Optimal Large-Scale Air Traffic Flow Management

Hamsa Balakrishnan
Massachusetts Institute of Technology
Cambridge, MA 02139
hamsa@mit.edu

Bala G. Chandran
Resilient Ops LLC
Winchester, MA 01890
bala.chandran@resilientops.com

Abstract

This paper presents an integer programming approach for solving large-scale air traffic flow management problems. Given flight-specific operating and delay costs, the proposed approach uses column generation to determine optimal trajectories in space and time in the presence of network and flight connectivity constraints as well as airport and airspace capacity constraints. Ground holds, airborne delays, reroutes and cancellations are considered as possible control actions. The approach is then extended to generate recourse strategies in the presence of uncertain capacity constraints, which are represented using probabilistic scenario trees. A scalable, parallel implementation of our approach is used to solve nation-scale examples from the United States, showing that the approach is fast enough for real-time implementation.

1 Introduction

Air traffic delays result in significant costs to airlines as well as passengers, and can cause massive disruptions in the air transportation network. The Joint Economic Committee, US Senate (2008) has estimated that domestic flight delays in the United States (US) in 2007 had a $31-40 billion impact on the economy. In 2013, over 20% of domestic air carrier flights were delayed by more than 15 minutes; the Bureau of Transportation Statistics (2014) estimates that 30% of these delays were weather-related, while another 12% were due to traffic volume.

Air Traffic Flow Management (ATFM) refers to the task of strategically modifying the departure times and trajectories of flights in order to address capacity-demand imbalances, which occur either when capacity is reduced, or when demand is high. The corresponding mathematical formulation is referred to as the Traffic Flow Management Problem (TFMP). Although airports have traditionally been the most capacitated elements of the National Airspace System (NAS) in the US, airspace sectors also experience congestion, especially during periods of bad weather. Airspace sector congestion is a more chronic problem in Europe, where it occurs in conjunction with airport congestion (Lulli and Odoni 2007).

The NAS serves approximately 20,000 scheduled air carrier operations on a typical high-demand day (Bureau of Transportation Statistics 2014), and this number is expected to grow significantly in the next decade (Joint Planning and Development Office 2007). Flight connectivity, when the same aircraft is used on multiple legs in sequence, further complicates the ATFM problem. Figure 1 shows the high levels of flight connectivity on a typical day in the NAS (only 6% of flights on the day had no connection while a typical aircraft performed 4–6 flights in a day). As a result of high levels of connectivity, accounting for downstream impacts of TFM actions is critical. Consequently, the problem cannot be solved by considering small geographical regions or time horizons in isolation without loss of efficiency. It is therefore desirable to develop scalable optimization approaches that can handle problems of a nationwide scale over time horizons spanning an entire day.

Uncertainty in capacities also poses a significant challenge to air traffic flow management. Airport and airspace sector capacities are greatly influenced by weather conditions, including visibility, winds and the location of storms. As a result, the exact capacity of an airport or sector is often not known with certainty hours ahead of time, when strategic traffic flow management decisions need to be made. This fact motivates the need for stochastic optimization algorithms for ATFM that explicitly account for capacity uncertainty.

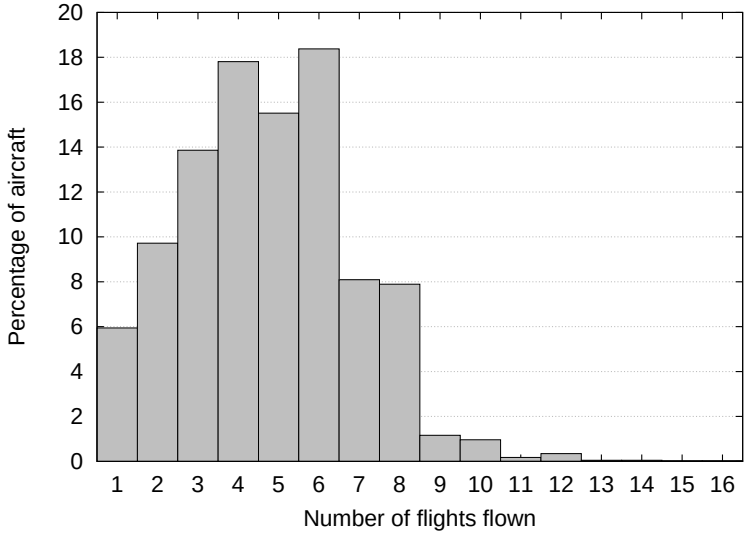


Figure 1: Histogram showing level of connectivity on July 8, 2013 with a total of 4,054 distinct tails and 19,217 flights (average of 4.75 flights per tail). Data from Bureau of Transportation Statistics (2014).

2 Related work

Since Odoni (1987) proposed the first mathematical formulation of the flow management problem in air transportation, many researchers have investigated approaches to address air traffic congestion. Bertsimas and Odoni (1997), Richard (2007) and Vossen et al. (2012) present comprehensive literature surveys on this topic. Helme (1992) formulated the NAS traffic management problem with airport and sector capacity constraints as a multi-commodity minimum-cost flow problem space-time network, but this approach was found to have weak computational performance. In their seminal paper on the TFMP, Bertsimas and Stock Patterson (1998) showed that the problem was $\mathcal{NP}$-hard and proposed a binary integer programming formulation that considered both airspace and airport capacities, but did not make routing decisions. A key contribution of this work was the development of a strong formulation of the TFMP in which many of the constraints were shown to be facet-defining, which led to good computational performance. Subsequently, Bertsimas and Stock Patterson (2000) used multi-commodity network flows to model rerouting. Their approach to solving the problem was to aggregate the flows, decompose the problem using Lagrangian relaxation to solve the linear program, and then apply rousing heuristics and solve a packing problem to generate individual flight paths.

In more recent work, Bertsimas et al. (2011) presented an extension of the formulation in Bertsimas and Stock Patterson (1998) to account for reroutes on an acyclic network. They developed valid inequalities for nodes in the network with in-degree or out-degree equal to 1, which were shown to be facet-defining for the problem. This model was shown to have good computational performance when applied to regional-sized and nation-sized instances.

Richard (2007) and Richard et al. (2011) developed a column generation approach to generate 4-dimensional trajectories for tactical ATFM problems. Their approach generates detailed routing and altitude guidance for flights to relieve localized short-term capacity imbalances, typically 30 minutes before overload.

In Sherali et al. (2003) and Sherali et al. (2006), the authors considered tactical air traffic control issues such as potential aircraft conflicts, as well as Collaborative Decision Making (CDM) between airlines. Similarly, Chaimatanan (2014) developed genetic algorithms to minimize aircraft trajectory interactions during air traffic flow management.

Aggregate models of the traffic flow management problem inspired by fluid flows have also been investigated

Reference	Control	Scale	Horizon/disc.	Run times
Deterministic instances				
Maugis (1995)	Ground holds; cancellations	4,743 flights; 1,153 sector-saturated time periods (no airport capacity limits)	1 day/5 min	2+ hours (no cancellations)
Bertsimas and Stock Patterson (1998)	Ground/air holds	1,002 flights; 18 airports; 305 sectors	8 hours/5 min	8+ hours
Bertsimas and Stock Patterson (2000)	Ground/air holds; limited rerouting	71 flights; 4 airports; 42 sectors	8 hours/5 min	4 min
Bertsimas et al. (2011)	Ground/air holds; rerouting network	6,745 flights; 30 airports; 145 sectors	8 hours/15 min	10 min
Wei et al. (2013)	Aggregate model; air holds	3,419 flight paths; 284 sectors	2 hours/1 min	21 min
This paper	Ground/air holds; unrestricted rerouting network; cancellations	17,500 flights; 370 airports; 375 sectors	24 hours/5 min	5 min
Stochastic instances				
Alonso et al. (2000)	Ground/air holds; max. delay 4 periods	160 flights; 4 airports; 5 sectors; 13 scenarios	4 hours/5 min	31 min
Marron (2004)	Ground/air holds; rerouting	148 flights; 40 sectors; 3 scenarios	Not spec./5 min	12 min
This paper	Ground/air holds; unrestricted rerouting network; cancellations	17,500 flights; 370 airports; 375 sectors; 5–25 scenarios	24 hours/10 min	5–20 min

Table 1: Comparison of the test cases presented in this paper, relative to the state-of-the-art. Note that run times are not necessarily comparable across these models given differences in computing environments, but are presented here for context.

(Menon et al. 2006, Sridhar et al. 2006, Sun et al. 2007, Sun and Bayen 2008). These models, known as *Eulerian* models, record aircraft counts in a region, and have been found to qualitatively reflect traffic in congested airspaces. Traffic flow management algorithms for these aggregate models have also been proposed, including distributed approaches using dual decomposition (Bayen et al. 2006, Sun et al. 2011, Wei et al. 2013).

Airport and airspace capacities can vary significantly with weather conditions, which are difficult to predict with certainty at long time horizons (Pfeil and Balakrishnan 2012). As a consequence, there is a need for approaches that can work under uncertainty (Odoni 1987). Given the difficulty in solving large-scale deterministic ATFM problems, the literature on solving stochastic TFM problems is typically limited to optimizing flows into a single capacitated airport under probabilistic scenario-tree forecasts of airport capacity (Richetta and Odoni 1994, Ball et al. 2003, Mukherjee and Hansen 2007). Liu et al. (2008) and Buxi and Hansen (2009) have developed techniques to determine probabilistic capacity profiles and scenario tree forecasts from historical data. Mukherjee and Hansen (2009) proposed a model that incorporated dynamic rerouting into the single-airport ground holding model. Nilim et al. (2002) and Nilim et al. (2003) modeled weather uncertainty as a stationary Markov chain, and proposed a dynamic programming algorithm to route aircraft. Marron (2004) proposed a column generation approach to the ATFM problem with rerouting, and demonstrated it on a regional-scale example with about 150 flights. Gupta and Bertsimas (2011) studied robust and adaptive optimization formulations of the ATFM problem to address capacity uncertainties. Andreatta et al. (2011) proposed a stochastic programming model for ATFM by aggregating aircraft flows between the same origin-destination pairs. Table 1 compares the test cases solved in this paper relative to the state-of-the-art.

In a practical implementation, the stochastic ATFM problem will be used to determine a strategic operational plan in the presence of uncertain forecasts. As new information is obtained, the deterministic ATFM problem will provide the tactical plan. In other words, both the stochastic and the deterministic ATFM solutions play an important role in the efficient operation of the air traffic system.

2.1 Contributions of this paper

This paper considers the problem of determining the optimal departure times, routes, speeds and cancellations for aircraft in the entire National Airspace System, given deterministic as well as probabilistic scenario-tree forecasts of airport and airspace capacities. The proposed column generation approach is demonstrated using national-scale examples drawn from operational data. Our computational experiments show that the proposed approach can determine very good integer solutions to the deterministic and stochastic ATFM problems, solving large-scale instances (24 hours at a 5-minute or 10-minute discretization, with ∼17,500 flights and realistic airport and airspace sector constraints) in approximately 5 minutes (deterministic capacities) and 5-15 min (stochastic capacities, depending on the number of scenarios), respectively. To the best of our knowledge, these experiments are among the largest realistic instances of the ATFM problem that have been optimized to date.

3 Model and Notation

3.1 Network structure

Our representation of the problem is a standard node-link network along which aircraft are routed. The model consists of the following components, as illustrated in Figure 2.

Node: A node can be either a physical location, corresponding to a region in the airspace, or a decision point. For example, in Figure 2, the "Hold" node represents a decision to hold an aircraft at the origin gate and not a physical movement of the aircraft, whereas the "Dep. fix" node is a physical location that the aircraft passes through.

Arc: An arc is a directed segment connecting two nodes. It is associated with a *strictly positive* minimum transit time, maximum transit time, and cost as a function of transit time.

Sector: A sector is a contiguous region of airspace. An arc is required to be fully contained within a sector (i.e., it cannot cross sector boundaries); therefore, a sector could be viewed as a collection of arcs. Nodes are introduced at the sector boundaries to ensure that arcs do not cross sectors, although a node could be present in the interior of a sector.

Tail: A tail refers to a physical aircraft.

Flight: A flight is a an operation of an aircraft from an origin to the destination; a tail may consist of multiple flights over a time horizon. For a given tail, the destination airport of a flight must be the origin airport of the successor/connecting flight. Each flight is associated with a set of arcs that form the network along which that aircraft can be routed from its origin to destination. The parameters of the arcs (minimum and maximum transit times and cost) are flight-specific.

Trajectory: A trajectory or "4D-trajectory" is a sequence of node-time combinations that represent the flight path of an aircraft. It implicitly specifies the arcs along the path and the transit times on those arcs.

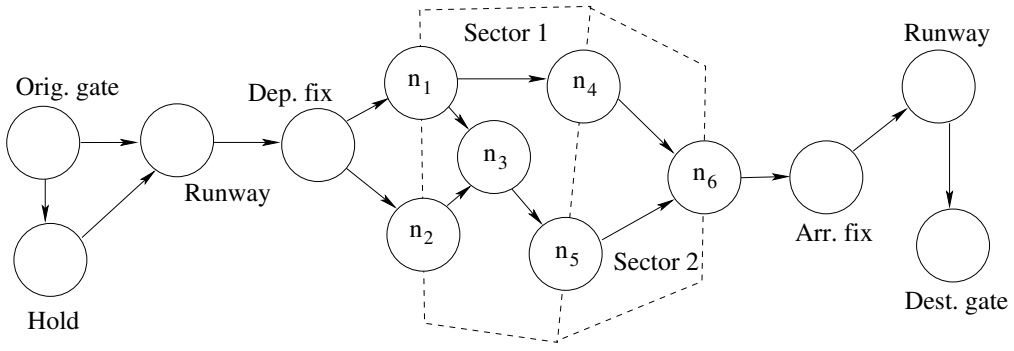


Figure 2: Network representation of the ATFM problem.

3.1.1 Network model of an airport:

While the interpretation of a node, arc, and sector is straightforward in the en-route airspace, our network model of an airport requires a deeper discussion. The airport network consists of an origin node (which could be viewed as the gate), represented by the "Origin" node in Figure 3.

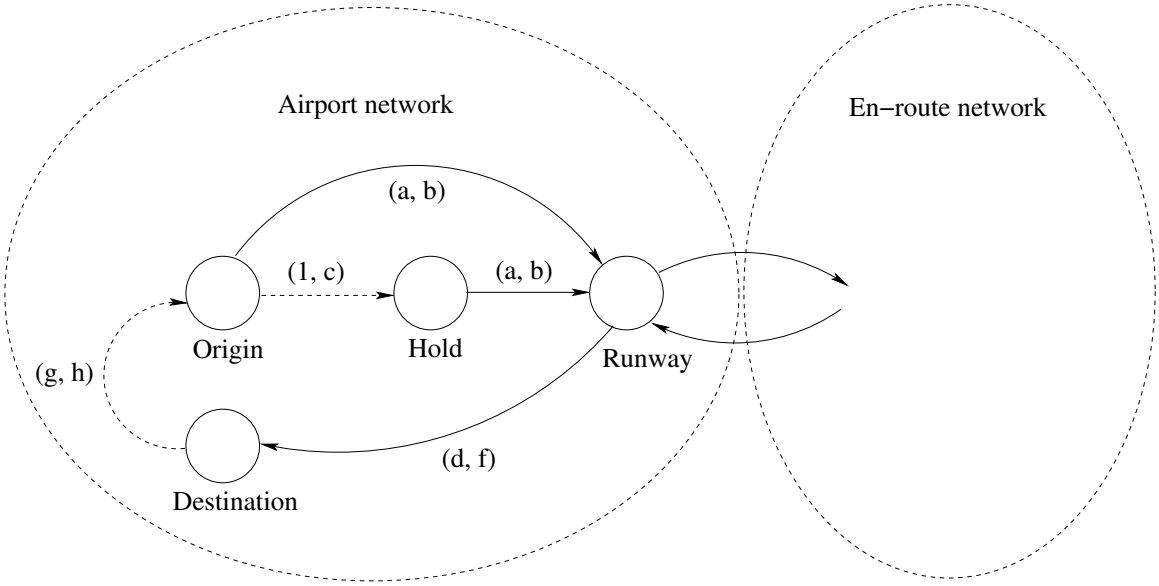


Figure 3: Network representation of an airport. Values in parentheses represent the minimum and maximum transit times on each arc. Physical movement of aircraft occur on arcs with solid lines.

