ABSTRACT

Title of Dissertation: IDENTIFICATION OF AIR TRAFFIC FLOW SEGMENTS VIA INCREMENTAL DETERMINISTIC ANNEALING CLUSTERING

Alex Thinh Nguyen,
Doctor of Philosophy, 2012

Directed By: Professor John S. Baras
Department of Electrical and Computer Engineering

Many of the traffic management decisions and initiatives in air traffic are based on “flows” of traffic in the National Airspace System (NAS), but the actual identification of the location and time of the flow segments are often left to interpretation based on observations of traffic data points over time. Having an automated method of identifying major flow segments can help to target traffic management initiatives, evaluate design of airspace, and enable actions to be taken on the collection of flights in a flow segment rather than on the flights individually.

A novel approach is developed to identify the major flow segments of air traffic in the NAS that consists of a robust method for partitioning 4-dimensional traffic trajectories into a series of great circle segments, and clustering the segments using an Agglomerate Deterministic Annealing clustering algorithm. In addition, a very efficient algorithm to incrementally cluster the segments is developed that takes into account the spatial and temporal properties of the segments, and makes the method very suitable for real-time applications. Further, an enhancement to the algorithm is provided that requires only a small subset of the segments to be clustered, drastically reducing the run time.

Results of the clustering technique are shown, highlighting various major traffic flow patterns in the NAS. In addition, organizing the traffic into the flow segments identified using the Incremental Clustering method is shown to have a potential reduction in the number of conflict points.

An application of the flow information is presented in the form of a Decision Support Tool (DST) that aids traffic managers in establishing and managing Airspace Flow Programs. In addition, the flow segment information is applied to a low-level form of aggregated traffic management, showing that aggregating flights into the flow segments and rerouting the whole flow segment can be efficiently performed as compared to rerouting individual aircraft separately, and can reduce the number of conflict points. Considerations for implementing these techniques in real-time systems are also discussed.

IDENTIFICATION OF AIR TRAFFIC FLOW SEGMENTS VIA INCREMENTAL DETERMINISTIC ANNEALING CLUSTERING

By

Alex Thinh Nguyen

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

Dedication

To my parents, Chien Nguyen and Bichdao Trinh,

My beloved wife Quan,

and

Lindsay, Courtney, Kaitlin

Acknowledgements

I am grateful to Professor John Baras for his enduring support, guidance, and vision. The breath of research activities and approach of open research has inspired me to explore many topics and apply methods across disciplines. I thank him for his guidance in always discovering the true purpose and benefit to whatever the problem is being tackled, and for teaching and instilling Systems Engineering principles.

I am also indebted to Professor Michael Ball for his teaching and expertise in Operations Research and Air Traffic Management. I am grateful to have him on my committee, and thank him for all the time he has made available for discussions of ideas, problem formulations, and approaches to solving them.

For all the suggestions, advice, and very useful comments, I am extremely appreciative of Professor David Lovell, and I thank him for all the discussions and for always providing new perspectives on my research and helpful ways to improve the work and its presentation.

I thank Professor Eyad Abed for always being so accommodating and taking the time to discuss my work, and the great advice and support in helping me through the process. I also thank Professor Bill Levine for his flexibility and great detailed comments towards improving my dissertation.

Through my research I have been fortunate to collaborate with many colleagues in academia, government, and industry who have helped further my work. In particular, I have enjoyed working with fellow HyNet and NEXTOR researchers. In addition, I thank Sherrie Callon of the FAA for all her invaluable controller expertise, out-of-the-box thinking, and very helpful comments.

My research would not have been possible without the support from Andras Kovacs at the FAA and Natee Wongsangpaiboon. Their understanding, encouragement, and sacrifice have enabled me to conduct my research.

Finally, the greatest sacrifice has been on the part of my family. I thank my mother for the drive to seek my doctorate, and I am eternally grateful to my wife Quan who has always been supportive at every step of the way, and provided the encouragement and inspiration during hard times. And, it is Lindsay, Courtney, and Kaitlin who have provided me the purpose and spirit.

Table of Contents

List of Figures

Chapter 1. Introduction

1.1 Background

The notion of the major "flows" of aircraft in air traffic has a very intuitive meaning, and is used extensively in describing traffic patterns or conducting traffic management initiatives based on them. However, the actual identification of the location and time of the major flow segments of traffic are often left to interpretation based on observations of traffic data points over time. Many of the traffic management techniques currently in use rely on human intuition of the traffic flows and some inefficiently manage individual flights without the knowledge of overall flows in the National Airspace System (NAS). This may yield inefficiencies in system performance and more scatter in trajectories, which could lead to more congestion and less throughput.

Controllers and air traffic managers today form a mental picture of the flows of traffic based on their extensive experience in observing the traffic patterns of a certain region over time. From seeing the trajectory of each aircraft displayed on the screen in the form of the current position, heading, speed, and other information, the controller or traffic manager is able to have an intuition about the general areas of major traffic and their direction. This results in variations of interpretations and no method of supporting the flows as a group in automation systems.

In order for automation systems to make use of this "flow," a definition for the flow is needed, and a method must be developed to efficiently identify them in near real-time.

1.2 Contributions of this Research

The main focus of this research is on a method to identify the major flows of air traffic. A definition of an air traffic "flow" is provided along with a technique for finding the locations and times of the major flows of traffic that consists of the following major components:

1) A method for identifying the great circle segments of each flight's trajectory,
2) A method for clustering these segments based on a Deterministic Annealing clustering algorithm.
3) A method for incrementally clustering these segments to take into account the spatial-temporal properties of the segments.

Further, a method for clustering only a subset of the segments is proposed as an enhancement to the algorithm to greatly increase the performance of the algorithm with minimal difference in clustering results.

Based on this, analysis is performed to show potential results and uses of the clustering technique. A method for efficiently determining the conflict location points of aircraft trajectories is presented, and used to show potential benefits in reducing the conflict points by clustering traffic into flow corridors.

Building on the knowledge of the identified air traffic flows, an application of the flow information is presented in the form of a Decision Support Tool (DST) that aids traffic managers in establishing and managing Airspace Flow Programs. In addition, the flow segment information is applied to a low-level form of aggregated traffic management, showing that aggregating flights into the flow segments and rerouting the whole flow segment can be efficiently performed as compared to rerouting individual aircraft separately, and can reduce the number of conflict points. Considerations for implementing these techniques in real-time systems are also discussed, including metering traffic at the entry into the flow corridors, and handling changes in the flow corridor's trajectory using a leader-based in-trail self-separation technique.

1.3 Organization of the Dissertation

Chapter 2 describes the NextGen air traffic management concepts that are pertinent to this dissertation, and discusses some of the relevant research that has been conducted recently and how this complements them. It also discusses the need for identifying the flows of traffic, and the overall clustering approach.

Chapter 3 describes the strategy for clustering segments in the spatial domain using a Deterministic Annealing clustering approach. The general Deterministic Annealing algorithm is first described, followed by the proposed agglomerate approach. In addition, metrics for evaluating the quality of the resulting clusters and determining the appropriate stopping condition for the algorithm are presented.

Chapter 4 discusses the time properties of the flight segments, and describes an Incremental Clustering algorithm to cluster portions of segments at a time that are temporally relevant during each increment. In addition, the scheme to partition the clusters by time is described along with analysis of results.

Chapter 5 describes some applications of the flow information determined from the clustering and time partitioning algorithms. A Decision Support Tool (DST) to provide the information to support Airspace Flow Programs is described. Then, an

Integer Programming model to reroute an entire flow of traffic rather than individual flights is shown. Data sources and methods for computing airspace weather avoidance fields are discussed. While not the primary focus of this research, a method for handling multiple simultaneous flows of traffic is described by prioritizing the traffic flows. The chapter also discusses considerations for implementing the algorithms on live traffic.

Finally, Chapter 6 provides a summary of the work and discusses additional future applications of the clustering problem that can be further researched.

Chapter 2. Air Traffic Flow Management

2.1 The Need for Identification of Traffic Flows

Knowledge of the location and times of major traffic flows is needed for the following purposes, among others:

1) To organize the airspace into structured routes (or "tubes") in near real-time to reduce controller workload and improve airspace utilization,
2) To apply traffic management initiatives to targeted areas that are needed the most,
3) To enable trajectories to truly be managed together as a flow rather than individual flights,
4) For airlines / pilots to either avoid the congested areas, or join tubes that may offer priority service,
5) To examine historical airspace utilization to better design airways and sectors.

The overall flow of the air traffic in the NAS is managed by the Air Traffic Control System Command Center (ATCSCC). Here, managers have access to weather information and flight plans of all flights in the NAS. From experience, traffic managers know where the main areas of traffic flow are and the portions of the day during which they appear. However, there currently does not exist an automated way of determining the flows in the NAS, and this is needed in order to feed future automation systems that manage air traffic flows.

Due to the current structure of the airways and the method for filing flight plans, the traffic may be somewhat predictable to an experienced controller or traffic manager. However, if new approaches come about for aircraft to fly 4-D Trajectory Based Operations and on more direct paths with less reliance on the current airways, these traffic patterns might become less predictable and even change throughout the day.

2.1.1 Flow Corridors

The Next Generation Air Transportation System (NextGen) is an effort underway by the Federal Aviation Administration (FAA) to transform and modernize the National Airspace System. [3] The goals for NextGen focus on significantly increasing the safety, security, and capacity of air transportation operations with new procedures and advances in technology.

One of the concepts of NextGen involving dynamic airspaces is that of establishing flow corridors or long tubes consisting of "bundles" of near-parallel traffic, as shown in Figure 1. This allows for a stream of aircraft with the same heading to fly through a portion of airspace reserved only for those flights. [1] Tubes would exist only at distinct locations and times throughout the airspace, and flights that want to use them would modify their flight plans to use a tube for the select portions of their flight.

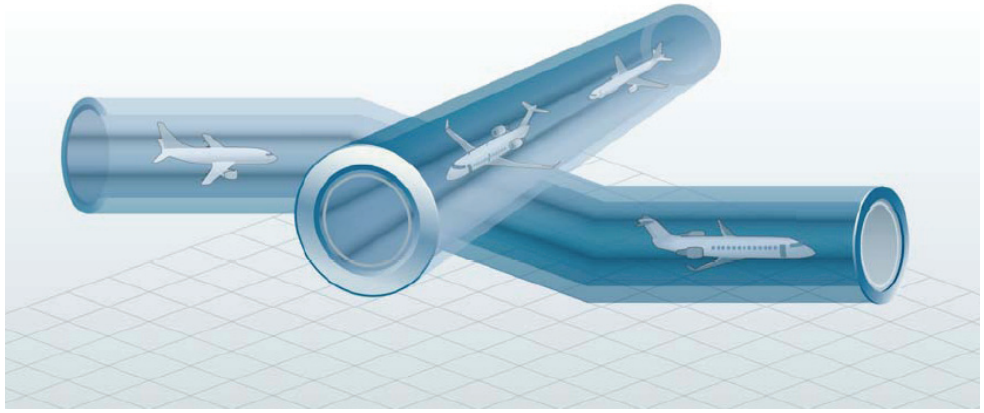


Figure 1. Flow corridors. Source: JPDO NextGen Concept of Operations. [1]

One of the key benefits envisioned in using flow corridors is the reduction in conflict location points, and that any conflicts that do occur can easily be resolved with small speed or trajectory adjustments. These conflict points are occurrences of two or more aircraft losing their horizontal or vertical separation minima. Since it is the controller's responsibility to ensure proper separation between aircraft, the controller's workload increases as the number of conflict points increases. To reduce the risk of aircraft losing separation minima, controllers may add more buffer between aircraft at complex areas, thus reducing capacity.

Because there is no cross traffic within this tube-shaped airspace, the airspace complexity and controller workload within the corridor is reduced, allowing for the separation between aircraft to be reduced thus providing for increased capacity within the tube. In addition, due to having less conflict points outside the tube, this also increases the airspace utilization of traffic around the tube. Finally, a single controller can manage the entire stream of aircraft throughout the tube without having to pass control to another. To make these corridors or tubes useful, they would be established along the optimum routes and altitudes where there is sufficient traffic to make it worthwhile, and may be dynamically shifted to avoid severe weather or other flow constrained areas. [1] In this dissertation, we will provide a means to determine where and when the major flows exist to target establishment of such tubes, and also examine the amount of conflict point reduction that could be achieved.

2.1.2 FCAs & Traffic Management Initiatives

Knowing the flows in the NAS helps to evaluate the impact of weather scenarios on the major travel routes. This in turn helps to identify the affected flows on which to focus various traffic management initiatives.

One of the methods to manage traffic through low-capacity areas caused by weather is through the Airspace Flow Programs (AFPs), which was implemented in 2006 to help address some of the limitations of the Ground Delay Programs (GDPs). [25] When there is severe weather, a Flow Constrained Area (FCA) is identified over the affected region as a line segment where flights that are scheduled to traverse it will be affected by the AFP. The geometry of the FCA is not generated dynamically, but instead selected from a set of predefined FCAs that is most appropriate for the expected weather traffic situation.

The selection of FCAs is based mostly on the traffic managers examining data and developing intuition on the most practical areas. Identification of the flows as part of this dissertation work can help to identify the major flows as consideration for implementing an FCA.

In [15], a model was developed for the assignment of dispositions to flights anticipated to be affected by an airspace flow program through an FCA. As shown in Figure 2, for each flight, the disposition can be either to depart as scheduled but via a secondary route that avoids the flow-constrained area, or to use the originally intended route but to depart with a controlled departure time and accompanying ground delay. The flights are assigned under a conservative scenario of maximal weather impact with the anticipation that the capacity through the flow-constrained area will increase after some future time once the weather activity clears. The model was a two-stage stochastic Integer Program that represented the time of this capacity windfall as a random variable, and determines expected costs given a second-stage decision, conditioning on that time. The model minimized the expected cost over the entire distribution of possible capacity increase times.

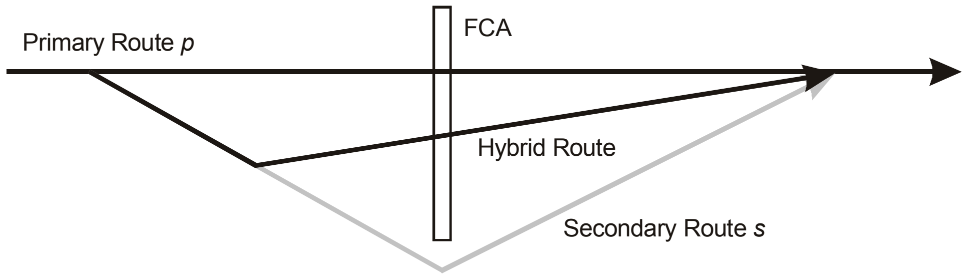


Figure 2. Reverting from secondary route back to primary route through FCA.[15]

To complement the model, this dissertation research can help identify major flow areas to target applying the stochastic optimization problem, or possibly help alleviate the need for longer-term forecasts, instead using more accurate shorter-term weather data repeated in time increments.

2.1.3 True Traffic “Flow” Management

From a near real-time perspective, being able to identify clusters of segments of trajectories allows the air traffic managers to evaluate entire flows of flights rather than individual aircraft. If there is a weather area or a temporary flight restriction imposed, it is easy to identify the individual flights that are affected without clustering, but managing the changes for each individual flight can be resource intensive. Instead, if segments of trajectories for multiple aircraft can be identified, then they can be moved in an aggregated manner, and using an algorithm that can achieve the best results for all the affected flights and the system overall, rather than just each individual aircraft. This offers a way to do system-wide optimization rather than just individual aircraft optimization.

For example, traffic flow techniques such as rerouting of aircraft around weather activity are handled on individual aircraft, not truly as a group. Because of this, it is possible that the individual aircrafts' reroutes around the weather can cause conflicts among themselves and lead to further delays.

Identification of the flows in the NAS provides a means for state space reduction of the air traffic and enable actions to be made on the aircraft groups, rather than individual aircraft.

2.1.4 Aiding User Route Preferences

Currently, airlines and pilots plan their routes based mainly on their preferred routes (for commercial flights) adjusted to take into account current wind and weather conditions to save fuel. Little is taken into consideration, or even known, about the trajectories of other aircraft that might be traversing the same portions of the trajectory as the one in question. If the users have knowledge of the flows of traffic, they may chose to avoid the congested areas in favor of longer, but faster, routes. Alternatively, if flow corridors (“tubes”) were to be established in certain high-flow areas, users might chose to go through them in favor of better flow management and reduced delays.

2.1.5 Analysis & Airspace Utilization

Most commercial aircraft currently file flight plans that place them along prescribed airways. Identification of the major flows of traffic can help the airways be designed in such a way as to align with the major flows, and reduce cross traffic over the most congested areas.

2.2 Clustering Traffic to Identify Flows

Because each flight individually files its own flight plan, and because these flight plans may take into account projected winds and weather conditions or restricted airspaces, there may be varying trajectories even for flights that fly between the same origin and destination around the same time.[5] In addition, flights with different origin or destinations may share a similar portion of airspace.

In Figure 3, the trajectories from the filed flight plans show flights traversing through a particular airspace, with varying origins and destinations. The goal here is to identify the heavy “flows” of traffic in the NAS by finding concentrations of segments of traffic that travel with a similar trajectory. The objective is to take 4-D trajectories from flight plans (in real time or historical) and identify where the overall flows of traffic are using clustering methods.

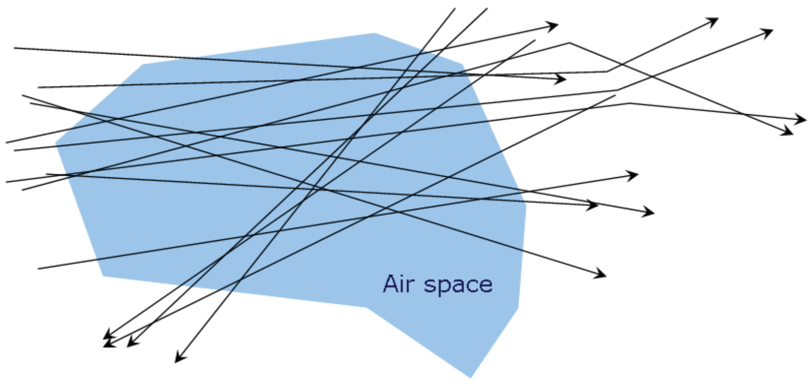


Figure 3. Filed trajectories from flight plan.

Note that as shown in Figure 4, we are not clustering entire trajectories, but rather segments of trajectories from flights that could have different origin and destination. Depending on the application, these trajectories could be historical (for post-analysis), or could be near real-time filed flight plans of flights that are about to takeoff.

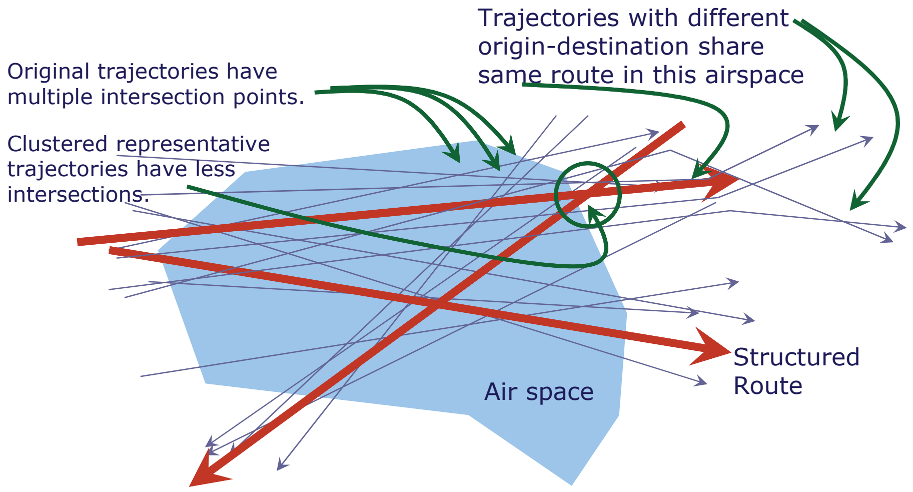

Trajectories with different origin-destination share same route in this airspace
Original trajectories have multiple intersection points.
Clustered representative trajectories have less intersections.
Structured Route
Air space

Figure 4. Clustering and consolidating flight segments with similar headings.

2.3 Related Work

Clustering of data points has been well-studied, and include a wide range of methods for various types of applications. [17] In the computer graphics arena, many methods exist for clustering of two-dimensional line segments, including one that uses an axiomatic approach that clusters lines invariant to changes in scale. [22] Efforts in clustering vehicular traffic patterns have included a coarse-to-fine approach where the trajectory is first smoothed, then features are extracted, and clustering is performed first at a coarse level then at a fine level. [24] In that approach, a histogram of the trajectory direction is formed that describes the directional distribution of each trajectory, and cluster together trajectories sharing similar trends in the histogram.

A framework was proposed to partition and group ground vehicle trajectories into clusters and generate a "representative trajectory" for each cluster based on the DBSCAN (Density Based Spatial Clustering of Applications with Noise) algorithm. [23][12] This provided a good overall framework for clustering portions of ground trajectories. Application of this model to aircraft trajectories would need to take into account aircraft heading, and also the timing of the aircraft trajectory segments.

There has also been some work performed in grouping similar trajectories as a whole. In [45], a Longest Common Subsequence (LCSS) model was used to match two sequences by allowing them to stretch, without rearranging the sequence of the elements but allowing some elements to be unmatched. This is a partial clustering approach that allows for some elements such as outliers to not be matched. The approach is useful for ground trajectories that have similar patterns that may be stretched, but may need to be adapted for use in clustering aircraft trajectories, since the stretching of trajectories causes changes in the geographical patterns of the flights, an important factor in the clustering.

Many of the past work on aircraft trajectories focused on clustering of entire flight trajectories between the same origin-destination pairs, rather than just portions of the trajectory. The method in [5] used historical flight data for purposes of analyzing the performance of the system and making adjustments for improvements to the routes. The method is intended for post-analysis of entire trajectories, rather than for near real-time applications of clustering segments of flight plans.

