Optimal Traffic Flow Scheduling Using High-Performance Computing

Prasenjit Sengupta*, Jason Kwan†, and P. K. Menon‡
Optimal Synthesis Inc., Los Altos, CA 94022-2777

Algorithms for the end-to-end optimized scheduling of aircraft to enhance the efficiency of the National Airspace System are developed. For a given set of flights and desired departing schedules, routes are constructed and unimpeded 4-dimensional trajectories are simulated. These trajectories serve as an input to a linear programming based approach, and result in optimized schedules that are deconflicted while assuring adherence to the system capacity constraints. For a large number of flights the computational effort is formidable and optimization coupled with the Dantzig-Wolfe decomposition technique has been found to be a suitable approach. Techniques for accelerating the decomposition and solver on emerging high-performance computing hardware are discussed. Acceleration results are reported for realistic nationwide traffic flow problems using a multi-threaded CPU implementation and a novel implementation on General-Purpose Graphics Processing Units. A multi-threaded CPU implementation and a novel implementation on General-Purpose Graphics Processing Units show acceleration over a state-of-the-art, open-source, decomposition-based solver. Acceleration observed can be up to 9 × for replicated flights and 3 × for realistic nation -wide traffic flow optimization examples.

I. Introduction

Traffic Flow Management (TFM) for aircraft operations broadly refers to techniques for flow control of multiple aircraft in the airspace to prevent congestion and conflict, while ensuring on-time performance. Several techniques for TFM have been presented in the literature to address its optimization using heuristics

* Research Scientist, 95 First Street Suite 240, Senior Member AIAA.
† Research Engineer, 95 First Street Suite 240.
‡ President and Chief Scientist, 95 First Street Suite 240, Fellow AIAA.

or by numerical techniques such as dynamic programming and genetic algorithms [1-16]. A key advancement in optimization-based methods for TFM is the Bertsimas-Stock Patterson (BSP) model [17,18], which poses the TFM problem as a binary integer program (IP). The Bertsimas-Lulli-Odoni (BLO) model [19] and further developments by Agustin *et al.* [20] (the AAEP model) introduce additional flexibility in the BSP model such as the possibility of multiple routes and fairer distribution of delays among different flights. The BLO and AAEP models formulate the deterministic TFM problem by using a graph abstraction of the airspace with nodes connected by one or more routes for all flights. The nodes can either represent waypoints in the airspace jet routes, or entry and exit points for Sectors or Centers. The IP mechanism can also be extended to the stochastic TFM problem with airspace capacity uncertainty, as shown in [21,22].

The foregoing formulations result in a very large scale, sparse optimization problem. In other words, the variables and constraints in the problem are numerous but each constraint typically is active on only a small subset of the variables. It has been shown the IP has a strong linear program (LP) relaxation [17], allowing for the use of standard LP solvers such as the simplex method, while resulting almost always in binary solutions. Recent work [23,24] has shown that techniques such as Dantzig-Wolfe (DW) decomposition [25] can exploit the unique block matrix structure of the LP and solve the problem using parallel computation, and in the process significantly reduce the computation time even for problems composed of a large number of flights over a large geographical area. Tandale *et al.* [26] have demonstrated an implementation of the decomposition on Graphics Processing Units. The results in that work were based on preliminary simulations which replicated a small set of flights multiple times to generate larger problems.

The work described in this paper is a significant advancement of the approach described in [23], and its contributions are as follows. First, it is based on the BLO and AAEP formulations of the TFM problem which enable more general formulations. The most significant generalization is the modeling of multiple routes to enable weather-related rerouting. Second, since data from traffic files typically are provided for a single route, automated route generation and trajectory simulation for multiple routes of each flight are performed. This additional step allows for additional flexibility in TFM studies. Third, in contrast with the use of the GNU Linear Programming Kit (GLPK) [27] by the work in [23], the solver discussed in this paper is an implementation of a sparse, two-phase, revised bounded simplex solver [28] which can be executed on High-Performance Computers (HPCs) such as Graphics Processing Units (GPUs) as well as multi-threaded, multi-core CPUs, and offers significant advantages in terms of memory usage, over the regular simplex formulation. A detailed discussion of the potentially exponential memory requirement of the DW algorithm and the utility of the revised simplex formulation to address this requirement is provided in [29]. The use of GPUs is motivated by the high degree of parallelization possible on these devices.

Finally, a branch-and-cut algorithm, which can reduce the computation time required to solve binary integer programs of the type observed in the BLO formulation, is also utilized. This algorithm leverages the acceleration of the multithreaded implementation of the solver.

An outline of the architecture is shown in Figure 1. The inputs to the system are the flight demand data, which can be either Future ATM Concept Evaluation Tool (FACET) [30] TRX files or Airspace Concept Evaluation System (ACES) [31] FDS files. These files contain information on the filed flight plan and departure time for a flight. Additional inputs include Base of Aircraft Data (BADA) [32] and ambient wind data in the form of Rapid Update Cycle (RUC) or Rapid Refresh (RAP) files.

The flight plan data is then used in conjunction with the Coded Instrument Flight Procedures (CIFP) database which consists of waypoints and jet routes utilized by flights in the NAS. For every flight, one or more routes are constructed; the option of multiple routes is provided to allow flights to better manage their schedule especially when certain areas of the airspace are made unavailable due to inclement weather. The routes and the demand data are then used as inputs to a high-fidelity, high-speed, parallelized simulation tool. This tool generates the data sets utilized by the BLO model to formulate the IP for scheduling, which is then solved using the HPC-based DW decomposition and LP solver tool.

This paper is organized as follows. Section II describes the procedure for translating flight demand data and NAS structural data into multiple routes for flights. Section III describes the simulation process for generating spatio-temporal trajectories for each flight. This data is used to construct constraints for the BLO model, which is described in Section IV. Section V discusses the computational approach for the parallel implementation of the DW decomposition and simplex solver, and the computational experience associated with a realistic use case comprising NAS-wide TFM operations. Conclusions and directions for future research are given in Section VI.

II. Generation of Routes Using Demand Data and NAS Structure

The airspace is defined by waypoints and jet routes connecting these waypoints. Any flight from an origin airport to a destination airport in the NAS has to follow a route that is composed of many such airway segments. Also, the preferred route should be optimal with respect to considerations such as shortest path or shortest flight time in the presence of wind. The present work employs the A* search algorithm on the airway graph network to find the shortest path between a pair of origin-destination airports.The following steps are involved in the A* search for the shortest route:

1. Parsing the TRX file to extract the origin-destination airport pairs for all flights
2. Creation of the NAS airway/waypoint graph network
3. Extracting the airway and waypoint data from the CIFP (Coded Instrument Flight Procedures) formerly known as the National Flight Database (NFD)

4. Generating the airspace connectivity model
5. Implementation of the A* search
6. Generating the nominal routes between the origin-destination pairs in the absence of adverse weather.

A. Parsing the TRX file and CIFP Data

The TRX file is used to identify the airport origin-destination pairs that are used in a particular scenario. A TRX file record contains the following information:

TRACK DAL1598 B752 384800 1182000 484 370 0 ZOA ZOA33
FP_ROUTE **SFO**./.SAC131031..LVZ.LENDY5.**JFK**/1131

The data of interest are the origin and destination airport, shown in bold font in the example above. In this case, the origin airport is SFO and the destination airport is JFK. For this work, only the origin and destination airports are extracted; all other information in the TRX record is ignored. It is assumed that the first and last fixes in the FP_ROUTE line of a TRX record are the origin and destination airports. No validation is performed to determine if the first and last fix names correspond to airports or some other fix. The CIFP enroute airways data is used to create the airspace connectivity matrix and is available as an XML file with airway records. Each airway record contains a sequence of fixes which defines the route.

B. Airspace Connectivity and A* Search

The airspace connectivity matrix is formed by iterating over each airway in the global airways map. For each airway, the fix sequence is iterated over and a '1' is set in the connectivity matrix for each consecutive pair of fixes in the route. A general A* search algorithm [33-35] is used to find the shortest path between two nodes. The cost associated with each path can be proportional to the great-circle distance along a path, and the use of this cost function results in a shortest-distance route between two airports. If no wind profile is considered, the shortest-distance route and minimum time route are identical under the assumption of a constant airspeed in the enroute segment. Alternatively, by utilizing cruise airspeed information from BADA and wind forecast information from RUC, a wind-optimal search can be performed which results in a shortest-time route between two airports. An example of shortest-distance routes obtained using the framework described here is shown in Figure 2. It is worth noting that the A* search is one of many methods that can be used to generate routes. A comprehensive investigation of methods used in practice is beyond the scope of this paper. Even if they are not necessarily used in practice,

the routes obtained from the A* search are operationally feasible since they are constructed from the CIFP and are suited to the development of realistic scenarios in order to demonstrate optimized scheduling.