From the origin node, the aircraft could proceed to the departure runway with minimum and maximum taxi times of a and b respectively. Note that this arc implies that the aircraft leaves the gate and any delay is accrued on the surface of the airport after pushback. The second option available at the gate is to hold the airport for at least 1 time unit and up to c time units (the minimum hold cannot be zero since otherwise it would be equivalent to the direct arc to the runway). If the aircraft is held, it "proceeds" to the gate node (although this is not a physical movement of the aircraft) and then to the runway with the same minimum and maximum times of a and b as before. The aircraft then leaves the airport at the runway and enters the en-route network until it gets to its destination.

At the arrival airport, an aircraft arrives at the runway, and then taxis to the gate with minimum and maximum transit times of d and f respectively. If the aircraft is continuing on to a next flight, it "travels" from the destination node to the origin node with a minimum transit time equal to the turnaround time g, and maximum transit time h which is the difference between the scheduled departure of the outgoing flight and the earliest arrival of the incoming flight. Instead of a single runway node, the model could alternatively consist of independent departure and arrival runways.

3.2 Constraints

There are two types of constraints that are imposed in our model, as follows.

3.2.1 Operational constraints:

These constraints are flight-specific and specify what actions may be performed by a flight. They include minimum and maximum transit times on arcs, maximum ground and air delay, minimum turnaround time between successive flights, and routing restrictions, which are imposed by limiting the set of arcs that a flight can use. The feasible actions vary by aircraft performance characteristics such as the nominal speed and altitude.

3.2.2 Capacity constraints:

Capacity constraints represent aggregate limits on flows in the network. There are two types of capacity constraints, as listed below.

Sector capacity constraints: These constraints limit the number of aircraft that can be in a sector at any time, and are driven by the geometry of the sector as well as air traffic controller workload (Vossen et al. 2012). On the surface, the taxi arcs (represented by the solid lines in Figure 3) collectively form the "surface" sector, and it is possible to apply a limit on the number of aircraft that are on the surface at any time. Such constraints may be necessary in order to avoid surface gridlock during extreme events.

Node throughput constraints: These constraints limit the flow through a node at any time. For example, miles-in-trail and minutes-in-trail constraints stipulate the minimum spacing between two aircraft (Vossen et al. 2012); if there is a spacing requirement of 2 minutes between successive aircraft, it translates to a node throughput of 30 aircraft per hour.

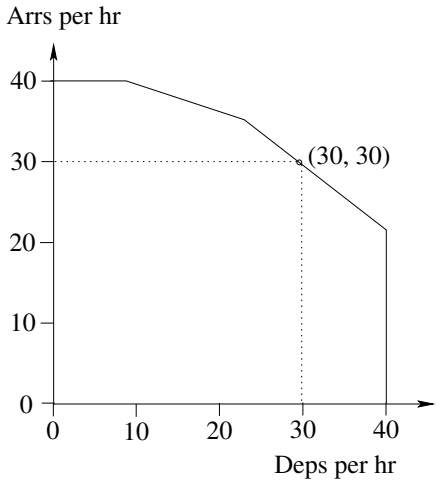


Figure 4: Example of runway capacity envelope with four segments.

The throughput constraints at some nodes, such as runways, are more complex since different types of operations (departures and arrivals) can occur at them. In that case, the node throughput constraint at any time is represented by a capacity envelope composed of segments that specify the tradeoff between arrival and departure operations. For example, the envelope shown in Figure 4 stipulates that if the runway is in a "departures-only" or "arrivals-only" operating mode, the limit on the number of operations is 40 per hour. However, in the case of mixed (arrival and departure) operations, the throughput may be higher. For instance, the runway is capable of handling 30 arrivals and 30 departures per hour. Since envelopes are typically convex (Gilbo 1993), the individual segments can be modeled as independent constraints, the intersection of which forms the capacity envelope.

3.3 Capacity uncertainty

Both airport and sector capacity depend on the weather conditions such as the visibility, cloud ceiling and the location of thunderstorms, and are therefore prone to uncertainty (Pfeil and Balakrishnan 2012). Buxi and Hansen (2009) have developed techniques to determine scenario tree forecasts of airport capacity from historical meteorological data.

Similar to the representation of uncertainty in Ground Delay Programs (Richetta and Odoni 1994, Ball et al. 2003, Mukherjee and Hansen 2007), we represent capacity uncertainty using scenario trees, an example of which is shown in Figure 5. In the example, scenario S_1 starts at 9 AM. At 9:30, scenario S_2 materializes with probability 0.4 and scenario S_3 with probability 0.6. If scenario S_2 materializes, then at 10:45 AM, scenario S_4 occurs with probability 0.3 and scenario S_5 with probability 0.7, and so on. The capacity of a sector or a

node during a scenario is represented by its *profile*, which specifies the capacity or envelope at each time during duration of the scenario.

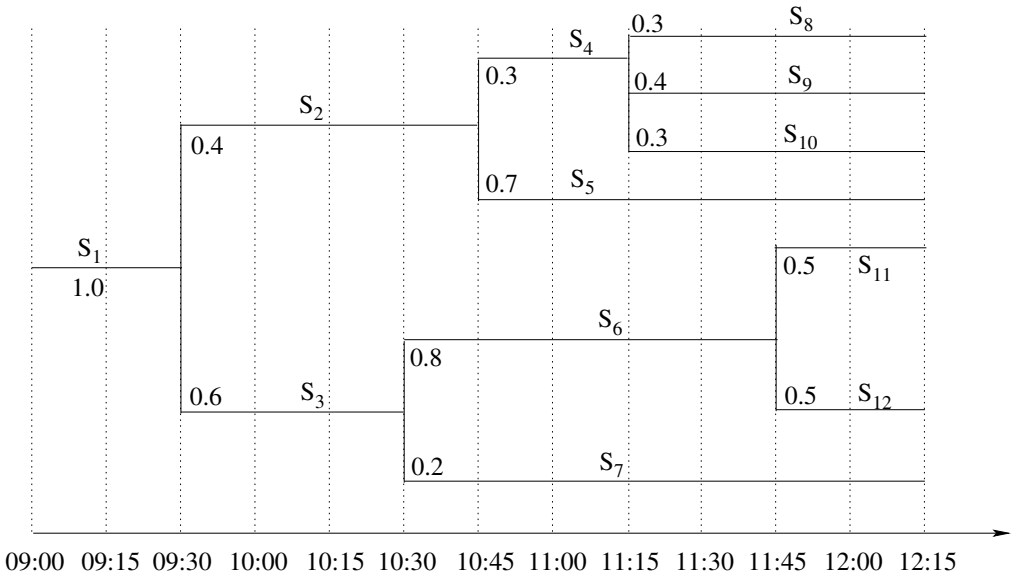


Figure 5: Example of a scenario tree with 12 scenarios.

When solving for the route for an aircraft in the presence of a scenario tree, our solution for an aircraft is a trajectory for each scenario that could unfold; such a set of trajectories is called a *trajectory tree*. The trajectory tree specifies the location of the aircraft at each time during a scenario and conditional statements on what actions to perform as each new scenario unfolds. For example, the trajectory tree for an aircraft given the scenario tree in Figure 5 might be "Leave the gate at 9:05, arrive at the runway at 9:15, arrive at the departure fix at 9:30; if scenario S_2 materializes, proceed toward node n_1 and arrive at 9:45, else if scenario S_3 materializes, proceed toward node n_2 and arrive at 10:05," and so on. The trajectory tree satisfies the "coupling constraint" that any decision can be based only on information that is available at the time that the decision is made, i.e., only on scenarios that have materialized up until that point.

3.4 Time-discretization

The time horizon is discretized in the mathematical formulation of the problem, implying that all transit times in the network are integer multiples of the time period, and all operations occur at a set of periodic epochs. Prior TFMP studies have typically used time periods of 15 minutes (Bertsimas and Stock Patterson 2000, Bertsimas et al. 2011), while we explore time discretizations of 5 to 10 minutes in our experiments.

Apart form uncertainty in capacities, there is some uncertainty in the ability of the aircraft to precisely follow a 4D-trajectory, also known as conformance uncertainty. Trajectory conformance is affected by several sources of uncertainty, including the capabilities of onboard equipment such as the Flight Management System (FMS), weather and wind conditions, pilot behavior, etc. In this paper, we choose not to model conformance uncertainty since some of this uncertainty is implicitly absorbed in the time discretization. Further, even in the current environment, air traffic controllers accommodate small deviations from the expected capacity using tactical maneuvers such as vectoring and speed changes, and conformance is expected to improve significantly in the future (Joint Planning and Development Office 2007).

The deterministic problem is stated as follows.

Problem statement (deterministic): Given a set of flights (and associated tails), and airport and airspace capacity constraints, identify a 4D-trajectory tree for each tail that maximizes the system-wide benefit (revenue plus cancellation penalty) minus costs (operating costs plus delay costs), and that obeys operational and capacity constraints for all time periods.

The stochastic problem statement is as follows.

> **Problem statement (stochastic):** Given a set of flights (and associated tails), a scenario tree and capacity profiles, identify a trajectory tree for each tail that maximizes the system-wide expected benefit (revenue plus cancellation penalty) minus expected costs (operating costs plus delay costs), and that obeys operational and capacity constraints for all time periods in all scenarios.

Although the deterministic TFMP is a special case of the stochastic problem in which the scenario tree has only one scenario with probability 1, we describe it as separate problem for purposes clarity of description.

4 Deterministic Traffic Flow Management

By contrast to prior models of the deterministic TFMP, rerouting in our model is not restricted to an enumerated set of simple paths from the origin to the destination (Bertsimas and Stock Patterson 2000), restricted to an acyclic network (Bertsimas et al. 2011), or limited in geographical scope (Mukherjee and Hansen 2009). Instead, our model allows route choices on a significantly larger network, with no restrictions on the network structure.

4.1 Mathematical program

We now describe our mathematical model in detail, using the following notation.

$\mathcal{F}$	Set of flights.
$\mathcal{L}$	Set of tails.
ℓ_f	The tail of flight f.
$\mathcal{S}$	Set of sectors.
$\mathcal{T}$	Set of time periods in the time horizon.
$\mathcal{N}$	Set of nodes in the network.
$\mathcal{A}$	Set of arcs in the network. An arc $a \in \mathcal{A}$ is defined by its start and termination nodes n_1 and n_2 as $a = (n_1, n_2)$, referred to as the *head* and *tail* of the arc respectively.
$\mathcal{A}_F(f)$	Set of arcs that can be used by flight $f \in \mathcal{F}$.
$\mathcal{A}_N(n)$	Set of arcs with head node $n \in \mathcal{N}$, also referred to as the "outgoing" arcs from node n.
$\mathcal{A}_S(s)$	Set of arcs in sector $s \in \mathcal{S}$.
s_a	Sector containing arc a.
$\mathcal{R}$	Set of all feasible 4D-trajectories, where a 4D-trajectory specifies the routing in space and time of a *tail*, and not that of a single flight. Specifying the trajectory implicitly specifies the tail. A trajectory need not include all flights for a tail. It could contain only the 4D-trajectory for an aircraft from its first flight until the first flight cancellation, and all flights with no trajectory (the first cancelled flight and all subsequent connections) are considered cancelled.
$\mathcal{R}_L(\ell)$	The set of feasible trajectories for tail ℓ.
$P(a,t)$	The number of aircraft on arc $a \in \mathcal{A}$ at time $t \in \mathcal{T}$.
$Q(a,t)$	The number of aircraft that enter arc $a \in \mathcal{A}$ at time $t \in \mathcal{T}$.
$\mathcal{J}(n,t)$	The set of segments in the capacity envelope of node n at time t.
ρ_r	The benefit (revenue plus cancellation penalties) minus costs (operating plus delay) of trajectory $r \in \mathcal{R}$.
C_f	The benefit (revenue plus cancellation penalties) of flight $f \in \mathcal{F}$.

Intersection between a trajectory and sector capacity constraint. Sector capacity constraints limit the total number of aircraft in a sector, which can be written as

$$\sum_{a \in A_S(s)} P(a,t) \leq B_{s,t} \qquad \forall s \in \mathcal{S}, t \in \mathcal{T} \tag{1}$$

where $B_{s,t}$ is the capacity of sector s at time t.

A trajectory $r \in \mathcal{R}$ is said to *intersect* with a sector capacity constraint for sector s at time t if the trajectory results in an aircraft being present in sector s at time t. A trajectory can intersect with at most one sector capacity constraint at any time (in the deterministic case). The set of trajectories that intersect with a capacity constraint representing sector s at time t is denoted by $\mathcal{R}_S(s,t)$.

Intersection between a trajectory and node capacity constraint. Node capacity constraints limit the throughput of aircraft through a node. Although the node capacity envelope at a certain time could consist of multiple segments, we treat each segment as an independent constraint, as the intersection of all segments at a certain time defines the envelope. Therefore, when we refer to a node constraint, we refer to one segment $j \in \mathcal{J}(n,t)$ of the envelope of a node n at time t. The node throughput constraint is linear, and is written as

$$\sum_{a \in A_N(s)} \sigma_{a,t,j} Q(a,t) \leq D_{n,t,j} \qquad \forall n \in \mathcal{N}, t \in \mathcal{T}, j \in \mathcal{J}(n,t) \tag{2}$$

where σ and D are constants that define the shape of the segment.

A trajectory $r \in \mathcal{R}$ is said to *intersect* with a node capacity constraint for node n at time t if the trajectory results in an aircraft entering an outgoing arc of node n at time t (i.e., the aircraft passes through node n at time t). The set of trajectories that intersect with a node capacity constraint representing envelope segment j of node n at time t is denoted by $\mathcal{R}_N(n,t,j)$.

To summarize, we have introduced the following additional notation.

$B_{s,t}$	The capacity of sector s at time t.
$D_{n,t,j}$	The right-hand-side of the linear constraint representing segment j of the capacity envelope for node n at time t.
$a_{r,t}$	The arc that an aircraft following trajectory r is on at time t. (This arc is unique since we are only dealing with the deterministic case with one scenario).
$\sigma_{a,t,j}$	The coefficient of arc a in the linear constraint representing segment j of the capacity envelope for the head node of arc a at time t
$\mathcal{R}_L(\ell)$	The set of trajectories of tail ℓ.
$\mathcal{R}_S(s,t)$	The set of trajectories that intersect with sector capacity constraint of sector s at time t.
$\mathcal{R}_N(n,t,j)$	The set of trajectories that intersect with segment j of the capacity envelope for node n at time t.

