Abstract
This paper describes automated algorithms for applying restrictions to air traffic to prevent one or more sectors of airspace from becoming overloaded. We evaluate the merits of both heuristic and classical optimization approaches, and we identify tradeoff issues that affect the selection of a particular algorithm for the airspace problem. Our initial results show that a hybrid algorithm combining both classical and heuristic approaches works best.

Introduction
Within the United States airspace, each sector has limited capacity to accommodate aircraft. Factors such as weather, equipment outages, traffic complexity, and controller workload all affect sector capacity.

The Traffic Management Coordinator (TMC) in the US typically monitors the en route congestion situation on the Traffic Situation Display using flight data from the Enhanced Traffic Management System (ETMS). The Monitor Alert Parameter (MAP) value provides an indication of when capacity might be exceeded [1]. The TMC then investigates potential overloads to assess whether or not the incoming traffic will present too much complexity or volume for the sector controllers. If so, then the TMC applies flow restrictions in the form of reroutes, en route spacing, or suppression of local departures (ground holding or altitude capping).

Observations and discussions with TMCs [2] indicate that existing decision support capabilities are not sufficiently effective for generating and evaluating traffic flow restriction options. Specifically, today’s process is manually intensive, highly cognitive, and relies too heavily on TMC experience. As a result, TMCs unknowingly over-restrict or under-restrict traffic. Additionally, the effect of restrictions on adjacent or surrounding sectors is not easily identifiable. These other sector counts may increase beyond their MAP values as a result of restrictions imposed to protect a sector (i.e., collateral damage).

To formulate a miles-in-trail (MIT) restriction that is needed to protect a sector, the TMCs assess the makeup of the traffic flow that will traverse the sector (e.g., the composition of overflights, internal departures, and arrivals). The TMC must then choose both the stream(s) to restrict and the size of the spacing requirement to produce the most effective results. For example, if the Cleveland Air Route Traffic Control Center’s (ARTCC’s) sector 48 is expected to exceed its sector monitor alert parameter of 14, then the Cleveland TMC may issue a restriction on an eastbound flow from Chicago ARTCC to reduce the sector complexity. In many cases, the flow that has the most aircraft gets the restriction since this is the easiest approach for preventing sector overload with minimal coordination in today’s system.

NASA researchers are developing a capability for the TMCs, called the Airspace Restriction Planner (ARP), designed to aid in the management of sector congestion [2]. The ARP allows TMCs to identify sectors that are going to be over-congested, visualize and detail traffic characteristics, and plan traffic flow management restrictions. The ARP is being developed and evaluated within the framework of NASA’s Future Air Traffic Management Concept Evaluation Tool (FACET) [3]. The combined application is referred to throughout this paper as FACET-ARP.

Human-in-the-loop simulations at Cleveland ARTCC confirmed the potential value of ARP, and also identified the need for greater automation to support congestion management involving multiple types of flow restrictions across multiple sectors [2]. For example, traffic that is the easiest to control using spacing programs (i.e., major flows) may be the least desirable traffic to impede. A combination of ground delays and time-based metering applied to departing and crossing traffic may be more effective to maximize throughput, but much harder to visualize and implement without additional decision-support tools.

Over the past two years, we have been formulating, developing, and testing a set of mathematical programming algorithms for controlling sector capacity levels. These algorithms will assist controllers in choosing appropriate...
combinations of control options in ARP, rather than performing a manual search of the solution space. The most promising algorithms will be embedded in FACET-ARP. This paper describes the algorithms, and it evaluates their performance and the merits of alternative approaches in algorithm design. The algorithms adapt time-based metering (TBM) concepts to manage en route congestion. While we use “sectors” as the regions of airspace in which demand is being controlled, the algorithms we describe apply equally well to flow constrained areas (FCAs), or any other volumes of airspace in which demand is measured and capacities are specified.

Alternative Approaches

Several models have appeared in the literature for aircraft routing and scheduling that minimize congestion. These approaches generally fall into two broad categories, (1) dispatching algorithms and (2) classical optimization. We have explored both techniques.