C. Generation of Shortest Paths around Adverse Weather

The A* search algorithm described above can be used to find shortest reroutes around adverse weather by performing the search on a modified graph. Assuming that the adverse weather is described by a polygon, all waypoints within the polygon and all airway segments intersecting the polygon can be removed. However, this method generates routes for all flights that skirt the adverse weather polygon closely creating congestion along the boundary of the polygon. In order to avoid congestion near the boundary, scaled versions of the adverse weather polygon can be used to generate multiple reroute options for every flight around the adverse weather as shown in Figure 3. Figure 4 shows the reroutes around the adverse weather polygons for all origin-destination airport pairs for all 35,000 flights in the TRX file for July 13, 2005. Note that the methodology handles multiple weather polygons and also both convex and non-convex polygons. In the latter case, waypoints lying outside the non-convex polygon but in the convex hull are not used because they will require an aircraft to turn back on its path. To avoid this operationally unrealistic scenario, non-convex weather polygons are replaced by their convex hull.

III. Generation of Nominal and Unimpeded 4D Trajectories

The present work utilized the Computational Appliance for Rapid Prediction of Aircraft Trajectories (CARPAT) software [36] for trajectory generation. CARPAT accepts flight demand data in the form of FACET [30] TRX or ACES [31] FDS files and can generate 4D trajectory predictions for 35,000 aircraft in the NAS over a 24-hour time horizon in less than 2.5 seconds. This is a significant acceleration over FACET-based simulations. CARPAT uses the ambient wind field from RUC and BADA [32] performance data for different aircraft types to generate accurate predictions. Multiple CARPAT trajectory predictions can be performed using minimum and maximum speeds from the BADA performance tables to obtain 1) earliest and latest entry times for every NAS resource and 2) minimum and maximum transit times for every resource. The trajectory predictor produces data required by the constraint generator, detailed in the next section. More specifically, an output of the predictor contains the earliest, latest, and nominal sector entry and exit times for each flight in the TRX file, which are obtained by simulating trajectories at the highest, lowest, and nominal groundspeed profiles, respectively. The outputs are grouped by alternative routes for each flight, and the groups are ordered by departure times. Within each group, the rows are ordered by the sequence in which the flight passes through each sector along its trajectory.

A significant advancement in this paper is the fact that due to the parallel nature of the unimpeded simulation over different flights, the data sets and constraints for each flight in the BLO model can also generated in parallel. Constraint generation for the optimization problem is discussed in the next section.

IV. Description of the BLO Integer Program and Constraint Construction

The BLO and AAEP models formulate the TFM problem in terms of the following optimization problem:

$$\begin{aligned} &\min c^{\top} x \\ &Ax \leq b \\ &x \in \{0,1\} \end{aligned} \tag{1}$$

Since the development of the optimization problem is largely the same in both [19] and [20], this paper uses the variable notation in the latter. The following sections describe the construction of the decision variables and the constraints.

A. Decision Variable and Data Sets

The variable of interest in IP formulation is denoted by $x_{f,XY}^{t}$, and is a binary variable, i.e. $x_{f,XY}^{t} \in \{0,1\}$. A value of 1 indicates that flight f (member of set $\mathcal{F}$), reaches Node Y from Node X, by time interval $t \in \mathcal{T}$ using an arc connecting the two nodes. Nodes X and Y belong to set $\mathcal{N}_f$ that is composed of all nodes on the route(s) of flight f. The sets $\mathcal{K}^d$ and $\mathcal{K}^a$ denote the set of nodes corresponding to departure and arrival airports, respectively. Since an airport in general is both a departure as well as an arrival airport, $\mathcal{K}^d \cap \mathcal{K}^a \neq \emptyset$. Let $k_f^d \in \mathcal{K}^d$ and $k_f^a \in \mathcal{K}^a$ denote departure and arrival airport nodes for flight f, respectively. Node $q\left(k_f^d\right)$ denotes the departure airport boundary, and $p\left(k_f^a\right)$ denotes the arrival airport boundary. The distinction between an airport node and its boundary node is noted in [20] and is used to construct an optimization problem which allows for a detailed model for ground holds and runway delays. For example, the difference between the actual and scheduled number of time units by which a flight arrives at the airport boundary is representative of the departure delay or amount of ground hold enforced on a flight.

The arc XY is a member of set $\mathcal{A}_f = \{XY | X, Y \in \mathcal{N}_f\}$ that is composed of all arcs on the route(s) of flight f. The set $\Gamma_f^+(X) = \{Y | XY \in A_f\}$ and $\Gamma_f^-(X) = \{Y | YX \in A_f\}$ are the set of nodes that have arcs from Node X and leading into Node Y respectively, for flight f.

Whereas sets $\mathcal{N}_f$ and $\mathcal{A}_f$ denote all possible nodes and all possible arcs for flight f, the sets $\mathcal{N}_f^* \subset \mathcal{N}_f$ and $\mathcal{A}_f^* \subset \mathcal{A}_f$ denote the nodes and arcs corresponding to the scheduled route of flight f. The variables $l_{f,XY}$, r_f, and d_f denote the travel time (number of time periods) for flight f over arc XY, the scheduled

departure time period, and the scheduled arrival time period, respectively. It is noted in [20] that $l_{f,k_f^d,q(k_f^d)} = l_{f,p(k_f^a),k_f^a} = 0, \forall k_f^d \in \mathcal{K}^d, k_f^a \in \mathcal{K}^a$. In other words, a flight reaches the departure airport boundary immediately after leaving the departure airport node, and a flight reaches the arrival airport node immediately after leaving the arrival airport boundary. It also follows that

$$r_f = d_f + \sum_{XY \in A_f^*} l_{f,XY} \tag{2}$$

In other words, the scheduled arrival time of the flight is given by the sum of the departure time and flight times along scheduled route segments. Furthermore, $\bar{l}_{f,XY}$ and $\underline{l}_{f,XY}$ denote the maximum and minimum number of time segments for flight f on arc XY.

The 0-1 BLO variables can be used to determine quantities of interest for TFM. For instance, the time segment in which the flight f reaches node n is denoted by $T_{f,n}$ and given by the following summation:

$$T_{f,n} = \sum_{X \in \Gamma_f^-(Y)} \sum_{t \in \mathcal{T}} t\left(x_{f,XY}^t - x_{f,XY}^{t-1}\right) \tag{3}$$

It follows from Eq. (3) that given the time of entry at a node and the time of entry at a preceding node, the number of time intervals required to travel on the arc connecting the nodes can be calculated. Additional quantities such as sector counts (given the arcs belonging to a sector) can also be calculated, as detailed in [19] and [20].

B. Constraint Formulation

The variables are linked with constraints resulting from the spatio-temporal representation of the graph. The so-called flight structure constraints define the continuity in time and space for a flight. The temporal continuity constraints [20] are represented by the following linear inequalities and equalities:

$$\begin{aligned} x_{f,XY}^{t-1} &\leq x_{f,XY}^t, \qquad t \in \mathcal{T}_{f,XY}^*, (X,Y) \in \mathcal{A}_f, f \in \mathcal{F} \\ x_{f,XY}^{t-1} &= x_{f,XY}^t, \qquad t \in \mathcal{T}_{f,XY} \setminus \mathcal{T}_{Xf,Y}^*, (X,Y) \in \mathcal{A}_f, f \in \mathcal{F} \end{aligned} \tag{4}$$

where $\mathcal{T}_{f,XY}^*$ is the set of feasible time units in which a flight f can reach Node Y from Node X over the arc connecting the two nodes, and $\mathcal{T}_{f,XY}$ is the smallest set of consecutive time intervals that contains $\mathcal{T}_{f,XY}^*$. These constraints state that if a flight was in node X by time period t, then this must also hold true for any later time period $t' > t$.