The traditional formulation of the TFMP has capacity and operational constraints, with binary decision variables that indicate whether a flight has reached a sector by a certain time period. By contrast, our formulation has only capacity constraints, and all the other constraints (minimum and maximum transit times, flight connectivity, turnaround times, etc.) are absorbed into the definition of the variable. This formulation results in fewer constraints but an exponentially greater number of variables. The decision variables are defined as follows:

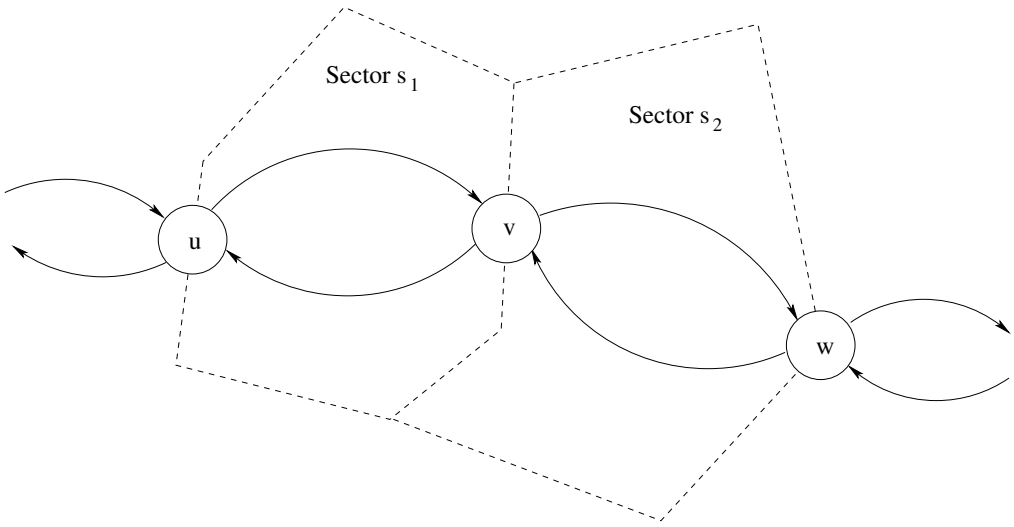
$$x_r = \begin{cases} 1 & \text{if trajectory r is chosen, and} \\ 0 & \text{otherwise} \end{cases} \qquad \forall r \in \mathcal{R} \tag{3}$$

The Integer Master Problem (IMP) may now be stated as follows.

$$\text{(IMP)} \qquad \begin{aligned}
\text{maximize} \quad z &= \sum_{r \in \mathcal{R}} \rho_r \, x_r && && (4)\\
\text{s.t.} \quad \sum_{r \in \mathcal{R}_L(\ell)} x_r &\leq 1 \qquad \forall \ell \in \mathcal{L} && (5)\\
\sum_{r \in \mathcal{R}_S(s,t)} x_r &\leq B_{s,t} \qquad \forall s \in \mathcal{S}, t \in \mathcal{T} && (6)\\
\sum_{r \in \mathcal{R}_N(n,t,j)} \sigma_{a_{r,t},t,j} \, x_r &\leq D_{n,t,j} \qquad \forall n \in \mathcal{N}, t \in \mathcal{T}, j \in \mathcal{J}(n,t) && (7)\\
x_r &\in \{0,1\} \qquad \forall r \in \mathcal{R} && (8)
\end{aligned}$$

Objective (4) maximizes the total benefit minus cost of all trajectories selected. Constraint (5) states that at most one trajectory may be selected for each tail. Constraints (6) and (7) are the sector and node capacity constraints respectively.

Example Consider two tails ℓ_1 and ℓ_2 going Left-to-Right and Right-to-Left respectively, as shown in Figure 6.



Tail	Trajectory (node, timestamp)
Tail ℓ_1 (Left-to-Right)	..., (u, 9:05), (v, 9:15), (w, 9:30), ...
Tail ℓ_2 (Right-to-Left)	..., (w, 9:00), (v, 9:15), (u, 9:25), ...

Figure 6: Illustrative example for the mathematical formulation.

Let the capacity of sector s_1 be 30 aircraft from 9:00 to 9:15 and 40 aircraft from 9:15 to 9:30, and the capacity of sector s_2 be 10 aircraft from 9:00 to 9:15 and 20 aircraft from 9:15 to 9:30.

Let the node capacity envelope be represented by two segments

$$\begin{array}{lrcl} \text{Segment } j_1: & 1.0\ (\text{Flow from v to w}) + 2.0\ (\text{Flow from v to u}) & \leq & 100\ \text{aircraft/hr} \\ \text{Segment } j_2: & 2.0\ (\text{Flow from v to w}) + 1.0\ (\text{Flow from v to u}) & \leq & 80\ \text{aircraft/hr} \end{array}$$

Given a discretization of 5 minutes, the right hand sides of the above node capacity constraints must be divided by 12 to get the equivalent constraint within a 5-minute interval. The resulting mathematical program is shown in Figure 7.

4.2 Solution approach

Since there are an exponentially large number of possible trajectories, we resort to column generation to solve the problem. The book by Desaulniers et al. (2005) contains a primer in column generation as well as a sampling of applications in various areas.

In order to solve the IMP, we first solve the linear programming relaxation of the master problem (henceforth referred to as LMP), i.e., the linear program obtained by replacing constraint (8) with the following:

$$x_r \geq 0 \qquad \forall r \in \mathcal{R} \tag{9}$$

An upper bound of 1 is not needed since the variables are implicitly bounded by constraint (5).

In column generation, we solve a restricted master problem (RMP) with a subset of the variables, and use the dual values from the solution of the restricted problem to identify new variables to add to the RMP. Each time we solve the RMP, we also generate an integer solution using a heuristic to estimate the optimality gap. The process terminates when no more variables can be added or when the optimality gap is sufficiently small. Once the LP has been solved (which gives us an upper bound on the integer solution of the LMP), we generate

x_1	x_2				
ρ_1	ρ_2	=	z		Objective
1		$\leq$	1	tail ℓ_1	At most one trajectory per tail
	1	$\leq$	1	tail ℓ_2	
1		$\leq$	40	s_1, 9:05-9:10	
1		$\leq$	40	s_1, 9:10-9:15	
	1	$\leq$	50	s_1, 9:15-9:20	
	1	$\leq$	50	s_1, 9:20-9:25	
	1	$\leq$	10	s_2, 9:00-9:05	Sector capacity
	1	$\leq$	10	s_2, 9:05-9:10	
	1	$\leq$	10	s_2, 9:10-9:15	
1		$\leq$	20	s_2, 9:15-9:20	
1		$\leq$	20	s_2, 9:20-9:25	
1		$\leq$	20	s_2, 9:25-9:30	
1	2	$\leq$	(100/12)	v, 9:15, segment j_1	Node capacity
2	1	$\leq$	(80/12)	v, 9:15, segment j_2	

Figure 7: Matrix representation of illustrative mathematical program formulation for a five-minute discretization.

an integer solution to the problem via a heuristic using the columns of the RMP (which gives us a lower bound on the optimal solution of the IMP). Finally, we solve the integer version of the RMP to generate the final solution to our problem. This process is illustrated in Figure 8. Since we are solving the IP on a subset of all the trajectories, this process does not necessarily solve the IMP to optimality however, the optimality gap is known at termination and our experiments show that the solutions obtained are of very high quality.

```
0  begin
1      Start RMP with empty variable set and set all duals to zero
3      Solve subproblem to find new variables
4      while new variables exist and time limit is not reached and optimality gap exceeds threshold do
5          Add new variables to RMP
6          Solve LP
7          Generate feasible solution using heuristic
8      end while
9      Solve IP with variable set of last LP
10 end
```

Figure 8: High-level description of the column generation process.

4.2.1 Solving the restricted master problem:

The restricted master problem is solved using IBM ILOG CPLEX. Our experiments, discussed in greater detail in Section 6, describe the specific settings we used to tune CPLEX's performance.

We denote the duals from the RMP as follows.

π_ℓ	The dual variable for constraint (5) for tail ℓ.
$\lambda_{s,t}$	The dual variable for constraint (6) for sector s at time t.
$\mu_{n,t,j}$	The dual variable for constraint (7) for segment j of the capacity envelope for node n at time t.

4.2.2 Solving the subproblem:

Given the duals of a RMP, a subproblem is solved to identify variables with positive reduced cost that can be added to the RMP (or determine that no such variables exist). Since the subproblems are separable by tail (the only coupling constraints across tails are the capacity constraints which are accounted for the RMP), we solve a

pricing subproblem independently for each tail to identify the best trajectory to add for it (or to determine that no trajectory exists). The subproblem for a tail is equivalent to a longest path problem on a directed acyclic network, and is solved using dynamic programming.

The following notation will be used in describing the subproblem:

$t^{\min}_{a,f}$	The minimum transit time (in periods) of flight f on arc a.
$t^{\max}_{a,f}$	The maximum transit time (in periods) of flight f on arc a.
$t^{\text{turn}}_{f_1,f_2}$	The minimum turnaround time (in periods) between connecting flights f_1 and f_2.
$\Omega(a,\tau,f)$	The operating cost of flight f on arc a with transit time of τ, which can be any arbitrary function. In our experiments, we model this as a linear function of τ, with the en-route arcs having the greatest cost, followed by taxi-in/out arcs, followed by ground hold arcs which have the lowest cost. We assume that the operating cost of a trajectory is separable across the arcs, i.e., the total operating cost of a flight equals the sum of operating costs on the arcs.
$\Delta(f,t)$	The total delay cost of flight f arriving at its destination at time t.

The network on which the subproblem is solved is called the space-time network, or ST-network, and consists of the following components.

ST-Nodes: These are nodes in the space-time network. There is a node for every feasible node-time-flight combination for each tail, which is defined by (n,t,f) representing the node $n \in \mathcal{N}$, time $t \in \mathcal{T}$, and flight $f \in \mathcal{F}$. An ST-Node (n,t,f) exists if and only if one of the Rules 1, 2, or 3 is satisfied.

Rule 1 *n is the origin of f* **and** *t is the earliest pushback period for f.*

Rule 2 *n is the destination of f* **and** *t is between the earliest and latest arrival times of f.*

Rule 3 *n is not an origin or destination* **and** *t is between the earliest pushback and latest arrival time of f.*

Rule 1 states that there is only one space-time node corresponding to the origin of a flight, and it exists at the earliest pushback time of the flight. Rules 2 and 3 state that the space-time nodes corresponding to a node that is not an origin of a flight must be between the earliest and latest arrival of the flight. Collectively, the three rules imply that any path for a flight through the space-time network must depart the origin at the earliest departure time and arrive at the destination between the earliest and latest arrival times of the flight.

ST-Arcs: An arc in the ST-network is directed between two ST-Nodes. An arc exists between two space-time nodes (n_1,t_1,f_1) and (n_2,t_2,f_2) if and only if one of the Rules 4–5 is satisfied.

Rule 4 $f_1 = f_2$ **and** $a = (n_1,n_2) \in \mathcal{A}_F(f_1)$ **and** $t^{\min}_{a,f_1} \le (t_2 - t_1) \le t^{\max}_{a,f_1}$.

Rule 5 $a = (n_1,n_2) \in \mathcal{A}$ **and** f_1 *connects to* f_2 *and* n_1 *is the destination of* f_1 **and** n_2 *is the origin of* f_2 **and** $t_2 - t_1 \ge t^{turn}_{f_1,f_2}$.

Rule 4 states that if the flights corresponding to the two ST-Nodes are the same (i.e., it is not a connecting flight), then the arc from n_1 to n_2 must exist in the flight's network, and that the difference in times between the two nodes must be between the min and max transit times of the arc. Rule 5 states that if the two flights are different, then it is a connecting arc and the difference in time between the two ST-Nodes must be between the turnaround time and the difference between the latest departure of the connecting flight and the earliest arrival of the incoming flight. Collectively, the two arcs capture the routing restrictions on the flights and the min and max transit times on each arc.

The weight or "length" of an ST-Arc for all turnaround arcs is zero. For all other ST-Arcs from node (n_1,t_1,f) to (n_2,t_2,f), the weights are calculated as:

$$\begin{aligned}\text{Weight} \;=\;& C_f \text{ (Benefit, applies only to arcs to the destination of flight } f) \\ & - \Delta(f,t_2) \text{ (Arrival delay cost, applies only to arcs to the destination of flight } f) \\ & - \Omega(a,(t_2-t_1),f) \text{ (Operating cost)} \\ & - \sum_{j \in \mathcal{J}(n_1,t_1)} \sigma_{a,t_1,j}\ \mu_{n_1,t_1,j} - \sum_{t=t_1}^{t_2-1} \lambda_{s_a,t} \text{ (Node dual cost + sector dual cost)}\end{aligned}$$

Finally, we introduce an arc with zero weight from the destination of each flight to a sink. These arcs allow the creation of partial trajectories where all flights in a tail beyond a certain destination are cancelled. Define an *ST-path* as a path from the origin node of the tail to the sink.

Lemma 4.1 *An ST-path exists if and only if it represents a trajectory that satisfies the operating constraints including minimum and maximum transit times, network connectivity, flight connectivity, turnaround time requirements, and the earliest pushback and latest arrival times of each flight.*

Proof Proof. The proof follows from construction of the network.

To prove the "forward" direction of the statement, i.e., that an ST-path in the network is a feasible 4D-trajectory, we observe that Rules 1–3 above impose earliest departure and latest arrival times on the path. Rule 4 restricts the path to belong to $A_F(.)$, which captures aircraft capability constraints and network connectivity requirements. Rule 4 also imposes minimum and maximum transit times on all arcs. Rule 5 captures flight connectivity for a tail and imposes restrictions on the turnaround time.

The proof of the "reverse" direction, i.e., that any feasible 4D-trajectory is a path in the network, is by contradiction. Suppose the trajectory of a tail is given by the (node, time, flight) sequence $\{(n_i, t_i, f_i)\}$ that is not represented in the ST-network. Then, (a) there must be a node (n_j, t_j) that is not a node in the ST-network or (b) there must be an arc from (n_j, t_j, f_j) to $(n_{j+1}, t_{j+1}, f_{j+1})$ that is not contained in the ST-network. However, no such node can exist because the network is constructed by exhaustively enumerating all possible node-flight-time combinations and including all nodes that satisfy the constraints. Similarly, no such arc can exist since the set of arcs is constructed by exhaustively examining all possible node combinations and only choosing those that satisfy the constraints. Therefore, the nodes and arcs in the ST-network are actually a superset of the nodes and arcs that are feasible to a 4D-trajectory. ■

Lemma 4.2 *For a given tail, the weight of an ST-path in the space-time network minus the tail's dual π_ℓ equals the reduced cost associated with the corresponding 4D-trajectory.*

Proof Proof. The reduced cost of the trajectory corresponding to a path is given by

$$\begin{aligned}\text{Reduced cost} \quad = \quad & \text{Sum of benefits across all flights} \\ & - \text{Arrival delay cost} \\ & - \text{Operating cost} \\ & - \text{Duals of constraints (6)} \\ & - \text{Duals multiplied by appropriate coefficient of constraints (7)} \\ & - \text{Dual of constraint (5)}\end{aligned}$$

The ST-path correctly captures the sum of benefits and arrival delay cost since the benefit C_f and cost $\Delta(f, t)$ of each flight f is accounted for in the arcs entering the destination of the flight, so any path passing through the destination of a flight picks up the benefit and delay cost associated with the flight. The operating costs are accounted for by associating the appropriate $\Omega(.)$ with each arc, and summing along the arcs of the path gives the total operating cost of the trajectory.

The cost of each arc also includes the sum of sector duals λ associated with the particular sector-time combination (since the arc-time combination maps to a sector-time combination). The node capacity duals are associated with each ST-node in the path (i.e., the time at which the flight passes through the node) through the product of σ and μ for each segment of the capacity envelope.

The weight of the ST-path therefore contains the benefit, delay cost, and dual costs associated with the capacity constraints. The reduced cost of the 4D-trajectory can thus be obtained by subtracting duals corresponding to the tail constraints (5) from the weight of the ST-path. ■

Corollary 4.1 *A trajectory with strictly positive reduced cost exists if and only if there is a path of length strictly greater than π_ℓ in the space-time network.*

Thus, in order to determine whether a trajectory exists in each subproblem, we solve a longest path problem on the ST-network and identify the 4D-trajectory associated with the longest path. Since the ST-network is a directed acyclic graph (each arc moves forward in time as all transit times are strictly positive), the longest path can be obtained by dynamic programming (Sedgewick and Wayne 2011).

Note that in addition to solving for the reduced cost on the path, we also obtain as a byproduct the objective function coefficient ρ of the path, which is used to construct the new column.

