the operational needs is the relative priority of scheduling the departing flights. A flight's priority in the given operation may change, and a new flight may even be added, with a set priority, to an already tentatively scheduled operation. Such changes can require recomputing the schedule, in part or even in entirety. To reflect the presence of priority, our scheduling algorithm will be based on the assumption that the flights are numbered in the order of priority and, therefore, need to be scheduled in that order. A high-level description of the following algorithm is given in appendix A.

The following interval notation will be used: (i) $[a, b]$ denotes the interval with the endpoints included: $a \le x \le b$; (ii) $]a, b[$ will denote the interval: $a < x < b$. The round parentheses, ( ), are reserved in this paper for ordered pairs.

A. The algorithm invariants

Assume the first $(f-1)$ flights have been scheduled; i.e., the schedules of the form (3)

$$S(1),\ S(2),\ \ldots,\ S(f-1) \tag{6}$$

have been computed to comply with all of (4), (5).

B. The generic step: scheduling the next flight

To compute $S(f)$, proceed as follows:

Step 1. For the first node n_f^1 (i.e., the departure node) on the route of f, compute the time window

$$\Big] -\infty,\ \mathtt{ETA}_f^{n_f^1} \Big[, \tag{7}$$

which is unavailable to flight f at that node because of constraint (4.a).

Step 2. For each $\mathtt{STA}_h^{n_h^i}$ occurring in the computed schedules (6), i.e. for every flight h such that $h < f$ and every node n_h^i on the route of flight h, *if this node also occurs on the route of* f, compute the time window

$$\Big[\mathtt{STA}_h^{n_h^i} - c_{h,f}^{n_h^i},\ \mathtt{STA}_h^{n_h^i} + c_{f,h}^{n_h^i} \Big], \tag{8}$$

which is unavailable to flight f at node n_h^i because of constraints (4.b).

Step 3. For each node n_f^i on the route of f at which the flight's STA is frozen, compute the time window

$$\Big] \mathtt{ETA}_f^{n_f^i}, +\infty \Big[, \tag{9}$$

which is unavailable to flight f at node n_h^i because of constraints (4.c).

How condition (9) effects a freezing of the STA: the unavailability of both the time windows (7) and (9) leaves the only possible STA for flight f at node n_h^i: the one given by (4.c).)

Step 4. Use the algorithm in Ref. [4] to compute the available time windows for f at the nodes on its route (the region shown green in Ref. [4, Fig. 2(c)]), using:

- the totality of the intervals (7), (8), (9) as the unavailable time windows (shown red in Ref. [4, Fig. 2]),
- the travel time bounds specified in (5.a) as the travel times (e.g., Ref. [4, Equations (5), (6), (7)]).

Step 5. The output of Step 4 is a collection of time windows of availability of nodes in r_f for flight f. Since a node on the route of f can have multiple such time windows (we can assume they are pairwise nonoverlapping), denote the j-th such window at node n_f^i by $A_f^{n_f^i,j}$.

Satisfaction of constraints (4): The computed time windows $A_f^{n_f^i,j}$ satisfy all the constraints (4.a, c) and satisfy all the pairwise constraints (4.b) as long as the other aircraft has index $< f$ (i.e., already has its schedule computed).

Step 6. To represent the feasible scheduling options for f, form a graph with the time windows $A_f^{n_f^i,j}$ as the nodes, and a window pair

$$\left(A_f^{n_f^i,j_1},\quad A_f^{n_f^{i+1},j_2} \right) \tag{10}$$

constituting an edge if and only if, (i), the two windows correspond to two consecutive nodes on the route of f, and, (ii), flight f can leave node n_f^i some time within window $A_f^{n_f^i,j_1}$ and reach n_f^{i+1}

3some time within window $A_f^{n_f^{i+1},j_2}$ in compliance with constraints (5) applied to all pairs of flights from among $1, 2, \ldots, f-1, f$.
An effect of this compliance: This compliance, together with the satisfaction of constraints (4) designed into the computation of the $A_f^{n_f^i,j}$'s, maintains the invariant formulated in section V-A.
Connection to the passing constraint (5.b): Two edges (10) corresponding to different flights and the same pair of nodes can be directly examined for satisfaction of this constraint.

