Solving High Fidelity, Large-Scale Traffic Flow Management Problems in Reduced Time

Joseph Rios[*]
NASA Ames Research Center, Moffett Field, CA 94035

Kevin Ross[†]
University of California at Santa Cruz, Santa Cruz, CA 95064

Traffic flow management aims to minimize total delay subject to various constraints. While there are several models to achieve this aim, those with the highest fidelity are typically the most useful but also the most computationally intensive. Often these high fidelity models can take hours or days to solve a nationwide, 2-3 hour planning horizon scenario. There is a need for a computationally feasible scheduling algorithm that achieves near-optimal performance on large, nationwide data sets if there is to be a useful tool for nationwide traffic flow management. To this end, this work describes how flights from a given traffic data set are each assigned a "congestion score." Based upon this score, the highest scoring flights (those involved in the most congestion) are selected for use as input to a well-accepted traffic flow management model. By choosing progressively larger key subsets of flights, the tradeoff between runtime and solution quality is examined. Results are promising as the scenarios studied showed that optimizing on a subset of only 20% of all flights produces a schedule that is within 1% of the global optimum from the original "master" problem, and this reduced scenario is solved about 14 times faster.

I. Introduction

At peak times in the United States, there are thousands of flights in the air and thousands more scheduled to depart within a few hours. Determining which of these flights to delay during excessive demand or disruptions (such as weather or system failures) is a difficult, computationally intensive task potentially requiring hours or days to obtain an optimal solution. This problem can be generically referred to as the Traffic Flow Management Problem (TFMP). While many uncertainties exist when attempting to model traffic flow, the promise of optimization applied to Traffic Flow Management (TFM) is ultimately increased efficiency in the system when compared to today's operations. Taking the time and resources to solve this problem optimally has shown promise for significantly reducing overall delay when compared to current methods.[1,2] TFM optimization could reduce reliance upon ground delay programs, ground stops and airspace flow programs as a more detailed view of which flights need to be held to meet capacity constraints is possible through models, such as the one used in this study. There are several models for solving the TFMP. They are generally categorized according to whether they aggregate the flights into flows[3–5] or treat them discretely[6–8] and whether they are deterministic or stochastic in nature. A major roadblock to the implementation and use of optimal models for the TFMP is the large computational effort that is required, but obtaining quality solutions to the TFMP is likely to gain importance as the traffic demand continues to increase in the near future.

This paper presents a method for solving the TFMP to near-optimality in significantly reduced time. The heuristic presented chooses a subset of flights and sectors deemed "important" to the model and relegates the remaining data to the "background" of the problem. The model of choice is non-aggregate (often called 'Lagrangian') and deterministic, referred to as the Bertsimas and Stock-Patterson (BSP) model.[6] This model

[*]Aerospace Engineer, Automation Concepts Research Branch, Mail Stop 210-10, Joseph.L.Rios@nasa.gov. Member AIAA.
[†]Assistant Professor, School of Engineering, 1156 High Street, kross@soe.ucsc.edu.

was published a decade ago, but recently there has been more research into its implementation and use.[1,8,9] This increased interest is likely encouraged by the availability of more powerful computing resources to solve problems based on this model. Even the original authors of the BSP model were interested in the computational costs of running it. A decade later, there still has not been a nationwide, fine-grained implementation shown to complete in a reasonable amount of time on common computing equipment. Showing that this model can produce a result in a short amount of time for large problems would be a key step toward demonstrating its feasibility for implementation in future decision support tools for traffic flow managers.

This paper is organized as follows. In Section II the new heuristic and the original model upon which it is based are described in detail. Next, in Section III a description of the data used in the validation experiments along with how they were acquired is presented. Following the presentation of the data, Section IV describes the experiments performed and their respective results. Finally, Section V offers concluding remarks including potential future research directions.

II. Model and Heuristic Reduction

In this section, the original BSP model is described. Following that description, a heuristic that selects reduced data sets to supply to the model is proposed. This heuristic reduces the problem size allowing for faster computation at the cost of a slight increase in delay costs.