Other aircraft clustering work that focused on parts of the trajectory has included [10], where "snapshots" of traffic position data were taken at regular time intervals, and clustering was performed by starting with a seed aircraft, and continually finding other proximate aircraft. The results were then concentrated to obtain the temporal evolution of the clusters. The work was later extended in [11] to three dimensions, utilizing airspace complexity measures instead of aircraft density.

In [38], a heat map was used for structure-based traffic abstraction to identify "standard" flows. Two approaches were identified, one using a Greedy algorithm to associate tracks together based on their traffic density heat map, and the other to identify the clusters by the ridges of the 3D heat map.

Recently, an approach was proposed to cluster Velocity Vector Fields (VVF) of traffic. [50] The actual clustering method used neighbor lists similar to that used in DBSCAN, and the associations consisted of 7 input parameters chosen based on design criteria. The final resulting clusters are filtered based on an 8th parameter. The study showed benefits to the user of reduction in delay time in using tubes. In this dissertation, a clustering approach will be presented relying on 2 parameters that can be intuitively chosen, and potential benefits to the system and controller will be shown in the form of reduced conflict location points.

2.4 Clustering Approach

The high-level approach for clustering air traffic is shown in Figure 5. The first step is to take a 4D trajectory and break it into great-circle segments. Then, portions of these segments are retrieved for a certain period of time for the Clustering algorithm to group them into clusters. Next, the clusters are partitioned at areas where the time gap between consecutive segments within the same cluster are greater than a threshold time. Processed segments are appended to the completed set of clusters, and the process repeats for the next batch of segments.

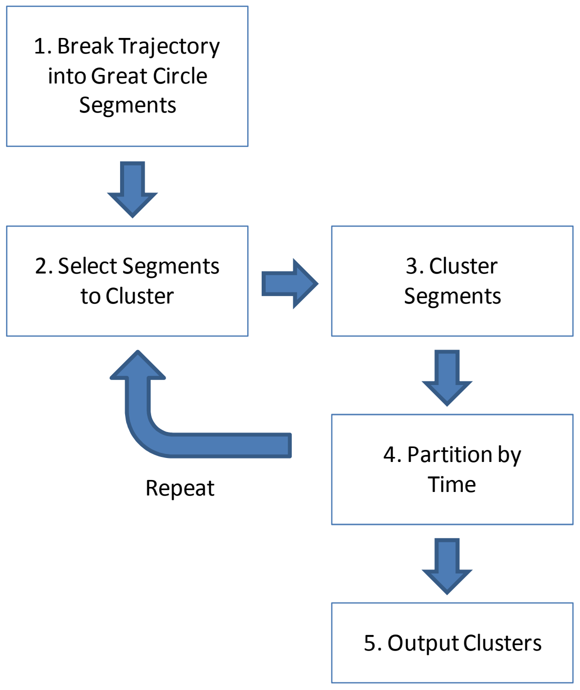

1. Break Trajectory into Great Circle Segments
2. Select Segments to Cluster
3. Cluster Segments
4. Partition by Time
5. Output Clusters
Repeat

Figure 5. High-level Clustering Steps.

In Chapter 3, the spatial aspects of the clustering process are discussed, and an algorithm for breaking the trajectory into great circle segments is presented to address Box 1 in the process diagram. The main contribution of Chapter 3 is a method for clustering segments using a Deterministic Annealing algorithm, which is Box 3 and is the core of the whole process. Chapter 4 discusses the temporal aspects of the segments, and provides an Incremental Clustering algorithm for selecting the segments in Box 2, Time Partitioning in Box 4, and assembling the output clusters in Box 5.

Chapter 3. Clustering Flight Segments

3.1 Generating Piece-wise Great Circle Segments

Because the goal is to find flows of traffic in the air space, we are interested in clustering portions of flights rather than entire flight trajectories between origin and destination. The first step then is to obtain the great circle segments for each flight.

The shortest route that an aircraft can fly between any two points is a great circle route, which is defined by the arc generated when intersecting a plane through the center of a sphere. As shown in Figure 6, a great circle is represented by the longitude ϕ_0 at which it crosses the equator, and the maximum latitude $\lambda_{\max}$ achieved by the circle.

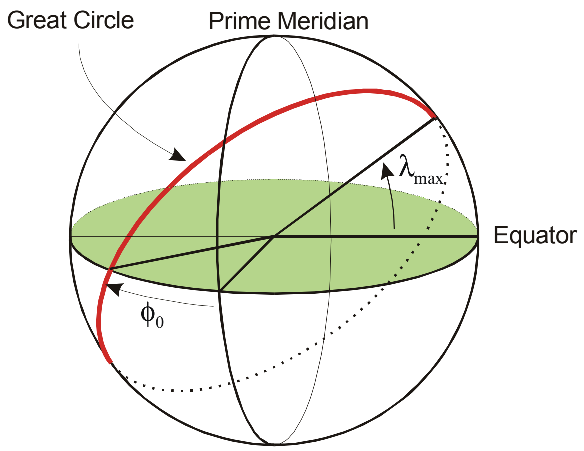


Figure 6. Great circle defined by the intersection of a plane through the center of a sphere.

Because of the structure of the NAS, aircraft do not actually fly a continuous great circle route between the origin and destination, but instead follow jet routes and make turns to take advantage of favorable winds for fuel efficiency, avoid areas of congestion, and avoid Special Use Airspace (SUAs). Between points on a trajectory, however, aircraft will generally fly along a great circle. Thus, a trajectory can be represented as a series of great circle route segments.

For analysis of traffic in the future or in near real-time, flight plans provide the most direct information on segments. A flight plan consists of a series of way points through which an aircraft plans to travel, with the portion in between representing a great circle segment that can be used for clustering. However, if flight plans are not available or if an analysis is performed on historical data, the as-flown trajectories must be used.

Because the trajectory does not have information on the turns imbedded in it, the cluster segments must be identified, which is the focus of this section. The as-flown trajectory consists of a series of position reports from radar data that is available about every minute. Work is underway on a notion of a Flight Object that would contain all the pertinent information about a flight in a consolidated package of data. This "object" would contain the positions and turns that an aircraft has made (or plans to make) and could provide more direct information of the aircraft's flight segments flown. However, until that is available, the series of great segments that an aircraft flew would have to be derived from the trajectory position reports, such as from ETMS (Enhanced Traffic Management System) data.

The 4-D trajectory of a flight is a series of *N* points $J = \{p_1, p_2, ..., p_N\}$, where each point $p_i = \{\lambda_i, \phi_i, h_i, t_i\}$ is a 4-dimensional position consisting of the latitude λ, longitude ϕ, altitude h, and time t. Let us define a fight segment as follows:

Definition 1. *A flight segment* $\mathbf{s} = (\lambda_1, \phi_1, h_1, t_1, \lambda_2, \phi_2, h_2, t_2)$ *is a continuous portion of an aircraft's trajectory from position* (λ_1, ϕ_1, h_1) *at time* t_1 *along a great circle arc to position* (λ_2, ϕ_2, h_2) *at time* $t_2 > t_1$.

Note that a great circle arc can be uniquely defined by two end points (λ_1, ϕ_1, h_1) and (λ_2, ϕ_2, h_2). As shown in Figure 7(a), we wish to partition the flight trajectory into $M < N$ great circle segments where the points associated with each segment represent a position along the great circle arc defined by the segment's end points. Each trajectory point must be associated with exactly one great circle segment. This problem can be solved via a Dynamic Programming approach similar to that used in [21] to approximate digital planar curves in vehicle trajectories with line segments and circular arcs. However, experience with air traffic data has shown that methods that "force" a fit of a series of points, either with an objective function by comparing weights of possible solutions or using constraints can succumb to bad data points. For example, it could break a segment (due to a bad data point) to minimize some weight when it really shouldn't.

Instead, since the goal is to find segments between turns, we will use an alternate approach described below that is based on finding turns in the trajectory. This is reasonable because in reality, aircraft fly great circle segments between turns. As will be described below, the great circle obtained is the one that minimizes the squared error for the associated points. With this method, any outlier points that appear between turns will be included into that great circle segment (as should be the case), possibly causing the error of that fit to increase. Rather than placing an explicit limit on that error, which may cause the segment to be improperly split at that outlier point, no limit is placed on the error of the fit, as the main criterion is the existence of a turn. This makes the method more resilient to position report errors and provides faster performance.

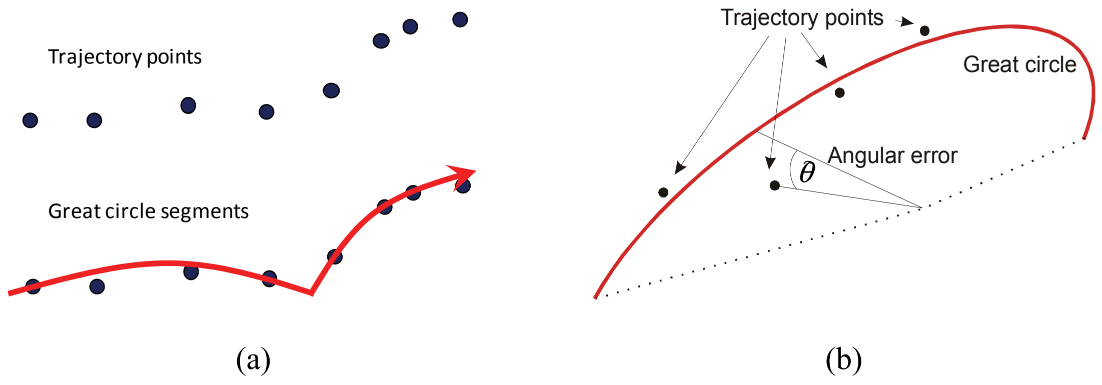


Figure 7. Forming great circle arcs from trajectory points.

The overall approach is to find the great circle fits of successive portions of the trajectory where there is no turn present. The fitness of a point $\mathbf{p}_k$ to a great circle is measured by the angular error between the points and the great circle plane as shown in Figure 7(b), and is computed using the dot product between the point and the unit normal to the great circle $G = \left(n_x, n_y, n_z\right)$:

$$e_k = \mathbf{p}_k \bullet G = p_{k,x}n_x + p_{k,y}n_y + p_{k,z}n_z$$

For a given set of consecutive points *i* through *j* in the trajectory, we wish to find the great circle G_{ij} that minimizes the angular error between the points and the great circle:

$$\min E_{ij} = \sum_{k=i}^{j} e_k^2 = \sum_{k=i}^{j} \left(\mathbf{p}_k \bullet G_{ij}\right)^2 \qquad i<j$$

For the G_{ij} solved from the least squares fit (method to be discussed later), let $\delta(G_{ij})$ be an indicator function such that:

$$\delta(G_{ij}) = \begin{cases} 1 \text{ if there is a turn in the trajectory away from } G_{ij} \text{ at point } j+1 \\ 0 \text{ otherwise} \end{cases}$$

Let $\omega_i(G_{ij})$ be the number of points represented by the great circle segment G_{ij} generated by the least squares fit of points starting at point *i* in the trajectory:

$$\omega_i(G_{ij}) = \left|\left\{ p_k \in G_{ij} \right\}\right| = j - i + 1$$

For a starting point *i*, we wish to continually expand the set of points associated with the current segment and calculate the associated great circle until a turn is detected:

$$\max_j \omega_i(G_{ij})$$

such that $\delta(G_{ij}) = 0$, $i < j \leq N$

The algorithm works by computing the great circle segment G_{ij} between points *i* and *j*, and detect if there is a turn in a look ahead range of points beyond *j* (calculated by the FIND_TURN_IN_LOOKAHEAD function). The turns are identified relative to the GC fit of the preceding points, otherwise identifying turns alone without a GC in mind would make it prone to breaking segments at bad points that look like turns. If there is no turn, the next point is added to the current segment, and the great circle is recalculated to minimize the error with the additional point. If a turn is found, then a new segment is established starting after the end of the current one.

Since an aircraft makes an arc as it turns, once a turn is identified, the intersection between the current great circle segment and the next is calculated. Then, the points before the intersection are associated with the current great circle segment, and those after are associated with the next. Figure 8 shows the overall TRAJ_FIT algorithm for determining the great circle segments. The method for calculating the great circle fit and the algorithm for the FIND_FURN_IN_LOOKAHEAD function used to detect a turn in the trajectory are discussed in the following sections.

```
TRAJ_FIT algorithm:

Input: Trajectory consisting of N points $\{p_1, p_2, .., p_N\}$
Output: Set of M great circle segments
$\left\{G_{1,r_1}, G_{r_1+1,r_2}, G_{r_i+1,r_{i+1}}, .., G_{r_{M-1}+1,r_M}, r_i \in R\right\}$, where $R = \{r_1, r_2, ..., r_M\}$ contains the
indices of the last trajectory point associated with each great circle, and
$1 < r_1$ and $r_M = N$.

    1. Initialize:
        a. $i = 1$
        b. $j = 2$
        c. $R = \varnothing$
    2. Find the longest great circle segment starting at point i without turns:
        a. Find the great-circle fit $G_{ij}$ from i to j
        b. Determine $\delta(G_{ij})$ using the function
           FIND_TURN_IN_LOOKAHEAD
        c. If $\delta(G_{ij}) = 0$, there is no turn, so extend the great circle
           segment by setting $j = j+1$
            i. Go to 2(a) if $j \le N$, otherwise stop
        d. If $\delta(G_{ij}) = 1$, a turn has been detected beyond j
            i. Find the corner between the current great circle and
               the next
            ii. Find the last point r before the corner, which will be
                the last point associated with the current segment.
                Update $R = R \cup \{r\}$
            iii. Start a new great circle segment by setting $i = r+1$,
                 $j = i+1$
            iv. Go to 2 if $j \le N$, otherwise stop
```

Figure 8. Algorithm TRAJ_FIT for finding great circle segments from trajectories.

3.1.1 Great Circle Fit

To calculate the great circle fit, the flight trajectory is first converted from the Geodetic into the Earth-Centered Earth-Fixed (ECEF) coordinate system using the following:

$$x = \left(N(\lambda) + h\right)\cos\lambda\cos\phi$$
$$y = \left(N(\lambda) + h\right)\cos\lambda\sin\phi$$
$$z = \left(N(\lambda)\left(1 - e^2\right) + h\right)\sin\lambda$$

where

$$N(\lambda) = \frac{a}{\sqrt{1 - e^2 \sin^2\lambda}},$$

and $a = 6378137$ and $e = 8.1819190842622\text{e}^{-2}$ are the semi-major axis and eccentricity of the Earth ellipsoid respectively.

Using the dot product error function defined in the previous section, for a consecutive series of points *i* through *j*, the total error between the points and a particular great circle is given by

$$E_{ij} = \sum_{k=i}^{j} e_k^2 = \sum_{k=i}^{j}\left(p_{k,x}n_x + p_{k,y}n_y + p_{k,z}n_z\right)^2$$

Setting $\frac{\partial E}{\partial n_x} = 0$, $\frac{\partial E}{\partial n_y} = 0$, and $\frac{\partial E}{\partial n_z} = 0$, we can solve for the great circle unit normal $G = \left(n_x, n_y, n_z\right)$ that provides the least squares fit of the set of trajectory points as

$$n_z = \frac{1}{\sqrt{\frac{1}{\rho_{xx}{}^2}\left(\beta\rho_{xy}\right)^2 + 2\beta\rho_{xy}\rho_{xz} + \rho_{xz}{}^2 + \beta^2 + 1}},$$

$$n_y = \beta n_z, \qquad n_x = -\frac{1}{\rho_{xx}}\left(n_y\rho_{xy} + n_z\rho_{xz}\right)$$

where

$$\rho_{xx} = \sum_{k=i}^{j} p_{k,x} p_{k,x} \qquad \rho_{xy} = \sum_{k=i}^{j} p_{k,x} p_{k,y} \qquad \rho_{xz} = \sum_{k=i}^{j} p_{k,x} p_{k,z}$$

$$\rho_{yy} = \sum_{k=i}^{j} p_{k,y} p_{k,y} \qquad \rho_{yz} = \sum_{k=i}^{j} p_{k,y} p_{k,z}$$

$$\beta = \frac{\dfrac{\rho_{xz}\rho_{xy}}{\rho_{xx}} - \rho_{yz}}{\rho_{yy} - \dfrac{\rho_{xy}{}^{2}}{\rho_{xx}}}$$

3.1.2 Turn Detection

The function FIND_TURN_IN_LOOKAHEAD finds the location of the next turn in the trajectory by examining a moving window of points and determining if the trajectory points begin to deviate from the current great circle according to a certain pattern. The window is initially placed at MAX_FIT_POINTS number of points before and after the current point in order to establish a pattern. The window is extended by MAX_LOOKAHEAD_POINTS in order to foresee upcoming turns. The values for MAX_FIT_POINTS and MAX_LOOKAHEAD_POINTS are determined experimentally and set at 5 and 5 respectively.

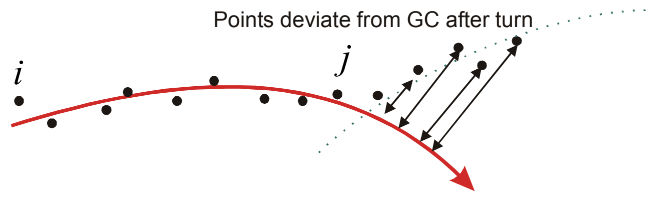


Figure 9. Deviation of points from the great circle after a turn.

Figure 9 shows a great circle fit (red arrow) between points *i* through *j*. Beyond *j*, as a turn approaches, the points on the trajectory will begin to deviate from the great circle. If the error itself was used to detect a turn, the process would be susceptible to errors in the position points. Instead, we observe that as a turn is made, the slope of the angular error $m_{k,k+1} = \frac{e_{k+1} - e_k}{t_{k+1} - t_k}$ will increase and then flatten as the turn is complete, as shown in Figure 10(a). However, the value of $m_{k,k+1}$ when it flattens depends on the turn angle, which varies. So instead, we examine that the second derivative of the error, $m'_{k,k+1} = \frac{m_{k+1} - m_k}{t_{k+1} - t_k}$, exhibits the pattern shown in Figure 10(b), where there is initially a low steady value, followed by a sharp peak (over which $m_{k,k+1}$ is increasing), and then a return back to the low value. The algorithm uses two thresholds, where a turn is detected if $m'_{k,k+1}$ is initially below THRESHOLD_LO, then above THRESHOLD_HI, and again below THRESHOLD_LO. The FIND_TURN_IN_LOOKAHEAD algorithm is shown in Figure 11.

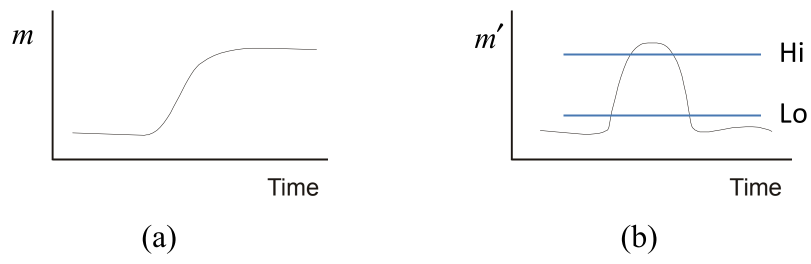


Figure 10. Patterns in the error's derivative and second derivative in detecting turns.

```
FIND_TURN_IN_LOOKAHEAD algorithm:

Input: $i, j, G_{ij}$
Output: $\delta(G_{ij}) = \{0,1\}$

    1. Set start point $s = \max(j - \text{MAX\_FIT\_POINTS} + 1, i)$
    2. Set end point $t = \min(j + \text{MAX\_FIT\_POINTS}, N)$
    3. Detect turn pattern:
        a. Calculate the second derivatives of the angular errors
           between points $p_s, .., p_t$ and $G_{ij}$.
        b. Determine if the second derivative starts below
           THRESHOLD_LO, then becomes greater than
           THRESHOLD_HI, then below THRESHOLD_LO again
            i. If true, then a turn is detected. Set $\delta(G_{ij}) = 1$
            ii. Otherwise, Set $\delta(G_{ij}) = 0$
```

Figure 11. Algorithm FIND_TURN_IN_LOOKAHEAD for finding turns.

3.1.3 Results

The algorithm is implemented in a C++ program using flight trajectories from the FAA's Enhanced Traffic Management System (ETMS). This required a separate tool to parse the data, filter bad points, and assemble the individual data elements into flights that, while not necessarily absent of all outliers, have reasonably accurate trajectories that can then be provided as input for the TRAJ_FIT algorithm. The focus of this work is on enroute airspace, where the trajectories are well behaved and do not exhibit holding patterns (except in an extremely small subset of likely scenarios) as may be encountered in the terminal airspace. Even if they do occur, however, because holding patterns are comprised of a series of a turn followed by a straight segment, and because the TRAJ_FIT method is based on turn detection, it is resilient to holding patterns. Nevertheless, extensive runs and associated comparisons with several data sets did not reveal any results that were contrary to intuition, or any strange results, and the computations on many examples showed that the method and algorithm are very robust to a variety of data inputs.

Figure 12 shows a sample of the series of great circle fits to the 4-D trajectory for a flight. The points are positions from ETMS data, and the great circle segments generating the least squares fit to the trajectory segments are shown as lines. (The line segments are shown extended slightly past each corner to aid in visualizing the line.) ETMS provides a roughly one-minute radar position of each flight, as well as filed flight plan information. The program TRAJ_FIT also identifies the corners, which are the intersections of the great circles. This example shows that the great circles generated by the algorithm fit the points reasonably well. The overall algorithm has a complexity of $O\left(n^2\right)$, however the average complexity is much less, since once a segment is identified, those points do not need to be revisited.