4.2.3 Lower and upper bounds:

At any given time, the objective value of the LP of the RMP is a lower bound on the optimal LP value. The reduced costs can be used to calculate an upper bound on the LP, as follows.

An upper bound. We now show that it is possible to calculate an upper bound on the LP value after each iteration.

Suppose we have a primal problem (P) and the corresponding dual (D)

$$
\begin{array}{lll}
(P) & & \\
\max & \sum_i c_i x_i & \\
\text{s.t.} & \sum_i a_{i,j} x_i \leq b_j & \forall j \\
& x_i \geq 0 & \forall i
\end{array}
\qquad\qquad
\begin{array}{lll}
(D) & & \\
\min & \sum_j b_j y_j & \\
\text{s.t.} & \sum_j a_{i,j} y_j \geq c_i & \forall i \\
& y_j \geq 0 & \forall j
\end{array}
$$

Once we solve a restricted master problem, we have values of x and y that are feasible to the above constraints and $\sum_i c_i x_i = \sum_j b_j y_j$ (in our case, primal is feasible and bounded, so dual exists and the objective functions of the primal and dual at optimality are equal).

The reduced costs associated with each variable (in the primal) are denoted by

$$\phi_i = c_i - \sum_j a_{i,j} y_j,$$

which can be rewritten as

$$c_i = \phi_i + \sum_j a_{i,j} y_j.$$

Suppose the value of the optimal LP is z^* and the optimal values of the variables x^*, then

$$
\begin{aligned}
z^* &= \sum_i c_i x_i^* \\
&= \sum_i x_i^* \left(\phi_i + \sum_j a_{i,j} y_j \right) \\
&= \sum_i x_i^* \phi_i + \sum_j y_j \sum_i a_{i,j} x_i^* \\
&\leq \sum_i x_i^* \phi_i + \sum_j y_j b_j.
\end{aligned}
$$

The value of $\sum_j y_j b_j$ is known – it is the value of the current objective function of the RMP. The value of $\sum_i x_i^* \phi_i$ can now be bounded as follows: in each sub-problem, we identify the most positive value of reduced cost among all trajectories for a given tail. Let $\phi_\ell^{\max}$ be the largest reduced cost of all trajectories for tail ℓ (or zero if the largest reduced cost is non-positive). Since the sum of x_i for a tail is bounded by 1 due to constraint (5), $\sum_i x_i^* \phi_i \leq \sum_\ell \phi_\ell^{\max}$. Thus, after the LP of each RMP has been solved, an upper bound on the optimal LP of the LMP is calculated as the current value of the objective function plus the sum of positive reduced costs across all tails.

The bounds on the LP solution are therefore given by:

$$
\begin{aligned}
\text{LP solution to RMP at column generation termination} &\leq \text{Optimal LP to Master Problem} \\
&\leq \text{LP upper bound}
\end{aligned}
$$

We use the gap between the lower and upper LP bounds to terminate the column generation process when the gap is below a threshold of 0.01%.

4.2.4 A heuristic:

After each LP is solved, we apply a simple heuristic to assess the optimality gap, and terminate if necessary. The heuristic is also useful for seeding the IP with a "warm start" feasible solution.

The heuristic proceeds in two stages given the solution to an LP. In the first stage, each variable is randomly rounded up or down depending on its value as long as it does not violate any constraints. If the value of the variable is $\alpha \in [0, 1]$, the variable is rounded up with probability α and down with probability $(1-\alpha)$. To ensure that variables with a value of zero or one in the LP relaxation have some probability of being rounded, the values of the variables are first truncated to the interval $[\epsilon, (1-\epsilon)]$ so that even variables with a value of 0 or 1 have a probability of ϵ of being rounded away from their LP value. If rounding up a variable causes a constraint to be violated, it is rounded down to zero.

In the second stage, the variables that were not selected in first stage (i.e., that were rounded down) are sorted by decreasing objective function value and greedily rounded up in the sorted order, as long as it does not violate any constraints. The procedure terminates when all variables have been examined.

Since the procedure is random in nature, the heuristic is run multiple times to generate a pool of feasible solutions, and the one with the highest objective value is chosen to seed the IP. We did not attempt to develop a more sophisticated heuristic as this simple heuristic appears to be very effective at generating high quality solutions with run times that are orders of magnitude less than solving the LP and IP. Further, most variables in the LP solution are integers, requiring few variables to be rounded.

We terminate the column generation process if the gap between the heuristic solution and the LP upper bound is less than 0.1%.

4.2.5 Solving the IP:

Once the RMP has been solved to optimality, we solve the IP version of the RMP, which is obtained by restricting the variables to be binary. While the solution to this IP could, in theory, be far from the optimal solution the the IMP, our experiments show that solving the IP of the RMP results in high quality solutions that are within 0.1% of the optimal solution to the IMP.

Since the IP is provided a heuristic solution to start with, CPLEX only needs to generate integer solutions when the heuristic is not within 0.1% of optimal (i.e., the column generation process terminated because the time limit was exceeded or the LP gap of 0.01% was hit without finding a feasible solution within 0.1%).

In our experiments, we observe that the IP is typically solved at the root node (i.e., with no branching) since the LP is tight and the heuristic solution is optimal or close to optimal.

The bounds for the optimal IP solution to the Master Problem are

$$\begin{aligned} \text{Heuristic IP} &\leq \text{IP solution to RMP} \\ &\leq \text{Optimal IP to Master Problem} \\ &\leq \text{Optimal LP to Master Problem} \\ &\leq \text{LP upper bound} \end{aligned}$$

In our experiments, we report on the optimality gaps, calculated as follows:

$$\text{LP Gap} = (\text{LP upper bound} - \text{LP solution to RMP})/\text{LP solution to RMP}. \quad (10)$$

$$\text{IP Gap} = (\text{LP upper bound} - \text{IP solution to RMP})/\text{IP solution to RMP}. \quad (11)$$

4.2.6 Parallel implementation:

Since the subproblem time is typically greater than the time to solve the RMP or the final IP, and the subproblem for each tail is independent of other tails, we can parallelize the subproblem computation to significantly reduce run times. The fact that the subproblem is easy to parallelize makes the algorithm extremely scalable. We note that while solving a regular IP can also be parallelized (and is done by commercial solvers), using our parallel implementation makes our algorithms scale *predictably* with the number of cores. In Section 6.3.2, we discuss the scaling performance of the algorithm with the number of parallel threads.

5 Stochastic Traffic Flow Management

We now describe our approach to solving the traffic flow management in the presence of capacity uncertainty represented via scenario trees, as described in Section 3.3.

The solution to the stochastic case is similar to that of the deterministic case, except that the solution is no longer a simple path through a space-time network, but is instead a tree in a space-time-scenario network.

The notation used in the stochastic case is described below. In general, we use a " $'$ " to distinguish between the stochastic variable/parameter and its deterministic counterpart; for example the decision variable in the stochastic formulation is denoted by x' while that in the deterministic formulation is denoted by x.

$\mathcal{G}$	The set of scenarios.
p_g	The unconditional probability that scenario g will materialize, i.e., it is the product of the conditional probabilities along the path from the root of the scenario tree to g. For example, in Figure 5, the unconditional probability of scenario S_4 is $0.3\times0.4\times1.0 = 0.12$.
$\mathcal{T}_G(g)$	The set of time periods spanned by scenario g. The start and end times of scenario g are denoted by t_g^{start} and t_g^{end} respectively.
$\mathcal{R}'$	Set of all feasible 4D trajectory trees. Each trajectory tree is tail-specific, and therefore, specifying it implicitly specifies the tail. A trajectory tree could potentially cancel a subset of flights under certain scenarios.
$P'(a,t,g)$	The number of aircraft on arc $a \in \mathcal{A}$ at time $t \in \mathcal{T}$ in scenario $g \in \mathcal{G}$.
$Q'(a,t,g)$	The number of aircraft that enter arc $a \in \mathcal{A}$ at time $t \in \mathcal{T}$ in scenario $g \in \mathcal{G}$.
$\mathcal{J}'(n,t,g)$	Set of segments in the capacity envelope of node n at time t in scenario g.
$B'_{s,t,g}$	The capacity of sector s at time t in scenario g.
$D'_{n,t,j,g}$	The right-hand-side of the linear constraint representing segment j of the capacity envelope for node n at time t during scenario g.
$\sigma'_{a,t,j,g}$	The coefficient of arc a in the linear constraint representing segment j of the capacity envelope for the head node of arc a at time t in scenario g.
$a'_{r,t,g}$	The arc that an aircraft following trajectory tree r is on at time t in scenario g.
$\mathcal{R}'_L(\ell)$	The set of trajectory trees of tail ℓ.
$\mathcal{R}'_S(s,t,g)$	The set of trajectory trees that intersect with sector capacity constraint of sector s at time t in scenario g.
$\mathcal{R}'_N(n,t,j,g)$	The set of trajectories that intersect with segment j of the capacity envelope for node n at time t in scenario g.
ρ'_r	The expected benefit (revenue plus cancellation penalties) minus expected costs (operating plus delay) of trajectory tree r.

5.1 Mathematical Program

As in the deterministic case, a variable represents whether or not a trajectory tree is selected.

$$x'_r = \begin{cases} 1 & \text{if trajectory tree r is chosen, and} \\ 0 & \text{otherwise} \end{cases} \qquad \forall r \in \mathcal{R}' \tag{12}$$

The Stochastic Integer Master Problem (SIMP) may now be stated as follows.

(SIMP)

$$\text{maximize} \quad z = \sum_{r\in\mathcal{R}'} \rho'_r\, x'_r \tag{13}$$

$$\text{s.t.} \quad \sum_{r\in\mathcal{R}'_L(\ell)} x'_r \le 1 \qquad \forall \ell \in \mathcal{L} \tag{14}$$

$$\sum_{r\in\mathcal{R}'_S(s,t,g)} x'_r \le B'_{s,t,g} \qquad \forall s \in \mathcal{S}, g \in \mathcal{G}, t \in \mathcal{T}_G(g), \tag{15}$$

$$\sum_{r\in\mathcal{R}'_N(n,t,j,g)} \sigma_{a'_{r,t,g},t,j,g}\, x'_r \le D'_{n,t,j,g} \qquad \forall n \in \mathcal{N}, g \in \mathcal{G}, t \in \mathcal{T}_G(g), j \in \mathcal{J}(n,t,g) \tag{16}$$

$$x'_r \in \{0,1\} \qquad \forall r \in \mathcal{R}' \tag{17}$$

Note that this formulation is almost identical to the deterministic formulation, except for the additional scenario index in the constraints. Objective (13) maximizes the total benefit minus cost of all trajectory trees

selected. Constraint (14) states that at most one trajectory tree may be selected for each tail. Constraints (15) and (16) are the sector and node capacity constraints respectively.

As in the deterministic case, we solve the linear relaxation of this master problem using column generation. We refer to the linear programming relaxation of the SIMP as SLMP and the restricted master problem as SRMP.

5.2 Solving the subproblem

Solving the subproblem is equivalent to solving for a maximum weighted tree in a space-time-scenario network using dynamic programming. Let the duals from the SRMP be denoted as follows.

π'_ℓ	The dual variable for constraint (14) for tail ℓ.
$\lambda'_{s,t,g}$	The dual variable for constraint (15) for sector s at time t during scenario g.
$\mu'_{n,t,j,g}$	The dual variable for constraint (16) for segment j of the capacity envelope for node n at time t during scenario g.

5.2.1 The space-time-scenario network:

The space-time-scenario network (henceforth referred to as the STS-network or simply as the network) is shown in Figure 9.

The network has two kinds of nodes:

- **Type-1 nodes** are defined by (n, t, f, g) corresponding to flight f reaching node n at time t during scenario g. A Type-1 node (n, t, g, f) exists if at least one of the Rules 6, 7, or 8 hold.

  Rule 6 *n is the origin of f* **and** *t is the earliest pushback period for f* **and** $t_g^{start} \leq t \leq t_g^{end}$.

  Rule 7 *n is the destination of f* **and** *t is between the earliest and latest arrival times of f* **and** $t_g^{start} \leq t \leq t_g^{end}$.

  Rule 8 *n is not an origin or destination* **and** *t is between the earliest pushback and latest arrival time of f* **and** $t_g^{start} \leq t \leq t_g^{end}$.

  Rules 6, 7, and 8 are analogous to Rules 1, 2, and 3 in the deterministic case respectively. Collectively, the three rules restrict the times at which node-time-scenario combinations can exist.

- **Type-2 nodes** are defined by (a, v, f, g) corresponding to flight f that is on arc a, having spent time v on the arc at the end of scenario g. These nodes (shown shaded in Figure 9) capture time spent that an aircraft has spent on an arc at the time of a scenario transition. A Type-2 node (a, v, g, f) exists if the following Rule 9 holds.

  Rule 9 *The head of arc a has a Type-1 node in scenario g* **and** *v is between 0 and $t_{a,f}^{\max}$, the maximum transit time of flight f on arc a.*

  Rule 9 states that the arc must originate in the state, and that the time spent on the arc must not exceed the transit time on the arc; it effectively limits the locations and time that an aircraft can be in at the time of a scenario transition.

The network has four kinds of arcs that connect each of the two types of nodes to each other.

- **Type-1 arcs** are from a Type-1 node to another Type-1 node. For example, in Figure 9, the Type-1 arc represents a movement of flight f from node n_1 at time t_1 to node n_2 at time t_2 during scenario g_1. Let $a \in \mathcal{A}$ denote the arc in the network from n_1 to n_2. A Type-1 arc from (n_1, t_1, g_1, f_1) to (n_2, t_2, g_2, f_2) exists if one of the Rules 10–11 hold.

  Rule 10 $g_1 = g_2$ **and** $a = (n_1, n_2) \in \mathcal{A}$ *and* $f_1 = f_2$ **and** $a \in \mathcal{A}_F(f)$ **and** $t_{a,f}^{\min} \leq (t_2 - t_1) \leq t_{a,f}^{\max}$.

  Rule 11 $g_1 = g_2$ **and** $a = (n_1, n_2) \in \mathcal{A}$ **and** $f_1 \neq f_2$ **and** f_1 *connects to* f_2 **and** n_1 *is the destination of* f_1 **and** n_2 *is the origin of* f_2 **and** $t_2 - t_1 \geq t_{f_1,f_2}^{turn}$.