II.A. Bertsimas-Stock Patterson Binary Integer Program

To perform optimal scheduling to minimize delay costs, the BSP model[6] was implemented. This model describes the TFMP with multiple airports and deterministic sector capacities. Each flight in the set of flights, $\mathcal{F}$, is described as an ordered list of sectors with earliest and latest feasible entry times for each of those sectors. Sectors in the flight path are denoted by $P(f,y)$ where f is the flight and y is the ordinal representing the sector in the flight path. For the purposes of the model, airports are considered as special cases of sectors. The objective function to be minimized is:

$$\sum_f [c_f^g g_f + c_f^a a_f]$$

where the c_f^g and c_f^a are the costs of holding flight f on the ground or in the air, respectively, for one unit of time. The air and ground delay for each flight f (a_f and g_f, respectively) are ultimately expressed in terms of the binary variables, w, through a substitution for a_f and g_f. These variables designate whether a flight has entered a given sector by a given time.

$$w_{ft}^j = \begin{cases} 1, & \text{if flight } f \text{ arrives at sector } j \text{ by time } t, \\ 0, & \text{otherwise.} \end{cases}$$

The problem is constrained by various capacity restrictions. Namely, airport departure $(D_k(t))$, airport arrival $(A_k(t))$, and sector $(S_j(t))$ capacities at time t, where k is in the set of airports, $\mathcal{K}$, and j is in the set of sectors, $\mathcal{J}$:

$$\begin{aligned} \textstyle\sum_{f:P(f,1)=k} \quad (w_{ft}^k - w_{f,t-1}^k) &\leq D_k(t) \\ \textstyle\sum_{f:P(f,\text{last})=k} \quad (w_{ft}^k - w_{f,t-1}^k) &\leq A_k(t) \\ \textstyle\sum_{f:P(f,i)=j,P(f,i+1)=j'} \quad (w_{ft}^j - w_{ft}^{j'}) &\leq S_j(t) \end{aligned}$$

There is a set of constraints that guarantees each flight spends at least the specified minimum amount of time in each of its sectors. A final set of constraints enforces the flight path requirements, i.e., the sectors are visited in the correct order. Given that $\min(f,j)$ is the minimum amount of time flight f spends in sector j, these two constraint sets are formalized here, respectively:

$$\begin{aligned} w_{f,t+\min(f,j)}^{j'} - w_{ft}^j &\leq 0 \\ w_{ft}^j - w_{f,t-1}^j &\geq 0 \end{aligned}$$

The original authors note how this model is extensible to various scenarios including rerouting of flights, modeling of ground-holding programs, continuing flights, modeling of banks of flights wherein one of several planes may be assigned to a given flight, and modeling specific airports that exhibit dependence between arrival and departure capacities.

Bertsimas and Stock-Patterson showed that the number of constraints in the problem is bounded above by

$$2\,|\mathcal{K}|\,|\mathcal{T}| + |\mathcal{J}|\,|\mathcal{T}| + 2\,|\mathcal{F}|\,DX \tag{1}$$

and the number of variables is bounded from above by

$$|\mathcal{F}|\,DX. \tag{2}$$

where D is the feasible lateness in terms of time slices, X is the longest flight path in the system in terms of number of sectors, and $\mathcal{T}$ is the set of time slices. In general $|\mathcal{F}|$ is much larger than the other variables in this expression. It is also worth noting here that the time resolution greatly affects the problem size. In this study, time slices of length 1 minute are used. A resolution of 5 or 15 minutes would generate many fewer variables (since D would be smaller) and constraints (since D *and* $\mathcal{T}$ would be smaller) for the same model time horizon and, thus, would be more quickly solved at the cost of reduced resolution.

For this study, the important point to recognize regarding these bounds on variables and constraints is their relationship with the number of flights, $|\mathcal{F}|$. If the number of flights doubles, both constraints and variables will essentially double as well. For an understanding of why this is important, several general references[10–12] discuss the complexity issues with linear programming. Essentially, the algorithm likely used to solve this problem (the simplex algorithm) can take $O(mn)$ steps at each iteration and will likely take about $2m$ iterations to solve[12] where n is the number of variables and m is the number of constraints. Note that these runtime measurements do not account for the binary nature of the variables in the BSP model, which may add significantly to the runtime. In general, the fewer constraints and variables in the system, the better the expected runtime performance. Solving this problem on a national scale can take hours or even days which is impractical. Therefore, the following heuristic is provided as a method for reducing the runtime without sacrificing significant solution quality.