Step 7. In the graph computed in step 6, find all paths from a time window $A_f^{n_f^1,j}$ corresponding to a departure of f to a time window $A_f^{n_f^{N_f},k}$ corresponding to an arrival of f.
Different objectives that allow being encoded as weights on this graph can be pursued by computing the path shortest in the sense of the chosen objective.

Step 8. Choose a path from those computed in step 7. It will be a sequence of time windows, indexed by the nodes on the route of f:

$$A_f^{n_f^1,j_1},\ A_f^{n_f^2,j_2},\ \ldots,\ A_f^{n_f^{N_f},j_{N_f}} \tag{11}$$

Step 9. From each time window $A_f^{n_f^i,j_i}$ in (11) select a desired $\mathtt{STA}_f^{n_f^i}$. We do it in two steps. The outcome of the first step determines whether the second step is necessary:

- At the first node on the route, assign $\mathtt{STA}_f^{n_f^1} = \mathtt{ETA}_f^{r(1)}$. For all subsequent nodes, assign the STAs according to the preferred nominal times (the quantities in []'s):
$$\mathtt{STA}_f^{r_f(k+1)} = \mathtt{STA}_f^{r_f(k)} + \left[\mathtt{ETA}_f^{r(k+1)} - \mathtt{ETA}_f^{r(k)}\right].$$
- If every computed $\mathtt{STA}_f^{r_f(k)}$ lies in the corresponding time window $A_f^{r_f(k),j_k}$, report the computed $S(f)$ and exit the algorithm. Otherwise, go to step 10.

Step 10. If one or more $\mathtt{STA}_f^{r_f(k)}$ fails to be in the corresponding time window $A_f^{r_f(k),j_k}$, then solve the Quadratic Program formulated in appendix B, which tries to find such STAs as make the flights travel at speeds as close to the nominal (highly preferred) as possible. The found solution gives the suitable $\mathtt{STA}_f^{r_f(k)}$'s.

VI. Sample problems and solutions

This section shows (tables I and II) two problem instances and the solutions computed by the algorithm of section V. The following properties are shared by all the instances:

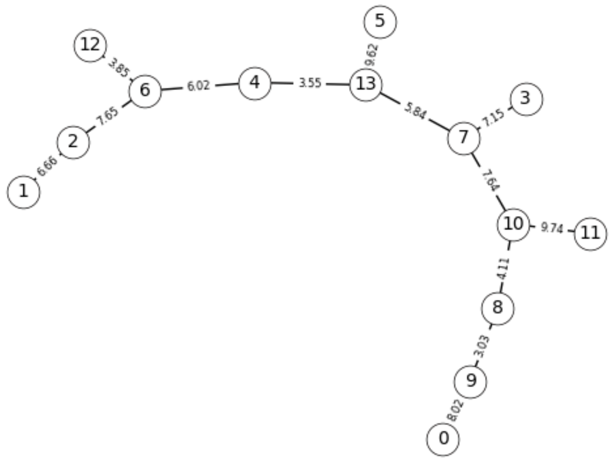
- *Units of measurement:*
  - Distance is measured in nautical miles (NMI).
  - Time is measured in seconds. **Note:** Despite the operational use of whole seconds, as opposed to fractions of seconds, the computed ETAs and STAs shown in the sample solutions are given to the 2nd place after the decimal point. This operationally infeasible accuracy is retained here so as to avoid letting the rounding cause loss of information about the computations. For operational use, rounding is desired and will require care to avoid violation of the separation constraint.
- *Parameters affecting air traffic:*
  - Only four types of aircraft can appear in the problem instance: [s]mall, [m]edium, [L]arge, [H]eavy. The minimal required distance separations between two aircraft at a node depend on the types of the leading aircraft (the one who traverses the node first) and the trailing one. These separations are specified, in nautical miles, in the form of a matrix, where the rows index the trailing aircraft, and columns the leading one:

	s	m	L	H
s	3.0	3.0	3.0	3.0
m	4.0	4.0	4.0	4.0
L	4.0	4.0	5.0	5.0
H	5.0	5.0	5.0	5.0

  - Constraints of the types (4.c) and (5.a) are not imposed. That is, passing is always allowed, and no flight has pre-existing STAs.
  - For each flight, the ETA at the first node of the route is provided as part of the problem and a required parameter for constraint (4.a). The ETAs for the flight's subsequent nodes are estimated by assuming the flight leaves the first node at the provided ETA and travels at the nominal speed of 450.0 KTS:
$$\begin{aligned}&\mathtt{ETA}^{\text{next node}}\\ =\ &\mathtt{ETA}^{\text{current node}} + 3600.0\frac{\text{distance betw. the nodes}}{450.0\text{ KTS}}.\end{aligned}$$
  - The flight scheduled ends up with all of its STAs equal to the corresponding ETAs.
- *The source and structure of the problem instances:*
  - The route network for each problem is a tree weighted by the internodal distances and generated using `random_tree` function [3].
  - Each route network shows only the connectivity of the nodes and the weights. The nodes (shown as circles) are not assigned any specific positions.
  - In each problem instance, all flights are routed to the same node,–the arrival node,–and this node has maximal degree. This is node 13 in the problem instance of table I and node 9 in the problem instance of table II.
  - Each flight's ETA at first node is chosen to make the flights contend for the arrival node.

There are some distinctions between the problem instances in tables I and II. In the 6-flight example of table I, the maximal degree of a node in the route network is 3, and multiple nodes have that degree. In the 9-flight example of

TABLE I: A 6-flight example.



SCHEDULE:						
Flight 0–Aircraf type: B777						
NODE:	0	9	8	10	7	13
ETA:	0.55	64.71	88.95	121.83	182.95	229.67
STA:	0.55	64.71	88.95	121.83	182.95	229.67
Flight 1–Aircraf type: B777						
NODE:	1	2	6	4	13	
ETA:	40.34	93.62	154.82	202.98	231.38	
STA:	43.63	96.91	158.11	206.27	234.67	
Flight 2–Aircraf type: B777						
NODE:	3	7	13			
ETA:	125.62	182.82	229.54			
STA:	135.75	192.95	239.67			
Flight 3–Aircraf type: B737						
NODE:	5	13				
ETA:	154.4	231.36				
STA:	167.71	244.67				
Flight 4–Aircraf type: B777						
NODE:	11	10	7	13		
ETA:	45.66	123.58	184.7	231.42		
STA:	63.91	141.83	202.95	249.67		
Flight 5–Aircraf type: B737						
NODE:	12	6	4	13		
ETA:	123.56	154.36	202.52	230.92		
STA:	147.31	178.11	226.27	254.67		

table II, only node (node 9) has degree 5, and all the other nodes have degree 3 or lower.

VII. Compatibility with a distributed system for managing arrival air traffic

Arrival traffic in a given TRACON [5] can reach high congestion. This, in turn, can lead to the necessity to re-compute a given arrival schedule quickly and reliably. If such computations and conformance monitoring are performed by a single software thread, this thread becomes a single point of failure. Therefore, a distributed system may be desirable for carrying out the scheduling described above.

Whereas each step of the algorithm formulated in section V, including the ordering by priority, can be executed by either a centralized system or a participant in a distributed system, an implementation of a distributed system would require further supporting functionality and offer new capabilities, which we now describe in more detail.

A. Required supporting functionality

The participants of a distributed system, analogous in some ways to instances of a USS (see, e.g., Ref. [7] and Refs therein), would each carry out some of the algorithm described in section IV, maintaining and sharing the required data with the other participants.

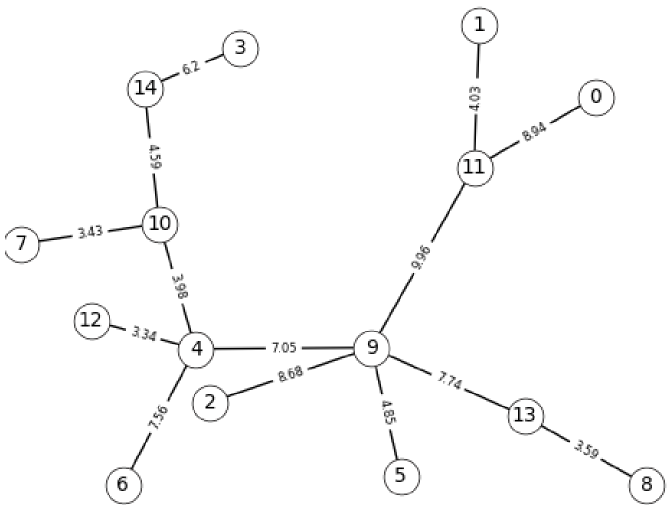
Each such participant, called here an *Arrival Fix Schedule Assistant (AFSA)*, is likely to be handling the schedule $S(f)$ for one or more flights f, and is unlikely to share the task of computing the schedule for a given individual flight with other AFSAs. With this stipulation, an AFSA handling the schedule for flight f is likely to be subject to the following requirements:

1) The ability to determine, for all other flights f' currently in the total schedule (i.e., flights handled by this or other AFSAs), whether f' has a higher or lower scheduling priority than f does.
2) The ability to query the other AFSAs for the presence of flights that (i) have already been scheduled and (ii) share one or more route nodes with the route of f.
3) The ability to determine whether all flights with higher scheduling priority than f have already been scheduled.
4) The ability to share its most recently (re)computed schedule $S(f)$ with the enquiring AFSAs.

B. New capabilities

Steps 9 and, if executed, step 10 of the algorithm in section V both involve selecting an `STA` (for a given flight at a given

TABLE II: A 9-flight example.



SCHEDULE:

Flight 0					
Aircraf type: B777					
NODE:	0	11	9		
ETA:	24.47	95.99	175.67		
STA:	24.47	95.99	175.67		
Flight 1					
Aircraf type: B737					
NODE:	1	11	9		
ETA:	64.86	97.1	176.78		
STA:	68.75	100.99	180.67		
Flight 2					
Aircraf type: B777					
NODE:	2	9			
ETA:	106.97	176.41			
STA:	116.23	185.67			
Flight 3					
Aircraf type: B737					
NODE:	3	14	10	4	9
ETA:	1.34	50.94	87.66	119.5	175.9
STA:	16.11	65.71	102.43	134.27	190.67
Flight 4					
Aircraf type: B737					
NODE:	5	9			
ETA:	135.86	174.66			
STA:	156.87	195.67			
Flight 5					
Aircraf type: B777					
NODE:	6	4	9		
ETA:	60.21	120.69	177.09		
STA:	83.79	144.27	200.67		
Flight 6					
Aircraf type: B737					
NODE:	7	10	4	9	
ETA:	61.81	89.25	121.09	177.49	
STA:	89.99	117.43	149.27	205.67	
Flight 7					
Aircraf type: B737					
NODE:	8	13	9		
ETA:	86.25	114.97	176.89		
STA:	120.03	148.75	210.67		
Flight 8					
Aircraf type: B777					
NODE:	12	4	9		
ETA:	91.71	118.43	174.83		
STA:	132.55	159.27	215.67		

node) from a permissible time window. The preferences in making this selection may vary from one flight operator to another. Therefore, if the system is so distributed that each AFSA is running its own implementation of the algorithm for each of its flights, this enables each AFSA to set their own preferences as to how they execute step 10 or select the objective function in problem (12) when executing step V[3]. This ability to set individual preferences is one way for an operator to influence how their flight is scheduled.

Another such capability is that of negotiating with those flights already scheduled for a contested time window. Development of a process for moderating and arbitrating such a negotiation is an open problem. It is likely to begin with an extensive discussion with the stakeholders (flight operators), aimed at learning their preferences, and to involve modeling of a market-like interaction, akin to game-theoretic models used in economics.

[3]A non-quadratic objective function will result in a problem that is not a Quadratic Program.

VIII. Summary

A distributed system for managing diverse air traffic will need the capability of scheduling the given arriving flights in compliance with the operational constraints. The scheduling algorithm proposed in this paper, while itself agnostic to whether it is run by a centralized system or a distributed one, was developed in the service of such an envisioned distributed system for managing air arrival traffic. Given a collection of flights, ordered by priority, and each with a route and an estimated time of departure, the algorithm schedules one flight at a time, satisfying not only the constraints on the individual flight's individual parameters, but also the separation constraints imposed by the already scheduled flights. The algorithm has a two-layer safety net: it first schedules the flight using a fast heuristic procedure, and then, if the schedule is infeasible, recomputes it using an optimization method. This algorithm design pinpoints the steps where a flight operator can exercise their individual scheduling preferences, which will generally differ from those of other operators'.