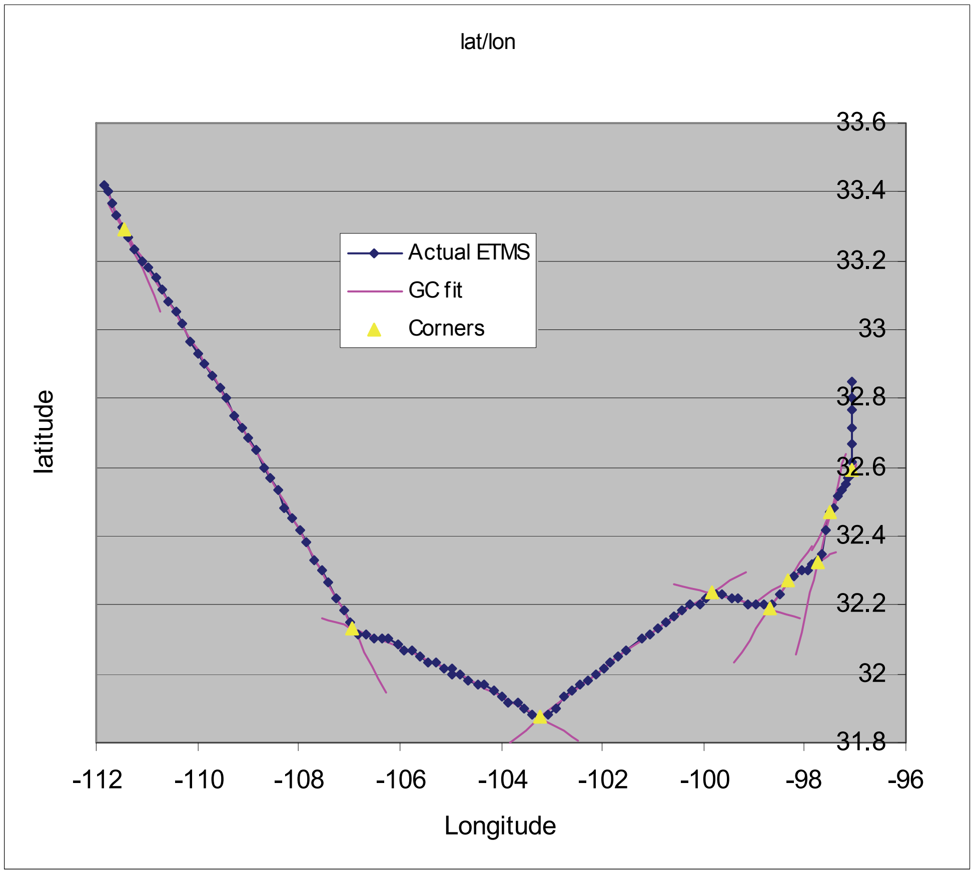


Figure 12. Results of fitting a flight's trajectory to a series of great circle segments.

3.2 Clustering of Segments

Clustering algorithms can be classified in a number of ways, one of which is by the overall types of partitioning, hierarchical, and locality-based. [17][29] Partitioning algorithms take a given set of *n* objects and group them into $k \leq n$ clusters. Examples of such methods include the *k*-means algorithm, in which each cluster is represented by the gravity center of the cluster (which may or may not be an actual point in the database), or the *k*-medoid algorithm, in which an actual point in the dataset located near the center of the cluster is chosen as the representative for the cluster. Because *k* is an input parameter, these methods generally require domain knowledge of the problem to determine in advance the number clusters desired. Often times, as in our application, this is not known in advance.

Locality-based methods examine the local conditions around each point and form clusters based on certain metrics applicable to the surrounding points. An example is DBSCAN (Density-Based Spatial Clustering of Applications with Noise), which, like other density-based algorithms, identifies clusters as dense regions of elements that are separated by regions of low densities of objects. [12][39] The algorithm forms a cluster if there is a minimum number of points in the cluster that have other points within a certain distance from them. It then grows the cluster as long as these conditions still satisfy. However, while this method works well for points, it is susceptible to anomalies when applied to line segments.

Because the ultimate goal of the clustering is to identify the major flows of air traffic in the NAS, each trajectory segment should belong to at most one cluster, even though segments could have traits in common to segments in other clusters. However, the clusters could have some attribute that may be similar to each other such as they occur in the morning when there is a push of originating aircraft in the NAS. So, for the most part these would involve partitional, exclusive clustering methods rather than overlapping techniques.

In order for this to be used in operations in near real-time, the algorithm will need to be able to dynamically and incrementally cluster new flight segments with existing clusters without having to re-cluster the entire data set each time. As flight plans get modified or new ones are filed, this will allow for the system to incorporate this new, updated information on flight plans in near-real time and identify how the flows will possibly change based on the information. The incorporation of the time element into the clustering algorithm will be an important factor.

3.2.1 Problem Setup

Given the set of flight segments from all flights determined from the previous section, the goal in this section is to group them into clusters. Using Lambert conformal conic projection, the flight segments are expressed as follows:

Definition 1b. *A flight segment* $\mathbf{s} = (x_1, y_1, t_1, x_2, y_2, t_2)$ *is a continuous great circle portion of an aircraft's trajectory from position* (x_1, y_1) *at time* t_1 *to position* (x_2, y_2) *at time* $t_2 > t_1$.

The following definitions related to a cluster of traffic segments are established:

Definition 2. *For a given set of n segments* $\{\mathbf{s}_1,..,\mathbf{s}_n\}$, *the representative trajectory* $\mathbf{q}$ *of the segments is the average position of the segments:*

$$\mathbf{q} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{s}_i$$

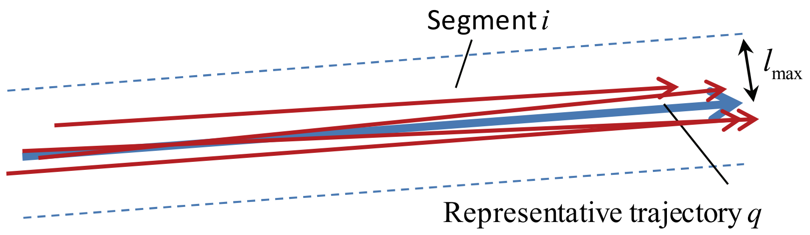


Figure 13. Segments that are part of the same cluster.

Definition 3. *As shown in* Figure 14*, the maximum perpendicular distance* $l_{p,q}$ *of a line segment* **p** *projected onto a representative trajectory* **q** *is defined by* $l_{p,q} = \max(r_1, r_2)$*, where*

$$r_1 = \frac{\left|(x_2^q - x_1^q)(y_1^q - y_1^p) - (x_1^q - x_1^p)(y_2^q - y_1^q)\right|}{\sqrt{(x_2^q - x_1^q)^2 + (y_2^q - y_1^q)^2}},$$

$$r_2 = \frac{\left|(x_2^q - x_1^q)(y_1^q - y_2^p) - (x_1^q - x_2^p)(y_2^q - y_1^q)\right|}{\sqrt{(x_2^q - x_1^q)^2 + (y_2^q - y_1^q)^2}}.$$

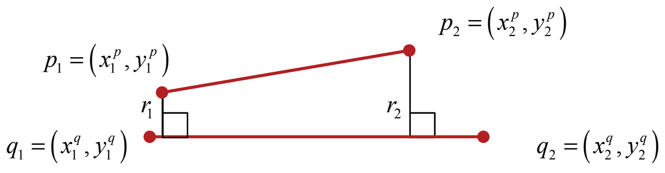


Figure 14. Maximum perpendicular distance.

Let $F = \{f_i\}$ be the set of all flights, where each flight *f* consists of a series of flight segments $f = \{\mathbf{s}_{f1}, \mathbf{s}_{f2}, ..., \mathbf{s}_{fg}\}$, and $\mathbf{S} = \bigcup_{f \in F} \{\mathbf{s}_{f1}, \mathbf{s}_{f2}, ..., \mathbf{s}_{fg_f}\}$ be the collection of flight segments from all flights. As shown in Figure 15, for a given maximum perpendicular distance $l_{\max}$, we wish to find a partitioning of **S** into clusters $C_k = \{s_i : l_{ik} < l_{\max} \forall i \in C_k\}$ such that for each cluster, the largest perpendicular distance between any segment in the cluster and the cluster's representative trajectory does not exceed $l_{\max}$.

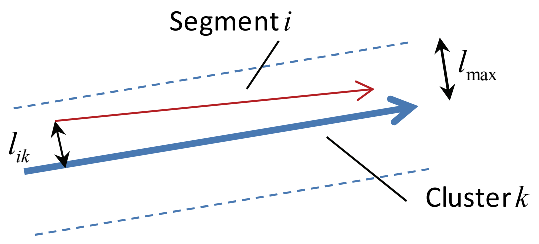


Figure 15. Boundary for segment to be associated with cluster.

For the clustering algorithm itself, we are not concerned with the number of segments associated with each cluster. The clustering just provides the best association of each segment to exactly one cluster. Then, depending on the specific use of the cluster information, the analyst can chose to select only clusters that have a certain minimum number of segments $n_{\min}$.

For this application, we focus on the enroute airspace, and consider only flight segments above 29,000 ft. For segments that span below and above this altitude, the segment is clipped and only the portion above 29,000 ft is kept. In enroute airspace, the controller must manage traffic along each altitude band within his sector. Although there are specific altitudes allocated for west-ward versus east-ward travel to separate the directions of traffic, the controller looking at 2-dimensional display still incurs workload in managing the separate altitude levels. Thus, in order to truly reduce a controller's workload, the entire band of altitude must be considered together during the clustering. Also, aside from fuel efficiency and aircraft performance, the selection of altitude by the pilot is somewhat arbitrary, and most commercial jets can fly at nearly any altitude range within the enroute airspace. So, the clustering is performed just on the segments' latitude and longitude, and not on altitude. Once the flows of traffic are determined, then the most desired altitudes can be examined and separate stacked tubes can be established at varying altitudes depending on the traffic volume.

3.2.2 Deterministic Annealing Overview

Clustering is an optimization problem with a cost function that is usually non-convex and therefore contains many poor local minima. Many methods have been used in an attempt to counter this, such as Simulated Annealing, where a stochastic search is performed on the solution space. The method models after the physical process of annealing in metallurgy, where the material is heated to high temperature and cooled in a controlled fashion to increase the size of the crystals in a manner that reduces its distortion. At each step in the simulated annealing algorithm, a new configuration of a possible solution is generated and its associated cost is computed. If the cost is lower, then the new configuration is accepted. If the cost is higher, the new configuration is accepted with a probability $e^{-D/T}$, where D is a measure of the cost of the solution, and T is the current temperature, to provide a means to help escape local minima. [27] The temperature is gradually lowered until bad solutions are no longer accepted and a minimum is reached. The probability that the simulated annealing algorithm terminates with the global optimal solution approaches unity as the annealing schedule is extended. However, in practical cases, the algorithm is often cut short due to time considerations, thus yielding only an approximate global solution.

Rather than randomly searching the solution space, Deterministic Annealing has been used to optimize the cost function in a deterministic manner at each temperature. The approach is to view the optimization from a “fuzzy” context, starting the problem with a high level of randomness, and slowly reducing this randomness similar to the annealing process by lowering the temperature *T* at each step in the problem. [37]

One of the main attractions of Deterministic Annealing is its ability to yield the effective number of clusters at each temperature, eliminating the need to know the output number of clusters at the start. Also, the method provides a natural way for elements to be associated to each cluster by balancing a certain amount of entropy with an acceptable distortion amount during the annealing process, allowing the user to essentially "dial" the amount of distortion tolerated to find the appropriate number of clusters.

For each element *x*, the clustering algorithm associates it with a cluster *y(x)* that best represents the elements. For a given distortion or distance measure $d(x, y(x))$, the total distortion across all segments and clusters is

$$D = \sum_{x} p(x) d(x, y(x))$$

For this application, *x* represents a flight segment, and we will use a quadratic Euclidean distance as the distortion measure $d(x, y) = |x - y|^2$. Since this calculation will need to be performed repeatedly, its simplicity is favored over the great circle distance. In addition, the algorithm is concerned more about the relative distance between the two elements rather than actual distance, and at small distances, the difference between Euclidean and great circle distance is negligible for our purposes. Further, using a Lambert conformal conic map projection provides very good estimates of great circle distances. Since each segment is defined by its two end points, the distance measure implicitly takes into account the angular distortion among segments. Instead of using the distance between elements and the cluster center, an alternative approach would be to minimize the distance between elements of the same cluster, but this would mean performing pair-wise clustering as was used in [20], a method that greatly increases problem complexity and runtime, and used mainly when pair-wise associations among data elements are needed.

If we were to minimize this distortion measure, the typical “hard” clustering would be achieved, where each element would be automatically assigned to the nearest cluster, making it more susceptible to being stuck in a local minimum. Instead, if the elements were assigned to the clusters only in probability, this leads to a “soft” clustering approach at allows for the best association to be made. In this context, the total distortion for the randomized partition would be

$$D = \sum_x \sum_y p(x,y)d(x,y) = \sum_x p(x) \sum_y p(y \mid x)d(x,y)$$

where $p(x,y)$ is the joint probability of x and y, and $p(y \mid x)$ is the conditional probability of y being associated with a given element x. With this distortion measure, we minimize D subject to a level of randomness measured by the Shannon entropy:

$$H(X,Y) = -\sum_x \sum_y p(x,y) \log p(x,y)$$

Since the level of randomness that we are willing to accept will be adjusted throughout the optimization, it will be included in the objective function as a Lagrangian as follows

$$\min F = D - TH$$

where T is the Lagrange multiplier. For large values of T, higher entropy is favored, while as T is lowered, lower entropy is traded to achieve lower distortion, which is ultimately what is desired. At any particular value of T, the maximum level of entropy is desired to minimize the objective function, which is consistent with Jaynes’s maximum entropy principle. This principle indicates that of all the possible probability states that satisfy a certain set of constraints, select the one that produces the maximum entropy, because if another state was chosen with lower entropy, then that state would implicitly take into account some additional constraint in order to reduce the entropy. [37]

To determine the entropy, we recognize that $H(X,Y)=H(X)+H(Y\,|\,X)$, where $H(X)=-\sum p(x)\log p(x)$ is the source entropy, which is independent of clustering and can be ignored. Also, $H(Y\,|\,X)=-\sum_x p(x)\sum_y p(y\,|\,x)\log p(y\,|\,x)$, giving

$$H(X,Y)=-\sum_x p(x)\sum_y p(y\,|\,x)\log p(y\,|\,x)$$

Minimizing *F* with respect to the association probabilities $p(y\,|\,x)$ gives the Gibbs distribution

$$p(y\,|\,x)=\frac{\exp\left(-\frac{d(x,y)}{T}\right)}{Z_x}$$

where $Z_x=\sum_y \exp\left(-\frac{d(x,y)}{T}\right)$. Substituting $p(y\,|\,x)$ into the objective function, the optimization then becomes that of minimizing *F*:

$$F^* = \min_{\{p(y|x)\}} F$$

$$F=-T\sum_x p(x)\log Z_x$$

$$F=-T\sum_x p(x)\log\sum_y \exp\left(-\frac{d(x,y)}{T}\right).$$

To determine the optimal location of the clusters, we minimize the Lagrangian with respect to the cluster locations $\{y\}$ by setting its gradients to zero, yielding

$$\sum_x p(x,y)\frac{d}{dy}d(x,y)=0 \qquad \forall y \in Y$$

For squared error distortion, the cluster location is solved as

$$y=\sum_x p(x\mid y)x=\frac{\sum_x p(x)p(y\mid x)x}{p(y)}$$

The standard implementation of Deterministic Annealing uses mass-constrained clustering in order to avoid the dependence on the number of codevectors generated and take into account the presence of other clusters. The system is assumed to have an unlimited supply of codevectors, and the association of element to cluster is scaled by a factor that ends up to being equal to the probability of the cluster. Thus, for mass-constrained clustering, the association probabilities become:

$$p(y\mid x)=\frac{p(y)\exp\left(-\frac{d(x,y)}{T}\right)}{\sum_y p(y)\exp\left(-\frac{d(x,y)}{T}\right)}$$

3.2.3 Deterministic Annealing "Cooling" Algorithm

The traditional deterministic annealing algorithm uses repeated iterations of a modified version (to account for "fuzziness") of *k*-means at each step. The basic two-step process of the *k*-means algorithm essentially determines for all entities which cluster each entity belongs to, then updates the centers of the clusters accordingly.

Initially at high temperature, maximal entropy is favored over minimal distortion, and all the elements are associated with the same, single cluster. As the temperature and the corresponding entropy is lowered, less randomness is allowed and elements become more associated with specific clusters. In order to continue to lower the total distortion, the clusters must be split to allow for the clusters to move towards the elements to which they are more closely associated. Without splitting, the distortion and cluster positions will remain constant with the elements merely being more associated with certain clusters as the entropy is lowered. In [37], these splits were found to occur when the critical temperature, T_c, of a cluster is reached, which is twice the largest eigenvalue $\lambda_{\max}$ of the covariance matrix $C_{x|y}$ of $p(x\,|\,y)$: $T_c = 2\lambda_{\max}$.

While at each temperature there might be a large number of clusters, many of them overlap in position and together produce only a certain effective number of clusters at that level of randomness. Thus, the process yields the effective number of clusters to the problem. As $T \to 0$, no randomness is allowed and each element is associated with its own cluster. In practice, we desire to stop the algorithm at an acceptable tradeoff between distortion and entropy, freeze the current effective number of clusters, and quickly lower the temperature to quench the set so as to obtain hard clustering and make each element be associated with a particular cluster with probability one.

In the general Deterministic Annealing cooling algorithm, either a minimum temperature or maximum number of clusters (or target total distortion amount) is specified that causes the algorithm to stop. However, in this application, those parameters are not known, so instead we use the maximum perpendicular distance $l_{\max}$ as the stopping condition. The adapted algorithm for use in this application is summarized as follows:

```
1. Set limits
    a. $T_{start}$: Starting temperature (just above critical temperature of first cluster)
    b. $\alpha < 1$: Cooling rate
    c. $l_{\max}$: Maximum spread (stopping condition)
    d. $\varepsilon$ : Perturbation between clusters
2. Initialize:
    a. $T = T_{start}$
    b. $K = 1$, $K_{\max} = |X|$ (where $K$ is the number of clusters)
    c. $y_1 = \sum_i x_i p(x_i)$
    d. $p(y_1) = 1$
3. Loop until convergence
    a. Update for $i = 1..K$
        i. Association step: Compute the association probabilities of each $x$ to each $y$:
           $$p(y_i \mid x) = \frac{p(y_i)\exp\left(-\frac{d(x,y_i)}{T}\right)}{\sum_{j=1}^{K} p(y_j)\exp\left(-\frac{d(x,y_j)}{T}\right)}$$, and
           $$p(y_i) = \sum_x p(x)p(y_i \mid x)$$
        ii. Update the centroids: Update $y \in Y$ to minimize
           $D = \sum_x \sum_y p(x,y)d(x,y)$:
           $$y_i = \frac{\sum_x xp(x)p(y_i \mid x)}{p(y_i)}$$
4. Check for convergence, and if not satisfied, go to 3)
5. For each cluster, if all the segments associated with the cluster are within $l_{\max}$ of the cluster, then break. Perform last iteration for $T = 0$ and stop.
6. If $K < K_{\max}$, check for cluster splits.
    a. For $i = 1,..,K$
        i. If $T \le T_{crit,i}$, then
            1. Create new cluster pair:, $y_{K+1} = y_i + \varepsilon$
            2. Divide up probability mass: $p(y_{K+1}) = p(y_i)/2$,
               $p(y_i) = p(y_i)/2$
            3. Increase number of cluster centers: $K = K + 1$
7. Cooling step: $T = \alpha T$
8. Go to 3)
```

To speed up the algorithm, the critical temperature of each cluster is calculated, and the temperature is skipped directly to the largest critical temperature of all clusters without having to incrementally lower in between. Once the split occurs, the pair of old and new clusters are perturbed slightly from each other by an amount ε so that as the temperature decreases, the two points will drift further apart, and separate clusters will be identified for each of the two centroids that used to belong to the same cluster.

Figure 16 shows a sample range of temperatures traversed for a Deterministic Annealing cooling algorithm. The process starts with one cluster at a high temperature ($T = 100$), and gradually decreases to lower the total distortion, until the stopping condition $l_{\max}$ is reached at $T = 24$. For low-cluster-to-element ratios ($1 < K \ll |X|$), the algorithm can be completed in relatively short time. However, in this application, where tight clustering is desired, ($1 \ll K < |X|$), the stopping condition can be reached quicker if a "reverse" approach is used that starts at low temperature and increases.

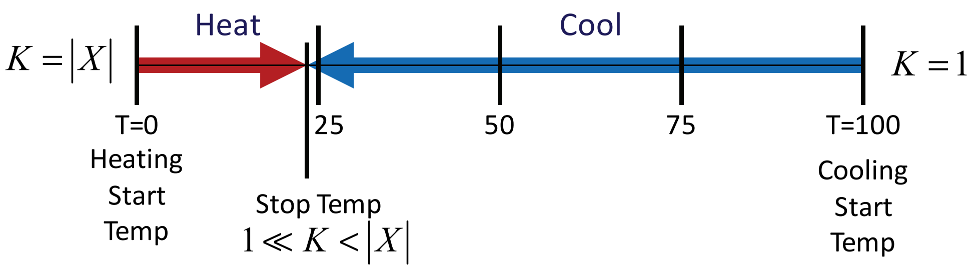


Figure 16. Cooling vs Agglomeration for varying stopping temperatures.

3.3 Deterministic Annealing Agglomeration Algorithm

For applications such as this where tighter clusters are desired, a reverse deterministic annealing or agglomeration approach is more advantageous. The basic principles of deterministic annealing still apply, where at each temperature, an optimization is performed that provides for the maximum entropy and minimal distortion. However, rather than starting with one cluster, the algorithm begins with a cluster for each element. More distortion is allowed as the temperature is raised, and as clusters move toward each other, they are merged.