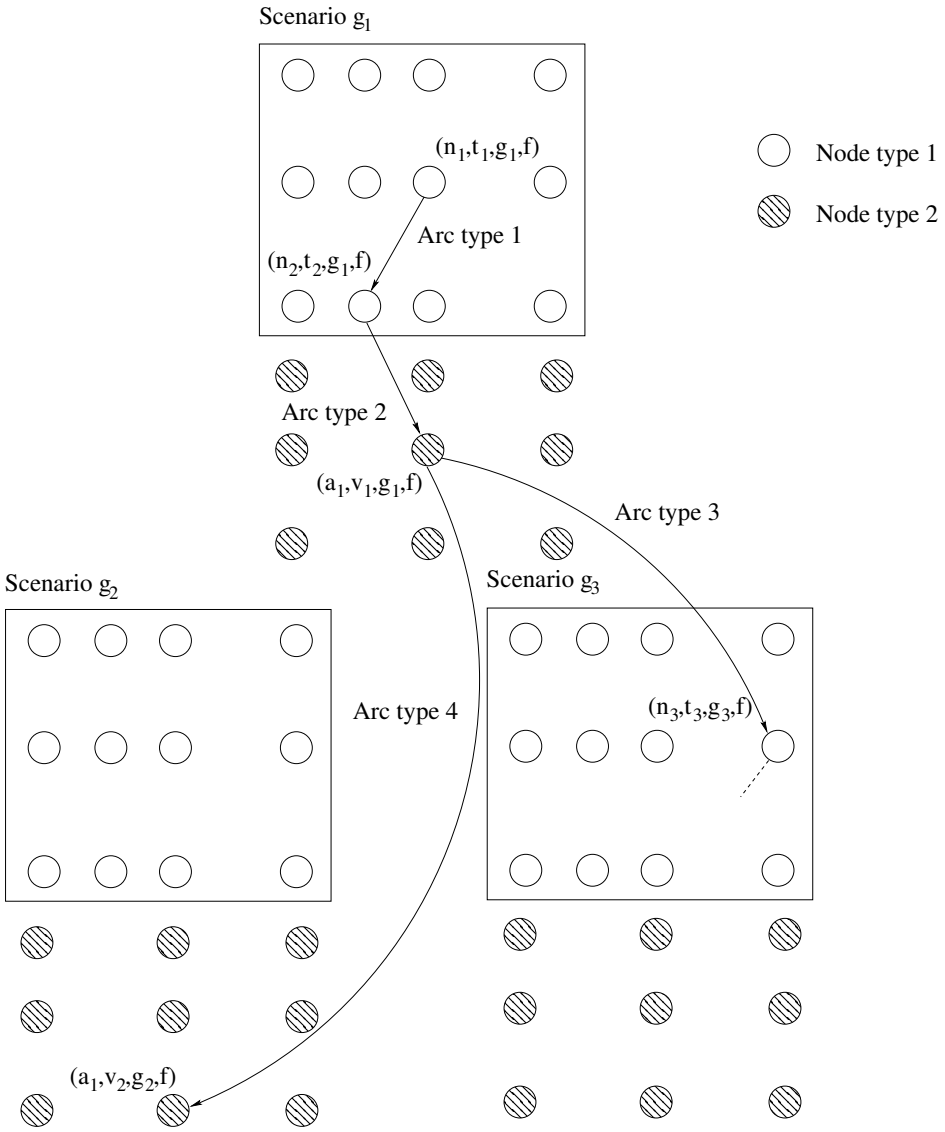


Figure 9: The space-time-scenario network for solving the stochastic subproblem.

Rules 10 and 11 collectively state that for a type-1 arc to exist, both end points must be in the same scenario, the arc must belong to a flight's network and must obey the transit time requirements and, if applicable, flight connectivity requirements.

The weight of a Type-1 Arc for all turnaround arcs is zero. For all other arcs from (n_1, t_1, g, f) to (n_2, t_2, g, f), weights are calculated as:

$$\begin{aligned}\text{Weight} \quad = \quad & p_g\ C_f \text{ (Benefit, applies only to arcs to the destination of flight } f) \\ & - p_g\ \Delta(f, t_2) \text{ (Arrival delay cost, applies only to arcs to the destination of flight } f) \\ & - p_g\ \Omega(a, (t_2 - t_1), f) \text{ (Operating cost)} \\ & - \sum_{j \in \mathcal{J}(n_1,t_1,g)} \sigma_{a,t_1,j,g}\ \mu_{n_1,t_1,j,g} - \sum_{t=t_1}^{t_2-1} \lambda_{s_a,t,g} \text{ (Dual cost)}\end{aligned}$$

Note that all non-dual costs are multiplied by the probability of the scenario, so that their summation will be the expected cost.

- **Type-2 arcs** are from a Type-1 node to a Type-2 node. For example, in Figure 9, the Type-2 arc implies

that flight f arrived at node n_2 at time t_2 during scenario g_1, and left node n_2 along arc a_1; at the end of the scenario, the aircraft had spent v_1 periods on the arc. A Type-2 arc from (n, t, g_1, f_1) to (a, v, g_2, f_2) exists if one of the Rules 12–13 hold.

Rule 12 $g_1 = g_2$ **and** n *is the head of arc* a **and** $f_1 = f_2$ **and** $a \in \mathcal{A}_F(f)$ **and** *the time spent on the arc* $v = t_g^{\text{end}} - t$ **and** $0 \le v \le t_{a,f}^{\max}$.

Rule 13 $g_1 = g_2$ **and** n *is the head of arc* a **and** $f_1 \neq f_2$ **and** $a \in \mathcal{A}_F(f)$ **and** *the time spent on the arc* $v = t_g^{\text{end}} - t$ **and** $0 \le v \le$ *(earliest departure of* f_2 *− earliest arrival of* f_1*).*

Rules 12 and 13 state the relationship between the time that the aircraft entered the arc t and the time spent on the arc at the end of the scenario; they also impose min and max transit time constraints as well as network routing and connectivity constraints on the arcs.

The weight of a Type-2 Arc for all turnaround arcs is zero. For all other arcs from (n, t_1, g, f) to (a, v, g, f), weights are calculated as:

$$\begin{aligned}\text{Weight} \quad = \quad & p_g \ \Omega(a, v, f) \text{ (Operating cost)} \\ & - \sum_{j \in \mathcal{J}(n,t_1,g)} \sigma_{a,t_1,j,g} \ \mu_{n,t_1,j,g} - \sum_{t=t_1}^{t_g^{\text{end}}} \lambda_{s_a,t,g} \text{ (Dual cost)}\end{aligned}$$

Note that these arcs are not associated with any benefit or delay costs since those are assessed only for arcs going into a destination Type-1 node.

- **Type-3 arcs** are from a Type-2 node to a Type-1 node. For example, in Figure 9, the Type-3 arc implies that at the end of scenario g_2, flight f had spent v_1 periods on the arc a_1. The aircraft then arrived at node n_3 at time t_3 during scenario g_3, where g_3 is a child of scenario g_1 in the scenario tree. A Type-3 arc from (a, v, g_1, f_1) to (n, t, g_2, f_2) exists if one of the Rules 14–15 hold.

Rule 14 g_1 *is the parent scenario of* g_2 **and** n *is the tail of arc* a **and** $f_1 = f_2$ **and** $a \in \mathcal{A}_F(f)$ **and** *the total time spent on the arc* $(v + t - t_{g_1}^{\text{end}})$ *is between* $t_{a,f}^{\min}$ *and* $t_{a,f}^{\max}$.

Rule 15 g_1 *is the parent scenario of* g_2 **and** n *is the tail of arc* a **and** $f_1 \neq f_2$ *and* $a \in \mathcal{A}_F(f)$ **and** *the total time spent on the arc* $(v + t - t_{g_1}^{\text{end}})$ *is between* t_{f_1,f_2}^{turn} *and (earliest departure of* f_2 *− earliest arrival of* f_1*).*

Rules 14 and 15 state the relationship between the time spent on an arc at the end of a scenario and the time it reaches the end of the arc in a child scenario, and ensures that this total time is within the min and max transit time of the arc; it also imposes network restrictions and connectivity requirements.

The weight of a Type-3 Arc for all turnaround arcs is zero. For all other arcs from (a, v, g_1, f_1) to (n, t, g_2, f_2), weights are calculated as:

$$\begin{aligned}\text{Weight} \quad = \quad & p_{g_2} \ C_f \text{ (Benefit, applies only to arcs to the destination of flight } f) \\ & - p_{g_2} \ \Delta(f, t_2) \text{ (Arrival delay cost, applies only to arcs to the destination of flight } f) \\ & - p_{g_2} \ (\Omega(a, (v + t_2 - t_{g_1}^{\text{end}}), f) - \Omega(a, v, f)) \text{ (Incremental operating cost)} \\ & - \sum_{t=t_{g_1}^{\text{end}}+1}^{t_2-1} \lambda_{s_a,t,g_2} \text{ (Dual cost)}\end{aligned}$$

Note that there are no node capacity dual costs associated with these arcs since those are assessed only if the arc is from a Type-1 node.

- **Type-4 arcs** are from a Type-2 node to another Type-2 node. For example, in Figure 9, the Type-4 arc implies flight f was on arc a_1 at the end of scenario g_1 having spent v_1 periods on the arc; it then continued on the same arc for the duration of the scenario g_2, and at the end of scenario g_2 had spent v_2 time periods on it. (Thus, $(v_2 - v_1)$ is the duration of scenario g_2. A Type-4 arc from (a_1, v_1, g_1, f_1) to (a_2, v_2, g_2, f_2) exists if the following Rule 16 holds:

Rule 16 g_1 *is the parent scenario of* g_2 **and** $f_1 = f_2$ **and** $a_1 = a_2$ **and** $v_2 = (v_1 + t_{g_2}^{\text{end}} - t_{g_1}^{\text{end}})$.

Rule 16 states the relationship between the time spent on an arc at the end of a scenario and the time spend by the end of a child scenario if the aircraft continues along that arc; it also imposes min and max transit times on these arcs and accounts for network routing restrictions and flight connectivity constraints. The weight of a Type-4 Arc for all turnaround arcs is zero. For all other arcs from (a, v_1, g_1, f) to (a, v_2, g_2, f), weights are calculated as

$$\begin{aligned} \text{Weight} \quad = \quad & -p_{g_2} \left(\Omega(a, v_2, f) - \Omega(a, v_1, f)\right) \text{ (Incremental perating cost)} \\ & - \sum_{t=t^{\text{end}}_{g_1}+1}^{t^{\text{end}}_{g_2}} \lambda_{s_a,t,g_2} \text{ (Dual cost)} \end{aligned}$$

Finally, arcs are added from each Type-1 node that is a flight destination to a sink (that would represent a cancellation of subsequent flights); note that these arcs could be used only for certain scenarios, so the solution could potentially cancel some flights under some scenarios and not others.

Collectively, the four arcs in Figure 9 would represent the following sequence of events. Flight f arrives at node n_1 at time t_1 during scenario g_1. It then proceeds to node n_2 at time t_2 during the same scenario. It then leaves n_2, proceeding along arc a_1. At the end of scenario g_1, the aircraft had spent v_1 time periods on the arc. At that time, if scenario g_2 unfolds, the aircraft stays on the arc until the end of scenario g_2, at the end of which it would have spent v_2 time periods on the arc. If scenario g_3 unfolds, the aircraft proceeds to node n_3, reaching it at time t_3.

The rationale for creating such a network representation is that the Type-2 nodes represent the the points at which new scenarios materialize, and therefore represent branching locations in the trajectory tree while the Type-1 nodes represent deterministic decisions that take place within a scenario. Define a *space-time-scenario forest* (henceforth referred to as an *STS-forest*) as a forest (a collection of disjoint trees) in the space-time-scenario network with the following properties.

Property 5.1 *Every Type-1 node that is the origin of the first flight in the tail is a root of a tree in the STS-forest (i.e., it does not have a parent node).*

This property ensures that a decision is made for the origin node of a tail for all scenarios that contain the earliest departure time of the first flight of the tail. We allow a tree to contain just the root node, in which case the interpretation for the singleton root is that all flights in the tail are cancelled in the scenario containing this root.

Property 5.2 *Every non-root Type-1 node has a parent if and only if it has a child, except for nodes that correspond to the destination of a flight, which can have a parent without having a child.*

This property ensures continuity of the flight if the node is not a destination; if a flight arrives at a node, it must depart from it. If the node represents the destination of a flight but does not have children, the interpretation is that all subsequent connecting flights are cancelled in the scenario that contains the node.

Property 5.3 *Every Type-2 node has children if and only if it has a parent.*

This property ensures that Type-2 nodes cannot be the root or the leaves of a tree; if they belong to a tree, then they must have a parent as well as children.

Property 5.4 *Each Type-2 node in scenario g that has a parent must have a child in every scenario that is a child of g in the scenario tree.*

This property ensures that there is a decision that is made for every possible realization of scenarios that occur at the end of g, and that a flight never gets "stranded" at the end of a scenario without guidance on where it should proceed next.

Define the weight of a forest to be the sum of arc weights of all the arcs in the forest *minus the dual π'_ℓ for tail ℓ*. The maximum weight STS-forest is defined as an STS-forest with the largest weight of all STS-forests.

Define the weight of a subtree (a connected subset of a tree) to be the sum of arc weights in the tree.

Lemma 5.1 *An STS-forest in the space-time-scenario network exists if and only if it represents a trajectory tree that satisfies the operating constraints including minimum and maximum transit times, network connectivity, flight connectivity, turnaround time requirements, and the earliest pushback and latest arrival time constraints of each flight.*

Proof Proof. The proof follows from construction of the network.

To prove the "forward" direction of the statement, i.e., that an STS-forest in the network is a feasible 4D-trajectory tree, we observe that Rules 6–8 above impose earliest departure and latest arrival times on the path, and also accounts for mapping time periods to scenarios. The arc rules require that all arcs belong to $A_F(f)$, and that the origin of a flight must be preceded by the destination of the previous flight. Thus, all network connectivity and flight connectivity constraints are accounted for in the network. We only need to show that a trajectory tree satisfies minimum and maximum transit times. If both end points of an arc are Type-1 nodes, then the transit times are valid by construction (per Rule 16). We only need to show that the transit time constraints between two nodes are satisfied even when the path passes through a Type-2 node. Consider a sequence starting at a Type-1 node, passing through one or more Type-2 nodes and ending at a Type-1 node, represented by $(n_1, t_1, g_1, f), (a, v_1, g_1, f), (a, v_2, g_2, f), ..., (n_2, t_2, g_{k+1}, f)$. We wish to show that $t^{\min}_{a,f} \leq (t_2 - t_1) \leq t^{\max}_{a,f}$. Considering the last and penultimate scenarios, the time spent on the arc is given by the time spent on the arc until the penultimate scenario v_k plus the additional time spent in the last scenario $(t_2 - t^{\text{end}}_{g_k})$. Let $\theta = v_k + (t_2 - t^{\text{end}}_{g_k})$ represent the time spent on the arc. Then, by Rule 14, $t^{\min}_{a,f} \leq \theta \leq t^{\max}_{a,f}$. Also, θ can be written as follows:

$$
\begin{aligned}
\theta &= v_k + t_2 - t^{\text{end}}_{g_k} \\
&= v_{k-1} + t^{\text{end}}_{g_k} - t^{\text{end}}_{g_{k-1}} + t_2 - t^{\text{end}}_{g_k} && \text{(Expanding } v_k \text{ using Rule 14)} \\
&= v_{k-2} + t^{\text{end}}_{g_{k-1}} - t^{\text{end}}_{g_{k-2}} + t^{\text{end}}_{g_k} - t^{\text{end}}_{g_{k-1}} + t_2 - t^{\text{end}}_{g_k} && \text{(Expanding } v_{k-1} \text{ using Rule 14)} \\
&= ... && \text{(Iteratively expanding } v \text{ and canceling terms)} \\
&= v_1 - t^{\text{end}}_{g_1} + t_2 \\
&= -t_1 + t_2 && \text{(By Rule 12)}
\end{aligned}
$$

Thus, we have shown that $\theta = (t_2 - t_1)$ and $t^{\min}_{a,f} \leq \theta \leq t^{\max}_{a,f}$. Therefore, the construction of the STS-network preserves minimum and maximum transit times.

The proof of the "reverse" direction, i.e., that any feasible 4D-trajectory tree is an STS-forest in the network, is by contradiction. Suppose there exists a 4D-trajectory tree that does not have an equivalent STS-forest in the network. Then, there exists a node-time-scenario combination that does not exist in the network or that there is no arc in the network between two nodes in the 4D-trajectory tree. However, no such node or arc can exist since the construction of the network exhaustively examines all node-time-scenario combinations to generate the nodes, and then examines all possible combinations of nodes to create arcs between them. The set of nodes and arcs in the STS-network is actually a superset of nodes and arcs that are feasible to a trajectory tree. ■

Lemma 5.2 *The weight of an STS-forest in the space-time-scenario network minus the tail's dual π'_ℓ equals the reduced cost of the corresponding 4D-trajectory tree.*

Proof Proof. The reduced cost of a 4D-trajectory corresponding is given by

$$
\begin{aligned}
\text{Reduced cost} \quad = \quad & \text{Sum of benefits across all flights in the tail} \\
& - \text{Arrival delay cost} \\
& - \text{Operating cost} \\
& - \text{Duals of constraints (15)} \\
& - \text{Duals multiplied by appropriate coefficient of constraints (16)} \\
& - \text{Dual of constraint (14)}
\end{aligned}
$$

The STS-path correctly captures the expected benefits and arrival delay cost since the expected benefit $p_g\ C_f$ and cost $p_g\ \Delta(f,t)$ of each flight f and scenario g is accounted for in the arcs entering the destination of the flight of each scenario, so any path passing through the destination of a flight picks up the benefit and delay cost associated with the flight. The operating costs are accounted for by associating the appropriate $p_g\ \Omega(.)$ with each arc, and summing along the arcs of the forest gives the total expected operating cost of the trajectory tree.