The spatial continuity constraints [20] are given by the following inequalities:

$$\sum_{Z\in\Gamma_f^+(Y)} x_{f,YZ}^{t+\underline{l}_{f,YZ}} \le \sum_{X\in\Gamma_f^-(Y)} x_{f,XY}^{t} \le \sum_{Z\in\Gamma_f^+(Y)} x_{f,YZ}^{t+\bar{l}_{f,YZ}}, \qquad t\in\mathcal{T}_{f,Y}, Y\in\mathcal{N}_f\backslash\{k_f^d,k_f^a\}, f\in\mathcal{F} \tag{5}$$

In the foregoing, $\mathcal{T}_{f,Y}$ denotes the set of all times units by which a flight f can reach Node Y from any other node along the route of that flight. Spatial continuity constraints force connectivity through a node.

The third set of constraints is composed of those that are derived from airspace capacity. To formulate the problem with capacity constraints, the sets $\mathcal{N}_f^{j+}$ and $\mathcal{N}_f^{j-}$ are defined for a flight f in the jth sector, as the set of nodes entering and leaving the jth Sector. The sector capacity constraints are given by the following [20]:

$$\sum_{f\in\mathcal{F}}\left[\sum_{Y\in\mathcal{N}_f^{j+}}\sum_{X\in\Gamma_f^-(Y)} x_{f,XY}^{t} - \sum_{Y\in\mathcal{N}_f^{j-}}\sum_{X\in\Gamma_f^-(Y)} x_{f,XY}^{t}\right] \le S_j^t, \qquad t\in\mathcal{T}, j\in\mathcal{J} \tag{6}$$

The foregoing equation counts the number of flights entering Sector j at time t, and subtracts from it, the number flights leaving the Sector at that time. This number is constrained to be less than the Sector capacity at that time, S_j^t, for a Sector $j\in\mathcal{J}$. Similar capacity constraints can be derived for airport arrival and departure capacity, but were not used in this work. Mechanisms to include arrival and departure capacity constraints are described in [17], which can also explicitly model scenarios where the arrival and departure capacity constraints are dependent on each other due to simultaneous operation on the same runways.

C. Cost Function Formulation

In the BLO model, the cost J has contributions from different components, depending on the modeling requirements of the problem. A comprehensive list is presented in [20], which not only includes the components presented in [19], but also introduces additional terms for greater flexibility in formulating TFM problems. In this work, the number of cancelled flights, overall flight ground delays, and airborne delays were penalized. These three cost function components, denoted by J_{cancel}, J_{ground}, and J_{airborne}, are listed as follows:

$$J_{\text{cancel}} = -\sum_{f\in\mathcal{F}} x_{f,k_f^d,q\left(k_f^d\right)}^{t}, \qquad t=\max\mathcal{T}_{f,k_f^d,q\left(k_f^d\right)} \tag{7}$$

$$J_{\text{ground}} = \sum_{f\in\mathcal{F}}\sum_{t\in\mathcal{T}_{f,k_f^d,q\left(k_f^d\right)}} c_{f,G}(t)\left(x_{f,k_f^d,q\left(k_f^d\right)}^{t} - x_{f,k_f^d,q\left(k_f^d\right)}^{t-1}\right) \tag{8}$$

$$J_{\text{airborne}} = \sum_{f \in \mathcal{F}} \left[\sum_{t \in \mathcal{T}_{f,p\left(k_f^a\right),k_f^a}} c_{f,T}(t) \left(x^{t}_{f,p\left(k_f^a\right),k_f^a} - x^{t-1}_{f,p\left(k_f^a\right),k_f^a} \right) \right. \\ \left. - \sum_{t \in \mathcal{T}_{f,k_f^d,q\left(k_f^d\right)}} c_{f,G}(t) \left(x^{t}_{f,k_f^d,q\left(k_f^d\right)} - x^{t-1}_{f,k_f^d,q\left(k_f^d\right)} \right) \right] \tag{9}$$

where $c_{f,T}(t) = c_T \cdot \left(t - r_f\right)$ and $c_{f,G}(t) = c_G \cdot \left(t - d_f\right)$ (with constant c_T and c_G) are coefficients such that each additional unit of delay from scheduled arrival and departure has a proportionately heavier penalty. Alternative formulations include the so-called superlinear cost function [19] with additional penalty on larger delays. Computational experiments described in [19] have shown that this results in a more equitable distribution of delays over the set of flights. It should be noted that although the cost function coefficients can be functions of the time unit, they are not functions of the decision variables, and consequently, the resulting cost function is still linear as shown in Eq. (1).

V. Solution using Parallelized Dantzig-Wolfe Decomposition

Prior work described in [23] shows that the BSP model exhibits the so-called primal block structure [37], which is shown Figure 5. This structure is also exhibited by the BLO model and the constitution of the blocks is discussed in Section V.A. The implementation details on parallel and high-performance computers are discussed in Section V.B. Nationwide TFM examples and computational experience for these problems are discussed in Section V.C.

A. Dantzig-Wolfe Structure

With reference to Figure 5, the constraints of the LP can be divided into master problem blocks D_1 through D_n, and sub-problem blocks F_1 through F_n, where n is the number of flights in the simulation. In other words, the LP can be written in the following form:

$$\begin{gathered} \min c_1^\top x_1 + c_2^\top x_2 + \cdots c_n^\top x_n \\ D_1 x_1 + D_2 x_2 + \cdots + D_n x_n = b_D \\ F_1 x_1 = b_{F_1} \\ F_2 x_2 = b_{F_2} \\ \vdots \\ F_n x_n = b_{F_n} \\ x_1, x_2, \ldots, x_n \geq 0 \end{gathered} \tag{10}$$

where the variable set x_f consist of all variables $x_{f,XY}^t$ for a flight $f \in \mathcal{F}$ in the simulation. The master problem constraints composed of block matrices D_1 through D_n consists of constraints that relate the variables of multiple flights, and are composed of capacity constraints shown in Eq. (6). The sub-problem constraints composed of block matrices F_1 through F_n consist of spatio-temporal constraints of individual flights and only relate variables associated with a single flight each, as shown in Eqs. (4) and (5). Note that in the foregoing equation the general LP form is assumed for the optimization problem, in which all constraints are equality constraints and inequalities are converted to equalities using slack, surplus, and artificial variables.

Let $P_f = \{x_f | F_f x_f = b_f, x_f \geq 0\}$. This set denotes the feasible values for the variables for the fth flight and can be rewritten as a convex combination of its extreme points and rays. In the context of the BLO formulation, the feasible region is bounded since all variables are bounded in the region [0,1], and as a consequence, P_f can be expressed as a convex combination of extreme points only. Let the extreme points of P_f be denoted by x_f^j, where $j \in \mathcal{J}_f$ and $\mathcal{J}_f$ is the set of indices iterating over the extreme points. The monolithic problem shown in Eq. (10) is rewritten as the so-called master program:

$$\begin{aligned} &\min \sum_{f\in\mathcal{F}} \sum_{j\in\mathcal{J}_f} \lambda_f^j c_f^\top x_f^j \\ &\sum_{f\in\mathcal{F}} \sum_{j\in\mathcal{J}_f} \lambda_f^j D_f x_f^j = b_D \\ &\sum_{j\in\mathcal{J}_f} \lambda_f^j = 1, \qquad \forall f \in \mathcal{F} \\ &\lambda_f^j \geq 0, \qquad \forall j \in \mathcal{J}, f \in \mathcal{F} \end{aligned} \tag{11}$$

where λ_f^j are decision variables.

Let the number of rows in the master problem blocks $D_{1,\dots n}$ be denoted by m_0 and the number of rows in each sub-problem block be denoted by $m_{1,\dots,n}$. Then, the number of constraints in the monolithic form of the problem shown in Eq. (10) is equal to $m_0 + \sum_{f\in\mathcal{F}} m_f$, whereas the master program shown in Eq. (11) consists of $m_0 + n$ constraints. The BLO formulation constraints are such that m_0 is significantly smaller than the number of monolithic constraints; however this is achieved by using an exponentially larger number of variables which correspond to the extreme points of all the sub-problems.

The advantage of the DW decomposition is that in spite of a large number of variables, at any given simplex iteration of the master program, a vast majority of the variables λ_f^j are zero and their corresponding columns in the master program simplex tableau are not used. By utilizing a process called delayed column

generation, only potentially useful columns are added to the master program iteration. Details of the algorithm can be found in [25,29] and are summarized here.