II.B. Problem Reduction Heuristic

For large data sets, the BSP model will produce tens of thousands to hundreds of thousands of constraints and variables in the linear program. A problem of this size can rarely be solved in a 15-30 minute time frame. Even on the most modern desktop equipment (3+ GHz processor(s), 8+ GB of memory), problems of this size can take anywhere from 1 hour to 24+ hours to solve exactly. As mentioned in Section II.A, choosing a lower time resolution can greatly reduce problem size. The goal of this study, however, is to solve large problems with high time resolution, which increases the size of D and $\mathcal{T}$. The resolution for this study is set at 1 minute. This implies that the location of the plane (which sector) is known/predicted at each minute rather than only every 5 or 15 minutes.

Since D is determined by the length of a time slice and X is determined by the flights in the system, the best way to reduce the number of variables *and* constraints in the problem and, thus, reduce the complexity of the problem as a whole is to reduce the number of flights, $|\mathcal{F}|$, that are being controlled in the system (see Equations 1 and 2). To this end, every flight in the system was assigned a "congestion score." This score is dependent on a more complete definition of congestion which is provided before defining the score itself.

Recall that the model is concerned with capacities at given times, so each sector/time pair, airport arrival/time pair, and airport departure/time pair has a certain capacity. These resource pairs will be referred to as *bins*. For each of these bins that are predicted to be over capacity, all involved flights are considered "impacted." Note that to find good or potentially optimal solutions, it is generally not sufficient to include in the optimization only the flights from the over-capacity bins. Often, (spatially or temporally) nearby bins need also be included in the data set to ensure good solutions. Consider, for example, a departure airport with capacity 10 flights per 15 minutes. If 11 flights are scheduled from 8:00 to 8:14, this bin is over capacity. All flights in this bin are considered impacted. Now if the airport has 10 departures scheduled for

8:15 to 8:29, one of the flights from 8:00 to 8:14 will have to be scheduled to depart after 8:29 since none of the flights departing from 8:15 to 8:29 are included in the model since their bin wasn't impacted. This will generate excessive delay, because the optimal solution is likely to hold the latest flight in the 8:00 to 8:14 bin to depart in the next bin and to bump one of 8:15 to 8:29 flights to depart after 8:29. This simple scenario leads to the following insight.

To obtain good solutions, temporally successive airport bins must be included in the data set to ensure that the excess demand might be accommodated appropriately. Essentially, when a bin, say α, is over capacity, we must include successive bins until the sum of the residual capacity of those successive bins can accommodate the excess demand from the earlier bin(s). Thus, if $cap(i)$ is the capacity of bin i and $occ(i)$ is the scheduled occupancy of bin i with bins strictly ordered temporally, we need to find the minimum n such that:

$$\sum_{i=\alpha}^{n}(cap(i) - occ(i)) \geq 0 \tag{3}$$

Having found n, all bins from α through $n-1$ should be considered "impacted" and all flights scheduled to use those bins are added to the list of impacted flights. Note that $n-1$ is used since bin n offers enough residual capacity to finally accommodate all flights from prior bins. The flights scheduled for departure from bin n will not need to be controlled as only the residual capacity of that bin will be needed.

Note that if departure capacities were the only capacity constraints in the system (i.e., there are no en route sector or airport arrival constraints), then the above methodology would guarantee the same optimal solution value as the original problem that includes all flights. However, since there is a strong relationship among all capacity constraints, the method described can only be considered a heuristic for finding a "good" set of flights to use in the reduced data set. For the experiments described in Section IV, airport bins were considered impacted using Equation 3 on all departure and arrival bins, but enroute bins were only considered impacted if they were actually impacted. Thus, if sector ZAU77 is over capacity at time 13:03UTC, all of the impacted flights are included in the reduced model data set, but temporally or spatially nearby sectors are not included unless they too are over capacity because it is not clear which nearby sectors should be included.