Dispatching Algorithms

A common heuristic approach to solving the Job Shop Scheduling Problem (JSSP) involves “dispatching rules.” It is worth noting that the terms “dispatching rule”, “scheduling rule”, “sequencing rule”, or even “heuristic” are often used synonymously in the literature. The premise is to sort jobs (i.e., flights) according to some criterion and then sequentially optimize each job in priority order. Typical sorting rules are Shortest Processing Time (SPT), First-Come-First-Served (FCFS), or desired completion time. Simulation studies [4] have shown that SPT is the best choice for optimizing the mean value of basic metrics such as total waiting time and system utilization, but can lead to excessively long waiting times for some jobs (translating to inequitably assigned delay). Dispatching rules can be a highly effective and efficient means of allocating the use of a single resource over time, especially when flights are processed in chronological order of demand [5]. For instance, Ration-by-Schedule (RBS) is an effective means of allocating runway usage in a ground delay program (GDP) by using a First-Scheduled-First-Served (FSFS) heuristic to sort flights.

Two heuristic approaches employing dispatching rules for aircraft routing and scheduling are the Collaborative Routing Coordination Tool (CRCT) [6][7] and Collaborative Routing Resource Allocation Tool (CRRAT) [5][8].

Optimization

In the classical optimization category, the sentinel formulations in the literature are the Traffic Flow Management Problem (TFMP) [9], in which ground delay and sector entry delays are determined for aircraft with no rerouting options, and the Traffic Flow Management Rerouting Problem (TFMRP) [10], in which aircraft rerouting is considered. TFMP and TFMRP are formulated as 0-1 integer-programming problems. The objective is to minimize total weighted delay, with constraints imposed upon departure and arrival capacities of airports, sector capacities, sector connectivities, and airport connectivities. The number of variables in these formulations grows rapidly as the time interval length is reduced, and this exposes tradeoffs between scenario scope and duration, modeling resolution, and computational feasibility.

The TFMP is equivalent in computational complexity to the JSSP [9], and therefore, is “NP-hard” in problem complexity [11]. However, Bertsimas and Stock have observed that the linear programming relaxation formulation almost always returns integer values, which provides some confidence that realistically sized problems might be solvable. The formulation of TFMP extends to the TFMRP, but with a potentially significant increase in the number of decision variables. For TFMRP, researchers [10] employ a hybrid, multi-step approach combining optimization and heuristic methods.

Comparable optimization techniques have been applied by Sherali et alia [12][13]. Their focus lies somewhere between traffic management and air traffic control. In addition to aircraft delay costs, the base model computes workload, conflict resolution penalties, and equity among carriers.

Our targeted area of application is more local in time and space than that of Bertsimas and considers fewer factors than Sherali. Though we share some techniques with them, our algorithms are designed to support a decision support tool that admits a rapid “what-if” capability. Hence, we are seeking more rapid solutions than those that would be obtained from large-scale optimization formulations.

Issues for Algorithm Selection

As a result of this research, we have identified a number of issues that explain why no single approach works well for all situations. These issues are referenced throughout this paper.

Issue A. Varying resource demands: Depending upon where an aircraft enters and exits a sector of airspace, dwell times in a sector can vary
For this reason, simple allocation rules or models that assume uniform resource utilization for all aircraft are inappropriate.

For example, Figure 1 shows a frequency distribution of dwell times for 328 flights passing through sector ZOB48 during a clear-weather, four-hour period in May of 2005. Note that the spike of nearly 30 flights with a dwell time of 10 minutes adds up to more total dwell time (roughly 300 minutes) than the sum of the 100+ flights in the first two histogram bins.

This lies in marked contrast to large-scale traffic flow management initiatives, e.g. ground delay programs, where one can assume that aircraft require (approximately) equal runway occupancy time, thereby allowing one to assign aircraft to uniform arrival slots. For airspace sectors, the problem is even more pronounced when capacity levels vary over time.

Issue B. Shifting arrival time sequences: A related problem arises when multiple sectors are considered. Flights may not only pass through volumes of airspace going in different directions, but may also enter different sectors according to different arrival sequences.

Issue C. Tradeoff between computational speed and granularity: Optimization techniques generally rely on the discretization of time into uniform intervals. If these time intervals are set large relative to flight dwell times, then the binary assignment of aircraft to time intervals wastes capacity. For instance, with five-minute time intervals, an aircraft requiring only one minute of dwell time wastes four minutes of capacity. On the other hand, if time intervals are highly refined, the number of decision variables and constraints scales with the number of time steps. Thus, when time discretization is applied, a happy medium between model fidelity and computational speed must be found.