Dual variables are associated with the master problem constraints, of which m_0 dual variables, denoted by σ, correspond to the capacity constraints, and n dual variables, denoted by $\pi_{1,\ldots,n}$, correspond to the convexity constraints for the n sub-problems. In the revised simplex tableau, the dual variables are obtained directly from the row of reduced costs. The fth sub-problem consists of the LP $\min \left(c_f^\top - \sigma^\top D_f\right) x_f, x_f \in P_f$. If the optimal cost to this problem is less than π_f, then the optimal solution is an extreme point, and a column $\left[\left(D_f x_f^j\right)^\top \; e_f^\top\right]^\top$ is generated, where e_f is a vector of length n consisting of zeros everywhere except for the fth entry, which is equal to 1. If the optimal cost is no less than π_f, no column is generated. Since the BLO variables are bounded, the optimal cost is always finite. The master problem reaches optimality when no columns are generated by any sub-problem.

Of importance is the fact that the fth sub-problem consists only of constraints given by the matrix F_f, and the solution to one sub-problem does not depend on the solutions to the other sub-problems. This motivates the use of parallel implementations of the DW decomposition.

B. Parallel Implementation on Multithreaded, Multicore, and High-Performance Computers

Parallelized implementation of the DW decomposition is achieved at two levels. In the first, "coarse" level of parallelization, each sub-problem is solved independently. A simplex tableau is created for every sub-problem and the pivoting operations on each tableau are executed in parallel. Constraint elimination is performed simultaneously with constraint generation for each sub-problem, in order to reduce the problem size. Single-variable inequalities and equalities are converted to variable bounds and fixed variables, respectively. Fixed variables are then removed from constraints. If a master problem constraint is such that it only involves variables from a single sub-problem, it is assigned to the corresponding sub-problem block matrix. Finally, redundant capacity constraints are removed by utilizing variable bounds, since all variables are bounded. The foregoing steps are repeated until the number of constraints does not change.

The process of sub-problem tableau generation is performed in parallel since the construction steps for one sub-problem do not require access to computer memory containing information of the other sub-problems. Parallel thread management on CPUs is performed using the Open Multi-Processing (OpenMP) framework, which uses compiler directives to assign a separate thread to the optimization of each sub-problem. On GPUs, parallelized code is developed using the Compute Unified Device Architecture (CUDA) framework, and parallel processes are invoked using CUDA 'kernels'. The CUDA compiler launches a separate kernel thread for each sub-problem in parallel.

The number of cores offered by GPUs is significantly larger; however they typically have a lower clock speed than CPUs. For example, the NVIDIA™ Titan GPU offers 2688 cores operating at 732 MHz, whereas an Intel® Xeon® processor family typically offers 2 cores with 12 concurrent threads each, operating at 2.4 GHz.

The second, "fine" level of parallelization is achieved at the simplex iteration stage. As noted before, the research in this paper utilized a two-phase, bounded, revised simplex algorithm. Details of the simplex method are given in [28]; it is sufficient to note here that the two-phase solver is aimed at addressing problems where an initial basic feasible solution from which simplex iterations are initiated, may not exist. For example, if the problem consists of constraints where $A_g x \geq b_g > 0$, then $x = 0$ is no longer an initial basic feasible solution, and the first phase of the solver identifies such a solution using artificial variables, and the second phase of the solver iterates on the basic feasible solution to achieve an optimal solution. The bounded simplex method explicitly models variable bounds, and does not pose them as constraints. This is useful from the perspective of the BLO model because every variable in the LP relaxation is bounded between 0 and 1, and adding these bounds would increase the number of constraints by the number of variables in the problem. The revised simplex algorithm differs from the regular simplex algorithm in that the tableau only consists of columns from the current basis at any iteration. While this can require additional computations at every iteration to generate the column corresponding to leaving variables, the memory requirement is significantly reduced. This approach is useful when the number of variables in the problem is significantly larger than the number of constraints; in this case the simplex method only requires storage for a square tableau matrix whose dimension is equal to the number of constraints.

The use of the revised simplex algorithm is motivated by the fact that the master program defined in Eq. (11) has an exponentially large number of variables but a significantly smaller set of constraints in comparison with the monolithic problem. Although the simplex algorithm is sequential in nature, fine-grain parallelization is used to perform simultaneous pivoting of all rows and the calculation of the reduced costs, within each sub-problem. Fine grain parallelization generally cannot be implemented in the CPU architecture but the CUDA framework on GPUs allows for the launch of sub-kernels from kernels, exploiting the large number of cores.

C. Computational Experience for NAS-Wide TFM Problems

The foregoing sections describe the different steps utilized to transcribe flight demand data into an optimization problem for the solver. This section describes the application of this approach to two use cases and discusses the computational experience for both problems. It should be noted at the outset that in both examples, the LP solution was exactly integral. Strength of the LP relaxation for very large problems has been noted in [23].

East Coast Flights

This example consists all flights arriving and departing in the eastern region of the continental United States in the 24 hour period on July 30, 2005. There are 4,366 flights in this simulation, and at a resolution of 3 minutes the resulting monolithic problem has 976,203 variables and 1,292,646 constraints. Of this, the master problem blocks are composed of 15,209 constraints, and the sub-problems consist of 224 variables and 293 constraints on an average. When decomposed, the variables and constraints for each flight constitute a separate sub-problem. Sector Monitor Alert Parameter (MAP) values were chosen as capacity constraints in this example, and the maximum amount of delay allowed by any flight was 25 minutes.

The problem described here is significantly larger than those described in [23] or later works demonstrating GPU-based decomposition [26], namely in terms of the number of master problem constraints. An attempt was made to reduce the number of constraints by removing inactive constraints but this approach does not work well for diverse schedules over a large time frame. In order to ensure that there is only one optimal solution, flights departing from and arriving at the same airport were assigned priorities based on their desired departure time, which was then used to scale their respective cost function coefficients.

The execution time on an Intel Xeon processor with 24 maximum concurrent threads at 2.4 GHz, and an NVIDIA Titan GPU with 2,688 cores at a clock speed of 732MHz is shown in Figure 6. A comparison is made with the wall clock time of dwsolver [23]; which also utilizes multithreading together with GLPK. Results are also verified against those obtained from dwsolver. No capacity constraints were found to be active due to the restricted geographical area, and the computation time is therefore equivalent to the amount of time required to optimize the sub-problems. A comparison with a monolithic problem was not pursued due to the infeasible time and memory requirements of the latter problem. Comparisons with restricted-license optimization software were also not performed but such comparisons are shown in [23].

Figure 6 shows that an OpenMP implementation of the multithreaded decomposition problem performs significantly better than the GPU implementation. It also shows an approximately $3\times$ acceleration over dwsolver. The fastest execution time for a problem of this size is 15 seconds.

Parallelization on GPUs, both at the coarser level of sub-problem assignment and at the finer level of pivoting operations, produces best results when the operation is data parallel as well as task parallel. While the TFM problem is task-parallelizable at a coarse level due to sub-problem independence, the only circumstance under which it is also task-parallelizable and data-parallelizable at a finer level is when every thread requires the same amount of memory and the same number of computational steps. The flights in a realistic example are all of different durations and lengths due to the diversity of flight lengths in the NAS. Consequently, the number of variables and constraints utilized to model a flight's trajectory can show significant variance. Not only does this result in each sub-problem simplex tableau requiring different

amounts of memory, each sub-problem also requires a different number of steps to reach optimality and to generate the column for the entering variables in the master program. Therefore, the GPU is not able to leverage parallelization at a level beyond that of separate sub-problem assignment and its overall performance is impacted by the relatively slower clock speed in comparison with CPUs. Only in the special case of identical flights does the GPU perform significantly better than a multithreaded CPU. In this case, acceleration of up to 9× was observed using the GPU-based solver, and for smaller problems larger acceleration values have been reported [26]. The fact that realistic problems, including the ones presented here, do not adhere to this ideal is also exhibited by the flattening of the execution time curve as a function of the number of concurrent threads in Figure 6. Although replicated flight examples are not expected in the real world, they can appear as a result of experiments which require stochastic analysis using Monte Carlo simulations. The GPU -based solver is beneficial for these problems.