Armed with this definition of congestion, a *congestion score* can now be defined. This definition relies on several independent parameters. The first parameter, p, is the percentage of capacity being used to define congestion. Typically, this parameter is 100% or, equivalently, 1.0. For clarity and depending on context, p may be defined as an actual percentage or its decimal equivalent. Though if p is decremented, more flights become included in the congestion of the system. For example, if 210 flights are involved in congestion when 100% of capacity is available, there will be *at least* 210 flights that would be involved in congestion if there were only 90% of maximum capacity available. This serves to collect flights that are not directly involved in congested sectors or airports but are using relatively dense resources. It will be described further in Section IV how p is allowed to vary from 1.0 down to 0.0 in 0.1-sized steps. The second parameter, r, is the ratio of cost assigned to airport congestion to cost assigned to sector congestion. For this study, that parameter is fixed at 2 because that value seemed to provide good results for these experiments. In certain scenarios, it may make more sense for r to have some other value. The final parameter is c, which is a large cost value. It serves to distinguish flights involved in congestion at various levels of p, so c should be chosen such that it is much greater than the number of resources (sectors and airports) used by any particular flight. In the experiments described in Section IV, a value of 10,000 was used for c. The congestion score, CS, of a flight, f, is defined thusly:

$$CS(f) = max_{p \in P}(1_{p,f}\left[(c \times p) + (r \times aCount_f) + sCount_f\right]) \tag{4}$$

where $aCount_f$ and $sCount_f$ are the number of congested airport bins and congested sector bins that flight f uses and $1_{p,f}$ is an indicator variable denoting whether flight f is actually involved in congestion with parameter p. As for the "max" function in CS, if P is the set of all p used in a given experiment, only the largest value of $CS(f)$ over the set P is kept.

Alternate CS functions may lead to different, potentially better, solutions. This study focused on the congestion score described by Equation 4, though developing and testing other congestion metrics would be necessary to determine if this CS function could be improved. Some other options for the CS function include summing up the CS over all values of $p \in P$ instead of taking the maximum CS over all $p \in P$. This would allow finer-grained differentiation between flights at a cost of increased computation. Another alternative might be to develop a definition of "congestion degree." If a flight is directly involved in congestion it could be considered "primarily congested," then all flights that interact with a primarily congested flight that are not themselves primarily congested may be considered "secondarily congested" and so on. Based on the degree of congestion and other factors, flights could then be ranked for selection. There are likely several other logical definitions of CS.

Since this paper is concerned with runtime values, it is worth noting that the process of ranking all flights and creating a reduced model data set is negligible compared to the runtimes required for the optimization. For instance, for a 9000+ flight, 180 minute scenario, a data set can be generated in under one minute (including the FACET simulation) on the hardware described in Section IV. For smaller scenarios, data set generation may take under 10 seconds. Little care was taken to improve these times, though there are likely opportunities to do so.

III. Data

A data set was chosen such that it represented a typical day with several resources operating at (or potentially over) capacity. The ASDI (Aircraft Situation Display to Industry)[13] data for Thursday, August 24th, 2005 was obtained. More specifically, the experiments described in Section IV used the historic data for that date starting at 13:15 UTC (9:15 AM EDT). This represents the fairly dense mid-to-late morning traffic on the east coast, as well as the earlier morning rush on the west coast. The data file was run for 180 minutes so that the scenario ended at 12:15 PM EDT. This project was focused on only domestic flights so all international traffic were excluded. Also, all flights that left a center and then re-entered the same center later without landing first were excluded – flights in this class were few. Flights through Canadian airspace were retained, but the Canadian sector constraints were ignored.

Every center was included to varying degrees due to the flights in the data set (i.e., more flights in the New York center than those in the Seattle center). More important to this model, the number of sectors ($|\mathcal{J}|$ in Equation 1) included was 974. No distinction is made between low, high, or superhigh sectors. Also adding to the complexity of the problem was the large number of airports, 905. While the vast majority of these airports do not serve as the binding constraints in the model, it is not clear, in general, which airports, sectors, and flights are actually "important" to the model for an arbitrary scenario. A smaller capacity airport, like Las Vegas for example, may actually influence global delays in a given data set more than a very large airport such as O'Hare in Chicago. Many of these 905 airports could have been eliminated from the problem by checking if less than, say, 5 flights used the airport over course of the planning horizon, though quite often these airports "fall out" of the problem due to the way flights are selected for inclusion to the reduced model.