Appendix A
A high-level description of the scheduling algorithm

Table III gives a high-level outline of the algorithm. In each part of the outline, we indicate the corresponding numbers of the steps in section V.

Appendix B
Quadratic Program

For n being the k-th node along the route of a given flight f (n is the node label, while k is the index of the node along the route of flight f), introduce the abbreviation

$$s_k = \mathtt{STA}^n_f.$$

Furthermore, let T^{nom}_k be the nominal travel time for f from node n to the next node on route, and $(\underline{a}_k, \overline{a}_k)$ the time window available to flight f at node n; these are the time windows (11), with

$$(\underline{a}_k, \overline{a}_k) = A_f^{n^k_f, j_k}.$$

Let $N = N_f$ be the length of the route.

The problem of scheduling flight f can then be written as the Quadratic Program

$$\left.\begin{array}{l} \epsilon\, s_1 + \frac{1}{2}\sum_{k\geq 2}\left(s_k - s_{k-1} - T^{nom}_{k-1}\right)^2 \longrightarrow \min, \\ \text{where } \epsilon \text{ is a small positive parameter,} \\ \\ \left.\begin{array}{rcl} s_k & \leq & \overline{a}_k \\ -s_k & \leq & -\underline{a}_k \end{array}\right\} \quad \text{for } k \geq 1 \end{array}\right\} \quad (12)$$

We now recast problem (12) in a format that corresponds to that of the Python package `cvxopt`.

The quadratic objective function is

$$\begin{aligned} & \epsilon\, s_1 + \tfrac{1}{2}\textstyle\sum_{k\geq 2}\left(s_k - s_{k-1} - T^{nom}_{k-1}\right)^2 \\ = \;& \epsilon\, s_1 + \tfrac{1}{2}\textstyle\sum_{k\geq 2}\Big[\left(s_{k-1} s_k\right)^2 + \\ - \;& 2\left(s_k - s_{k-1}\right) T^{nom}_{k-1} + \left(T^{nom}_{k-1}\right)^2\Big] \end{aligned}$$

The quadratic part of this sum has the tridiagonal matricial form

$$\begin{aligned} & \tfrac{1}{2}\textstyle\sum_{k\geq 2}\left(s_{k-1} - s_k\right)^2 = \\ & = \tfrac{1}{2}\left[s_1 s_2 \,\ldots\, s_N\right] \begin{bmatrix} 1 & -1 & & \ldots & & \\ -1 & 2 & -1 & & \ldots & \\ & -1 & 2 & -1 & & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & \ldots & & -1 & 2 & -1 \\ & \ldots & & & -1 & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix}. \end{aligned}$$

Letting $x = \left[s_1\ s_2\ \ldots\ s_N\right]^T$, we see that problem (12) has the form

$$\begin{array}{ll} \frac{1}{2}x^T P x + q^T x & \longrightarrow_x \min, \\ G^T x \leq h & \end{array}$$

(there are no linear equality constraints), where

$$P = \begin{bmatrix} 1 & -1 & & \ldots & & \\ -1 & 2 & -1 & & \ldots & \\ & -1 & 2 & -1 & & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & \ldots & & -1 & 2 & -1 \\ & \ldots & & & -1 & 1 \end{bmatrix},$$

$$q = \begin{bmatrix} T^{nom}_1 \\ T^{nom}_2 - T^{nom}_1 \\ T^{nom}_3 - T^{nom}_2 \\ \vdots \\ T^{nom}_{N-1} - T^{nom}_{N-2} \end{bmatrix},$$

$$G = \left[\begin{array}{c|c} I_N & \\ \hline & -I_N \end{array}\right] \quad (I_N = \text{identity matrix of dimension } N),$$

$$h^T = \left[\overline{a}_1 \quad \overline{a}_2 \quad \ldots \quad \overline{a}_N \quad -\underline{a}_1 \quad -\underline{a}_2 \quad \ldots \quad -\underline{a}_N\right].$$

For this problem class, the `cvxopt` Python module [1] has a numerical quadratic programming solver that can be called as follows:

```
from cvxopt import solvers
sol = solvers.qp(P, q, G, h)
```

TABLE III: A high-level outline of the scheduling algorithm of section V.