Metro Flights

This example consists of all flights arriving and departing into one of the 40 major airports in the NAS. As shown in the previous example, a discrete interval of 3 minutes was chosen over a 24 hour period on July 30, 2005. This example consists of 8,522 flights or sub-problems. The total number of variables resulting from the BLO formulation is 2,149,850, and the total number of constrains is 2,832,558. Of these, the master problem consists of 34,668 constraints. On an average, each sub-problem consists of 252 variables and 328 constraints, which is of the same order as the sub-problems in the previous example. However, the number of sector capacity constraints is more than double the number of constraints modeled in the East Coast example and indicates the larger number of sectors in which capacity is enforced. This problem consists of 907 unique sectors in which capacity constraints (MAP values) are enforced, and constitutes a significant portion of the NAS. As an indicator of the size of the problem, the frequency of the number of flights which have a given number of links is shown in Figure 7. In this example, the route of a flight is discretized such that a link connects an entry and exit node at a sector along the route of a flight. Arbitrary weather polygons as shown in Figure 4 were introduced in this example, and as a consequence, some flights have up to 3 alternative routes.

Optimization results show that approximately 2% of the flights were delayed by 2 time units and less than 1% of the flights were delayed by 3 time units. Since airborne delays are made to incur a larger penalty than ground delays, all the delays observed appeared on the ground. The appearance of delays shows that some of the capacity constraints are active, which is also indicated by a small number of master problem iterations. These do not contribute significantly to the computation time. Execution time for this problem as a function of the number of threads is shown in Figure 8. The best-case performance is observed to be 28 seconds, which also represents a speedup of approximately 3 × over dwsolver. In this case too, the multithreaded CPU implementation is a significant improvement over the GPU implementation due to the

causes mentioned in the previous example. As noted in Figure 7, the distribution of sub-problem size shows a relatively large variance in comparison with the mean. For geographically-restricted problems, the variance may be smaller and as a consequence, each sub-problem may be of similar size, but the number of steps required for the simplex algorithm to reach an optimal solution to the sub-problems can be different.

It is worth noting that the revised simplex formulation pursued in this paper required approximately 11% of the memory utilized by the GLPK-based DW decomposition (5GB in comparison with 45GB). Larger examples could not be tested due to a 6GB limit on the NVIDIA Titan card used in this research.

VI. Conclusions

This paper describes an end-to-end system for TFM optimization, inputs to which are the nominal flight schedules in standard format (e.g. TRX files), and the outputs are the optimized schedules which account for airspace capacity constraints. The approach considers the possibility of multiple routes due to weather-avoidance strategies and shows acceleration over the state-of-the-art solutions. The acceleration can be as high as 9× that of dwsolver when the sub-problems in the DW decomposition are data-parallel, i.e. all sub-problems require the same amount of memory and execute the same number of simplex iterations. However, in real-world examples in which the number of variables and constraints can vary over the different flights in the simulation, the OpenMP-based multi-threaded implementation on CPUs is significantly faster than GPU implementation. Performance is also impacted by the large number of master problem constraints when considering realistic, NAS-wide problems. In this case, the CPU implementation shows an acceleration of 3× over dwsolver whereas the GPU implementation is slower than dwsolver. It should also be noted that in a NAS-wide example consisting of 8,522 flights from the 40 major US airports, the approach presented in this paper requires approximately 11% of the memory required by dwsolver. In the best case the nationwide, 40 major-airport problem for a 24-hour schedule at a resolution of 3 minutes, was solved in 28 seconds.

Research has yielded considerable insight in the parallel nature of LP solvers and other optimization methods. For instance, the use of a system with multiple GPUs to surmount memory limitations and to address larger problems is an avenue of future research. While the simplex method is well-suited to the Dantzig-Wolfe decomposition, there is evidence in the literature supporting the use of Interior Point methods. The latter category may offer certain advantages over simplex. For example, it was observed that the GPU implementation is impacted when the sub-problems do not execute the same number of steps. While the simplex can be terminated at a fixed number of steps, this approach is generally arbitrary and provides no guarantees on the optimality of the solution. On the other hand, interior point methods can provide an estimate of the duality gap which is an indicator of solution optimality when calculations are terminated after a fixed number of iterations. Assessing the use of other types of optimization techniques is

therefore a useful exercise. The generic nature of the solver and decomposition itself lends itself to a large variety of schedule optimization problems in different industries. More specifically, the approach shows promise for the integrated arrival and departure scheduling problem in the terminal area and airport surface. The computation times observed in the examples suggest guidelines for selecting the time horizon and resolution for applications in this domain.

Acknowledgments

This research was supported by NASA Contract No. NNX12CA05C, with Dr. Joseph L. Rios of NASA Ames Research Center serving as the Technical Monitor.

References

[1] Andreatta, G. and Romanin-Jacur, G., "Aircraft Flow Management Under Congestion," *Transportation Science*, Vol. 21, No. 4, Nov. 1987, pp. 249—253.
doi: 10.1287/trsc.21.4.249

[2] Helme, M. P., "Reducing Air Traffic Delay in a Space-Time Network," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Chicago, IL, Vol. 1, Oct. 1992, pp. 236—242.
doi: 10.1109/ICSMC.1992.271770

[3] Terrab, M. and Odoni, A. R., "Strategic Flow Management for Air Traffic Control," *Operations Research*, Vol. 41, No. 1, Jan.-Feb. 1993, pp. 138—152.
doi: 10.1287/opre.41.1.138

[4] Richetta, O. and Odoni, A. R., "Solving Optimally the Static Ground-Holding Policy Problem in Air Traffic Control," *Operations Research Society of America*, Vol. 27, No. 3, Aug. 1993, pp. 228—238.
doi: 10.1287/trsc.27.3.228

[5] Vranas, P. B., Bertsimas, D. J., and Odoni, A. R., "The Multi-Airport Ground-Holding Problem in Air Traffic Control," *Operations Research*, Vol. 42, No. 2, Mar.-Apr 1994, pp. 249—261.
doi: 10.1287/opre.42.2.249

[6] Richetta, O. and Odoni, A. R., "Dynamic Solution to the Ground-Holding Problem in Air Traffic Control," *Transportation Research Part A: Policy and Practice*, Vol. 28, No. 3, May 1994, pp. 167—185.
doi: 10.1016/0965-8564(94)90015-9

[7] Vranas, P. B., Bertsimas, D. J., and Odoni, A. R., "Dynamic Ground-Holding Policies for a Network of Airports," *Transportation Science*, Vol. 28, No. 4, Nov. 1994, pp. 275—291.
doi: 10.1287/trsc.28.4.275

[8] Rikfin, R. M., *The Single Airport Static Stochastic Ground Holding Problem*, Master's Thesis, Massachusetts Institute of Technology, 1994.

[9] Hoffman, R. L., *Integer Programming Models for Ground-Holding in Air Traffic Flow Management*, Ph.D. Dissertation, University of Maryland at College Park, 1997.

[10] Navazio, L. and Romanin-Jacur, G., "The Multiple Connections Multi-Airport Ground-Holding Problem: Models and Algorithms," *Transportation Science*, Vol. 32, No. 3, Aug. 1998, pp. 268—276.
doi: 10.1287/trsc.32.3.268

[11] Grabbe, S., Sridhar, B., and Mukherjee, A., "Central East Pacific Flight Scheduling," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Hilton Head, SC, Paper AIAA 2007-6447, Aug. 2007.
doi: 10.2514/6.2007-6447

[12] Mukherjee, A. and Hansen, M., "A Dynamic Stochastic Model for the Single Airport Ground Holding Problem," *Transportation Science*, Vol. 41, No. 4, Nov. 2007, pp. 444—456.
doi: 10.1287/trsc.1070.0210

[13] Liu, P.-C. B., Hansen, M., and Mukherjee, A., "Scenario-Based Air Traffic Flow Management: From Theory to Practice," *Transportation Research Part B: Methodological*, Vol. 42, Nos. 7-8, Aug. 2008, pp. 685—702.
doi: 10.1016/j.trb.2008.01.002

[14] Sridhar, B., Grabbe, S., and Mukherjee, A., "Modeling and Optimization in Traffic Flow Management," *Proceedings of the IEEE*, Vol. 96, No. 12, Dec. 2008, pp. 2060—2080.
doi: 10.1109/JPROC.2008.2006141

[15] Mukherjee, A. and Hansen, M., "A Dynamic Rerouting Model for Air Traffic Flow Management," *Transportation Research Part B: Methodological*, Vol. 43, No. 1, Jan. 2009, pp. 159—171.
doi: 10.1016/j.trb.2008.05.011