Issue D. Delay cost evaluation: Models driven by delay minimizing objective functions require an expression of ground delay costs and air (en route) delay costs. Often, air delay is deemed considerably more expensive than ground delay (e.g. by a factor of two). These relative costs are not easily evaluated by heuristics, whereas optimization formulations can directly make that tradeoff by incorporating relative weights upon ground and en route delays in the objective function. Both approaches suffer from the fact that, in practice, delay costs are not uniform over aircraft.

Issue E. Single sector vs. multi-sector approaches: A modeling approach that works well on a single sector of airspace may not work well when demand needs to be controlled over multiple sectors.

Issue F. Interconnectivity among sectors: In particular, delays imposed on flights prior to entering one sector may trigger over-demand situations in other sectors (the “whack-a-mole” problem). This discourages use of heuristics that solve demand-capacity problems one sector at a time.

Algorithm Descriptions

We formed and evaluated four models (algorithms) for restricting aircraft flow:

- Single-pass dispatching rule
- Multi-pass dispatching rule
- Mathematical program
- Composite algorithm (combining a linear program and a dispatching rule heuristic)

The first two algorithms fall into the heuristic category, and the third falls into the optimization category. The fourth algorithm is a hybrid method utilizing both optimization and heuristic techniques. Some of the four algorithms can be applied only to a single sector, and some can be applied only to multiple sectors. (This will become clear as we discuss the details of the algorithms.) However, the composite algorithm, which we consider the most effective, can be applied to either. The following sections present each of the four algorithms in detail.

Single-pass Dispatching Rule

We found dispatching rules appealing because of their intuitive simplicity, short runtime, and compatibility with Collaborative Decision Making (CDM) principles (e.g. allowing the airlines to directly interact with the decision process).

In the single-pass dispatching rule algorithm, all flights are sorted in time order, then processed one at a time. The results presented later in this paper use an “entry-time minus accrued delay” sorting method to closely mimic the RBS algorithm used in GDPs [8]. The software also supports an
“entry-time” sorting method. In general, any prioritization rule can be used.

As flights are processed, capacity is reserved in the sectors they transit, leaving only the residual capacity for assignment to flights further down the list. Sector capacity is treated as a continuous-time resource to avoid the inefficiencies due to time discretization described above.

Delays are assumed to be assigned upstream of the set of sectors of interest (i.e. on the ground for pre-departure flights and en route for airborne flights). Formally, the following inputs are required:

- A set of scheduled flights, indexed by \( i = 1, \ldots, N \).
- A set of resources (sectors) comprising the region of interest, indexed by \( s = 1, \ldots, S \).
- Sector capacities \( C_i(t) \). Capacities are represented as continuous functions of time. We will use the notation \( C_i([t_1, t_2]) \) to denote the segment of the capacity function between times \( t_1 \) and \( t_2 \).
- An ordered sequence of sectors \( k = 1, \ldots, K_i \) and an associated set of dwell times \( \{\tau_{ik}\} \) for each aircraft \( i \) along its planned route. The transit times are generated by FACET-ARP, and represent times to traverse the set of resources along the flight’s route.

In the two dispatching algorithms, we consider only sectors contained in a region of specified interest. Other segments of a flight’s route are not directly considered by the algorithm, but will, of course, be affected if ground delays or upstream delays are assigned to that flight.

In the single-pass dispatching rule, flights are sequenced according to a priority rule. We prioritize flights by their earliest arrival time, defined as the earliest time that a flight enters the region, i.e. any one of the sectors \( \{1, \ldots, S\} \). (In principle, this is similar to the ration-by-schedule algorithm applied in GDPs in the US.) Flights are processed in order, but in the single-pass version, each flight is processed only once.

When a flight is processed, the algorithm iteratively computes and converges on an upstream delay to be assigned to that flight, along with associated entry and exit times for each of the sectors in the flight’s path. This upstream delay is computed such that no sectors’ capacities are violated, while considering the sector assignments of flights assigned in prior algorithm iterations. Let

\[
\Delta_i^m = \text{assigned delay to flight } i \text{ after the } m^{th} \text{ iteration.}
\]

\[
RC_{i,s}(t) = \text{the residual capacity function for sector } s \text{ after the } i^{th} \text{ flight has been assigned sector entry times, where } RC_{0,s}(t) = C_s(t).
\]