What is being done	**In which steps in section V**
For each node n on the route of f, compute the time windows unavailable to flight f at n because of the flight's ETA and because of the separation requirement with the already scheduled flights' STAs. These time windows are determined by constraints (4).	1, 2
If the `ETA` at the first node on the route of f is frozen, add the corresponding time window to the list of the time windows unavailable to f at that node.	3
Using the algorithm in Ref. [4], update the unavailable time windows to include the limitations on controllability and reachability, as explained in section IV.	4
Because the speed of f is bounded, if f reaches a node in a given time window, the next node may not be reachable in some of its available time windows. For every two consecutive nodes n_f^k, n_f^{k+1} on the route of f, consider each pair $$\Big(\text{ time window at } n_f^k, \quad \text{time window at } n_f^{k+1}\Big) \qquad (13)$$ and record the pair if the latter window is attainable from the former. Regard these pairs as links in a graph, which will be called *the scheduling graph for f*, and with the available time windows being the nodes of the scheduling graph.	5, 6; window pair (13) is window pair (10)
For each link in the scheduling graph, define a "weight" that reflects the stakeholders' and arbiter's preferences. The scheduling graph for f becomes a *weighted graph* [6]. Compute a *shortest path* in this graph.	7, 8
The shortest path computed in the scheduling graph for f is a sequence of available time windows, which will be called *weight-optimal time windows for f*. Each weight-optimal time window is assigned to one of the nodes n_f^k,. Each next weight-optimal time window is reachable from the previous one (i.e., from the one at the previous node). Within the weight-optimal time window at each n_f^k, compute an $\mathtt{STA}_n^{n_f^k}$ using the following heuristic procedure: • At the first node on the route of f, let $\mathtt{STA}_f^1 =$ (starting time of weight-optimal time window at this node). • At all subsequent nodes n, let $\mathtt{STA}_f^{n+1} = \mathtt{STA}_f^n +$ (the nominal travel time for f from n to $n+1$). • If possible, subtract the same positive time length from all the `STA`s, to "remove the slack"; i.e., to make sure that at least one of them is at the first time instant of its time window.	9
If all the `STA`s thus computed fit into the corresponding weight-optimal time windows, report them as the computed schedule for f. Otherwise, solve a Quadratic Program (appendix B) for such `STA`s as satisfy the constraints and result in travel speeds for f that are as close to the nominal as possible.	10

References

[1] M. S. Andersen, J. Dahl, and L. Vandenberghe. CVXOPT: A python package for convex optimization, version 1.1.5. *seas.ucla.edu/~vandenbe/publications/coneprog.pdf*, 2012.

[2] FAA-E-3025. *Time-Based Flow Management (TBFM) system specification document (SSD)*. Federal Aviation Administration, Washington, D.C., 2014.

[3] A. Hagberg, P. Swart, and D. Schult. Exploring network structure, dynamics, and function using networkx. *http://networkx.github.io*, 2008.

[4] L. Meyn. A closed-form solution to multi-point scheduling problems. In *AIAA Modeling and Simulation Technologies Conference*, page 7911, 2010.

[5] D. Mulfinger and A. Sadovsky. Design and evaluation of a stochastic time-based arrival scheduling simulation system. In *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, including the AIAA Balloon Systems Conference and 19th AIAA Lighter-Than*, page 6874, 2011.

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

[7] J. L. Rios, I. S. Smith, P. Venkatesan, D. R. Smith, V. Baskaran, S. M. Jurcak, S. K. Iyer, and P. Verma. UTM UAS service supplier development: Sprint 2 toward technical capability level 4. *NASA Technical Memorandum*, 2018.

[8] A. S. Tanenbaum and M. Van Steen. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[9] S. Torres and K. L Delpome. An integrated approach to air traffic management to achieve trajectory based operations. In *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, pages 3E6–1. IEEE, 2012.

[10] K. Tumer and A. Agogino. Distributed agent-based air traffic flow management. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 255. ACM, 2007.