[16] Xue, M., and Zelinski, S., "Optimal Integration of Departures and Arrivals in Terminal Airspace," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 1, Jan.-Feb. 2014, pp. 207—213.
doi: 10.2514/1.60489

[17] Bertsimas, D. and Patterson, S. S., "The Air Traffic Flow Management Problem with Enroute Capacities," *Operations Research*, Vol. 46, No. 3, May-Jun. 1998, pp. 406—422.
doi: 10.1287/opre.46.3.406

[18] Bertsimas, D. and Patterson, S. S., "The Traffic Flow Management Rerouting Problem in Air Traffic Control: A Dynamic Network Flow Approach," *Transportation Science*, Vol. 34, No. 3, Aug. 2000, pp. 239—255.

doi: 10.1287/trsc.34.3.239.12300

[19] Bertsimas, D., Lulli, G., and Odoni, A., "An Integer Optimization Approach to Large-Scale Air Traffic Flow Management," *Operations Research*, Vol. 59, No. 1, Jan.-Feb. 2011, pp. 211—227.
doi: 10.1287/opre.1100.0899

[20] Agustin, A., Alonso-Ayuso, A., Escudero, L. F., and Pizarro, C., "On Air Traffic Flow Management with Rerouting. Part I: Deterministic Case," *European Journal of Operational Research*, Vol. 219, No. 1, May 2012, pp. 156—166.
doi: 10.1016/j.ejor.2011.12.021

[21] Agustin, A., Alonso-Ayuso, A., Escudero, L. F., and Pizarro, C., "On Air Traffic Flow Management with Rerouting. Part II: Stochastic Case," *European Journal of Operational Research*, Vol. 219, No. 1, May 2012, pp. 167—177.
doi: 10.1016/j.ejor.2011.12.032

[22] Sengupta, P., Tandale, M. D., and Menon, P. K., "Risk-Hedged Traffic Flow Management under Airspace Capacity Uncertainties," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 5, Sep.-Oct. 2014, pp. 1487—1500.
doi: 10.2514/1.G000412

[23] Rios, J., and Ross, K., "Massively Parallel Dantzig-Wolfe Decomposition Applied to Traffic Flow Scheduling," *Journal of Aerospace Computing, Information, and Communication*, Vol. 7, No. 1, Jan. 2010, pp. 32—45.
doi: 10.2514/1.45606

[24] Wei, P., Cao, Y., and Sun, D., "Total Unimodularity and Decomposition Method for Large-Scale Air Traffic Cell Transmission Model," *Transportation Research Part B: Methodological*, Vol. 53, Jul. 2013, pp. 1—16.
doi: 10.1016/j.trb.2013.03.004

[25] Dantzig G B., and Wolfe P., "Decomposition Principle for Linear Programs," *Operations Research*, Vol. 8, No. 1, Jan.-Feb. 1960, pp. 101—111.
doi: 10.1287/opre.8.1.101

[26] Tandale, M. D., Wiraatmadja, S., Vaddi, V. V., and Rios, J. L., "Massively Parallel Optimal Solutions to the Nationwide Traffic Flow Management Problem," *AIAA Aviation Technology, Integration, and Operations Conference*, Los Angeles, CA, Paper AIAA 2013-4349, Aug. 2013.
doi: 10.2514/6.2013-4349

[27] Makhorin, A., "GNU Linear Programming Kit, Version 4.34," http://www.gnu.org/software/glpk/glpk.html.

[28] Maros, I., *Computational Techniques of the Simplex Method*, Kluwer Academic Publishers, The Netherlands, 2003.

[29] Bertsimas, D., and Tsitsiklis, J. N., *Introduction to Linear Optimization*, Athena Scientific, Belmont, MA, 1997, Chap. 6.

[30] Bilimoria, K. D., Sridhar, B., Chatterji, G. B., Sheth, G., and Grabbe, S., "FACET: Future ATM Concepts Evaluation Tool," *3rd USA/Europe Air Traffic Management R&D Seminar*, Naples, Italy, June 2000.

[31] Raytheon ATMSDI Team, "Airspace Concept Evaluation System Build 2 Software User Manual," *NASA Ames Research Center*, Moffett Field, CA, November 2003.

[32] Base of Aircraft DAta (BADA) http://www.eurocontrol.int/eec/public/standard_page/proj_BADA.html

[33] Wikipedia Page on the A* search algorithm: http://en.wikipedia.org/wiki/A*_search_algorithm

[34] Russell, S., Norvig, P., "Artificial Intelligence: A Modern Approach," Ed. 3, *Prentice Hall*, 2009

[35] Heineman, G. T., Pollice, G., Selkow, S., "Algorithms in a Nutshell: A Desktop Quick Reference," *O'Reilly Media Inc.*, 2009.

[36] Tandale, M. D., Wiraatmadja, S., Menon, P. K., and Rios, J. L., "High-Speed Prediction of Air Traffic for Real-Time Decision Support," *AIAA Guidance Navigation and Control Conference*, Portland OR, 8-11 August, 2011.

[37] Tebboth, J. R., *A Computational Study of the Dantzig-Wolfe Decomposition*, Ph.D. Dissertation, University of Buckingham, United Kingdom, 2001.