In [16], this method was used based on a maximum-minimum entropy solution for robot self-organization, and was found to be better at escaping local minima than other fuzzy clustering methods. For our application, there are many advantages to agglomerative deterministic annealing, including the following:

- Faster than cooling algorithm if the number of elements actually included in cluster is small compared to total elements or if there is a low distortion requirement.
- Segments can be merged or eliminated as the algorithm goes along, speeding up computation as the algorithm continues.
- Avoids explosion of clusters as could be experienced with the traditional deterministic annealing using cooling, because there's never more clusters than needed.

The process starts with each segment being its own cluster, and the distortion is gradually increased to allow the clusters to merge. The agglomerate / "heating" algorithm is as follows:

```
1. Set limits
    a. $T_0$ : Starting temperature (slightly greater than 0)
    b. $\alpha < 1$ : Heating rate
    c. $l_{\max}$ : Maximum spread (stopping condition)
    d. $\varepsilon$ : Threshold for merging clusters
2. Initialize:
    a. $T = T_0$
    b. $K = |X|$, $K_{\min} = 1$ (where $K$ is the number of clusters)
    c. $Y = X$  (Each element is its own initial cluster)
    d. $p(y_i) = p(x_i)$
3. Loop until convergence
    a. Update for $i = 1..K$
        i. Association step: Compute the association probabilities of each $x$
           to each $y$:
           $p(y_i \mid x) = \dfrac{p(y_i)\exp\left(-\dfrac{d(x, y_i)}{T}\right)}{\sum_{j=1}^{K} p(y_j)\exp\left(-\dfrac{d(x, y_j)}{T}\right)}$, and
           $p(y_i) = \sum_x p(x) p(y_i \mid x)$
        ii. Update the centroids: Update $y \in Y$ to minimize
           $D = \sum_x \sum_y p(x, y) d(x, y)$ :
           $y_i = \dfrac{\sum_x x p(x) p(y_i \mid x)}{p(y_i)}$
4. Check for convergence, and if not satisfied, go to 3)
5. If $K > K_{\min}$, check for cluster merges:
    a. For $i = 1,..,K$
        i. For $j = i+1,..,K$ If the inter-cluster distance < $\varepsilon$, merge clusters:
            1. Take the average position: $y_i = (y_i + y_j)/2$
            2. Merge the probability mass: $p(y_i) = p(y_i) + p(y_j)$,
            3. Remove merged cluster: $y_j = \varnothing$, $K = K - 1$
6. For each cluster, if all the segments associated with the cluster are more than $l_{\max}$
   of the cluster, then break. Perform last iteration for $T = 0$ and stop.
    a. Keep cluster associations from previous iteration where $l_{ij} \le l_{\max}$
7. Otherwise, continue to raise temperature: $T = T/\alpha$
8. Go to 3)
```

As the temperature increases, the clusters move away from their original positions, generating more distortion. Figure 17 shows the variation of the total distortion (D), number of clusters (K), and the critical temperatures of a sample of seven clusters (C1 through C7). In the figure, the total distortion (D) monotonically increases, while the critical temperatures of the individual clusters fluctuate. As neighboring elements are allowed to influence a cluster more, the cluster center moves towards those elements, temporarily causing an increase in its critical temperature. But as the cluster settles into its new position with more even influence from other elements, its critical temperature decreases. As two clusters approach each other, their centers converge, and their critical temperatures become the same. At this point, the clusters can be merged, eliminating one of the clusters. There is a smooth transition for both the total distortion and the critical temperatures of the clusters as the system temperature rises.

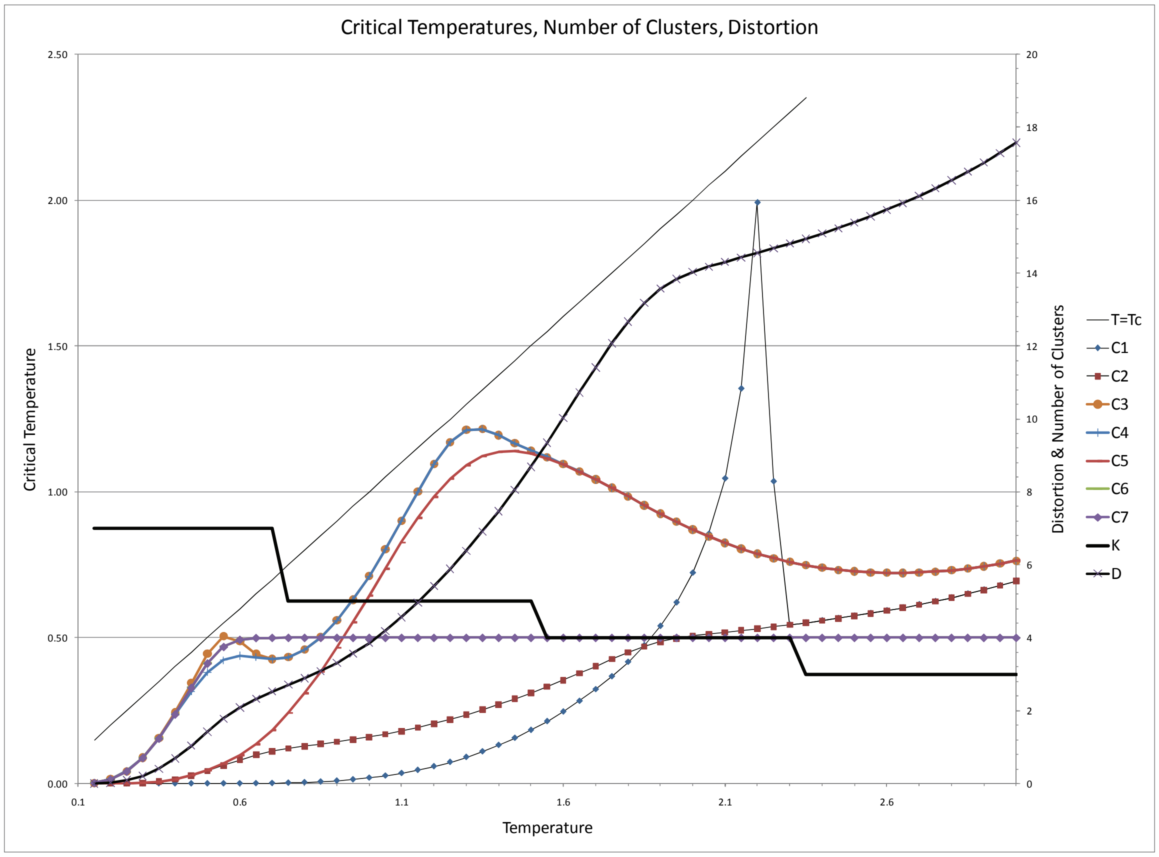


Figure 17. Total distortion and critical temperature for agglomerate algorithm.

In the cooling approach, however, as the temperature is decreased, in order to achieve a lower distortion, the clusters must split. The splitting event is determined when the distortion becomes flat and the minimum is no longer stable. During this time, the system is essentially strained in that it wants to achieve a lower distortion at the lower temperature, but cannot due to the lack of more clusters. Once the split occurs, there is a sharp decrease in distortion. Essentially, the cooling approach hangs on to a cluster longer than it would in the agglomerate approach, which will have more effective clusters at any point in system temperature than the cooling approach. Under the cooling algorithm, cluster splits occur when the system temperature reaches the critical temperature of a cluster, but as shown in Figure 17 for the agglomerate approach, the critical temperatures are always below the system temperature.

3.4 Clustering Comparison

3.4.1 Clustering Similarity Measure

While increased performance is desired, we also wish to ensure that the results of this agglomerate algorithm are similar to the cooling algorithm. Let $X = \{x_i,..,x_N\}$ be the set of all segments being clustered, $A = \{A_1,..,A_m\}$ be the result of a clustering producing *m* clusters, and $B = \{B_1,..,B_n\}$ be the result of a clustering producing *n* clusters, where each cluster $A_i, B_j \subset X$ represents the segments that are associated with the cluster.

For any two clusters A_i and B_j produced from different clustering methods based on the same set of elements, the Jaccard index $J(A_i, B_j)$ is a measure of similarity and diversity between the two, and is defined as:

$$J(A_i,B_j) = \frac{|A_i \cap B_j|}{|A_i \cup B_j|},\ 0 \le J(A_i,B_j) \le 1.$$

If A_i and B_j contain the same set of elements, then $J(A_i,B_j)=1$, while if A_i and B_j have no elements in common, then $J(A_i,B_j)=0$.

Using the $p(y|x)$ produced from each clustering run to provide the association between segments and clusters, we can generate the sets A and B. These associations will be "hard" associations, where each element will belong to exactly one cluster. The resulting clusters from the clustering algorithm are unordered, so it is impossible (and not meaningful) to determine which cluster from one clustering run is associated with the "same" cluster from another run. The overall similarity between the clustering outcomes *A* and *B* is defined as follows:

$$S_{A,B} = \frac{1}{\max(m,n)} \sum_{\forall i,j} J(A_i, B_j)$$

3.4.2 Performance & Similarity Comparisons

The algorithms were implemented using Matlab, and run on a workstation with an Intel Core i7 3.4 GHz CPU with 12GB of RAM.

Several algorithm modifications were evaluated for differences in performance and clustering results. Figure 18 shows a comparison of three of them: 1) Cooling, without skipping temperatures, 2) Cooling, with skipping to next predicted Critical Temperature, and 3) Agglomerate.

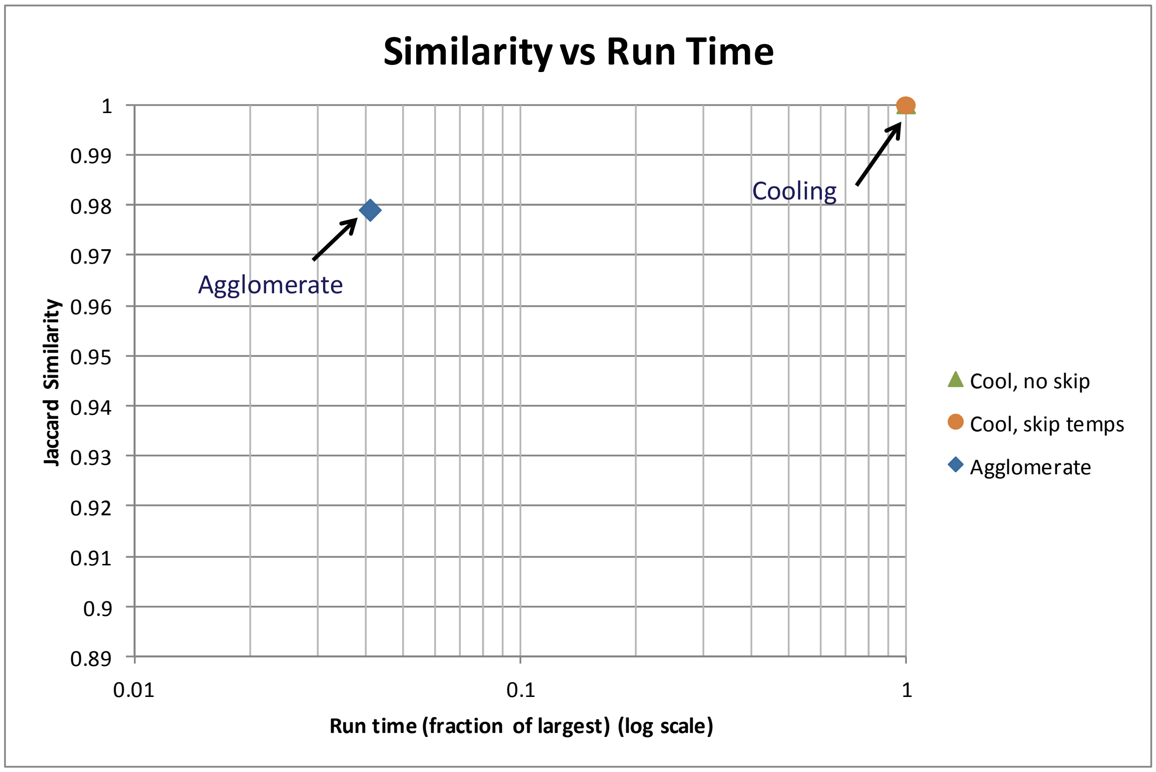


Figure 18. Comparison of Clustering Algorithms.

The clustering results are compared for similarity against the cooling algorithm with no skipping of temperatures. The results show that for cooling, skipping temperatures to the largest critical temperature when the next cluster split will occur does not have a considerable impact on performance nor similarity. However, using the agglomerate algorithm can reduce the runtime to just 4% of the cooling algorithm, with very miniminal difference in clustering results (Jaccard similarity of about 0.98).

The difference in clustering output from the cooling versus the agglomerate approach can be attributed to a couple of factors. First, the starting state is different, leading to variation in the stopping state. Second, the cooling algorithm relies on splitting clusters when the critical temperature is reached, causing jumps in the cluster positions that wouldn't occur in the same way if the temperature direction was reversed with the Agglomerate algorithm.

Figure 19 shows another perspective of the processing time for the traditional Deterministic Annealing cooling method versus the Agglomerative heating approach. For this application, the Agglomerate heating approach is significantly more efficient. While the process starts off slow due to a large number of clusters, it speeds up quickly as clusters are reduced.

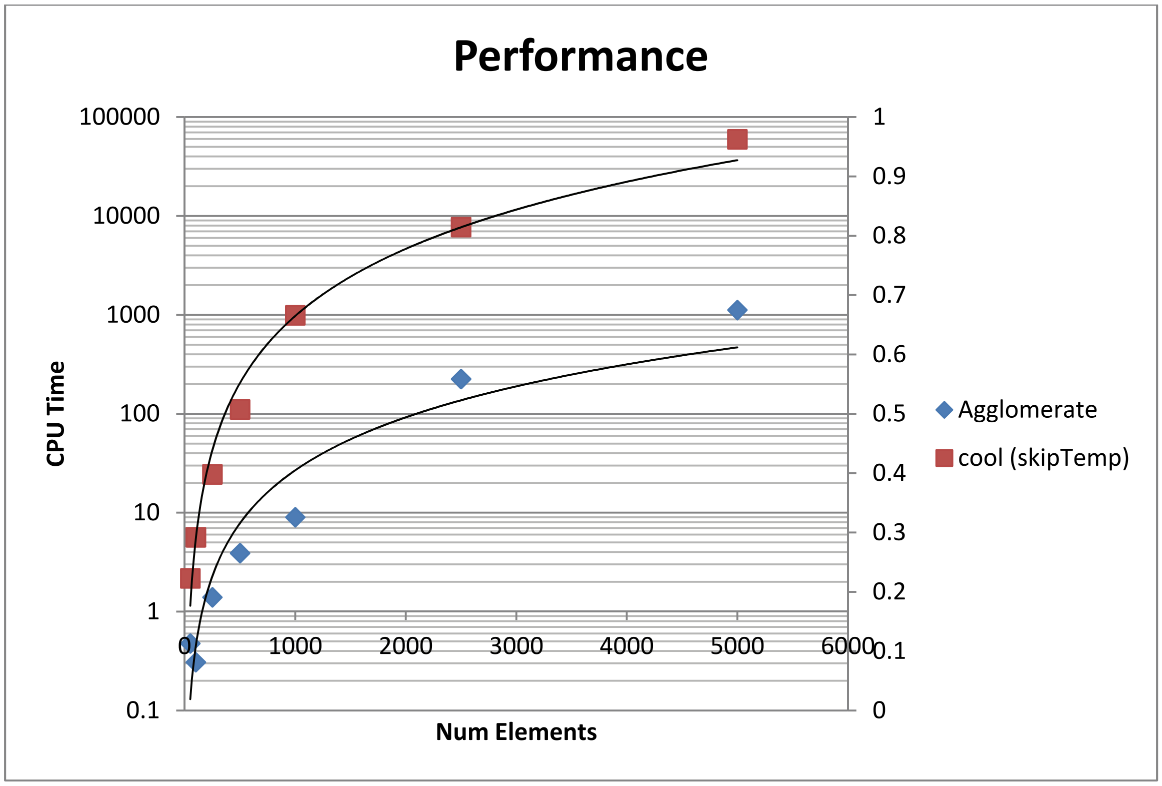


Figure 19. Processing time for Cooling vs Heating Algorithms

3.4.3 Lmax Parameter Selection

Deciding on a value for the stopping condition $l_{\max}$ is important in producing the desired overall clustering result. The parameter $l_{\max}$ must be as monotonic as possible with the overall total distortion D_{total}, otherwise variations in $l_{\max}$ will not produce consistently better or worse results in the same direction. To determine the monotonicity between the two, the total distortion D_{total} is obtained from clustering based on varying values of $l_{\max}$, and the Spearman correlation coefficient ρ is computed for the resulting set. This coefficient provides a measure of the strength of monotone association between two variables, and is obtained by taking the sample of n scores X_i and Y_i, obtaining their ranks x_i and y_i in their ordered positions. The coefficient is defined as follows:

$$\rho = 1 - \frac{6\sum_i (x_i - y_i)^2}{n(n^2 - 1)}$$

For the plot in Figure 20, $l_{\max}$ and D_{total} produce a Spearman's correlation coefficient of $\rho = 0.9929$, indicating that $l_{\max}$ has a strong monotonic association with D_{total} and can be used a stopping condition for the Deterministic Annealing algorithm.

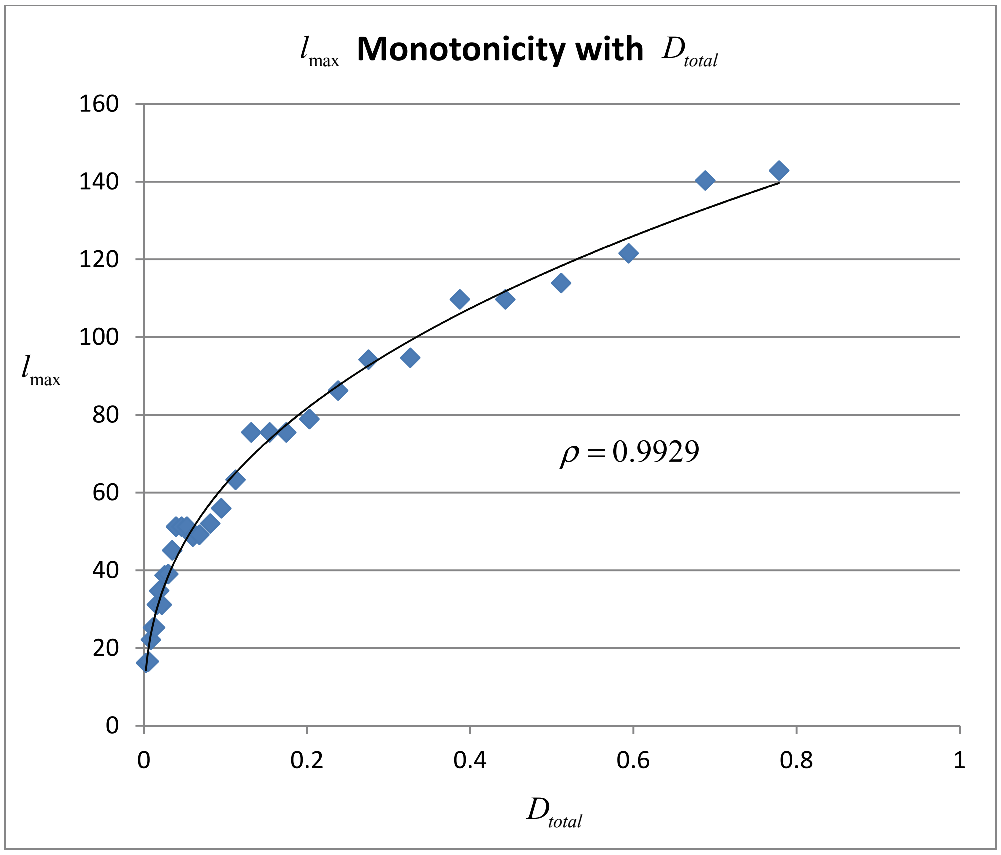


Figure 20. Relation of Lmax to Dtotal.

The quality of the clustering result and impact on the chosen $l_{\max}$ parameter are evaluated against the following inter- and intra-cluster measures:

Inter-cluster variability:

As a measure of the variability between clusters, for a particular clustering result $Y = \{y_j\}$, the cluster isolation is defined as the average distance between a cluster and its next closest cluster:

$$I_Y = \frac{1}{K} \sum_{j=1}^{K} \min_{i \neq j} (y_j - y_i)$$

Intra-cluster variability:

As a measure of the variability within each cluster, for a particular clustering result Y, the overall intra-cluster spread is defined as the maximum distance between the elements and their corresponding cluster centers, averaged across all clusters:

$$P_Y = \frac{1}{K}\sum_{j=1}^{K} \max_{i \in C_j}(y_j - x_i).$$

Ideally, results that produce clusters with greater distinction between clusters and small spread of elements within the clusters are desired. However, these are competing qualities. To compare the results based on these two qualities, they are first normalized, and the intra-cluster quality's scale is reversed so that larger values for both qualities are desired. The *Isolation* $\hat{I}_Y$ normalized against all clustering results $Y \in \mathbf{Y}$ is then:

$$\hat{I}_Y = \frac{I_Y - \min\limits_{Y \in \mathbf{Y}}(I_Y)}{\max\limits_{Y \in \mathbf{Y}}(I_Y) - \min\limits_{Y \in \mathbf{Y}}(I_Y)}$$

The *Compactness* $\hat{C}_Y$ serves as a measure of the intra-cluster quality, and is defined as:

$$\hat{C}_Y = 1 - \frac{P_Y - \min\limits_{Y \in \mathbf{Y}}(P_Y)}{\max\limits_{Y \in \mathbf{Y}}(P_Y) - \min\limits_{Y \in \mathbf{Y}}(P_Y)}$$

So, larger values of both isolation and compactness are desired. In Figure 21, these measures are plotted against the number of clusters that are produced from varying values of $l_{\max}$. Greater $l_{\max}$ produces less clusters, and the clusters tend to be more isolated but less compact. Lesser $l_{\max}$ values produce more clusters that are compact but less isolated.