The cost of each arc also includes the sum of sector duals λ associated with the particular sector-time-scenario combination (since the arc-time-scenario combination maps to a sector-time-scenario combination for all four types of arcs). The node capacity duals are associated with each Type-1 node in the path (and are accounted for in each Type-1 and 2 arcs) through the product of σ and μ for each segment of the capacity envelope in each scenario.

The weight of the ST-path contains the benefit, delay cost, and dual costs associated with the capacity constraints. The reduced cost of the 4D-trajectory tree can thus be obtained by subtracting π'_ℓ, the duals corresponding to the tail constraints (14), from the weight of the ST-path. ■

Corollary 5.1 *A trajectory with positive reduced cost exists if and only if there is an STS-forest of weight strictly greater than π'_ℓ in the space-time network.*

5.2.2 Dynamic programming algorithm:

We now describe a dynamic programming algorithm to compute a maximum weight STS-forest. Let $J_1(n,t,g,f)$ be the weight of a subtree rooted at a Type-1 node (n,t,g,f), and $J_2(a,v,g,f)$ be the weight of a subtree rooted at a Type-2 node (a,v,g,f). All nodes in the subtree other than the root satisfy Properties 5.1–5.4 since it is a subtree of a valid STS-forest; the root satisfies the properties only if it is an origin of the tail. The weight of a subtree rooted at a node is also referred to as the *label* of that node.

In order to simplify the description of the DP, we introduce the following notation. We refer to any space-time-scenario node simply as a node; this could be either a Type-1 or Type-2 node.

$G^+(u)$	The set of child scenarios of the scenario containing node u.
$V_1^+(u,g)$	The set of Type-1 nodes adjacent to node u in scenario g, i.e., there is an arc from u to each node $v \in V_1^+(u,g)$. If the scenario is not explicitly specified, $V_1^+(u,.)$ represents the set of Type-1 nodes within the same scenario as u respectively.
$V_2^+(u,g)$	The set of Type-2 nodes adjacent to node u in scenario g, i.e., there is an arc from u to each node $v \in V_2^+(u,g)$. If the scenario is not explicitly specified, $V_2^+(u,.)$ represents the set of Type-2 nodes within the same scenario as u respectively.
$\omega_{u,v}$	The weight of the arc form u to v.
FDEST(ℓ)	The set of STS-nodes that correspond to the destination nodes of flights belonging to tail ℓ.
TDEST(ℓ)	The set of STS-nodes that correspond to the destination nodes of the last flight of tail ℓ.

The dynamic program performs the following recursions:

$$J_2^*(u) = \sum_{g \in G^+(u)} \max \left\{ \max_{v \in V_1^+(u,g)} \{J_1^*(v) + \omega_{u,v}\}, \max_{v \in V_2^+(u,g)} \{J_2^*(v) + \omega_{u,v}\} \right\} \tag{18}$$

Equation (18) calculates the maximum weight subtree rooted at a Type-2 node by finding the best adjacent node in each of the scenario's children, and summing across these nodes.

$$J_1^*(u) = \max \left\{ \max_{v \in V_1^+(u,.)} \{J_1^*(v) + \omega_{u,v}\}, \max_{v \in V_2^+(u,.)} \{J_2^*(v) + \omega_{u,v}\} \right\} \qquad \forall u \notin \text{FDEST}(\ell) \tag{19}$$

Equation (19) states that the maximum-weight subtree rooted at a Type-1 node u is the maximum over all outgoing adjacent nodes v of the weight of arc (u,v) plus the maximum weight subtree rooted at node v.

$$J_1^*(u) = \max \left\{ 0, \max_{v \in V_1^+(u,.)} \{J_1^*(v) + \omega_{u,v}\}, \max_{v \in V_2^+(u,.)} \{J_2^*(v) + \omega_{u,v}\} \right\} \qquad \forall u \in \text{FDEST}(f) \tag{20}$$

Equation (20) is similar to the previous equation, except that it allows for the label of a destination of a flight to be zero; the interpretation of this is that the destination does not have any positive weight subtrees below it, which implies that all subsequent flights below that node are cancelled.

$$J_1^*(u) = 0 \qquad \forall u \in \text{TDEST}(\ell) \tag{21}$$
$$J_1^*(u) = -\infty \qquad \forall u \notin \text{TDEST}(\ell) \tag{22}$$

Equations (21) and (22) impose the boundary condition is that the weight of the subtree rooted at the destination of the tail is zero (for all scenarios where this node exists), and the initial weights of all other nodes is a large negative number.

The pseudocode for the dynamic programming algorithm is illustrated in Figure 10.

```
    Input: A space-time-scenario network
    Output: A maximum-weight STS-forest
0   begin
1       Initialize labels of all nodes based on Equations (21) and (22)
2       Set status of all scenarios in scenario tree to 0.
3       while status of root of scenario tree is 0 do
4           Get scenario farthest from the root with status 0
5           Compute labels of the Type-2 nodes in the scenario using Equation (18)
6           for each time period t from scenario end to scenario start do
7               Compute labels of Type-1 nodes in time period t of the scenario using Eqn. (19) and (20)
8           end for
9           Set status of scenario to 1
10      end while
11      Return the weight of forest as sum of labels of the tail's origin nodes.
12  end
```

Figure 10: Dynamic programming procedure to calculate the maximum weight STS-forest.

Theorem 5.1 *The algorithm described in Figure 10 and Equations* (18)–(22) *correctly compute the weight of the maximum weight STS-forest.*

Proof Proof. First, we observe that the boundary conditions in Equations (21) and (22) are correct since the subtree below the destinations are empty, and therefore have zero weight. The correctness of recursion in Equation (19) is proved by the properties of longest paths on directed acyclic graphs, and is not described here (see Sedgewick and Wayne (2011) details). Equation 20 follows from the correctness of longest paths in acyclic networks and the option available at destinations nodes to allow cancellations (i.e., have an empty subtree rooted at the destination with zero weight).

We now prove the correctness of Equation 18 by contradiction. Suppose the labels of $J^*(v)$ are correct for all v until a particular iteration and there exists some u such that

$$J_2^*(u) > \sum_{g \in G^+(u)} \max \left\{ \max_{v \in V_1^+(u,g)} \{J_1^*(v) + \omega_{u,v}\}, \max_{v \in V_2^+(u,g)} \{J_2^*(v) + \omega_{u,v}\} \right\} \tag{23}$$

Given that the weight of the optimal subtree rooted at u has value $J_2^*(u)$, the values of the labels of the children of u in the tree can be calculated as $J_2^*(u) - w_{u,v}$ for all children v in the optimal subtree. This implies that there exists some v such that $J_2^*(u) - w_{u,v} > J_1^*(v)$ or $J_2^*(u) - w_{u,v} > J_2^*(v)$ for Equation (23) to be valid. However, this violates our assumption of the optimality of all labels of v.

Given that the recursions work assuming all iterations up until that point have computed the correct labels, and that the boundary conditions are accurate, we conclude that the recursions accurately calculate the labels of all nodes. ■

5.3 Bounds

Apart from the difference in the subproblem, the rest of the procedure is identical to the deterministic case. The bounds derived in Section 4.2.3 still hold since we did not make any specific assumptions about the deterministic

case. The column generation process terminates when the LP gap is within 0.01%, the IP gap is within 0.1%, or the time-limit is reached, whichever happens first.

The two-stage heuristic described in Section 4.2.4 and the bounds are applicable to the stochastic case since no assumption was made in the heuristic that was specific to the deterministic formulation.

6 Computational experiments

In this section, we demonstrate the algorithms described so far on a set of examples drawn from operational data sets. We also illustrate the impact of different levels of time-discretization, as well as the number of parallel threads used in the computation.

6.1 Description of data

Flight schedule data from Bureau of Transportation Statistics (2014) was used in the computational experiments. The 10 days with the most traffic in July 2013 were selected. Table 2 presents a brief description of the data.

	Date	**# BTS flts.**	**# OD pairs**	**# Advisories**	**# Impacted apts.**
1.	7/12/13	19,391	3,750	18	11
2.	7/19/13	19,381	3,748	39	14
3.	7/26/13	19,375	3,744	12	7
4.	7/08/13	19,361	3,750	24	11
5.	7/15/13	19,354	3,745	10	5
6.	7/22/13	19,352	3,745	18	6
7.	7/11/13	19,349	3,749	21	9
8.	7/29/13	19,341	3,742	12	3
9.	7/18/13	19,329	3,743	13	10
10.	7/25/13	19,313	3,741	15	6

Table 2: Description of test case days, showing the numbers of flights in the BTS and optimization data, distinct tail numbers, OD pairs, GDP and GS issue and revision advisories, and airports impacted by these TMIs.

6.1.1 Network model:

The model considered consists of 370 airports, as represented in Bureau of Transportation Statistics (2014). The airspace is divided into 375 sectors using a 15×25 grid, according to latitude and longitude (Figure 11). Each sector has four nodes at which aircraft aircraft can enter and exit the aircraft (denoted by E, F, G, and H in Figure 11, right). Nodes are connected to each other as shown by bi-directional links. Each airport is connected by bi-directional links to each of the four nodes of the sector that contains the airport.

The nominal routes corresponded to the shortest paths between the origin and destination airports. The rerouting network was generated so as to include all paths within an ellipse, with the origin and destination airports as the foci. The eccentricity of the ellipse was varied so as to be larger for short-haul flights, and smaller for the trans-continental flights. For example, the rerouting network for a SFO-DCA flight had 463 nodes and $2,582$ arcs, while that of a DCA-BOS flight had 40 nodes and 174 arcs.

6.1.2 Flight schedules:

The total number of scheduled flights on each of these days was between 19,300 and 19,400. Flights that had a latest time of arrival past 0900 hours (Zulu) the next day are not considered in the optimization. Flights with missing tail numbers are assumed to not connect, and are assigned unique tail numbers. The optimization problems therefore consider approximately 17,500 flights per day, with approximately 4,200 unique tails.

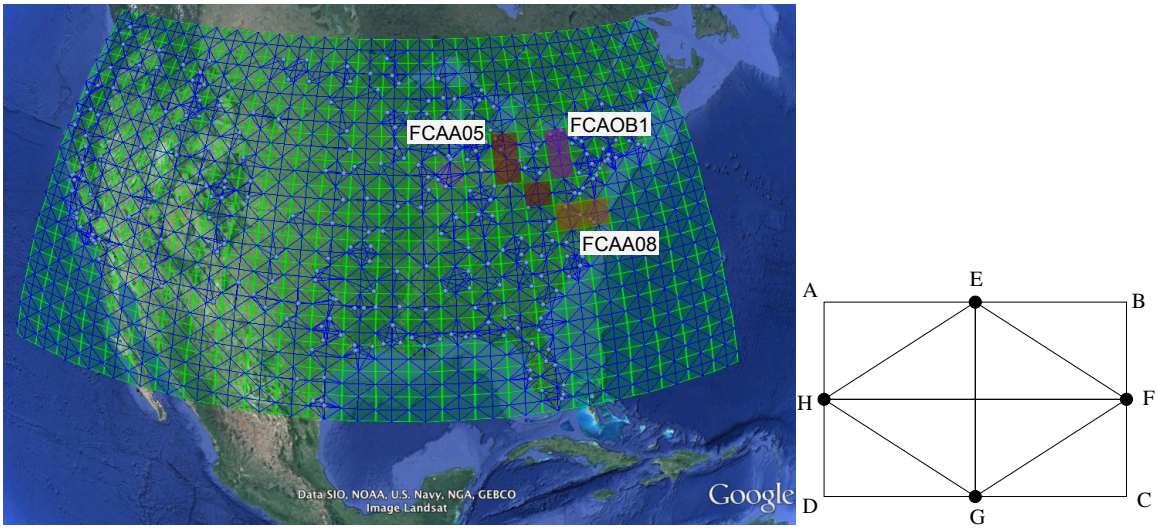


Figure 11: Airspace network model, showing the airspace sector boundaries (green), links (blue), and airports (light blue shaded circles). The shaded sectors denote candidate Flow Control Areas (FCAs) where capacity is reduced. The figure at the right shows the structure of each sector.

6.1.3 Airport and sector capacities:

The airport capacities under Visual Meteorological Conditions (VMC), Instrument Meteorological Conditions (IMC) and Marginal Conditions for the top 35 airports (excluding Honolulu, HNL) were obtained from Federal Aviation Administration (2004). Other airports were constrained only in the event of a Ground Stop or GDP that impacted that airport based on information from TMIs (Traffic Management Advisories). Data on Traffic Management Initiatives (TMIs) such as Ground Delay Programs (GDP) and Ground Stops (GS) on these days was extracted from the FAA's Advisory Database (Federal Aviation Administration 2014). For example, Figure 12 shows the Airport Arrival Rate (AAR) at Atlanta (ATL) on an example day, with Ground Stops and Ground Delay Programs highlighted.

The nominal capacity of each sector was determined by solving the optimization problem with no sector capacity constraints and VMC capacities at all airports; the capacity was set to higher value of 30 aircraft and the maximum occupancy seen in the optimal solution. The average sector capacity was 31 aircraft, while the maximum was 65 aircraft. Our airspace sector capacities are larger than the en-route sector capacities seen in the actual system for two reasons: The sectors are larger than typical en-route sectors and include all flight levels (while the en-route sectors are divided into low, high and super-high sectors), and they include congested terminal-areas as well (which are typically subdivided into as many as 8 arrival and departure sectors). Reduced sector capacity constraints were imposed during Airspace Flow Programs (AFPs) in the corresponding Flow Control Areas or FCAs (Federal Aviation Administration 2014) by scaling down the capacities in the associated sectors according to the advisories issued. The maximum departure delay allowed for a flight was assumed to be 4 hours (Federal Aviation Administration 2012). All data used in the experiments were drawn from publicly available sources.

6.1.4 Objective function:

The proposed approach can accommodate a wide range of objective functions that vary by unit cost of delay, by flight, and by airspace sectors or airports. The objective function used in our experiments was modeled as follows.

Revenue: The revenue for each flight is modeled as a constant plus a linear term in the transit time.

Delay cost: The delay (i.e., difference between the arrival time and the earliest arrival time) cost is modeled as a super-linear function of the delay. This form of the delay cost function ensures a more equitable distribution

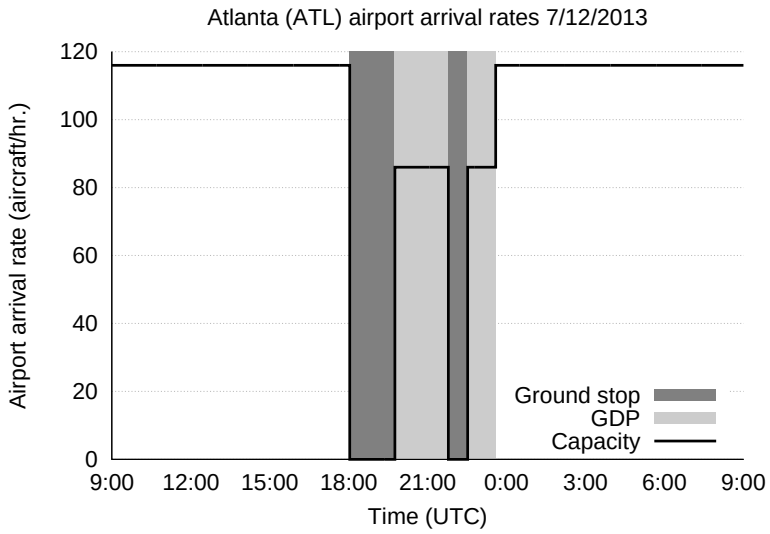


Figure 12: Arrival capacity profiles for an airport (ATL).

of delay, since it favors assigning moderate amounts of delay to two different flights over assigning a large amount of delay to one, and a small amount to the other Vossen and Ball (2006).