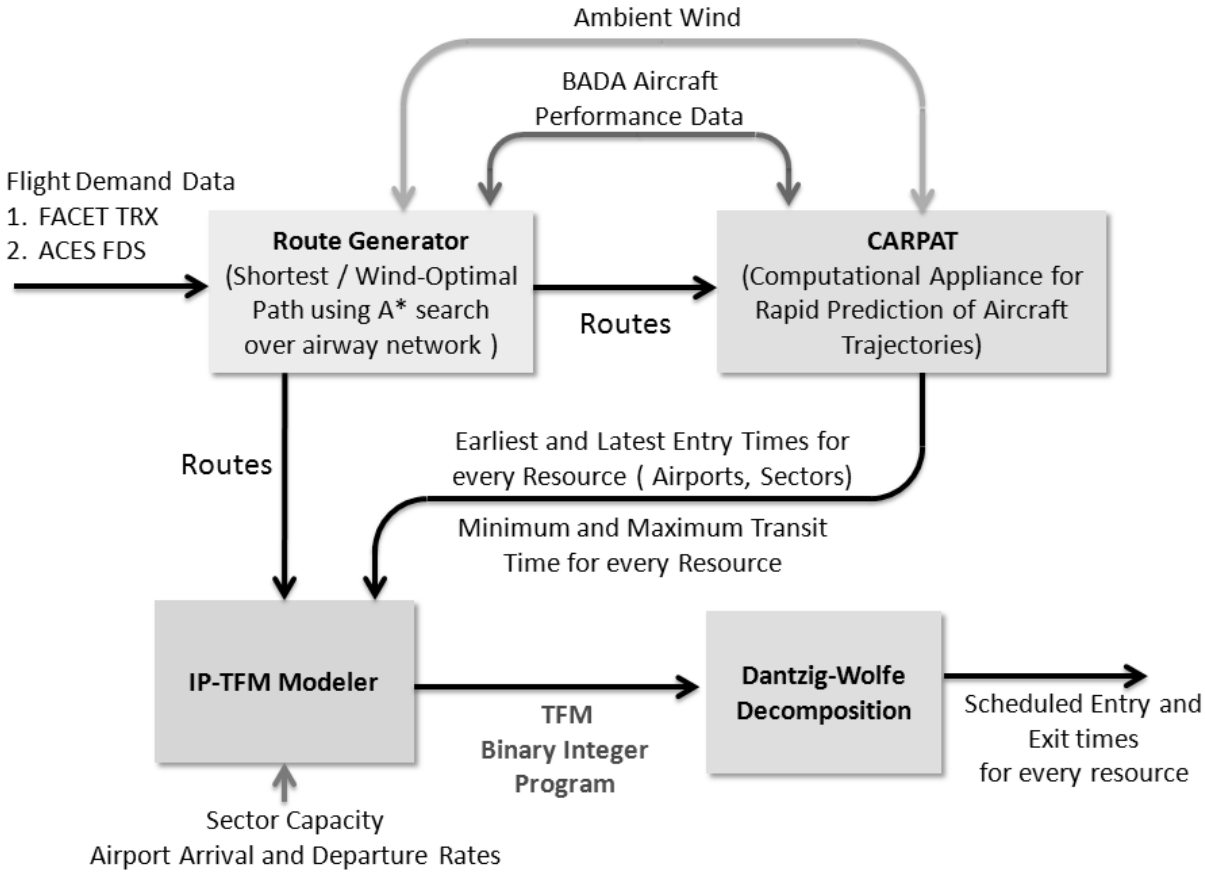

Figure 1. Conceptual Overview of the Traffic Flow Optimization Tool

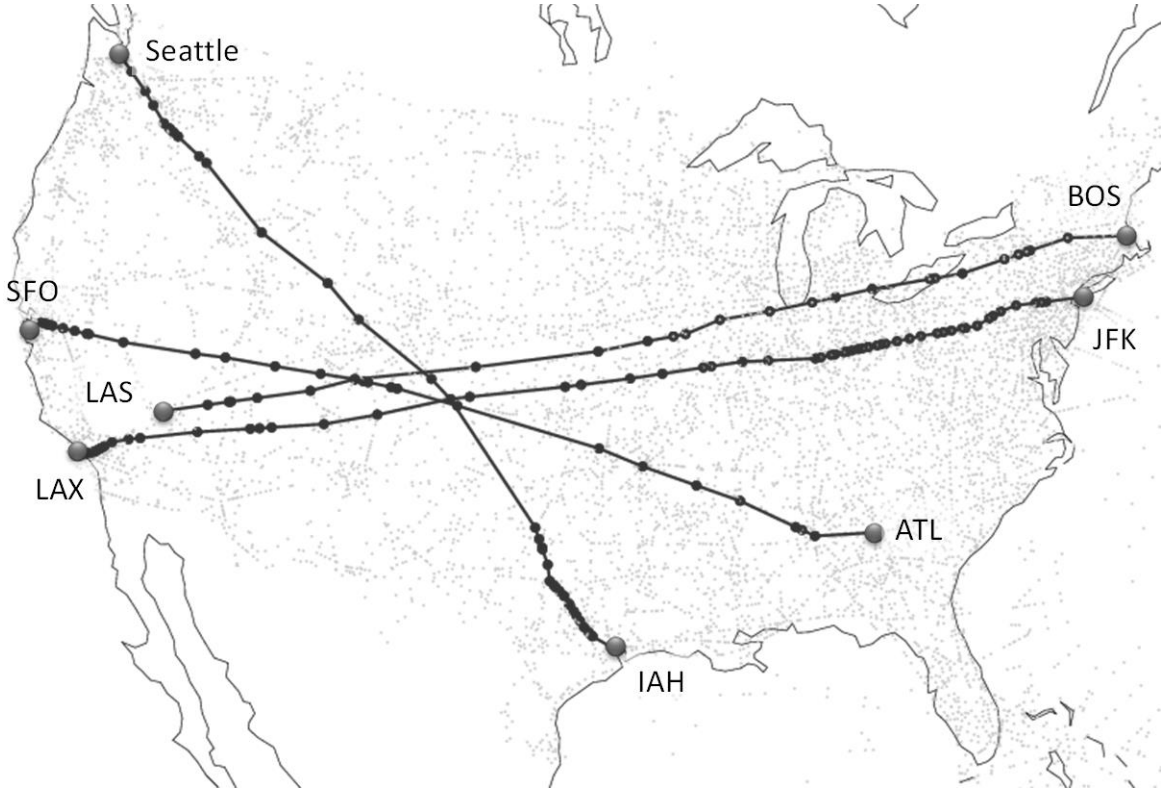

Figure 2. Shortest Routes between Origin-Destination Airport Pairs Generated by the A* Search