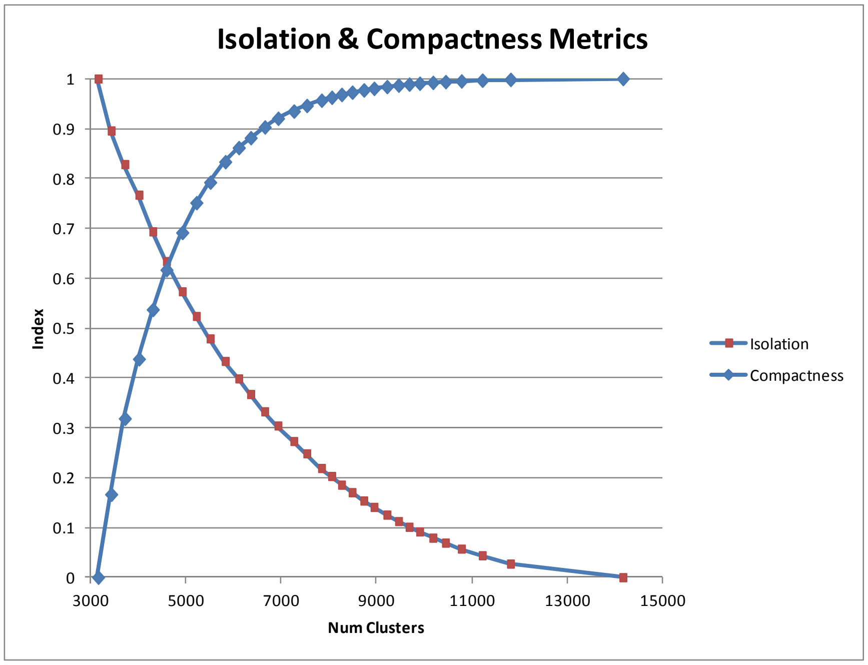


Figure 21. Cluster Compactness & Isolation Metrics.

These two measures are plotted against each other in Figure 22, with the "knee" in the curve observed at around $(C,I)=(0.95,0.25)$, which corresponds to a maximum perpendicular distance of about $l_{\max} \approx 50\text{nmi}$. This is the point where a small movement away from the point will cause a greater improvement in one parameter while drastically sacrificing the other, and would likely be of most interest to a decision maker as a tradeoff between the two measures. However, operational considerations for choosing a value of $l_{\max}$ include the maximum latitudinal deviation from and aircraft's original route, the amount of spread in flight segments that are associated with a cluster, and the amount of angular deviation that might result. It is likely that in actual operations, values for $l_{\max}$ would be further restricted, to about 20 nmi or less.

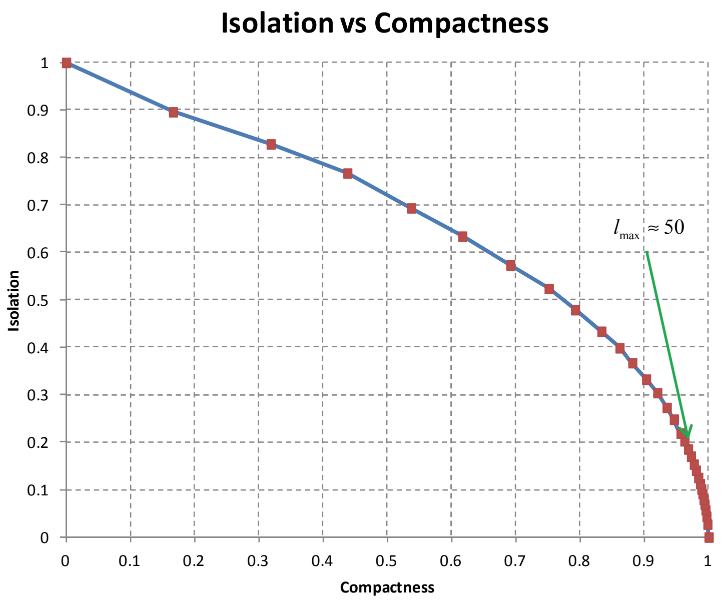


Figure 22. Cluster Isolation vs Compactness.

Chapter 4. Agglomerate Incremental Subset Clustering

4.1 Incremental Clustering Algorithm

In the next two sections, the temporal aspects of the segments are incorporated into the clustering algorithm. Two factors are important in this regard: the effect of time on the existence of a segment in the clustering, and the effect of the temporal separation between consecutive segments within the same cluster on whether the segment should be included with the cluster. This section address the first, with Incremental Clustering, and the following section addresses the second, with Time Partitioning.

If a given segment starts at time t_1 and ends at time t_2, another segment should be considered for being part of the same cluster only if it exists within that time frame. To address the spatial-temporal nature of the segments, the time could be included into the distance measure between clusters. However, this would cause the time and space to influence each other, leading to cases where segments that are very close in time could be clustered together even though their distances are far apart. While there may be distance functions and constraints created that may mitigate this, for this application we desire for the clustering to be influenced only by the spatial properties of the segments that exist during the same time period. So, the approach is to make incremental clusterings, and only include segments for consideration that are active within that time increment.

The second factor, temporal separation, determines whether the time gap between two consecutive segments is sufficiently small to consider them together. Let t_i^{start} and

t_i^{end} be the start and end times of segment *i*, respectively. Then, the following definitions are established:

Definition 4. *For a given cluster with segments sorted by the segment start time* t_i^{start}*, segments i and i+1 are considered consecutive if their ordered positions are adjacent, with* $t_i^{start} < t_{i+1}^{start}$.

Definition 5. *For any two consecutive segments from the same cluster such that* $t_i^{start} < t_{i+1}^{start}$*, the time gap between them is defined to be* $\tau_{i,i+1} = t_i^{end} - t_{i+1}^{start}$.

The time gap is defined with respect to consecutive segment end and start rather than start and start because the segment duration could greatly affect the time gap between the segments.

Definition 6. *For a given maximum time gap threshold,* $\tau_{\max}$*, a* *<u>flow</u>* *is a cluster such that all the time gaps between consecutive segments are not greater than* $\tau_{\max}$.

If the $\tau_{i,i+1}$ between any two consecutive segments *i* and *i*+1 of the same clusters is not greater than $\tau_{\max}$, then the two segments are considered part of the same flow, otherwise the cluster is partitioned off into two separate flows.

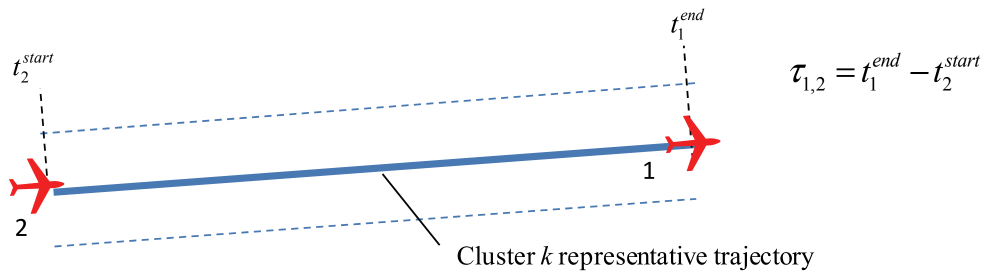

t_1^{end}
t_2^{start}
$\tau_{1,2} = t_1^{end} - t_2^{start}$
1
2
Cluster *k* representative trajectory

Figure 23. Sample flow showing time gap.

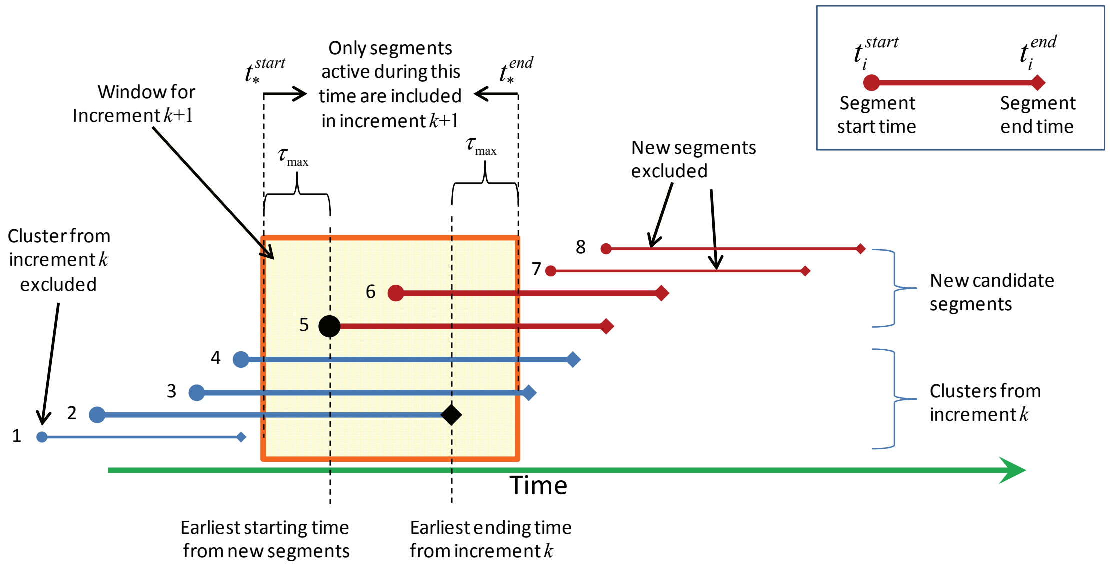
Definition 7. *A clustering increment is a time window defined by the start time* t_*^{start} *and end time* t_*^{end}.

Definition 8. *A segment* x_i *is active within an increment time window if* $t_*^{start} \le t_i^{end}$ *and* $t_i^{start} \le t_*^{end}$.

The Incremental Clustering algorithm operates by identifying segments that are active within a time window, clustering them, and advancing the time window for the next increment. Figure 24 shows how segments are chosen for clustering within increment $k+1$. The output from the clustering performed in increment *k* are the cluster numbers 1-4. Segments 5-8 are the next set of segments for consideration to be included into increment $k+1$.

The start of the time window for increment $k+1$, t_*^{start}, is set to be $\tau_{\max}$ before the start of the earliest new segment (segment 5). Any clusters (cluster 1) from increment *k* that end before this time have no chance of being associated with the new segments and can be excluded as being completed.

The end of the time window for increment $k+1$, t_*^{end}, is set to be $\tau_{\max}$ beyond the remaining clusters from increment *k* that has the earliest cluster ending time (cluster 2). The cluster with the earliest ending time is chosen so that the minimum set of temporally overlapping segments are used, and the threshold amount $\tau_{\max}$ is added because segments from increment $k+1$ that start before this time have the potential of being combined as part of that cluster. New segments (segments 7, 8) that start after the end of the time window t_*^{end} are not yet temporally relevant to those under consideration, and will not be included in this increment. So, for increment $k+1$, the clustering algorithm will be performed for the segments contained in clusters 2-4 retained from increment *k* along with the new segments 5, 6.



t_i^{start}
t_i^{end}
Segment start time
Segment end time
t_*^{start}
t_*^{end}
Only segments active during this time are included in increment *k*+1
Window for Increment *k*+1
$\tau_{\max}$
$\tau_{\max}$
New segments excluded
Cluster from increment *k* excluded
New candidate segments
Clusters from increment *k*
1
2
3
4
5
6
7
8
Time
Earliest starting time from new segments
Earliest ending time from increment *k*

Figure 24. Segments included in Incremental Clustering.

The algorithm for Incremental Clustering is summarized below. (The Time Partitioning step is explained in the subsequent section.)

1. Set limit: $\tau_{\max}$
2. Initialize:
    a. $X_{complete} = \varnothing$
    b. $t_*^{start} = t_1^{start}$
    c. $t_*^{end} = t_1^{end} + \tau_{\max}$
    d. Obtain initial set of segments: $X_0 = \{x_i : t_i^{start} < t_*^{end}\} = \{x_1..x_j\}$
3. Perform clustering for increment *k* to obtain output set of clusters Y_k
4. Perform Time Partitioning
5. Identify active segments for increment *k*+1:
    a. Set the window start time to be $\tau_{\max}$ before the start of the earliest new segment: $t_*^{start} = \min(t_i^{start}) - \tau_{\max}, i > j$
    b. Exclude clusters from increment *k* that end before t_*^{start}, and their respective segments, as these are considered complete: $Y_{k,exclude} = \{y_i \in Y_k : t_i^{end} < t_*^{start}\}$ and $X_{k,exclude} = \{x_i \in Y_{k,exclude}\}$. Append this

to the completed set of clusters: $Y_{complete} = Y_{complete} \cup Y_{k,exclude}$ and $X_{complete} = X_{complete} \cup X_{k,exclude}$.

   c. Retain remaining clusters from increment *k*: $X_{k,retain} = X_k \setminus X_{k,exclude}$
   d. Set the window end time to be $\tau_{\max}$ after the end time of the earliest ending cluster from increment *k*: $t_*^{end} = \max(t_i^{end}) + \tau_{\max}$, $x_i \in X_{k,retain}$
   e. Find new segments that start before t_*^{end}: $X_{new,retain} = \{x_i : t_i^{start} < t_*^{end}\}, i > j$
   f. Segments to cluster in increment $k+1$: $X_{k+1} = X_{k,retain} \cup X_{new,retain}$

6. Repeat steps 3-5 until complete
7. Assemble completed set of output clusters: $Y_{complete} = Y_{complete} \cup Y_k$

4.2 Partitioning Clusters by Time

Once the clustering by geometry in Step 3 of the Incremental Clustering algorithm is completed, the resulting clusters need to be further split based on time in Step 4. We are interested in identifying a continuous stream of flights through a corridor, and so will consider a flight to be part of a cluster as long as it enters the corridor within $\tau_{\max}$ from which the previous flight leaves the corridor. If the corridor is unused for more than $\tau_{\max}$, it will be considered closed once the last flight leaves.

Although segments from the previous increment that end before $\tau_{\max}$ of the start of the new segments were already excluded from the current increment, time partitioning is still needed after clustering because cluster associations might have changed from the current round of clustering, or new segments could have been associated together that have time gaps greater than $\tau_{\max}$.

Figure 25 shows an example with 6 segments that belong to the same cluster resulting from the geometric clustering done using Deterministic Annealing. During the Time Partitioning phase, segments 4-5 would be grouped together since they overlap in their occupancy times and $\tau_{i,i+1} < 0 < \tau_{\max}$. Segment 3, while it ended before segment 4 started, would still be considered part of the segments 4-6 since although greater than zero, the time gap is still less than $\tau_{\max}$, $0 < \tau_{3,4} < \tau_{\max}$. However, segments 1 and 2 will be partitioned off separately since their time gaps from segment 3 are too great, $0 < \tau_{\max} < \tau_{i,i+1}$.

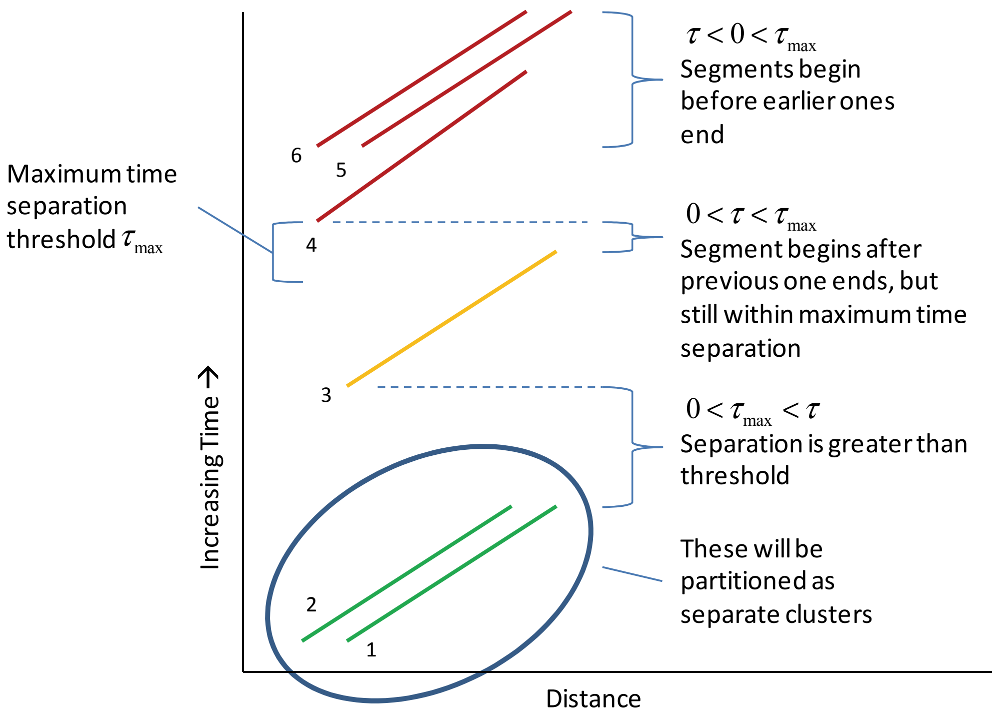

$\tau < 0 < \tau_{\max}$
Segments begin before earlier ones end
$0 < \tau < \tau_{\max}$
Segment begins after previous one ends, but still within maximum time separation
$0 < \tau_{\max} < \tau$
Separation is greater than threshold
These will be partitioned as separate clusters
Maximum time separation threshold $\tau_{\max}$
Increasing Time →
Distance
1
2
3
4
5
6

Figure 25. Time partitioning concept.

The Time Partitioning algorithm is as follows:

1. For each cluster do:
   a. Determine the time gaps between each consecutive segment in the cluster: $\tau_{i,i+1} = t_i^{end} - t_{i+1}^{start}$.
   b. Let $G = \{i : \tau_{i,i+1} > \tau_{\max}\}$ represent the set of segment indices that have gaps with the subsequent segment greater than $\tau_{\max}$. For each gap $g_k \in G$,

partition the group of segments into separate clusters:
$X_k = \left\{ x_{g_{k-1}+1} .. x_{g_k} \right\}, X_{k+1} = \left\{ x_{g_k+1} .. x_{g_{k+1}} \right\}$.

4.2.1 Clustering & Time Partitioning Samples

To show the clustering and partitioning results, a sample of segments from the southwest region of the United States was clustered. The results of the geometrical clustering phase are shown in Figure 26 with the following parameters:

- Lateral spreading threshold: $l_{\max} = 20$ nmi
- Maximum time separation: $\tau_{\max} = 20$ min
- Minimum number of segments per cluster: $n_{\min} = 3$

Cluster centers are shown as bold dotted lines, and for ease of readability, only those that have at least the minimum number of segments are shown. The segments that belong to these clusters are shown in color as thin solid lines, while segments that don't belong to any qualifying clusters are shown in grey.

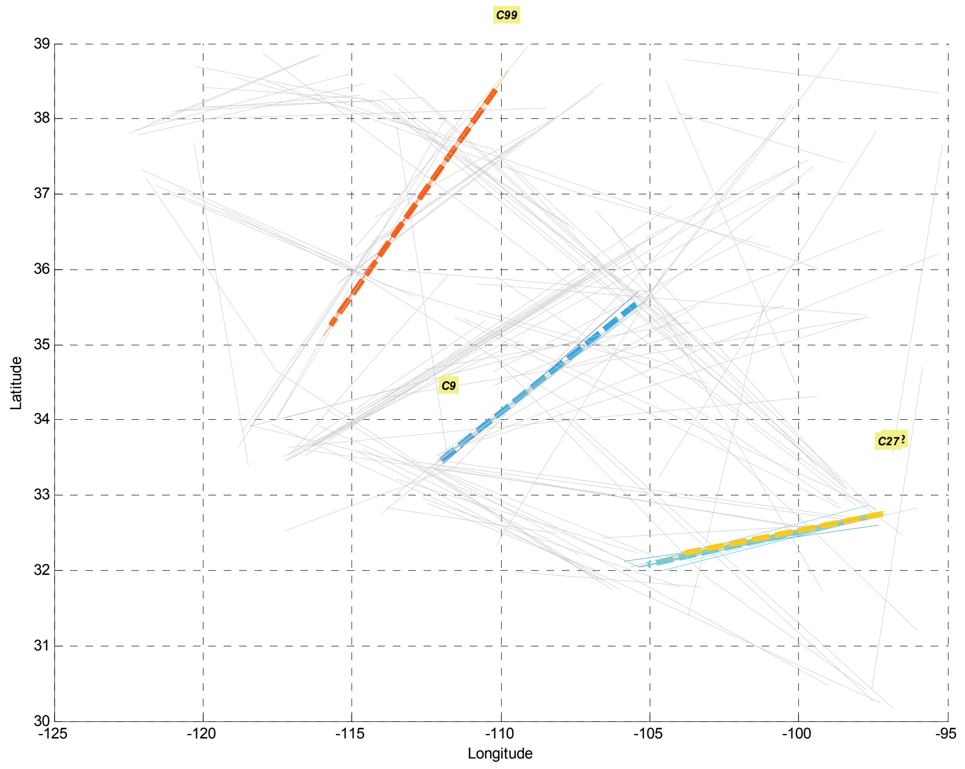


Figure 26. Spatial clustering results (lat/lon).

To examine the distribution in time, the same segments above are plotted with time on the vertical axis as shown in Figure 27, and colored according to the segments' position in time (blue is earlier, red is later). Note that cluster C10 on the left consists of 3 segments, one in green ("A"), and two in blue. The time between the end of the latest blue segment and the start of the green segment is about 40 minutes, which is greater than the threshold of $\tau_{\max} = 20$ minutes. Thus, in the time partitioning phase, the green segment will be separated from the other two into its own cluster. However, because of the display criteria that each cluster have at least 3 segments, this and the other 2 segments will not be shown in the next figure.

Cluster C8 also has a segment in red that is separated from the remaining ones in teal and blue. With the red segment partitioned into a separate cluster, the remaining segments will be shown in the subsequent diagrams since there are still 3 remaining segments.