The iterations begin with the undelayed values from FACET-ARP:

\[
z_{ik}^0 = \text{planned arrival time for flight } i \text{ at sector } k, \quad \text{and } \Delta_i^0 = 0.
\]

Then, at each iteration \( m \), the minimum arrival time at each sector \( k \) along the flight’s path is computed, based on the earliest possible arrival times from the previous iteration:

\[
z_{ik}^m = \min \left\{ t \geq z_{ik}^{m-1} \mid RC_{i-1,k}([t, t + \tau_{ik}]) \geq 1 \right\},
\]

and

\[
\Delta_i^m = \Delta_i^{m-1} + \max_{k=1,\ldots,K_i} \left( z_{ik}^m - z_{ik}^{m-1} \right).
\]

When \( \Delta_i^m = \Delta_i^{m-1} \), the iterations have converged and the delay is determined.

**Multi-pass Dispatching Algorithm**

The input to the multiple-pass algorithm is the same as the input to the single-pass algorithm. The multiple-pass algorithm divides a multiple sector problem into a sequence of single-sector problems, to be solved iteratively. In this sense, it can be viewed as a greedy algorithm. It uses the dispatching rule described in the previous section, starting with the sector with the most excess demand. It then solves one sector at a time, delaying flights until all sectors are within their capacity.

Each time a sector problem is solved, all sector demands are recomputed to take into account all assigned delays. The dispatching rule is then applied to the sector with the most excess demand, and the process is repeated until all excess demand is eliminated. This algorithm is designed to mitigate the shifting arrival time problem described previously as an issue for algorithm selection (Issue B). However, delays imposed when solving a single-sector problem may result in over-demand situations reappearing in previously-solved sectors (Issue F), so that some sector problems may have to be solved multiple times before capacity constraints are satisfied in all sectors. However, since flights only receive additional delays at each step of the algorithm, there is no chance that the same problem will reappear, and the algorithm will converge.
Mathematical Program

In order to understand the mathematical program and the hybrid algorithm, we must first present the optimization formulation that underlies them both. In addition to the usual sector capacity constraints, we have constraints on the maximum ground or en route delay that can be assigned to any single flight, and take into account the relative costs of ground delay and airborne delay.

Input:

Here we introduce the notation and input data to the optimization formulations.

\[ i = 1, \ldots, N \] will be used to index flights.
\[ j = 1, \ldots, J \] will be used to index time intervals.
\[ s = 1, \ldots, S \] will be used to index sectors.

The set of sectors in the problem will be denoted by \( \mathcal{S} \). The subset of \( \mathcal{S} \) along the route for flight \( i \) will be denoted by \( \mathcal{S}_i \).

For each flight \( i \), \( \tau_i \) denotes the required transit (or dwell) time in sector \( s \), expressed in an integer number of time intervals. For example, if the time interval length is 10 seconds and it takes 3 minutes for flight \( i \) to transit the sector, then \( \tau_i \) would be 18 time intervals. As noted above, it will be assumed that all flight delays occur before a flight enters the sector.

For each flight \( i \), \( T_i \) denotes the earliest time that the flight can enter sector \( s \). We measure flight delays for flight \( i \) relative to this time. Thus, if the solution specifies a 1-minute delay for flight \( i \) and the time interval length is 10 seconds, the entry time for flight \( i \) would become \( T_i + 6 \).

We use the variables \( \alpha_i \) as indicators of which flights are in the air at the start time of the problem, and which flights are on the ground:

\[ \alpha_i = \begin{cases} 
1 & \text{if flight } i \text{ is enroute at time zero} \\
0 & \text{if flight } i \text{ has not departed at time zero} \end{cases} \]

The scalar \( \alpha_i \) denotes the cost of airborne delay relative to ground delay. For example, if we take airborne delay to be twice as costly as ground delay, then \( \alpha_i = 2 \).

Finally, we denote the capacity of sector \( s \) at time \( j \) by \( C_{js} \). This quantity represents the residual capacity after all exempt flights have been taken into account, and, as the notation indicates, may vary from time period to time period.

Decision Variables:

The problem is formulated in terms of the decision variables \( \{x_{js}\} \), where

\[ x_{js} = \begin{cases} 
1 & \text{if flight } i \text{ enters sector } s \text{ during interval } j \\
0 & \text{otherwise} \end{cases} \]

Thus, if \( x_{js} = 1 \), the flight will dwell in the sector during the interval \([j, j + \tau_i]\).