Operating cost: En-route, taxi, and gate-hold operating costs were modeled as a linear in the time with relative weights of 1.0, 0.5, and 0.1 respectively.

The weights of the various parameters was set such that the net revenue (revenue minus costs) for a flight associated with a delay of 4 hours was approximately zero (i.e., all else being equal, the optimization model is indifferent to delay or cancellation if the delay is around 4 hours).

6.1.5 Stochastic instances:

For a given number of scenarios, we first generated a random tree in which each node other than a leaf has two children, with the required number of total nodes. The length of each scenario was uniformly distributed from 1 period to the length of the horizon, and then scaled so that the longest path in the tree equalled the horizon length. The end times of all leaf states were set to equal the end of the time horizon. The capacity profile of each scenario was scaled from the deterministic capacity by a random factor that was uniformly distributed in $[0.8, 1.2]$.

While it is unlikely that a realistic scenario will require developing 25-scenario recourse strategies for all flights in the NAS (since uncertainty is typically more localized geographically and temporally), we examine the extreme case to test the limits of our algorithm.

6.2 Experimental setup

The experiments were run on a Mac laptop with a 2.3 GHz Intel Core i7 processor with 4 physical cores and 8 virtual cores, and 16GB of RAM. The code was written in C, and interfaced with CPLEX 12.5 via the Callable Library.

6.2.1 CPLEX settings:

Since the number of rows in our models typically exceed the number of columns, we employed the primal simplex method, which was found to perform the best. We also turned on the "PreDual" parameter in CPLEX, which first converts the problem to its dual before solving it (note that this is different from applying the dual simplex method). This is recommended by CPLEX for problems where the number of rows exceeds the number of columns.

When solving the sequence of LPs, CPLEX by default uses the optimal basis of the last run as the starting basis for a new run. For the stochastic problems, CPLEX was additionally directed to perform a presolve at the beginning of each run by setting the "advanced start indicator" parameter to 2. For stochastic problems, the "PreDual" setting was found to not make a significant difference, and was set to its default value (in which CPLEX automatically determines whether or not to convert to the dual formulation).

6.2.2 Number of instances tested:

Our deterministic algorithm was run on 10 instances, one corresponding to each day in Table 2.

In our stochastic experiments, we tested instances with 5, 10, 15, 20, and 25 scenarios. For each day and scenario, we generated two different instances with different random seeds, which resulted in different tree structures and capacity profiles. Thus, our stochastic algorithm was run on 100 instances (10 days and 5 values of the size of the tree, each with two different capacity profiles).

6.2.3 Termination conditions:

The column generation process was terminated when the LP gap was within 0.01% or the IP gap was within 0.1% or the time limit was reached, whichever occurred first. The time limit for all problems was set to 30 minutes, although this limit was not relevant since all instances were solved within the time limit.

6.2.4 Times reported:

The times reported are wall/clock times (and not CPU times). The reason for reporting wall times is to be able to compare the impact of parallelization on the process. The process of parallelization requires dynamically allocating memory stacks, etc. that may not be captured under CPU user time, and therefore CPU time may underestimate the work done by the process. Since the machine used was dedicated to solving these problems, 100% of the memory and CPU resources were available to the algorithms (therefore, if only one thread were used, the CPU and wall times should be equivalent).

The following times are reported:

Subproblem time is the wall time spent in solving the subproblems. This includes operations to extract the dual information from the LP, solve the shortest paths/dynamic programming and create the necessary data structures to add variables to the LP.

LP time is the time spent solving the linear programs.

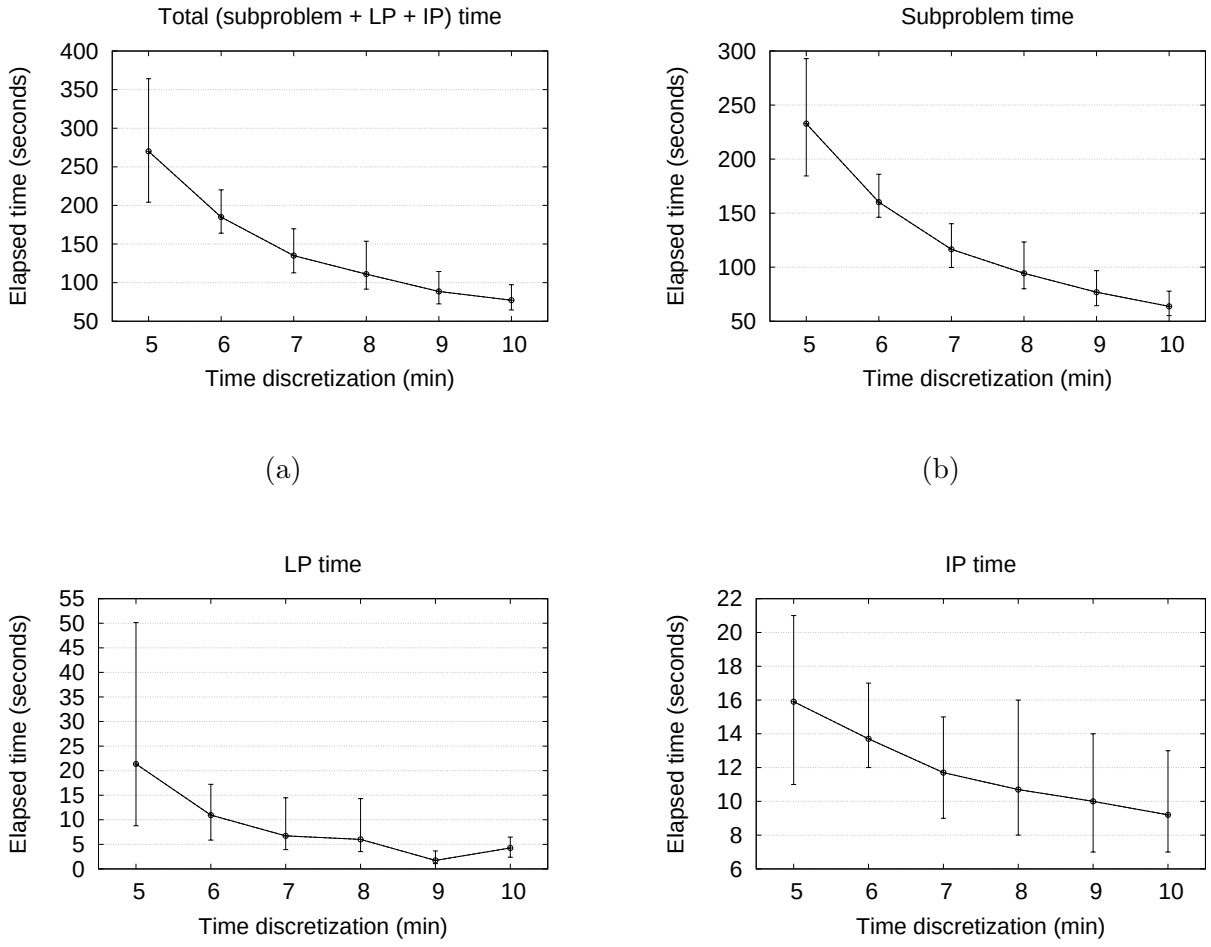
IP time is the time spent solving the IP.

Times associated with running the heuristic are not reported here since they are insignificant compared to the other times. In each case, the time limit for the heuristic was limited to 10% of the time of the LP solution time; in all our instances tested, the total heuristic time was under 5 seconds.

6.3 Deterministic TFMP results

6.3.1 Effect of time-discretization on run times:

Time discretization has a significant impact on run times. As discussed in earlier sections, having too small a discretization value is not only computationally more difficult, but may also be unnecessary given uncertainty in a flight's ability to conform precisely to a 4D trajectory. Having too large a discretization value results in loss of fidelity and efficiency if the discretization value is comparable to the sector transit time. We tested time discretizations of 5 to 10 minutes for all 10 days, the results if which are shown in Figure 13.



(a) (b)

(c) (d)

Figure 13: Run times as a function of discretization. Each figure shows the average, min, and max run times across the 10 days for each value of discretization. (a) Shows the total run time (LP + IP + sub-problem computation, (b) Shows subproblem run times, (c) Shows LP run times, and (d) Shows IP run time.

The results show that, as expected, the average and variance of run times increase as the length of the time period decreases. Interestingly, going from a discretization of 5 minutes to 6 minutes results in the average run time decreasing from 260 seconds to 180 seconds. While we advocate using a 5-minute discretization, the results show that if a faster solution is required, a modest decrease in discretization can drive significant reductions in run time.

6.3.2 Effect of parallelization on run times:

One attractive feature of our algorithm is that solving the subproblems can be easily parallelized. In this section, we attempt to quantify this benefit by examining the run times for different levels of parallelization. We solved all 10 deterministic instances with 1, 2, 4, 6, and 8 parallel threads (the machine we ran our experiments is capable of running up to 8 threads in parallel); the results are presented in Figure 14.

We see that increasing the number of threads from 1 to 4 decreases the total run time by about a factor of 3. However, beyond that, going to 6 processors only results in some marginal benefit, and there is almost no

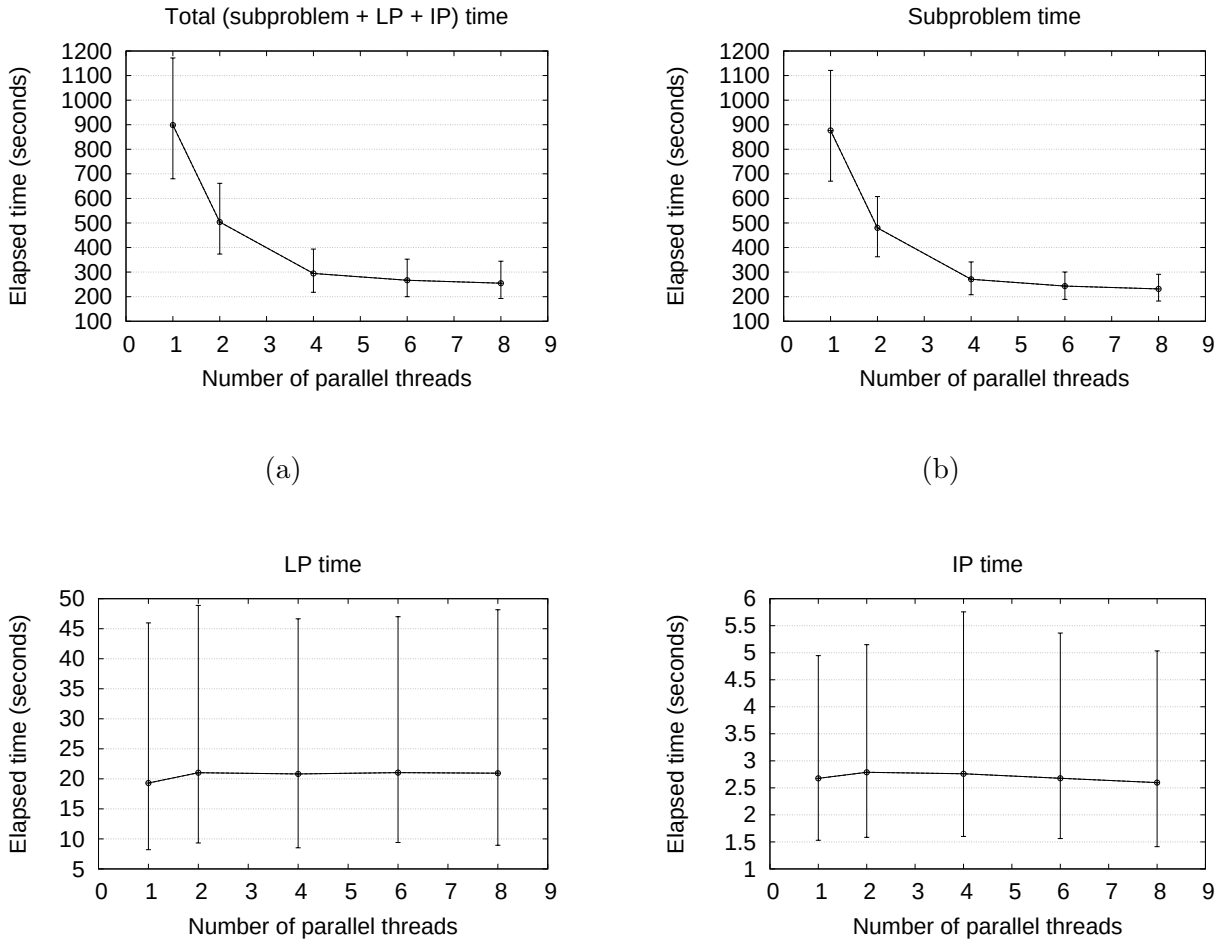


Figure 14: Run times as a function of number of parallel threads. Each figure shows the average, min, and max run times across the 10 days for different values of the number of threads. (a) Shows the total run time (LP + IP + sub-problem computation, (b) Shows subproblem run times, (c) Shows LP run times, and (d) Shows IP run time.

benefit from moving from 6 to 8 processors. This behavior is because of the overhead required in creating and managing the parallel threads.

The conclusion we draw from these experiments is that the sub-problem times initially scale almost linearly with the number of cores, but the benefit drops sharply as the number of threads approach the machine capacity. Since our experiments were run on a laptop that is not designed for high-performance parallelism, we believe that running our algorithms on workstations or cloud instances specifically designed for multi-core applications will yield significantly better run times.

6.3.3 Detailed results with 5-minute time-discretization:

We now describe in detail the results on running the algorithms on the 10 deterministic instances for a 5-minute discretization and 8 threads. Table 3 summarizes the results, and contains the following columns:

Subproblem iterations is the number of batches of subproblems solved before the column generation process

terminates (equals the number of master problems solved to generate dual values).

Subproblem and LP time are the clock/elapsed time to solve the subproblem and LP respectively, as described in Section 6.2.4.

LP and IP gap are the LP and IP gap at termination, as described in Equations (10) and (11) respectively.

Number of variables, constraints, and non-zeros shows the number of variables, constraints, and non-zeros in the final RMP.

Number of flights and tails represents the number of flights and tail numbers in the problem.

Ground hold and air delay is the total ground holding and total air delay across all flights. The ground delay of a flight is calculated as the actual takeoff time minus the earliest takeoff time with no capacity constraints. The total flight delay is the actual arrival time minus the earliest arrival time of a flight in the absence of any capacity constraints. The air delay is the total delay minus the ground delay. The delay minutes and cancellation rate are presented here only to show the relative "difficulty" of solving each day, and should not be used to make a comparison to actual delays observed on those days.

Number of cancelled and rerouted flights represent the number of flights that are cancelled and rerouted respectively. A flight is said to be rerouted if the set of arcs in its optimal trajectory is different from the flight's shortest path.