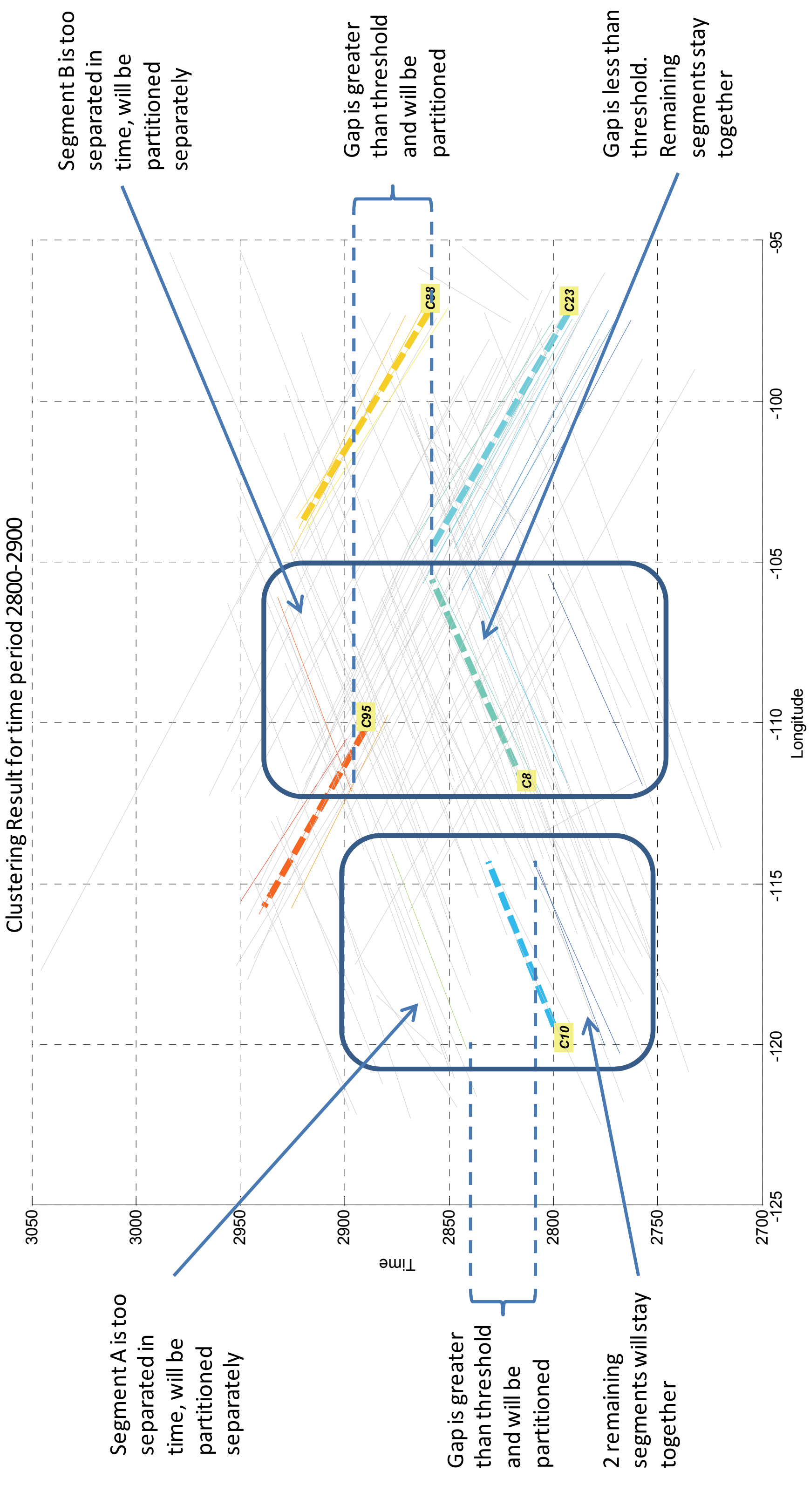


Figure 27. Spatial clustering results (showing time domain).

Figure 28 shows the results upon completion of the time partitioning phase. Note that the former cluster C10 with 3 segments is no longer shown.

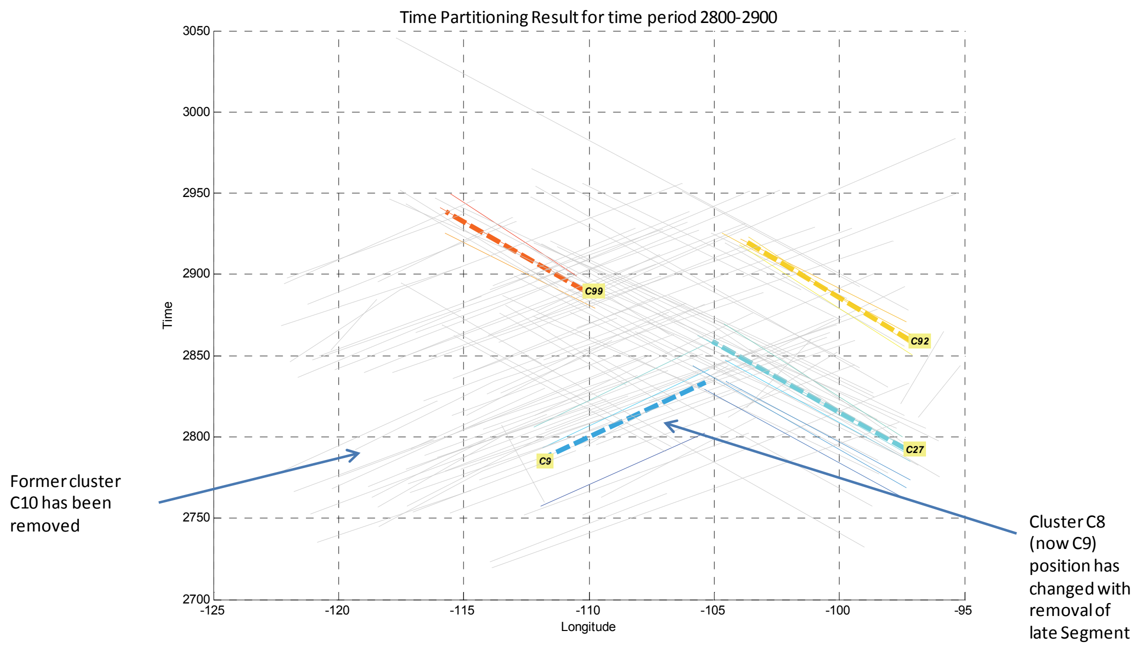


Figure 28. Time partitioning results.

4.2.2 Tmax Parameter Selection

For the time partitioning phase, the choice of the parameter $\tau_{\max}$ affects how densely grouped the segments are in time. Large values of $\tau_{\max}$ allow more time gap between flights in the cluster and decrease the overall flow rate through the cluster, while small values of $\tau_{\max}$ increase the flow rate, as shown in Figure 29. The value of $\tau_{\max}$ also indirectly affects the number of clusters. Large values of $\tau_{\max}$ tends to keep clusters intact, producing larger clusters, while small values of $\tau_{\max}$ cause more partitioning of clusters and produce smaller clusters.

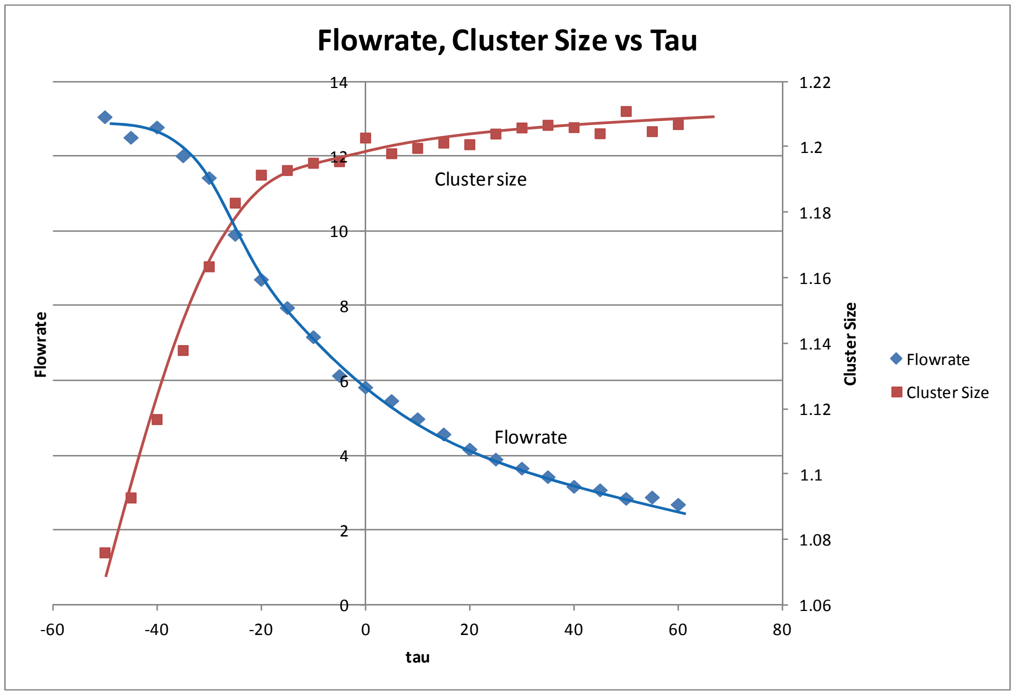


Figure 29. Flowrate and Cluster size vs tau.

For the sake of efficiency in managing the flows of traffic, larger clusters with higher flowrates are desired. To tradeoff between these two opposing criteria, time partitioning was performed for varying values of $\tau_{\max}$, and the corresponding flowrate and cluster size for each $\tau_{\max}$ were plotted against each other in Figure 30. The knee in the curve corresponds to a value of $\tau_{\max} \approx -20\,\text{min}$. Slightly larger $\tau_{\max}$ will cause the flowrate to drop quickly, while slightly smaller values will cause the cluster size to drop quickly.

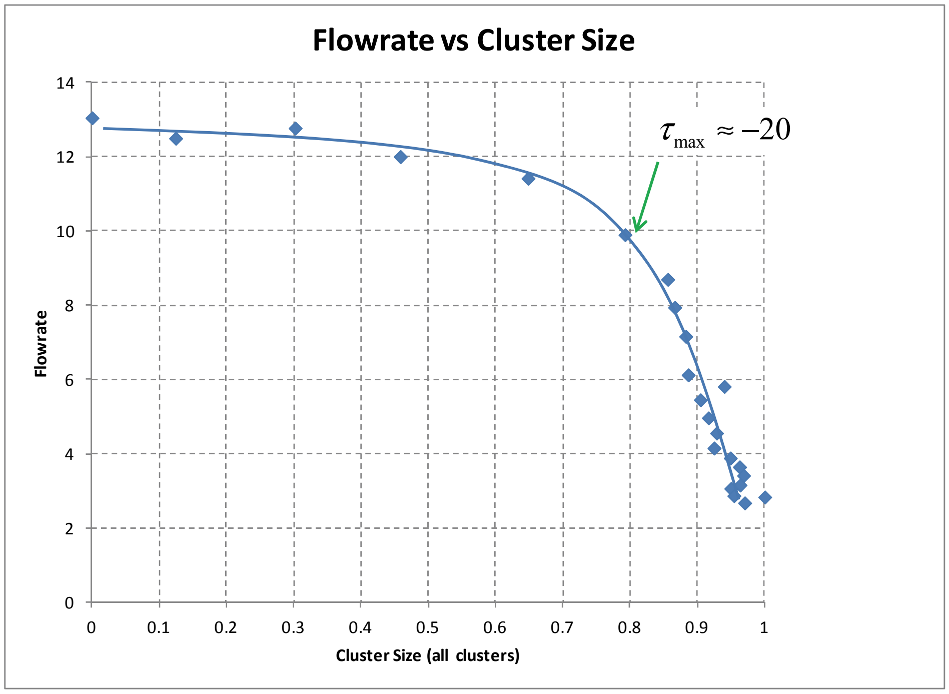


Figure 30. Flowrate vs Cluster Size.

Also of interest is how selection of $\tau_{\max}$ will also impact the number of non-singleton clusters (those with two or more elements), and how long the same geographic cluster might occur again at a later time. If the last segment assigned to the cluster leaves the tube at time t_{end}^{last}, the fact that the cluster is partitioned off means that there are no other segments arriving until at least $t_{end}^{last} + \tau_{\max}$. Let g be the time gap between $t_{end}^{last} + \tau_{\max}$ and the arrival of the next flight at time $t_{end}^{last} + \tau_{\max} + g$. If g is very small, it would not be worthwhile to close the tube, only to have to reopen it just a bit later. Figure 31 shows the number of non-singleton clusters and g for varying values of $\tau_{\max}$. The data shows that g remains fairly unaffected by $\tau_{\max}$, remaining constant at around 3.5 hours, meaning that a value of $\tau_{\max}$ can be chosen without fear that it would cause premature closing of a flow. The graph also shows that as long as $\tau_{\max} \geq -20$, the choice of $\tau_{\max}$ will not cause an increase in non-singleton clusters.

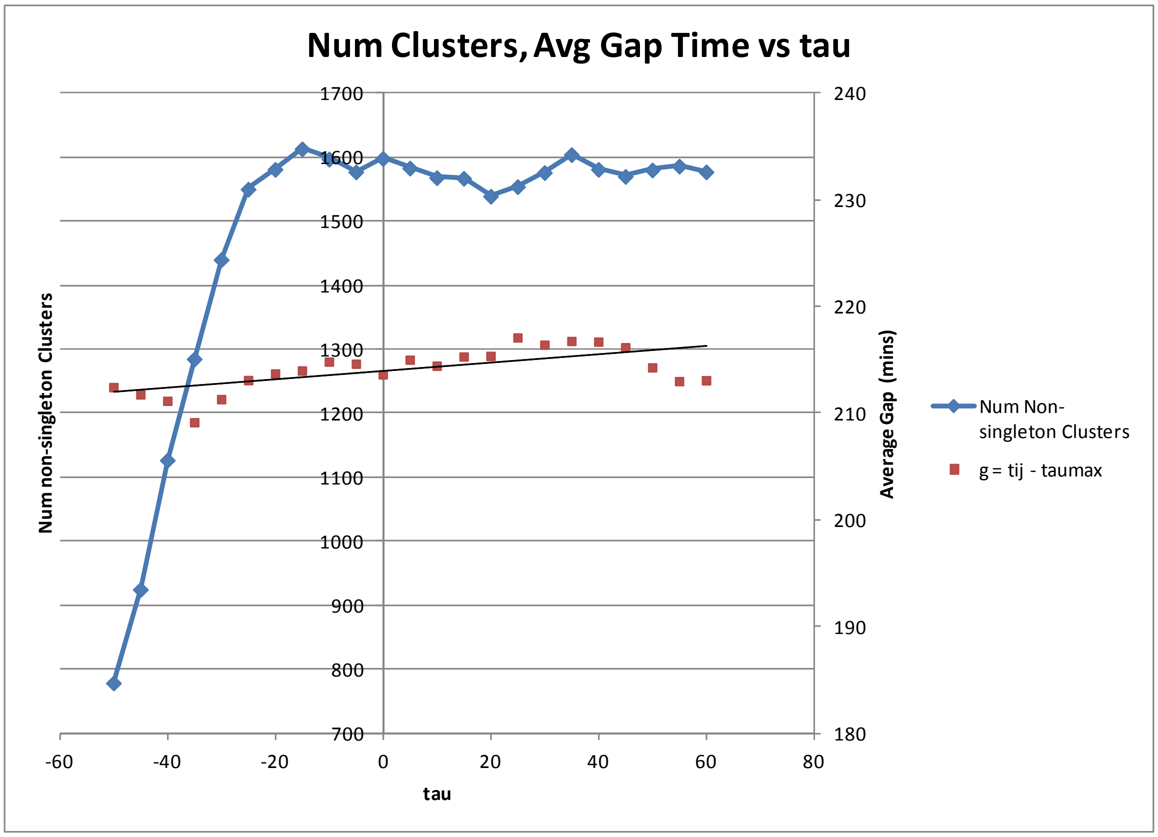


Figure 31. Num Clusters, Gap vs Tau.

4.3 Subset Clustering

To further improve performance, we modify Step 3 in the Incremental Clustering algorithm to cluster only the elements from the previous clustering that are close in proximity to the new elements, and combine the results in the end.

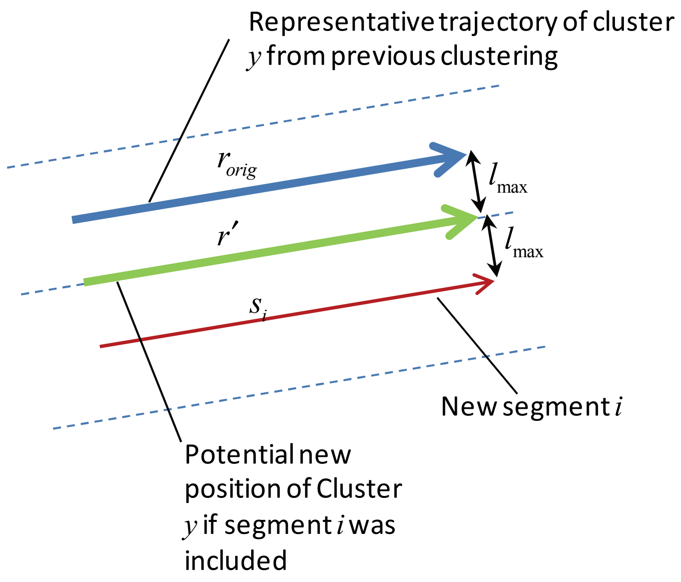

Representative trajectory of cluster
y from previous clustering
r_{orig}
$l_{\max}$
r'
$l_{\max}$
s_i
New segment i
Potential new position of Cluster y if segment i was included

Figure 32. Influence of new segments on previous clusters.

In Figure 32, r_{orig} is the representative trajectory of a cluster from previous clustering, and s_i is a new segment being clustered. If a segment is associated with an existing cluster, the cluster's representative trajectory will move towards the segment by some amount, depending on the "mass" of segments the cluster currently has. The maximum movement will occur when there is only one segment, in which case the representative trajectory will move to the average position of the two segments. Due to the requirement that all segments are within $l_{\max}$ of the representative trajectory, the new segment cannot be more than $l_{\max}$ of the new position of the representative trajectory, and thus cannot be more than $2l_{\max}$ from the original representative trajectory.

The localized clustering algorithm will be based on this fact, and include only clusters from previous clustering that are within $2l_{\max}$ of each new segment. Let $S^{prev} = \{s_i^{prev}\}$ be the set of segments from previous clustering, which are grouped into $C^{prev} = \{C_k^{prev}\}$ clusters, and $S^{new} = \{s_i^{new}\}$ be the set of new segments to be clustered with the previous ones. The Subset clustering algorithm is as follows:

1. For each new segment $s_i^{new} \in S^{new}$, identify the set of clusters from previous clustering $\mathrm{C}_{i,close}^{prev} = \{C_k^{prev} : l_{i,k} < 2l_{\max}\}$ that has a perpendicular distance to the new segment of less than $2l_{\max}$. These are considered the "close" clusters.
2. For each new segment s_i^{new}, identify the set of segments $S_{i,close}^{prev} = \{s_j^{prev} \in \mathrm{C}_{i,close}^{prev}\}$ that are represented by the previous close clusters. Let $S_{close}^{prev} = \bigcup_i S_{i,close}^{prev}$ be the set of all segments from previous clustering whose clusters are close to any of the new segments.
3. Let $S^{subset} = S^{new} \bigcup S_{close}^{prev}$ be the union of the set of new segments, and previous segments whose clusters are close.
4. Perform clustering for S^{subset}.
5. Combine the sets S^{subset} and $S^{prev} \setminus S_{close}^{prev}$.
6. Continue with incremental clustering algorithm.

Figure 33 shows a sample of subsetting, where the colored dotted lines are clusters from the previous iteration of clustering, and the solid black lines are new segments being added. Grey lines are segments from the previous iteration that are being excluded in the current increment of clustering. Cluster 12 (in red) is far from any of the new segments, and so will be excluded from the subset. The remaining two clusters (orange and light blue) are close to new segment 1, so will be included in the subset with the new segments 2 and 3 that will actually be clustered.

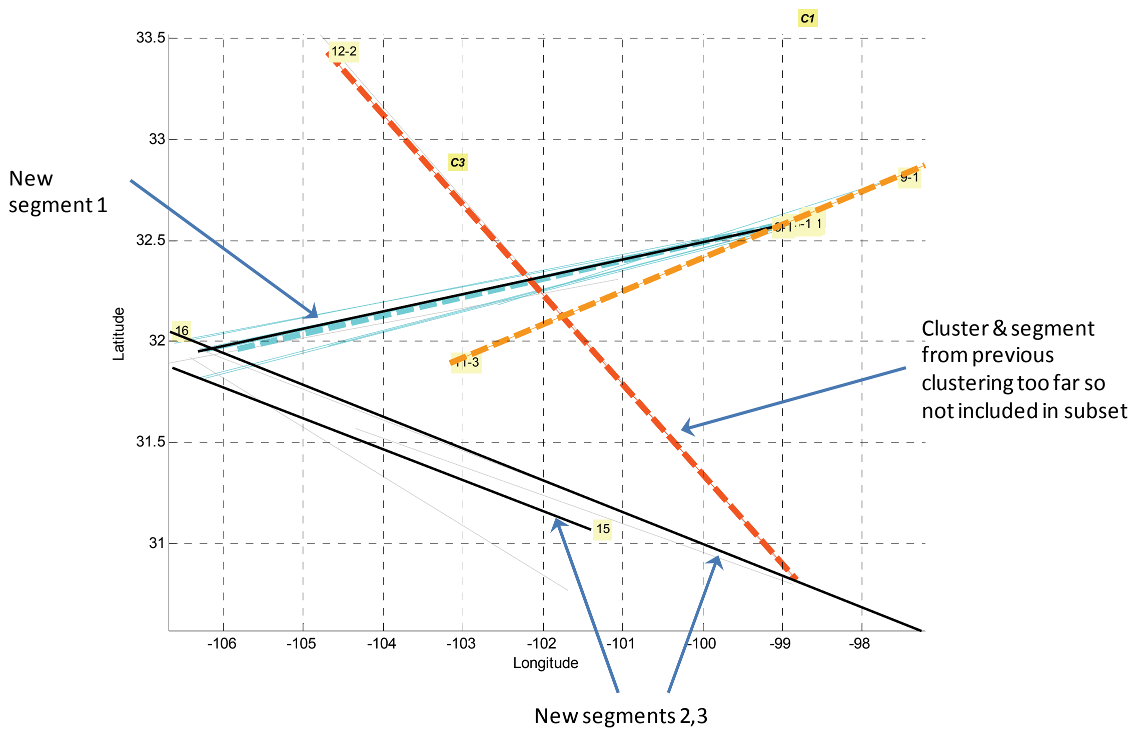


Figure 33. Clustering of subset.