Since exactly one value of \( \{x_{js}\} \) is nonzero for each value of \( i \) and \( s \) \((s \in \mathcal{S}_i)\), the total number of flights in the sector \( s \) at time \( j \) can be written as

\[ N_{js} = \sum_{i=1}^{N} \sum_{j=\max(j-\tau_i,1)}^{j} x_{iks} \]

where we set \( \tau_i = 0 \) if \( s \not\in \mathcal{S}_i \).

Finally, the delay to flight \( i \) can be written as

\[ d_i = \sum_{j=\tau_i}^{j} (j \cdot x_{js}) - T_i \quad \text{for all } s \in \mathcal{S}_i, \]

which is the number of time intervals between the time that a flight could have first entered the sector and the time it actually enters the sector. Note that these constraints must be included for all \( s \in \mathcal{S}_i \) to maintain flight spacing through sectors that are not included in the problem statement.

In the mathematical program below, we minimize the total weighted flight delay subject to flight entry and sector capacity constraints:

Minimize \[ \sum_{i=1}^{N} d_i (1 + \alpha_i \omega_i) \] (1)

subject to

\[ d_i = \sum_{j=\tau_i}^{j} (j \cdot x_{js}) - T_i \quad \text{for all } s \in \mathcal{S}_i \] (2)

\[ \sum_{i=1}^{N} \sum_{j=\max(j-\tau_i,1)}^{j} x_{js} \leq C_{js} \quad \text{for all } j, s \in \mathcal{S} \] (3)

\[ \sum_{j=1}^{j} x_{js} = 1 \quad \text{for all } i, s \in \mathcal{S}_i \] (4)

\[ x_{js} = 0 \quad \text{if } j \leq T_{is} \quad \text{for all } i, s \in \mathcal{S}_i \] (5)

\[ x_{js} \leq 1 \quad \text{for all } i, j, s \in \mathcal{S}_i \] (6)

\[ d_i \leq \Delta_{\text{max}}. \] (7)

Constraint set (2), which has already been discussed, maintains the order in which each flight must transit its designated sectors and assigns the amount of delay in reaching the first of those sectors. Constraint set (3) enforces sector capacities. Constraint set (4) imposes sector entry constraints. Constraint set (5) enforces minimum entry times (optional), while (6) provides the
necessary bounds on the decision variables. As posed, this is a linear program (LP). It becomes an integer program (IP) when we replace (6) with:

$$x_{ij} \text{ binary for all } i, j, s \in \mathcal{I} \quad (6^*)$$

The LP will, in general, not produce a feasible solution to the capacity control problem due to fractional values for the decision variables. The IP will produce a feasible solution, but we have found that it requires computation times that are too long to incorporate into a real-time decision aid.

**Composite Algorithm**

The composite algorithm utilizes the linear program just described as its first step to solving the problem. An optimal solution to the linear program is a lower bound on the optimal integer solution, and much easier to find. However, it may contain fractional decision variable values, thus splitting sector entry times over multiple time intervals.

An important finding of our research is that these instances are relatively rare. Moreover, we developed a heuristic to restore fractional values from the optimal linear program. For example, if the LP solution assigns 1 minute of delay to half of a flight and 3 minutes of delay to the other half of the flight, then the “derived” solution is to assign 2 minutes of delay to the flight, which may no longer be feasible. The multi-pass dispatching rule applied to the “derived” solution produces a feasible integer solution. In [14], we illustrate some of the properties and advantages of this hybrid approach. One such finding is that when a large number of small time intervals is used in the linear program, the hybrid integer solution may, in fact, be an improvement over an “optimal” integer programming solution based on a coarser time discretization (Issue C).

**AARP Development and Testing**

The Automated Airspace Restriction Planner (AARP) is a stand-alone MATLAB module that interfaces with FACET-ARP. The integration of these components is as follows: The FACET-ARP application simulates flight data and computes projected demand for airspace resources. This demand is then exported to the AARP as the beginning state of an airspace rationing problem. The AARP solves the problem using any of the algorithms discussed in this paper. The AARP solution is then loaded into FACET-ARP and simulated as constraints upon individual flights.

The AARP allows the user to visualize flight and flow data and to modify sector capacities, flight exemptions, or optimization parameters. To solve the optimization problems, we call the GNU Linear Programming Kit (GLPK). The dispatching algorithms are implemented directly in MATLAB.