To extract the data in the appropriate form, the Future ATM Concepts Evaluation Tool (FACET)[14] was used. FACET is capable of many functions, but for this study, its capabilities of playing back and simulating historical data and recording statistics were most relevant. Easing the burden of collecting data and controlling simulations was the newly implemented FACET Application Programming Interface (API) (based on a research tool called CARAT#[15]), which allows the experimenter to interface with FACET through either Java or Matlab code in order to extract information about elements in the system. The default sector capacities known as Monitor Alert Parameters (MAP) and airport capacities available through FACET were used for the experiments. If ever there was an airport without explicit capacities in FACET, a default value of 10 was used for arrival and departure capacity per 15 minutes. Often if an airport did not have explicit capacities, it was seldom used and quite small and, thus, not binding in the problem.

IV. Experiments and Results

Initially, the plan for exercising this model reduction heuristic involved setting parameter p in Equation 4 to 100% and solving the resulting reduced model for all flights involved in this congestion. These experiments incorporated roughly 15% of the flights in the system and resulted in solutions that were within 20% of optimal. To obtain a better (closer to optimal) solution, the decision was made to lower p and accept a larger overall percentage of flights in the system based on their respective $CS(f)$ values. Thus the following parametric study was implemented.

The set P was defined as $\{1.0, 0.9, 0.8, ..., 0.1, 0.0\}$. Every flight in the system, therefore, was assigned a congestion score, CS since every flight would be affected by *at least* the $p = 0.0$ scenario. Based upon these scores, the list of flights was ordered from highest CS to lowest CS, resulting in a list of the flights involved in the most congestion down to the least congestion. Then the top z% of the flights were taken to compose the reduced model where z was varied from 10% to 100%, while the remainder of flights were appropriately included in the background via capacity reductions. The subsequent delay costs and runtimes from each value of z were collected for comparison. For this parametric study, the set of all z's will be referred to as $\mathcal{Z}$.

For all experiments, the BSP model was implemented using the AMPL modeling language[16] and solved using the CPLEX 11.0 solver.[17] The "mipgap" parameter was set to 0.5%. This parameter allows CPLEX to return the first binary solution that it finds that is within 0.5% of the known relaxed optimal solution. The "relaxed" solution is found by ignoring the constraints that dictate the decision variables be binary. The machine used for experimentation was a dual quad-core Xeon with a clock speed of 2.66 GHz and 32GB of available memory. No experiments in this study were limited by lack of memory.

z-value (inclusion %)	Number Flights	Delay Cost	% Increase in Delay	Solve Time (min)	% Runtime Improvement
10	463	382	0.0	0.55	97.0
20	926	382	0.0	1.70	90.9
30	1390	382	0.0	3.03	83.7
40	1853	382	0.0	4.68	74.9
50	2316	382	0.0	6.83	63.3
60	2779	382	0.0	9.00	51.7
70	3242	382	0.0	11.18	40.0
80	3706	382	0.0	12.93	30.6
90	4169	382	0.0	15.97	14.3
100	4632	382	0.0	18.63	0.0

Table 1. Results from the 40 minute planning horizon, nominal capacity scenario.

z-value (inclusion %)	Number Flights	Delay Cost	% Increase in Delay	Solve Time (min)	% Runtime Improvement
10	795	-	-	-	-
20	1590	1237	0.4	14.60	92.8
30	2385	1235	0.2	26.73	86.9
40	3180	1232	0.0	62.62	69.3
50	3975	1232	0.0	90.80	55.5
60	4770	1232	0.0	123.48	39.5
70	5565	1232	0.0	150.50	26.2
80	6360	1232	0.0	171.58	15.9
90	7155	1232	0.0	188.63	7.5
100	7950	1232	0.0	203.95	0.0

Table 2. Results from the 120 minute planning horizon, nominal capacity scenario.

To test the performance of the model reduction heuristic, a set of scenarios was generated using the flight data noted in Section III. Initially, nominal capacities were considered for two different planning horizons, 40 and 120 minutes. The 40-min planning horizon allows for verification of the overall method while the 120-min horizon represents a time more suitable for actually planning traffic initiatives in a strategic manner. For the shorter planning horizon, the delay values for $z = 10\%$ and $z = 100\%$ were equal. This implies that in simple, short scenarios, looking at only the 10-20% most congested flights and locking the rest in the background will likely provide an excellent solution to the global problem. Results for the 40 minute, nominal scenario are summarized in Table 1. For the 120 minute, nominal scenario, the global optimum was found when $z \geq 40\%$. The results for $z = 30\%$ and $z = 20\%$ were within 1% of the global optimum. Note that the solution for $z = 10\%$ was infeasible. This occurs whenever the percentage of impacted flights is greater than z. Essentially, z must be greater than or equal to the percentage of impacted flights at nominal capacity. This situation is easily uncovered before running the optimization. The results for the 120-minute, nominal scenario are summarized in Table 2. It is interesting to note that a large-scale, high fidelity scenario can be solved under 15 minutes where and within 1% of optimality, whereas before solving the problem would take nearly 2.5 hours.