4.3.1 Performance & Similarity Comparisons

The Subset clustering algorithm was evaluated against the traditional (cooling) Deterministic Annealing algorithm and the Agglomerate algorithm for both similarity using the Jaccard Index, and CPU performance for the same sample. Figure 34 shows a comparison of the algorithms.

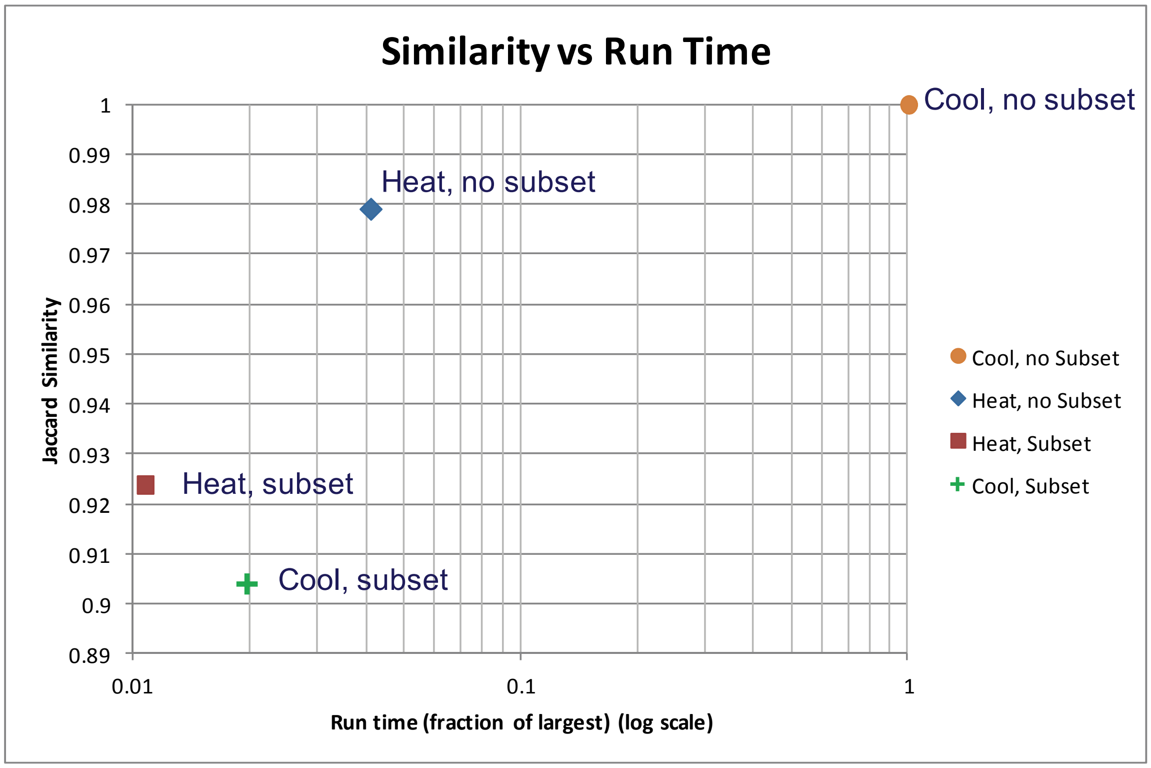


Figure 34. Comparison of Clustering Algorithms.

The result of the cooling Deterministic Annealing algorithm with full clustering (Subset clustering not used) was used as the baseline for the comparisons, which is given a Jaccard Similarity index of 1 by default. Recall that doing agglomerate clustering can reduce the runtime to just 4% of the cooling algorithm, with very miniminal difference in clustering results (Jaccard similarity of about 0.98). If we use the agglomerate algorithm and only cluster the subset of previous clusters, the run time can further be reduced to just 1% of the cooling algorithm, with still a high similarity of 0.92. Using the Subset technique on the cooling algorithm was worse, resulted in longer runtime and less similarity compared to Agglomerate Subsetting. Results from using subset clustering vary slightly from full clustering due to the interaction between the segments at the boundaries of the subset and the surrounding segments that are not included in the subset. The full clustering can account for this interaction, while the subset clustering cannot. However, the dramatic improvement in performance is worth the minimal difference in clustering results.

Figure 35 shows the number of total segments (in blue) that are included for each time increment in the Incremental clustering Algorithm. This includes segments from the previous round of clustering, as well as new segments for the increment. At its peak, each increment contains about 750 segments. However, only about 1/10th of them actually need to be clustered as part of the subset (in red).

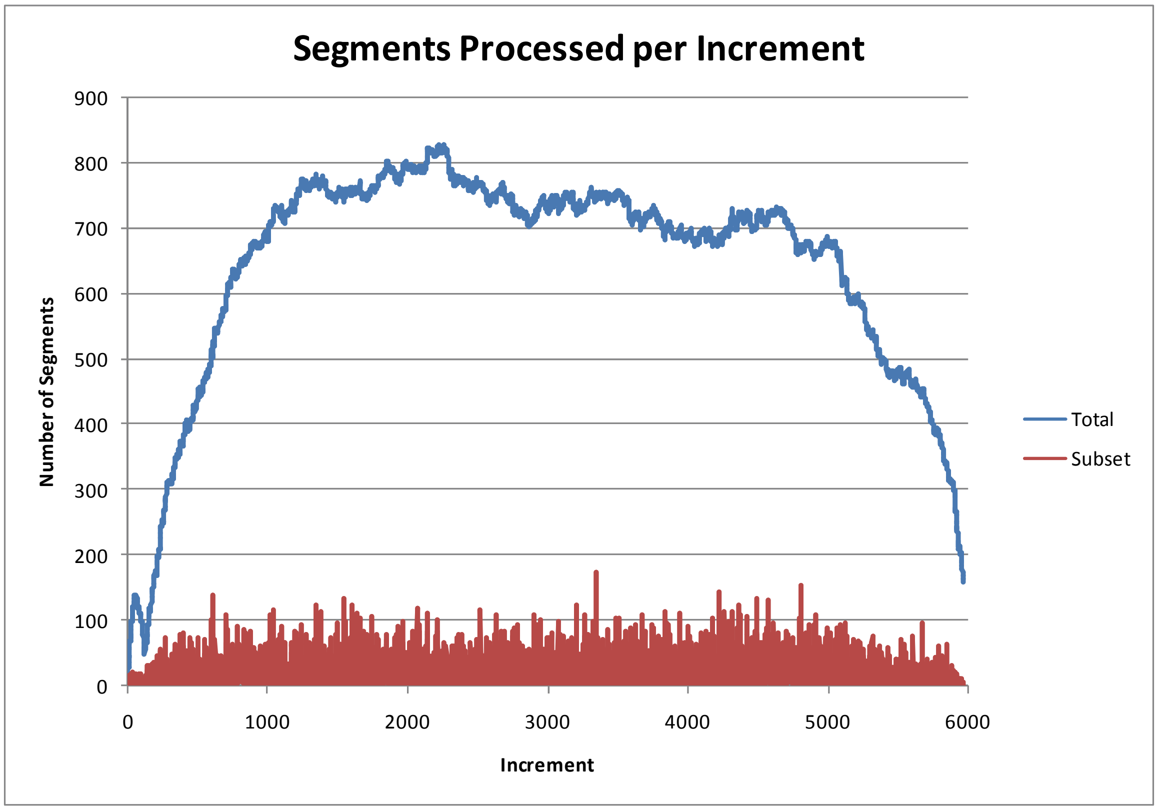


Figure 35. Segments Processed per Clustering Increment.

4.4 Clustering Results

Prior to clustering, the flight segments were filtered to include only segments in the following ranges:

Altitude range: ≥ 29,000 ft (enroute traffic)

Segment duration: 30 minutes to 180 minutes

Maximum distance: 2,000 nmi

For now, we investigate only the enroute portion of traffic, and ignore short segments since implementing traffic management initiatives would not be worth the effort. Segments that are too long are also ignored since they are less likely to occur and span too great of a region.

The entire clustering algorithm was implemented in Matlab, and ran on a Windows 7 workstation with an Intel Core i7 3.4 GHz CPU with 12 GB RAM. Clustering all of the flight segments for the entire CONUS, which consisted of 14,160 segments over 24 hours of data, using $l_{\max} = 20$ nmi and $\tau_{\max} = -20\,\text{min}$ took 16 minutes. This very fast performance means that it is very adaptable to real-time clustering of traffic.

Some intuition on the traffic patterns can be gathered from the directional information that is imbedded in the segment definitions. Figure 36 shows the traffic in the continental United States (CONUS) for a day and a half during clear weather, with each flight segment colored according to time, which is shown on the Z axis. The evening between the days is in the middle, when red-eye flights departing from the west coast to the east coast can be seen starting around 1AM Eastern Time (10PM Pacific Time). Also note that traffic on the east coast starts around 6:30AM, gradually increases west-ward, and likewise end earlier in the east and move towards the west as the day ends.

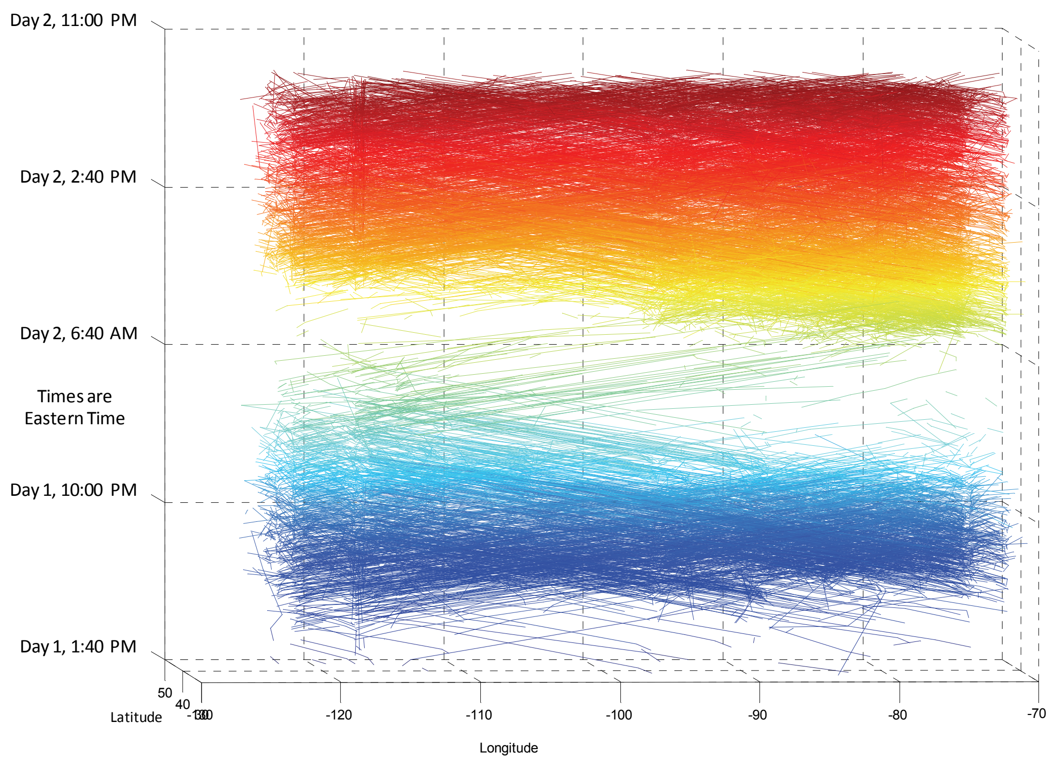


Figure 36. CONUS flight segments

Figure 37 shows the average heading for each hour of the day, along with a histogram of the number of segments that start within the hour for a typical clear weather day. (For ease of viewing East/West directions, the heading here is defined with West corresponding to -90° and East 90° from North.) We can see that the traffic day begins in the 6 o'clock hour Eastern Time, when traffic begins to grow and heads predominately westbound. As the morning develops, eastbound traffic from the west coast increases as the morning over there starts, and the peak traffic is found between the hours of 9AM-9PM, with the traffic not having a clear prominent average heading. As traffic decreases after 9PM, the red-eye flights from the west coast traveling eastbound can be seen during the low traffic hours of around 1AM through 6AM.

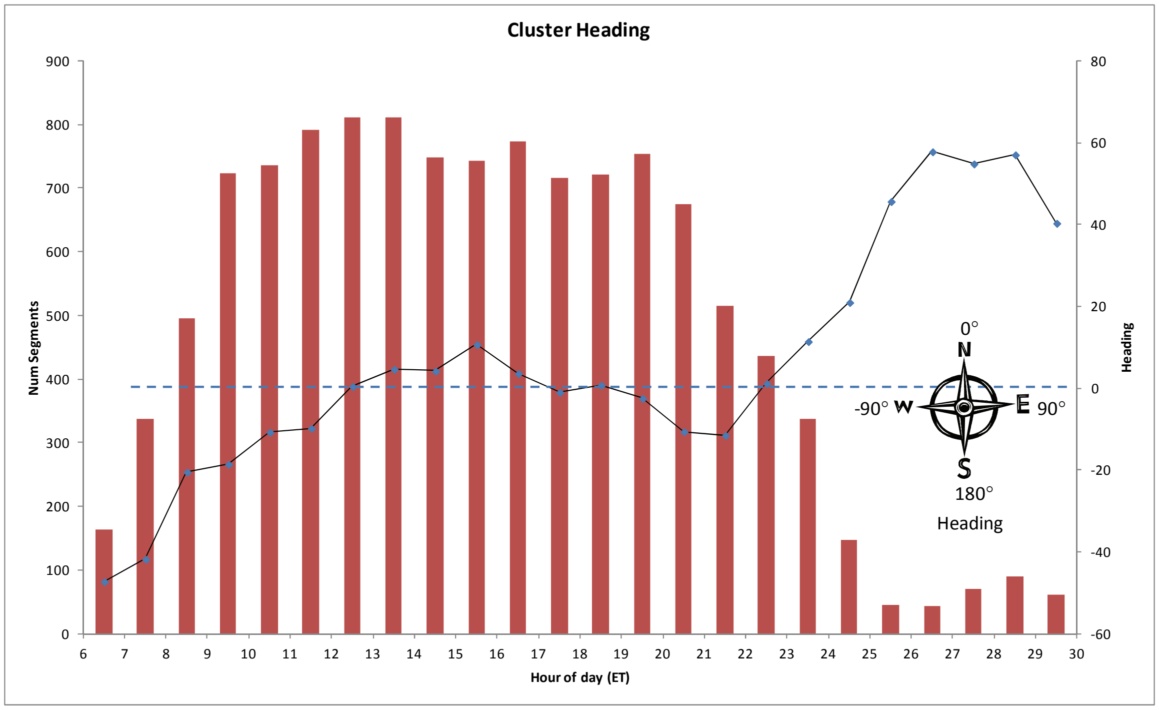


Figure 37. Cluster Heading & traffic volume.

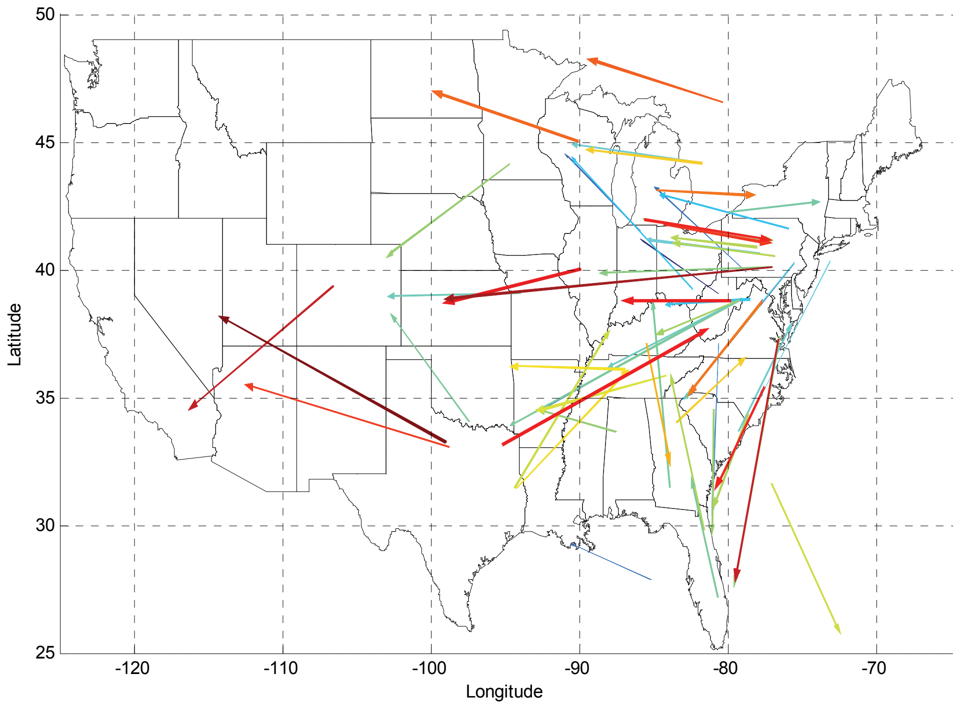


Figure 38. Morning Flows.

Figure 38 shows the flows resulting from the clustering algorithm, for the morning hours between 6:30AM - 9:30AM ET with at least 3 flights. The colors of the arrows indicate time of day, with blue at the start, yellow/green in the middle, and red at the end. The width of the arrows indicates the amount of aircraft contained in the flow. Note the predominance of the traffic at the east coast, with their heading mainly towards the west. Figure 39 shows the flows between 9:30AM - 9:30PM ET with at least 3 flights, when there is high volume and no distinct overall heading, and Figure 40 shows the evening flows between 1:00AM-6:00AM ET with at least 2 flights. These show the expected traffic flow patterns, with concentrations of flows located in the mid Atlantic region, and along the eastern corridor. In addition to providing information on historical major flows of traffic to aid the FAA in designing airways and pilots in planning around congested routes, knowledge of the flows of traffic can also be used in real-time traffic management. Pilots may elect to avoid the congested areas, or join the tubes that may be established during those locations and times. Traffic managers can plan traffic management initiatives accordingly.

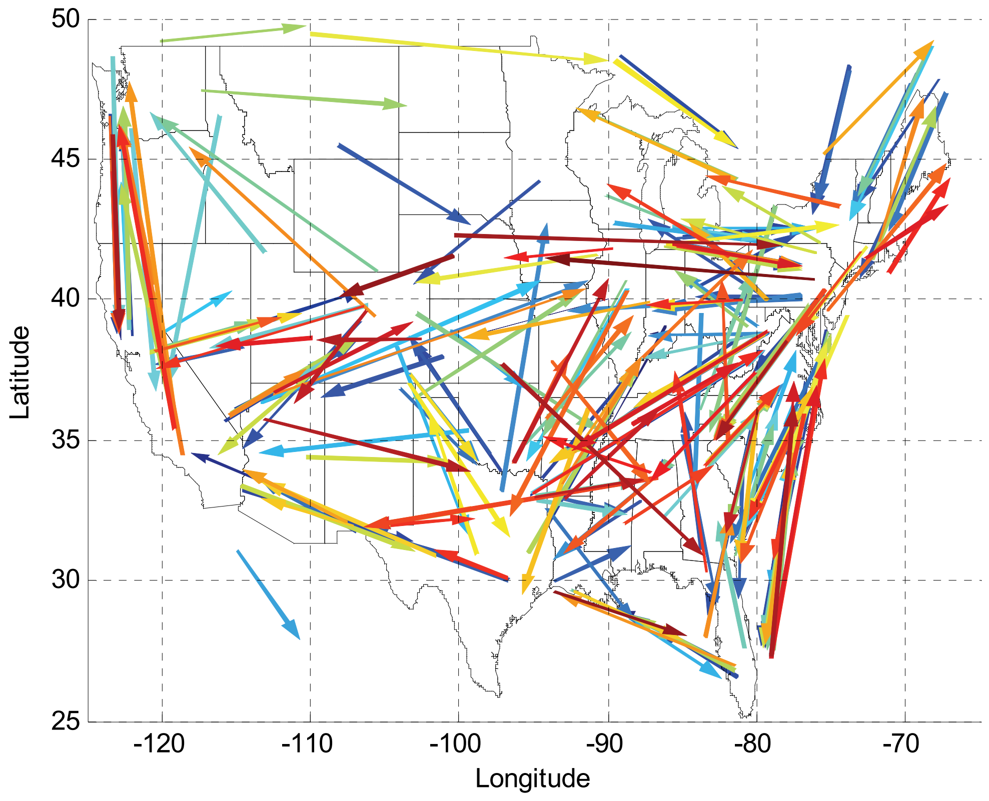


Figure 39. Peak Flows.