**Figure 2** is taken from the AARP user interface window. It shows occupancy levels and flight demand for an individual sector. Note that this sector’s demand exceeds its capacity. The AARP is also capable of displaying excess demand events, decomposing total sector demand into subflows and/or exempt flights, and invoking all of the optimization algorithms described in this paper.

**Figure 3** shows FACET-ARP after having loaded the flight delays computed by the AARP. The “before” and “after” demand profiles for a ZOB48 test problem are shown in **Figure 2** and **Figure 3**, respectively. These profiles are meant only to exhibit the software and are taken from the scenario described in the next section.

**Test Scenarios**

In order to test the algorithms, we selected a high-volume region of the NAS, Area 4 of Cleveland Center (ZOB), used traffic demand data for a clear-weather day taken from the ETMS database, and applied a variety of capacity
functions ranging from normal operations to severe weather conditions.

On the date chosen for the test scenario, the ZOB Area 4 sectors above 24,000 feet are ZOB45, ZOB47, ZOB48, and ZOB49. We additionally consider ZOB07 as it lies just below ZOB47 and is also above 24,000 feet. This set of five sectors is referred to as “Area 4” in this paper. These sectors handle traffic to and from a large number of major airports, including:

- arrivals and departures for Detroit (DTW),
- arrivals and departures for Pittsburgh (PIT),
- departures for Cleveland (CLE),
- arrivals and departures for Cincinnati (CVG),
- eastbound flows to the New York City airports and westbound flows to Chicago,
- and traffic flows between the Midwest and Philadelphia (PHL) plus the three Washington DC airports, as well as other overflights.

Filed flight plan data was extracted from the ETMS database for the 4-hour period on May 17, 2005 between 1800Z and 2159Z. The flight plans were input to FACET-ARP, and the trajectory projection algorithms in FACET were used to generate the traffic demand data.

May 17, 2005 was a clear weather day, selected to represent nominal traffic levels. In order to artificially construct excess demand scenarios, we used two different methods for constraining sector capacity. First, we lowered MAP levels by constant amounts (2, 4, or 6) simultaneously for all sectors. Second, we create a more complex problem by using traffic levels from May 17 together with capacity constraints derived from a severe weather day.

We ran both the 5-sector case and a single-sector case using a historically congested sector – ZOB48. A total of 977 flights entered Area 4 during the sample 4-hour period. Of those, 328 passed through ZOB48.

Severe weather capacities are derived from a scenario on June 14, 2005. On this day, a line of convective activity crossed ZOB from the west during the same 4-hour period of the day for which the May 17 traffic data was taken. Hourly snapshots of the weather are shown in Figure 4.

For analysis purposes, we extracted realized historical traffic levels from archived Enhanced Traffic Management System (ETMS) data, averaged those levels over each 1-hour period, and used 85% of those averaged levels as target capacities for resource allocation. This resulted in a total of 870 flight-minutes of excess demand in Area 4 during the 4-hour period, as shown in Figure 5 on the following page. (The red lines in the figure are the hourly target capacities.) The most over-capacitated sector is ZOB49, but all of the sectors experience at least some excess demand.

For each test scenario, we ran the dispatching rule and composite algorithms. Unlike the dispatching rules, the composite algorithm allows additional constraints to be included in the problem formulation. We include a 20-minute air delay constraint as an arbitrary but reasonable goal in assigning flight delays to manage congestion.

Figure 4. Severe Weather Impacting ZOB Center on June 14, 2005
The results are summarized in Table 1. The table contains values for total delay minutes, total number of flights delayed, maximum ground delay, and maximum en route delay, for each of the analysis cases.

Among the observations that we can make based on the data in the table are the following:

- Total delay increases nonlinearly with the amount of excess demand. As target capacity levels are decreased, excess demand increases from 19 flight minutes to 870 flight minutes. Meanwhile total delay minutes increase from 22 minutes (about 1.2 times excess demand) to 4,672 minutes (5.4 times excess demand) for the best of the solutions.

- As the over-demand situation becomes more acute, the composite algorithm performs much better than either dispatching rule. There is about a 29% improvement in total assigned delay in the severe weather case (4,672 minutes vs. 6,005 minutes). The composite approach achieves this improvement by identifying and shifting in time those flights that tend to consume unduly large portions of the capacity during low-capacity time periods (see reference [14] for elaboration).