To study this heuristic's performance on reduced capacity scenarios, a weather event affecting the San

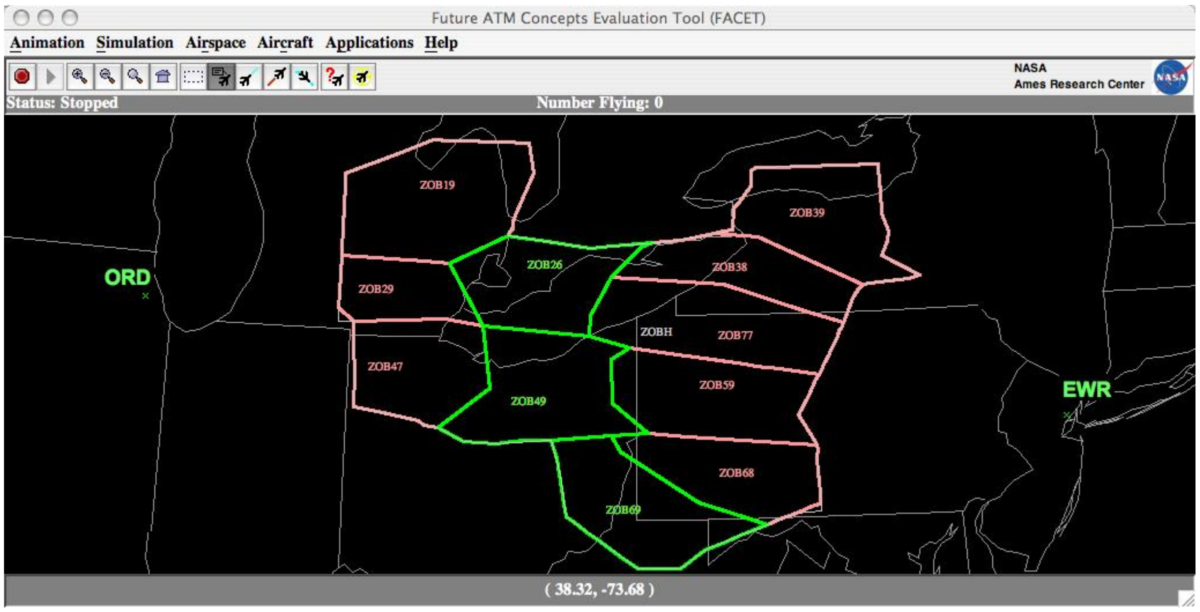
Future ATM Concepts Evaluation Tool (FACET)
Animation Simulation Airspace Aircraft Applications Help
NASA Ames Research Center
Status: Stopped
Number Flying: 0
ZOB19
ZOB39
ZOB26
ZOB38
ORD
ZOB29
ZOBH
ZOB77
ZOB47
ZOB59
EWR
ZOB49
ZOB68
ZOB69
(38.32, -73.68)

Figure 1. The high-altitude Cleveland sectors. The sectors modeled with reduced capacity in the weather scenario experiments (ZOB26, ZOB49, ZOB69) are highlighted in green. State boundaries are visible in the background.

Francisco Bay Area was roughly modeled. This region, in particular San Francisco International Airport (SFO), is often impacted by a marine stratus layer in the early to mid morning hours.[18] To roughly simulate this effect, the arrival capacity of SFO was reduced from a rate of 60 per hour to a rate of 32 per hour. This situation is typically handled with a combination of ground delay programs and miles-in-trail restrictions. Simultaneous with (though independent of) the disruption at SFO, the capacities of three contiguous en route sectors in the Cleveland center (ZOB26, ZOB49 and ZOB69) were reduced to 40% of their nominal MAP values. This roughly simulates a small weather pattern in the busy New York-Chicago corridor. The position of these sectors in relation to Chicago's O'Hare Airport and Newark International Airport is depicted in Figure 1. The same flight demand data was used as in the previous experiments and the planning horizon was kept at 120 minutes. The results are presented in Table 3. Note that the best solution was obtained for $z = 70\%$, which is initially surprising and is discussed in more detail below. Overall, the gains in computation speed appear to outweigh the loss in optimality when comparing the small z cases to the large z cases.