Date	Sub-prob. iterations	Sub-prob., LP times (seconds)	LP gap IP gap (%)	# Cons./# Vars. # Non-zeros	# Flights # Tails	Ground hold Air delay (minutes)	# Cancelled # Rerouted
7/08/2013	22	292.97	0.008	131,856 / 19,287	17,603	33,060	381
		50.14	0.008	1,970,546	4,272	8,245	657
7/11/2013	17	238.42	0.007	132,177 / 19,529	17,592	28,140	704
		16.28	0.094	1,923,413	4,453	7,655	492
7/12/2013	16	217.69	0.002	131,865 / 18,111	17,607	25,415	278
		22.30	0.002	1,765,003	4,275	9,885	517
7/15/2013	16	204.30	0.011	131,743 / 14,815	17,598	16,670	46
		22.17	0.011	1,535,680	4,159	3,520	377
7/18/2013	16	235.47	0.004	131,735 / 13,934	17,571	12,770	32
		17.97	0.084	1,435,384	4,151	3,195	346
7/19/2013	12	184.39	0.008	131,903 / 14,225	17,599	23,825	815
		8.79	0.008	1,417,841	4,168	7,620	470
7/22/2013	18	242.73	0.005	132,206 / 17,364	17,595	20,975	287
		18.97	0.069	1,755,866	4,310	4,525	456
7/25/2013	19	250.20	0.008	131,851 / 15,247	17,556	17,535	355
		11.52	0.008	1,585,386	4,123	55,35	422
7/26/2013	17	240.38	0.001	131,713 / 18,031	17,593	23,245	183
		23.88	0.001	1,880,184	4,092	5,950	513
7/29/2013	16	221.39	0.002	131,769 / 14,496	17,586	18,960	649
		21.65	0.002	1,495,767	4,185	5,875	461

Table 3: Summary of results for the deterministic ATFM problem with 8 threads and 300 second discretization.

The results show that the run-times of the solution process show very little variability from day to day. In particular, the subproblem time per iteration (total sub-problem time divided by number of iterations) is remarkably consistent at ∼14–16 seconds per iteration. This is a very desirable quality in any solution process since it appears that the run-times are somewhat independent of the severity of the capacity constraints or of the total delay or cancellation rate, making the performance of the algorithm predictable. The fact that the LPs typically have significantly greater rows than columns and the non-zero density is about 100 non-zeros per column explains why the CPLEX parameters described in Section 6.2.1 work well for the problem.

As mentioned earlier, these computational experiments assumed the nominal runway capacity envelopes from Federal Aviation Administration (2004) during periods with no Traffic Management Initiatives with explicit airport arrival rates. Recent work by Pyrgiotis et al. (2013) suggests that these nominal capacity estimates may

be quite optimistic. In order to quantify the operational benefits of Air Traffic Flow Management, future work would incorporate refined estimates of runway capacity, as well as other delay cost functions (Bloem and Huang 2011).

6.4 Stochastic TFMP results

We ran our stochastic TFMP algorithm on 100 instances (10 days with 5 different number of states, each with two random instances), the results of which are shown in Table 4. A time-discretization of 10 min was used since the memory requirements of a 5-min discretization was beyond the capabilities of our machine when the number of scenarios was large. All instances with 5 states were solved to within 4 minutes. An optimal solution was found within 15 min even when there were 25 scenarios in the forecast. We note that such extreme problems in which recourse strategies are required for all flights in the NAS for 20-25 scenarios are unlikely to arise in practice; we present the results for completeness and to demonstrate the scaling properties of our algorithm.

The sub-problem time appears to scale somewhat linearly with the number of scenarios; this bodes well for solving problems with a larger number of scenarios if necessary. The number of sub-problem iterations is approximately the same for all five scenario sizes, implying that the increase in run time is due to the size of each subproblem, as expected.

7 Conclusions

This paper presented a new approach to solving large-scale air traffic management problems, in both deterministic and stochastic settings. The possible control actions considered included ground and airborne delays (through speed changes), rerouting over a large network, and cancellations. Airport and airspace sector capacity constraints were considered, along with flight connectivity constraints. Using nation-scale examples drawn from US domestic flight data, we demonstrated a scalable, parallel implementation of our algorithm on instances with about 17,500 flights/day, 370 airports and 375 airspace sectors. Our results show that the proposed approach is capable of solving deterministic nation-scale examples (for a 1-day time window and a 5-min time-discretization) to optimality in about 5 min. We also illustrated how larger values of time-discretization yield even faster run times with our approach. Stochastic cases of the nation-scale ATFM problem with probabilistic scenario-tree forecasts were solved at a 10-min time-discretization. For cases of up to 25 scenarios, optimal solutions were determined within 20 min.

Our computational experiments, based on the largest instances of the air traffic flow management problem solved to date, show that the proposed approach is fast enough for real-time implementation. The easily parallelizable nature of the approach, in addition to having computational benefits, has the potential to enable distributed, yet collaborative, decision-making among the different airlines.

References

A. Alonso, L. F. Escudero, and M. T. Ortuno. A stochastic 0-1 program based approach for the air traffic flow management problem. *European Journal of Operations Research*, 120:47–62, 2000.

G. Andreatta, P. Dell'Olmo, and G. Lulli. An aggregate stochastic programming model for air traffic flow management. *European Journal of Operations Research*, 215:697–704, 2011.

M.O. Ball, R. Hoffman, A.R. Odoni, and R. Rifkin. A stochastic integer program with dual network structure and its application to the ground-holding problem. *Operations Research*, 51(1):167–171, Jan.-Feb. 2003.

A.M. Bayen, R. Raffard, and C. Tomlin. Adjoint-based control of a new Eulerian network model of air traffic flow. *IEEE Transactions on Control Systems Technology*, 14(5):804–818, 2006.

D. Bertsimas and A. Odoni. A critical survey of optimization models for tactical and strategic aspects of air traffic flow management. Technical Report CR-97-206409, NASA, 1997.

D. Bertsimas and S. Stock Patterson. The air traffic flow management problem with enroute capacities. *Operations Research*, 46(3):406–422, May-June 1998.

D. Bertsimas and S. Stock Patterson. The traffic flow management rerouting problem in air traffic control: A dynamic network flow approach. *Transportation Science*, 34(3):239–255, August 2000.

Date	Metric	Number of scenarios				
		5	**10**	**15**	**20**	**25**
7/08/2013	Sub-problem time (sec.)	214.48	484.30	823.46	1029.25	1000.18
	LP time (sec.)	32.39	13.76	15.75	22.40	11.05
	IP time (sec.)	7.77	5.32	7.79	12.03	4.37
	# subproblem iterations	13	15	17	17	14
7/11/2013	Sub-problem time (sec.)	205.31	453.11	734.83	1036.78	1033.94
	LP time (sec.)	38.67	17.21	16.41	31.87	11.66
	IP time (sec.)	6.74	5.63	8.08	14.53	4.47
	# subproblem iterations	12	13	14	16	14
7/12/2013	Sub-problem time (sec.)	195.88	415.46	673.35	867.46	1106.98
	LP time (sec.)	16.79	8.25	10.22	14.75	11.98
	IP time (sec.)	3.74	3.05	7.71	10.15	5.88
	# subproblem iterations	12	13	14	14	16
7/15/2013	Sub-problem time (sec.)	191.50	441.39	653.58	911.22	1003.72
	LP time (sec.)	42.12	14.13	14.29	21.94	10.76
	IP time (sec.)	8.19	8.88	5.74	16.38	14.13
	# subproblem iterations	11	13	12	14	12
7/18/2013	Sub-problem time (sec.)	197.68	348.34	605.33	774.70	930.34
	LP time (sec.)	33.48	8.76	10.34	13.34	8.35
	IP time (sec.)	4.50	4.12	5.11	7.39	6.38
	# subproblem iterations	11	9	11	11	10
7/19/2013	Sub-problem time (sec.)	185.75	365.50	560.56	762.92	855.70
	LP time (sec.)	32.17	8.78	8.83	11.60	7.34
	IP time (sec.)	4.24	2.69	6.12	7.11	5.16
	# subproblem iterations	10	10	10	10	9
7/22/2013	Sub-problem time (sec.)	208.31	432.20	719.22	1086.23	1129.24
	LP time (sec.)	38.05	10.30	13.22	23.64	13.38
	IP time (sec.)	6.44	3.73	4.01	7.55	2.85
	# subproblem iterations	12	13	15	18	18
7/25/2013	Sub-problem time (sec.)	211.33	358.64	526.02	827.59	798.29
	LP time (sec.)	49.18	10.11	8.86	15.63	6.22
	IP time (sec.)	5.53	5.59	6.67	11.33	2.71
	# subproblem iterations	12	10	9	12	8
7/26/2013	Sub-problem time (sec.)	213.96	446.95	663.86	960.95	1189.10
	LP time (sec.)	77.68	18.20	14.70	22.13	12.80
	IP time (sec.)	10.18	7.46	6.43	9.89	11.71
	# subproblem iterations	12	13	12	14	15
7/29/2013	Sub-problem time (sec.)	191.12	381.30	597.43	754.53	801.45
	LP time (sec.)	44.74	11.50	12.00	14.81	7.15
	IP time (sec.)	9.05	5.38	3.37	7.33	3.29
	# subproblem iterations	11	11	11	10	8
Overall	**Sub-problem time (sec.)**	201.53	412.72	655.76	901.16	984.89
	LP time (sec.)	40.53	12.10	12.46	19.21	10.07
	IP time (sec.)	6.64	5.18	6.10	10.37	6.09
	# subproblem iterations	11	12	12	13	12

Table 4: Summary of results for the stochastic ATFM problem with 8 threads and 600 second discretization.

D. Bertsimas, G. Lulli, and A. Odoni. An integer optimization approach to large-scale air traffic flow management. *Operations Research*, 59(1):211–227, January-February 2011.

M. Bloem and H. Huang. Evaluating delay cost functions with airline actions in airspace flow programs. In *Ninth USA/Europe Air Traffic Management Research and Development Seminar (ATM2011)*, June 2011.

Bureau of Transportation Statistics. Airline On-Time Statistics and Delay Causes, 2014. URL `http://www.transtats.bts.gov/`.

G. Buxi and M. Hansen. Generating probabilistic capacity profiles from weather forecast: A design-of-experiment approach. In *Ninth USA/Europe Air Traffic Management Research and Development Seminar (ATM2011)*, 2009.

S. Chaimatanan. *Planification stratégique des trajectoires d'avion*. PhD thesis, University of Toulouse, 2014.

G. Desaulniers, J. Desrosiers, and M.M. Solomon (eds.). *Column Generation*. Springer, 2005.

Federal Aviation Administration. Airport capacity benchmark report, 2004.

Federal Aviation Administration. The Business Case for the Next Generation Air Transportation System, 2012.

Federal Aviation Administration. Air Traffic Control System Command Center advisories database, 2014. URL `http://www.fly.faa.gov/adv/advADB.jsp`.

E. P. Gilbo. Airport capacity: Representation, estimation, optimization. *IEEE Transactions on Control Systems Technology*, 1(3):144–154, September 1993.

S. Gupta and D. Bertsimas. Multistage air traffic flow management under capacity uncertainty: A robust and adaptive optimization approach. In *51st AGIFORS Annual Symposium and Study Group Meeting*, October 2011.

M. P. Helme. Reducing air traffic delay in a space-time network. In *IEEE International Conference on Systems, Man and Cybernetics*, 1992.

Joint Economic Committee, US Senate. Your Flight has Been Delayed Again: Flight Delays Cost Passengers, Airlines, and the US Economy Billions, 2008.

Joint Planning and Development Office. *Concept of Operations for the Next Generation Air Transportation System*, June 2007.

P.-C. B. Liu, M. Hansen, and A. Mukherjee. Scenario-based air traffic flow management: From theory to practice. *Transportation Research Part B*, 42:685–702, 2008.

G. Lulli and A. Odoni. The European Air Traffic Flow Management Problem. *Transportation Science*, 41(4): 431–443, November 2007.

J. B. Marron. The stochastic air traffic flow management rerouting problem. Master's thesis, Massachusetts Institute of Technology, 2004.

L. Maugis. Mathematical programming for the air traffic management problem with en-route capacities. Technical Report CENA/R95-022, CENA, 1995.

P. K. Menon, G. D. Sweriduk, T. Lam, G. M. Diaz, and K. Bilimoria. Computer-aided Eulerian air traffic flow modeling and predictive control. *AIAA Journal of Guidance, Control and Dynamics*, 29:12–19, 2006.

A. Mukherjee and M. Hansen. A dynamic stochastic model for the single airport ground holding problem. *Transportation Science*, 41(4):444–456, 2007.

Avijit Mukherjee and Mark Hansen. A dynamic rerouting model for air traffic flow management. *Transportation Research Part B: Methodological*, 43(1):159 – 171, 2009.

A. Nilim, L. El Ghaoui, and V. Duong. Robust dynamic routing of aircraft under uncertainty. In *Proceedings of the 21st Digital Avionics Systems Conference*, 2002.

A. Nilim, L. El Ghaoui, and V. Duong. Multi-aircraft routing and traffic flow management under uncertainty. In *Proceedings of the 5th US/Europe Air Traffic Management R&D Seminar*, pages 23–27, Budapest, Hungary, 2003.

A. R. Odoni. The flow management problem in air traffic control. In A. R. Odoni, L. Bianco, and G. Szego, editors, *Flow Control of Congested Networks*, pages 269–288. Springer-Verlag, Berlin, 1987.

D. M. Pfeil and H. Balakrishnan. Identification of robust terminal-area routes in convective weather. *Transportation Science*, 46(1):56–73, February 2012.

Nikolas Pyrgiotis, Kerry M. Malone, and Amedeo Odoni. Modelling delay propagation within an airport network. *Transportation Research Part C: Emerging Technologies*, 27:60–75, 2013.

O. Richard. *Régulation court terme du trafic aérien et optimisation combinatoire: Application de la méthode de génération de colonnes*. PhD thesis, Institut National Polytechnique de Grenoble, 2007.

O. Richard, S. Constans, and R. Fondacci. Computing 4D near-optimal trajectories for dynamic air traffic flow management with column generation and branch-and-price. *Transportation Planning and Technology*, 34 (5):389–411, 2011.

O. Richetta and A. R. Odoni. Dynamic solution to the ground-holding problem in air traffic control. *Transportation Research Part A*, 28:167–185, 1994.

R. Sedgewick and K. Wayne. *Algorithms*. Pearson Education, 2011. ISBN 9780132762564. URL `http://books.google.com/books?id=idUdqdDXqnAC`.

H. D. Sherali, R. W. Staats, and A. A. Trani. An Airspace Planning and Collaborative Decision-Making Model: Part I – Probabilistic Conflicts, Workload, and Equity Considerations. *Transportation Science*, 37(4): 434–456, November 2003.

H. D. Sherali, R. W. Staats, and A. A. Trani. An Airspace-Planning and Collaborative Decision-Making Model: Part II – Cost Model, Data Considerations, and Computations. *Transportation Science*, 40(2):147–164, 2006.

B. Sridhar, T. Soni, K. Sheth, and G.B. Chatterji. Aggregate flow model for air-traffic management. In *Journal of Guidance, Control, and Dynamics*, volume 26, pages 992–997, 2006.

D. Sun and A.M. Bayen. Multicommodity Eulerian-Lagrangian large-capacity cell transmission model for en route traffic. *Journal of Guidance, Control, and Dynamics*, 31(3), 2008.

D. Sun, I. S. Strub, and A. Bayen. Comparison of the performance of four Eulerian network flow models for strategic air traffic management. *Networks and Heterogeneous Media*, 2(4):569–594, December 2007.

D. Sun, A. Clinet, and A. M. Bayen. A dual decomposition method for sector capacity constrained traffic flow optimization. *Transportation Research Part B*, 45:880–902, 2011.

T. W. M. Vossen, R. Hoffman, and A. Mukherjee. Air Traffic Flow Management. In C. Barnhart and B. C. Smith, editors, *Quantitative Problem Solving Methods in the Airline Industry*, volume 169 of *International Series in Operations Research & Management Science*, pages 385–453. Springer Science+Business Media, LLC, 2012.

Thomas Vossen and Michael Ball. Optimization and mediated bartering models for ground delay programs. *Naval Research Logistics*, 53(1):75–90, February 2006.

P. Wei, Y. Cao, and D. Sun. Total unimodularity and decomposition method for large-scale air traffic cell transmission model. *Transportation Research Part B*, 53:1–16, 2013.