- The numbers of delayed flights increase roughly in proportion to excess demand. In minimizing the total delay, the composite algorithm delays more flights than the dispatching rules. From an equity standpoint, this can be considered desirable as it tends to spread delay over a larger

Table 1. Analysis Results

<table>
<thead>
<tr>
<th>Case</th>
<th>Excess Demand</th>
<th>Single Pass</th>
<th>Multiple Pass</th>
<th>LP and Multiple Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full MAP</td>
<td>19</td>
<td>30</td>
<td>22</td>
<td>29</td>
</tr>
<tr>
<td>MAP-2</td>
<td>79</td>
<td>122</td>
<td>114</td>
<td>120</td>
</tr>
<tr>
<td>MAP-4</td>
<td>286</td>
<td>443</td>
<td>478</td>
<td>478</td>
</tr>
<tr>
<td>MAP-6</td>
<td>710</td>
<td>3229</td>
<td>3155</td>
<td>3244</td>
</tr>
<tr>
<td>WX Capacities</td>
<td>870</td>
<td>6605</td>
<td>6463</td>
<td>6672</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case</th>
<th>Excess Demand</th>
<th>Single Pass</th>
<th>Multiple Pass</th>
<th>LP and Multiple Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP-4</td>
<td>12</td>
<td>14</td>
<td>14</td>
<td>22</td>
</tr>
<tr>
<td>MAP-6</td>
<td>47</td>
<td>50</td>
<td>50</td>
<td>85</td>
</tr>
<tr>
<td>WX Capacities</td>
<td>111</td>
<td>368</td>
<td>368</td>
<td>273</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case</th>
<th>Excess Demand</th>
<th>Single Pass</th>
<th>Multiple Pass</th>
<th>LP and Multiple Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP-4</td>
<td>12</td>
<td>14</td>
<td>14</td>
<td>22</td>
</tr>
<tr>
<td>MAP-6</td>
<td>47</td>
<td>48</td>
<td>48</td>
<td>48</td>
</tr>
<tr>
<td>WX Capacities</td>
<td>111</td>
<td>62</td>
<td>62</td>
<td>98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case</th>
<th>Excess Demand</th>
<th>Single Pass</th>
<th>Multiple Pass</th>
<th>LP and Multiple Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP-4</td>
<td>12</td>
<td>14</td>
<td>14</td>
<td>22</td>
</tr>
<tr>
<td>MAP-6</td>
<td>47</td>
<td>48</td>
<td>48</td>
<td>48</td>
</tr>
<tr>
<td>WX Capacities</td>
<td>111</td>
<td>62</td>
<td>62</td>
<td>98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case</th>
<th>Excess Demand</th>
<th>Single Pass</th>
<th>Multiple Pass</th>
<th>LP and Multiple Pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP-4</td>
<td>12</td>
<td>14</td>
<td>14</td>
<td>22</td>
</tr>
<tr>
<td>MAP-6</td>
<td>47</td>
<td>48</td>
<td>48</td>
<td>48</td>
</tr>
<tr>
<td>WX Capacities</td>
<td>111</td>
<td>62</td>
<td>62</td>
<td>98</td>
</tr>
</tbody>
</table>

The results are summarized in Table 1. The table contains values for total delay minutes, total number of flights delayed, maximum ground delay, and maximum en route delay, for each of the analysis cases.

Among the observations that we can make based on the data in the table are the following:

- Total delay increases nonlinearly with the amount of excess demand. As target capacity levels are decreased, excess demand increases from 19 flight minutes to 870 flight minutes. Meanwhile total delay minutes increase from 22 minutes (about 1.2 times excess demand) to 4,672 minutes (5.4 times excess demand) for the best of the solutions.
collection of flights. The range of delays is also much greater with the composite algorithm. Since the latter incorporates a constraint on en route delay, assigned ground delays can become much larger in order to obtain a feasible solution. Note that the 20-minute en route delay constraint is exceeded in the three most severe cases. The LP portion of the composite algorithm produces a solution that conforms to the constraint, but adjustments made during the dispatching rule portion of the algorithm start with that solution and apply additional delays as needed to obtain an integer feasible solution.