z-value (inclusion %)	Number Flights	Delay Cost	% Increase in Delay	Solve Time (min)	% Runtime Improvement
10	795	-	-	-	-
20	1590	1472	0.5	15.70	88.3
30	2385	1469	0.3	26.37	80.3
40	3180	1468	0.3	58.22	56.6
50	3975	1465	0.1	90.68	32.4
60	4770	1467	0.2	69.93	47.8
70	5565	1464	0.0	149.08	-11.2
80	6360	1467	0.2	102.32	23.7
90	7155	1470	0.4	110.70	17.4
100	7950	1466	0.1	134.10	0.0

Table 3. Results from the 120 minute planning horizon, reduced capacity at SFO and 3 en route Cleveland sectors.

This weather scenario results require further explanation. When optimally solving the BSP model, as z increases the delay costs should be non-increasing. The basic notion is that as more flights are controlled, better (or at least not *worse*) solutions can be discovered. The reason that the delay values fluctuate slightly within Table 3 has to do with the mipgap discussed above. Specifically, for $z = 70\%$, the first binary solution found that was within the mipgap tolerance provided a delay cost of 1464, which was reportedly within 0.12% of the relaxed, optimal solution. For comparison, when $z = 80\%$ the solution providing a delay cost of 1467 was within 0.32% of the relaxed, optimal solution. This small difference is enough to account for the fact that the $z = 70\%$ solution was "better" than the $z = 80\%$ solution. If both scenarios were allowed to continue computing to find their respective optimal binary solutions, the expectation is that the delay cost for the $z = 80\%$ case to be less than or equal to that for the $z = 70\%$ case. Also of note is the negative runtime "improvement" of the $z = 70\%$ case. The runtime improvement was measured in relation to the $z = 100\%$ case. To obtain an optimal solution that was within the defined mipgap, the $z = 70\%$ case needed to enter an extra solving phase that the other cases were able to avoid. To solve linear programs with binary variables, CPLEX solves the "relaxed" version (i.e., all binary variable are considered real variables bound between 0.0 and 1.0) of the problem first. In all cases but the $z = 70\%$ case, the relaxed solution consisted of binary values for all variables. When non-binary variable are present after this initial phase (the simplex algorithm as discussed in Section II.A), CPLEX enters another solving phase (typically called "branch-and-bound") to assign appropriate values to the variables. This additional phase is typically computationally difficult, hence the longer runtime for the $z = 70\%$ case.

Since this is a study involving the importance of runtimes, there is a final note regarding the cost of converting an AMPL model into an instance that CPLEX can work to solve. AMPL offers a measurement of this conversion time through a variable called "`_ampl_time`," while the actual time spent to solve the model is stored in a variable called "`_solve_time`." The reported times in the result tables are the sums of these two variables. Overall, `_ampl_time` is often greater than `_solve_time`, especially on the smaller models. This cost of translating the mathematical model into data structures for CPLEX is significant and, thus, important to remember when designing tools for TFM.

V. Conclusion

This study has shown that a good heuristic for ranking flights based upon their congestion can aid in selecting a subset of flights on which to optimize to obtain a global solution. The results indicate that significant runtime gains are possible at a small degradation in solution quality. Further study will be necessary to determine the applicability of the method to all traffic flow management scenarios, but at the outset, it seems promising that most scenarios could benefit from such a scheme. This study demonstrates that the computation gains hold even as traffic density increases.

An open question is how to actually choose the best parameters (such as r, z or the step size for the set P) for a given scenario. If enough scenarios are run, then a database of results would offer some data mining opportunities for determining parameters for a new scenario. Then, if a new scenario could be recognized as similar (in terms of weather patterns, projected traffic, etc.) to an archived scenario (or scenarios), parameters could be determined by analyzing those of the known, archived scenario(s). Another approach might be to simultaneously launch several instances of a given problem with varying parameters. This would be feasible given the ubiquity of commodity computing resources capable of handling this algorithm. Then as the deadline for a decision approaches, a traffic flow manager could choose the best solution from the instances that have completed. For example in Table 2, instances defined by all values of $z \in \mathcal{Z}$ could be simultaneously launched on separate machines or cores within a machine. If the decision deadline were 15 minutes in the future, the best (and only) available solution would be from the $z = 20\%$ instance, whereas if the decision deadline were 30 minutes, the best available solution would then be the $z = 30\%$ solution. Note that without the heuristics defined in this work, the optimal solution would only be available after waiting nearly 3.5 hours.