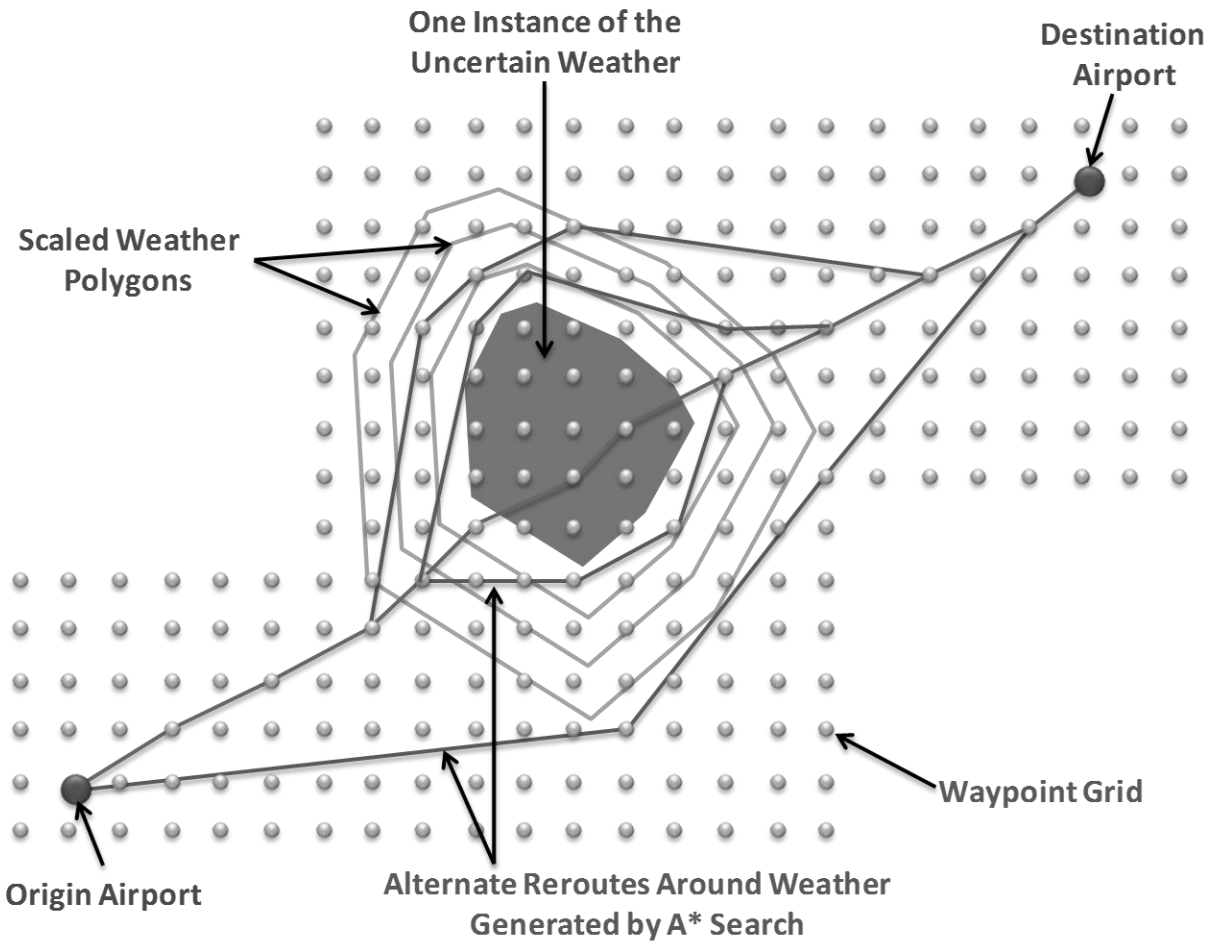

Figure 3. Generation of Alternate Reroutes Given the Adverse Weather Polygon

Figure 4. Reroutes Around Multiple Adverse Weather Polygons in the NAS

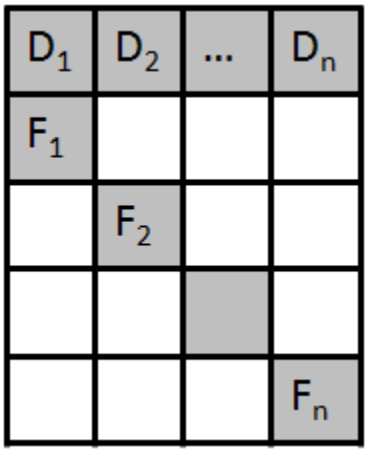

Figure 5. Primal Block Angular Structure of the TFM Constraint Matrix

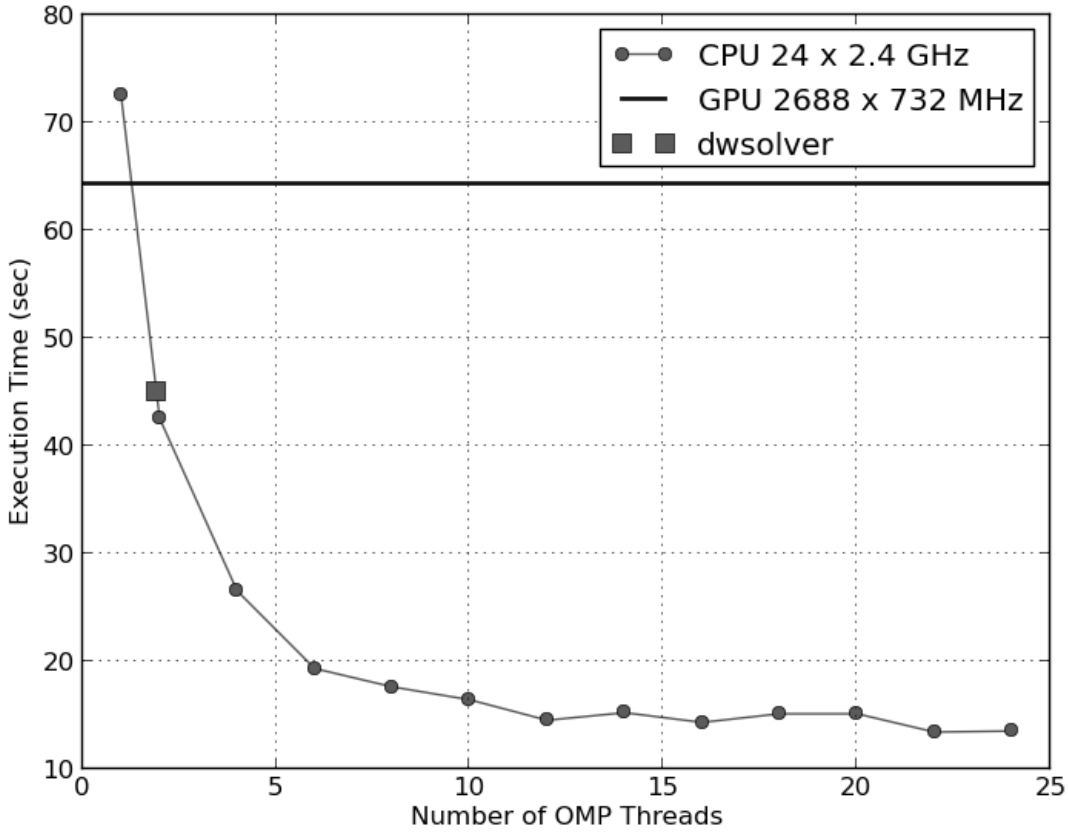

Figure 6. Execution Time Trend as a Function of OpenMP Threads for The East Coast Example

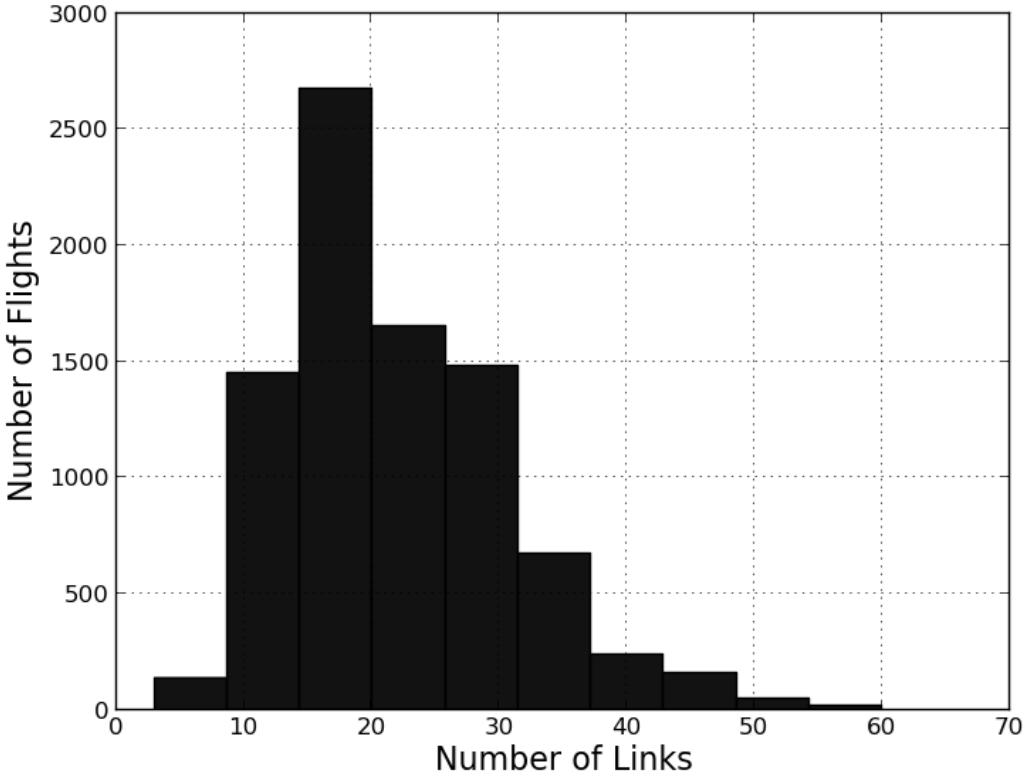

Figure 7. Distribution of the Number of Links Constituting a Flight's Path in the Metro Airport TFM Example

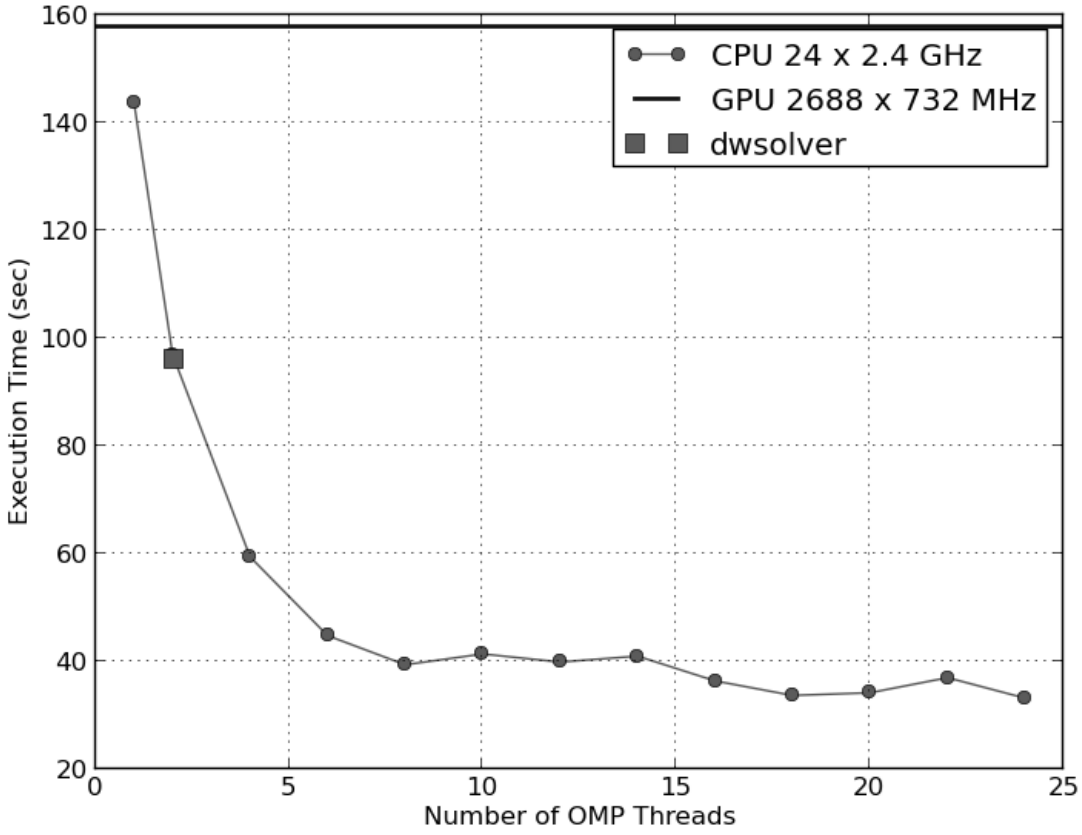

Figure 8. Execution Time Trend as a Function of OpenMP Threads for The 40-Airport Example