- There is little difference between the two dispatching rules in any of the cases. (Obviously they are same algorithm when applied to a single sector.) This is in contrast to the results for a Chicago Center (ZAU) severe weather case reported in [14], where the multiple-pass algorithm performed much better than the single-pass algorithm. We expect that the data in the ZOB case does not exhibit as much variability in arrival time sequences from sector to sector, but we have not examined the data in detail to confirm that hypothesis.

Conclusions and Future Work

Through this research, we have identified the need for a capability to better manage sector congestion, particularly within the presence of time-based metering. An airspace restriction planner decision support tool has been developed in concert with field observations and subject matter expert discussions to develop a prototype capability. The primary focus of this paper is the development of automation algorithms behind that software; these will allow traffic managers to rapidly sort through and choose appropriate restrictions to curtail excess demand on sectors of airspace.

We formulated models spanning both heuristic and optimization approaches and tested these on historical flight data. Our preliminary results indicate that the composite algorithm, which combines heuristics with a linear program, strikes a suitable balance between optimal solution and fast run time. Further experimentation is required using faster solver software, such as CPLEX. Nonetheless, we have shown that difficult demand problems can be solved in reasonable computational time e.g. several hours of traffic demand for an ARTCC could probably be solved in a minute or so of computational time.

We plan the following next steps:

1. Enhance algorithms to include multiple degrees of freedom. These degrees of freedom will include: rerouting, metering, tunneling, altitude capping, and assigned departure rates.
2. Conduct human-in-the-loop simulations to demonstrate the benefits of automated restrictions compared with current human generated restrictions. These simulations will include multiple sectors and traffic conditions from ZOB. The effectiveness will be assessed using measures such as delay statistics (average, maximum, minimum), equity between air-borne and ground-based delays, and number of aircraft delayed.

Acknowledgment

This work was sponsored by the National Aeronautics and Space Administration’s Advanced Air Transportation Technologies Project.

Key Words


References

Rerouting”, Proceedings of the 3rd USA/Europe ATM R&D Seminar, Santa Fe, NM, December.


[14] Burke, Jason, Ray Jakobovits, Steven Lent, 2006, Regional Metering (RM) Software Development and FACET-Based Modeling and Simulation: Automated Airspace Restriction Planner Algorithms and Historical Analysis, Metron Aviation Report, July. For document access, contact Parimal Kopardekar (pkopardekar@mail.arc.nasa.gov).

Author Biographies

Ray Jakobovits is an independent consultant specializing in optimization methods, stochastic modeling, and financial mathematics. He has over 30 years of experience in commercial, aviation and defense applications of optimization theory, including the development of numerous tactical decision support systems that embed large-scale optimization and distributed optimization techniques. He holds a B.S. in Mathematics and a M.S. in Operations Research from Stanford University, and a Ph.D. in Operations Research from Cornell University. He has held research staff positions with Metron Inc. and the Institute for Defense Analyses. Prior to becoming a consultant, he was president of Metron Management Company, a financial engineering and hedge fund management firm.

Parimal Kopardekar works as an Associate Principal Investigator of Dynamic Airspace Configuration technical area at the NASA Ames Research Center. Prior to his current assignment, he served as a Project Manager of the Strategic Airspace Usage project and sub-project manager under the Advanced Air Transportation Technologies project. Prior to working at NASA, he worked for the Federal Aviation Administration where he conducted research and development activities in the area of air traffic management. He has published numerous journal and conference papers in the area of air traffic management. As an adjunct faculty member at Rutgers and Drexel Universities, he taught graduate-level courses. He holds Ph.D. and M.S. degrees in Industrial Engineering and Bachelor of Engineering in Production Engineering.

Jason Burke is a senior analyst and software engineer at Metron Aviation Inc. (a consulting firm specializing in researching and developing solutions to air traffic management problems). Jason specializes in the integration of simulation and decision support tools and has served as software lead on numerous NASA and FAA projects. He holds a M.S. in Systems Engineering from the University of Maryland and a B.S. in Computer Engineering from the University of Dayton.

Robert Hoffman is a principal analyst at Metron Aviation, Inc. He holds a B.S. in mathematics from Boston University and a Ph.D. in applied mathematics with a concentration in operations research from the University of Maryland. Currently, he is involved in modeling and analysis of strategic ATM initiatives specializing in resource allocation and optimization issues. Dr. Hoffman is also a Visiting Research Scientist at the University of Maryland’s Institute for Systems Research. He has published numerous journal and conference papers in the area of air traffic management and optimization, for which he has received multiple awards.