Future work in this area of accelerating TFM optimization will involve other techniques such as parallelization. The entire NAS, for example, could be partitioned physically[19] or possibly temporally.[1] These parallelization techniques may be used in conjunction with the model reduction described here. The hope would be that nationwide, high fidelity TFM optimization could be reliably completed in under 15 or 30 minutes on a continual basis.

References

[1]Grabbe, S., Sridhar, B., and Mukherjee, A., "Central East Pacific Flight Scheduling," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Hilton Head, South Carolina, August 2007.

[2]Mukherjee, A., Grabbe, S., and Sridhar, B., "Alleviating Airspace Restriction through Strategic Control," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, August 2008.

[3]Sridhar, B., Soni, T., Sheth, K., and Chatterji, G., "An Aggregate Flow Model for Air Traffic Management," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Providence, Rhode Island, August 2004.

[4]Menon, P., Sweridul, G., and Bilimoria, K., "New Approach for Modeling, Analysis, and Control of Air Traffic Flow," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 5, 2004, pp. 737–744.

[5]Bayen, A. M., Raffard, R. L., and Tomlin, C. J., "Adjoint-based control of a new Eulerian network model of air traffic flow," *IEEE Transactions on Control Systems Technology*, Vol. 14, No. 5, September 2006, pp. 804–818.

[6]Bertsimas, D. and Patterson, S. S., "The Air traffic Flow Management Problem with Enroute Capacities," *Operations Research*, Vol. 46, No. 3, May-June 1998, pp. 406–422.

[7]Bayen, A. M., Grieder, P., Meyer, G., and Tomlin, C. J., "Lagrangian Delay Predictive Model for Sector-Based Air Traffic Flow," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 5, 2005, pp. 1015–1026.

[8]Lulli, G. and Odoni, A. R., "The European Air Traffic Flow Management Problem," *Transportation Science*, Vol. 41, No. 4, November 2007, pp. 431–443.

[9]Rios, J. and Ross, K., "Delay Optimization for Airspace Capacity Management with Runtime and Equity Considerations," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Hilton Head, South Carolina, August 2007.

[10]Megiddo, N., *Advances in Economic Theory*, chap. 6, Cambridge University Press, 1987.

[11]Bertsimas, D. and Tsitsiklis, J. N., *Introduction to Linear Optimization*, Athena Scientific, Belmont, Massachusetts, 1997.

[12]Hillier, F. S. and Lieberman, G. J., *Introduction to Operations Research*, McGraw-Hill, 8th ed., 2005.

[13]"Aircraft Situation Display To Industry: Functional Description and Interface Control Document," Tech. Rep. ASDI-FD-001, Volpe National Transportation Center, U.S. Department of Transportation, June 2005.

[14]Bilimoria, K., Sridhar, B., Chatterji, G., Sheth, K., and Grabbe, S., "FACET: Future ATM Concepts Evaluation Tool," *ATM2000*, Napoli, Italy, June 2000.

[15]Menon, P., Diaz, G., Vaddi, S., and Grabbe, S., "A Rapid-Prototyping Environment for En Route Air Traffic Management Research," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, CA, August 2005.

[16]Fourer, R., Gay, D., and Kernighan, B., "AMPL: A Mathematical Programming Language," 1989.

[17]ILOG, Inc., "ILOG CPLEX," http://www.ilog.com/products/cplex/, April 2007.

[18]Ivaldi, C. and Reynolds, D., "Upgrade and Technology Transfer of the San Francisco Marine Stratus Forecast System to the National Weather Service," *12th Conf. on Aviation, Range and Aerospace Meteorology*, Atlanta, GA, January 2006.

[19]Rios, J. and Ross, K., "Parallelization of the Traffic Flow Management Problem," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Honolulu, Hawaii, August 2008.