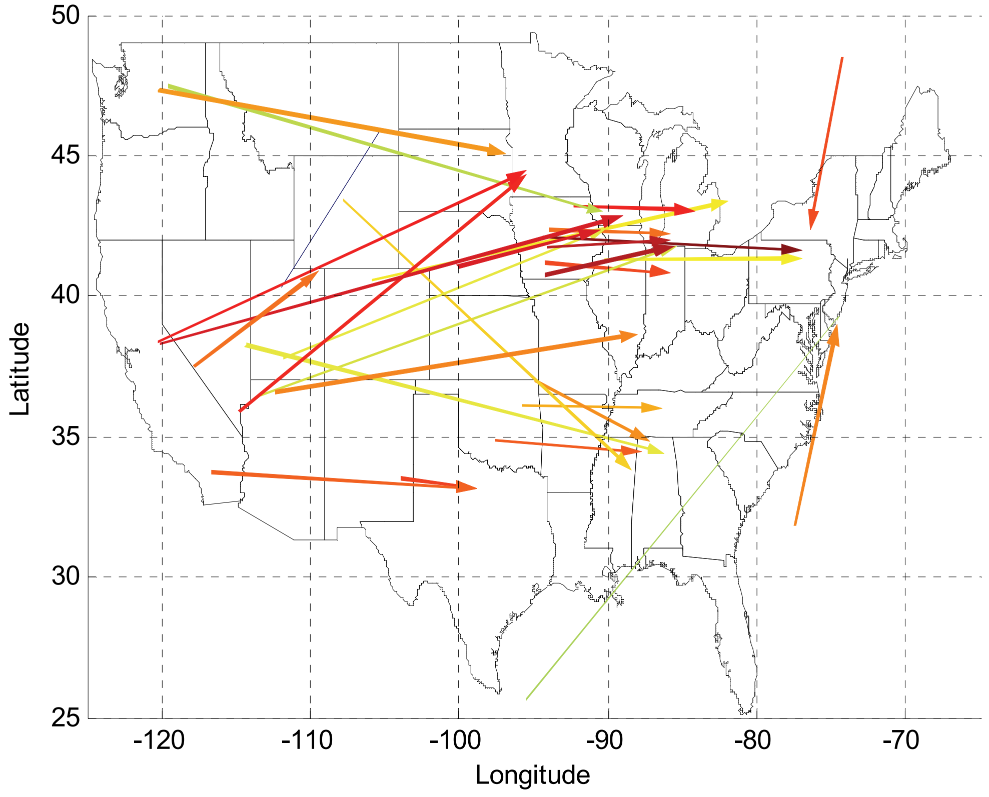


Figure 40. Evening Flows.

Figure 41 shows the flows throughout the entire day that have more than 5 segments, overlaid on the plot of position "TZ" points from ETMS data. The positions of the flows match reasonably well with the corresponding traffic position points, as well as the actual flows experienced by controllers. Note though, that high concentrations of position points do not necessarily translate into flows, since the actual trajectories at those points might not follow patterns that resemble flows. Clustering of traffic allows us to determine the directionality of the traffic that would otherwise not be possible from observing the traffic position points alone. Figure 42 shows the same flows with the time domain (times are minutes since midnight), where some of the same flows can be seen repeating several times a day.

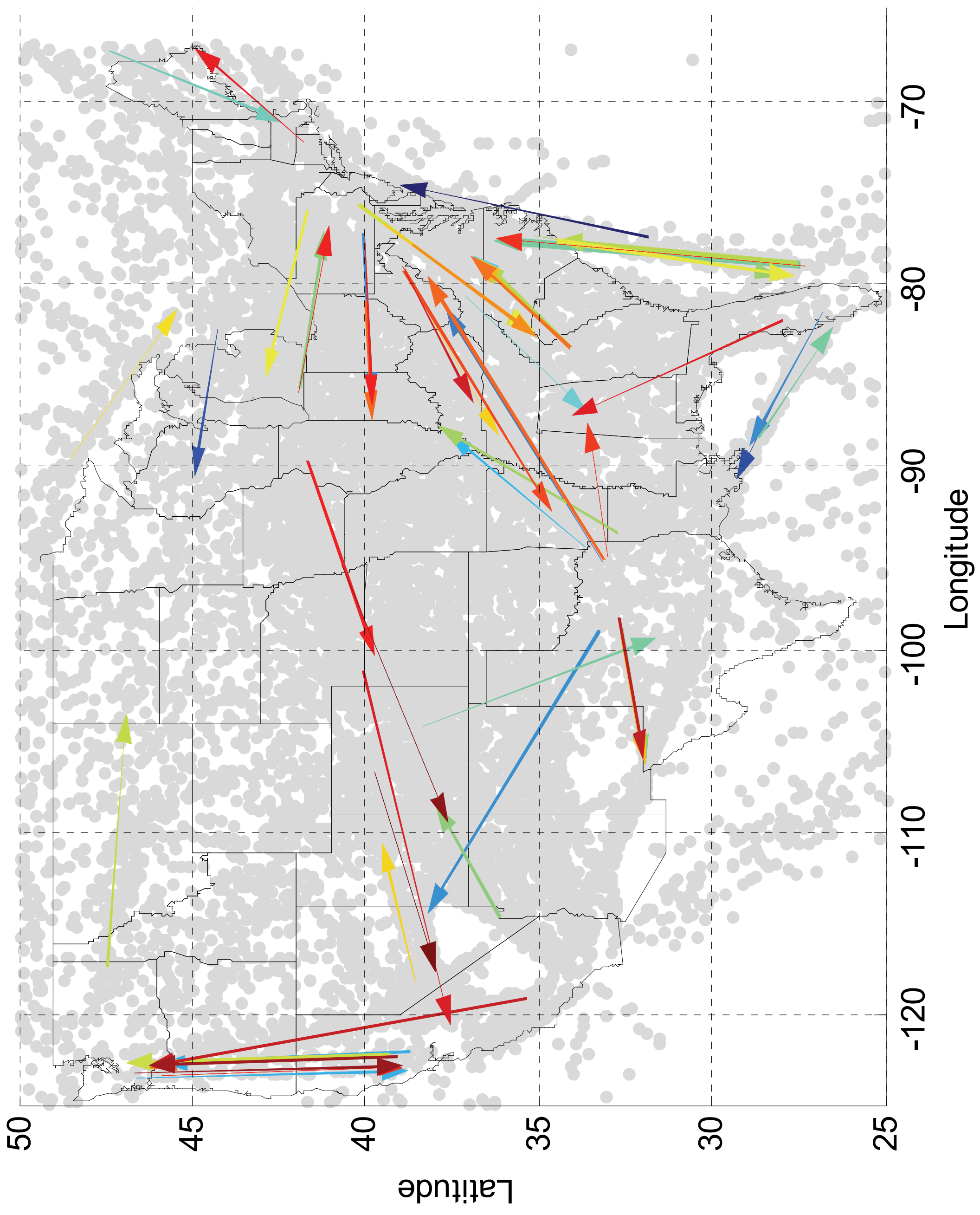


Figure 41. Major Flows for entire day overlaid on TZ points.

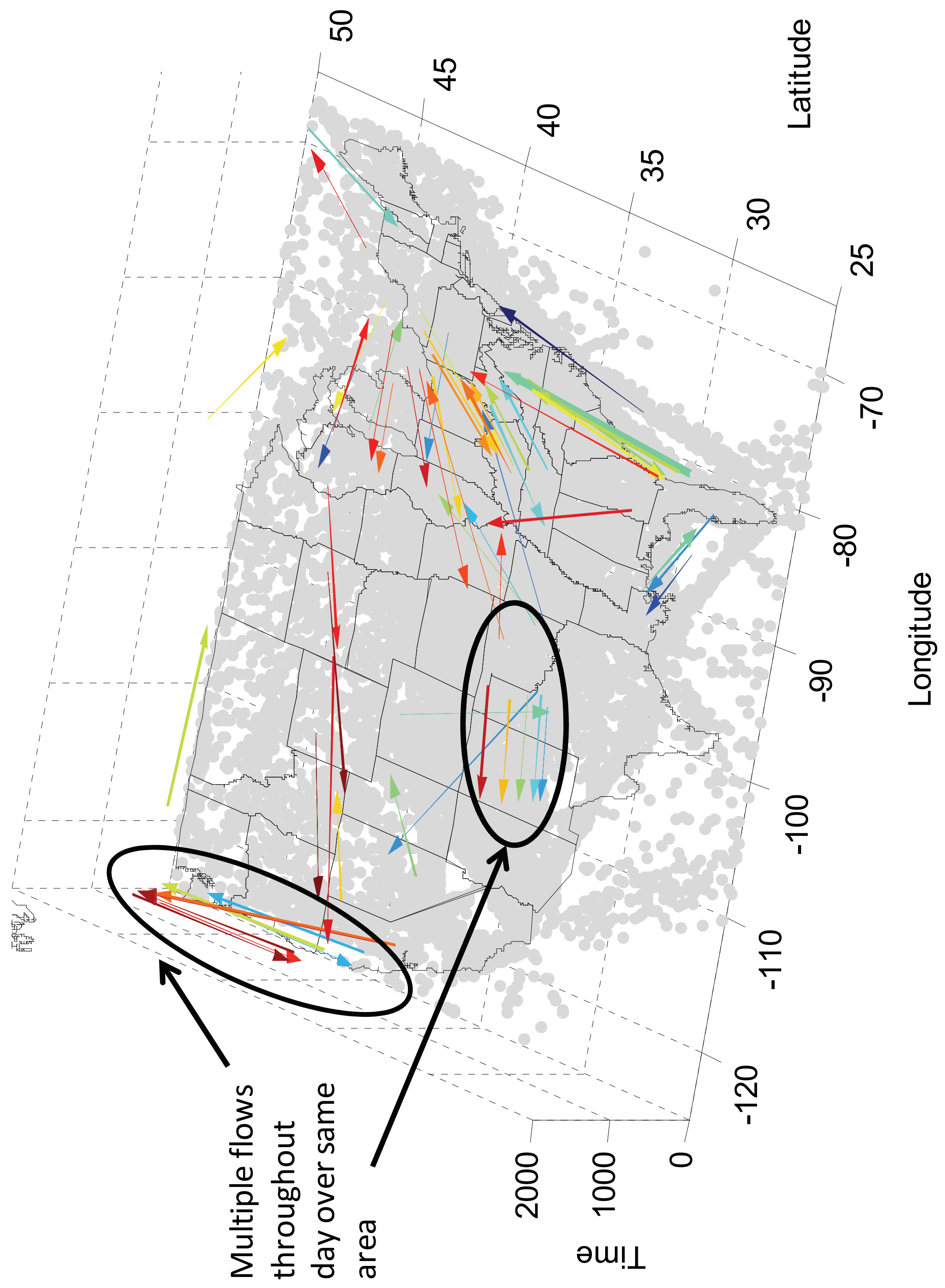
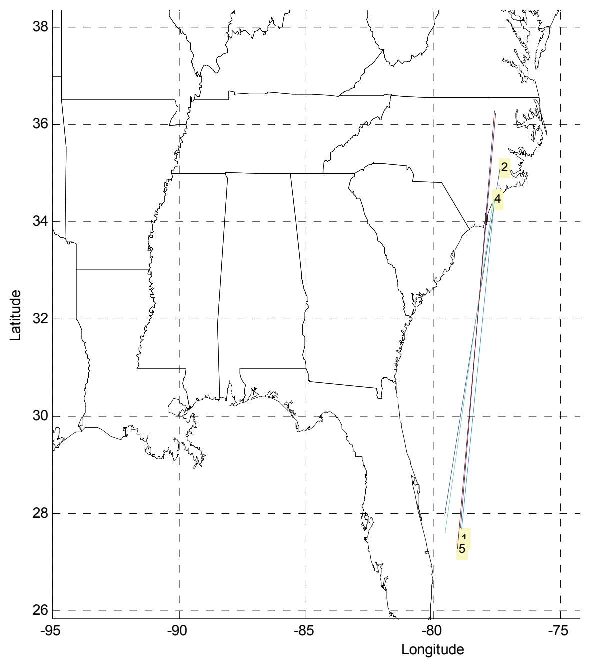


Figure 42. Major Flows for entire day showing repeating flows over time.

Figure 43 highlights the flows between Florida and the mid-Atlantic. (The ID number for each flow is placed at the start of the flow.) The plot of longitude versus time shows the breakout of the flows, indicating that there are 3 north-bound flows (1,3, and 5) over the same area throughout the day, and two south-bound flows (2, 4) that take place in between the first two north-bound flows.



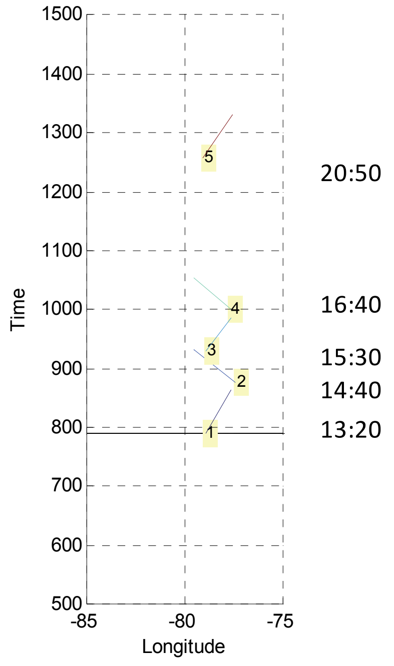


Figure 43. Eastern corridor flows.

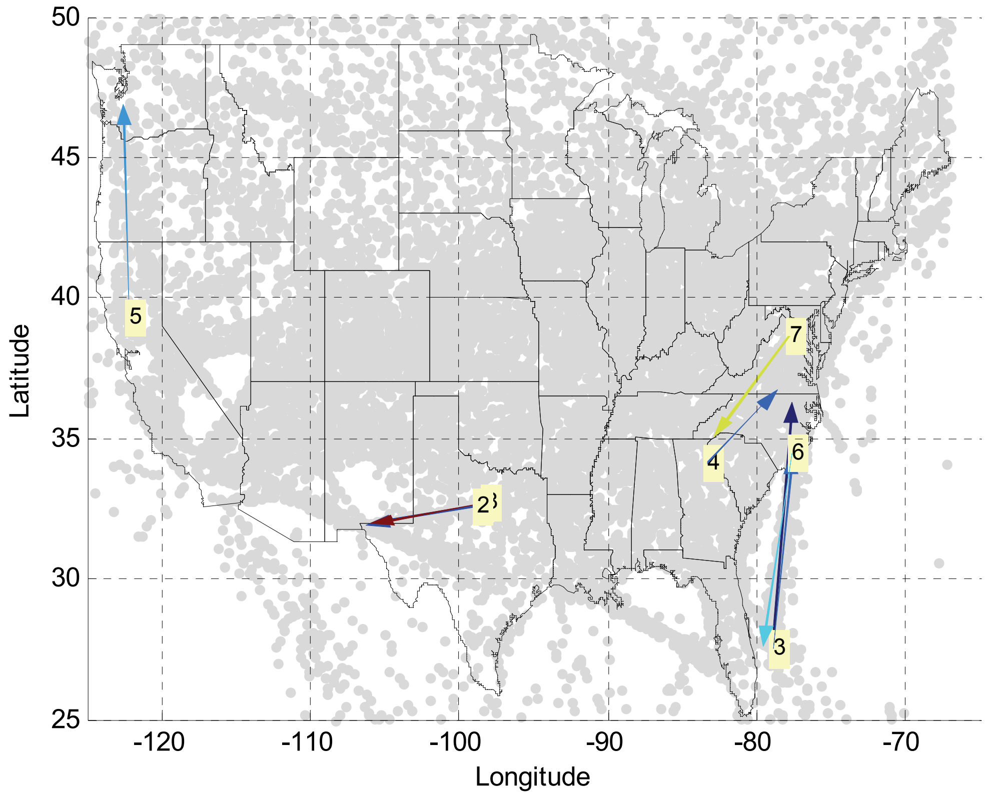


Figure 44. Top Flows.

Figure 44 shows the top set of flows that have 10 or more segments, and Table 1 shows the corresponding clusters with their origin destination information. Note that the flights that are part of the clusters are from varying origin-destination pairs. The "All ODs" column shows the origin-destination pairs followed by the number of aircraft segments that belong to that pair. Table 2 shows the length, duration, and flow information for these top flows. This flow information can be used for selecting the most applicable flows to target traffic initiatives.

Table 1. Origin-Destination Summary of top Flows.

	Num Segs	Head-ing	Unique OD pairs	% in main OD	Predominant OD	All ODs
1	32	8.72	14	25%	KFLL-KLGA	BCT-KBWI 1; BCT-KIAD 1; KFLL-KBWI 1; KFLL-KDCA 3; KFLL-KEWR 1; KFLL-KIAD 1; KFLL-KLGA 8; KMIA-KBWI 1; KMIA-KDCA 6; KMIA-KLGA 5; KPBI-KDCA 1; KPBI-KEWR 1; KPBI-KIAD 1; KPBI-KLGA 1
2	11	-95.0	7	27%	KDFW-KLAX	KDAL-KPHX 1; KDFW-KBUR 1; KDFW-KLAX 3; KDFW-KPHX 2; KDFW-KPSP 1; KDFW-KTUS 2; KDFW-ONT 1
3	14	10.5	12	14%	KFLL-KBOS	BCT-BLM 1; KFLL-ACY 1; KFLL-KBOS 2; KFLL-KEWR 1; KFLL-KJFK 1; KFLL-KPHL 1; KMIA-CYUL 1; KMIA-KJFK 1; KPBI-KBOS 1; KPBI-KEWR 1; KPBI-KHPN 2; KPBI-KJFK 1
4	10	61.2	8	20%	KATL-KEWR	KATL-KBOS 1; KATL-KDCA 1; KATL-KEWR 2; KATL-KHPN 1; KATL-KIAD 1; KATL-KLGA 1; KATL-KPHL 2; KBHM-KLGA 1
5	12	-2.59	6	33%	KSFO-CYVR	CCR-CYVR 1; KOAK-KSEA 2; KSFO-CYVR 4; KSFO-KSEA 2; KSJC-KSEA 2; KSMF-KSEA 1
6	10	-164	9	20%	KLGA-KFLL	KBWI-KFLL 1; KBWI-KMIA 1; KEWR-KFLL 1; KHPN-KFLL 1; KISP-KFLL 1; KJFK-KMIA 1; KLGA-KFLL 2; KPHL-KMIA 1; LIMC-KMIA 1
7	10	-125	10	10%	KALB-KATL	KALB-KATL 1; KBDL-KATL 1; KBOS-KATL 1; KBOS-KPDK 1; KBWI-KATL 1; KEWR-KATL 1; KEWR-KFTY 1; KHPN-KATL 1; KLGA-KATL 1; KPVD-KATL 1
8	12	-95.5	9	25%	KDFW-KLAX	KATL-KPHX 1; KATL-KSAN 1; KDAL-KELP 1; KDFW-KELP 1; KDFW-KLAX 3; KDFW-KPHX 1; KDFW-KSAN 2; KDFW-ONT 1; KDFW-SNA 1

Table 2. Flow Statistics.

	Num flight Segments	Length (nmi)	Flow rate	Start Time	End Time	Duration (hours)
1	32	533	2.80	7:40 AM	8:18 PM	12.6
2	11	379	5.72	1:28 PM	4:17 PM	2.8
3	14	422	7.61	2:32 PM	5:18 PM	2.8
4	10	275	8.57	3:00 PM	4:43 PM	1.7
5	12	457	6.09	2:45 PM	5:40 PM	2.9
6	10	423	7.92	4:00 PM	6:12 PM	2.2
7	10	321	8.46	6:37 PM	8:29 PM	1.9
8	12	399	5.77	9:47 PM	12:59 AM	3.2

4.5 Conflict Points Comparison

To help controllers ensure safety and proper separation between aircraft, the controller display has tools that show the projected path of each aircraft out to 10 to 20 minutes into the future. Conflict points are locations where two aircraft are projected to be within the minimum separation standards during this time if not acted upon. Thus, the controller must identify these situations and vector one or more aircraft where this is encountered to avoid a potential loss of separation. An aircraft's protective zone is represented as a cylinder as shown in Figure 45. The specific parameters for minimum separation vary according to certain conditions, but in general is represented by a minimum lateral separation φ of 5 nmi, and altitude separation γ of 1,000 ft where there is Reduced Minimum Vertical Separation (RVSM).

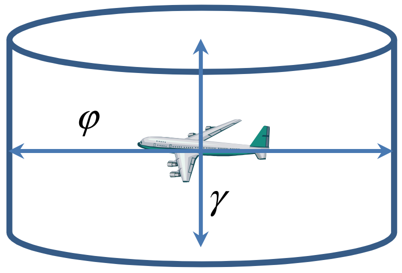


Figure 45. Aircraft protective zone.

To determine when two aircraft have the potential of entering each other's zone, their tracks are projected forward assuming each aircraft's own constant speed. As shown in Figure 46, the point at which they have the closest separation is not necessarily at the point at which their tracks cross due to their changing position over time.

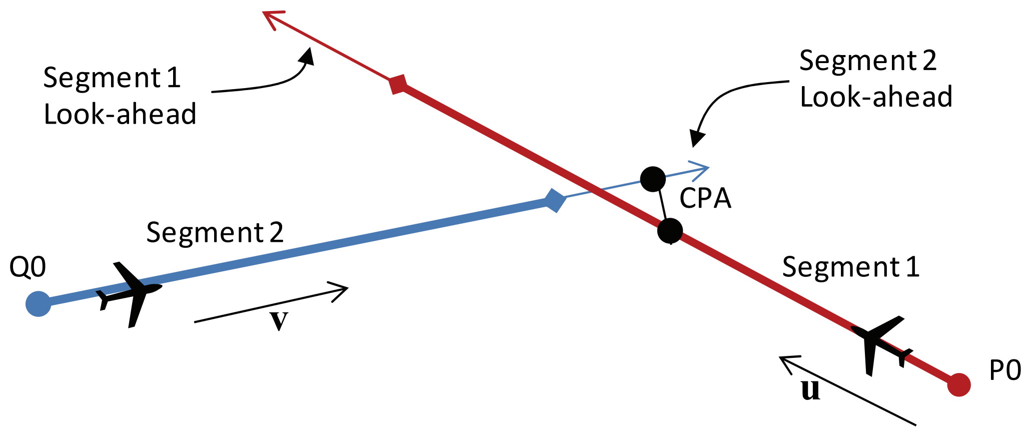


Figure 46. Conflict between two aircraft.

Let P_0 be the position of aircraft 1, and Q_0 be the position of aircraft 2, both at $t = 0$. Then, $P(t) = P_0 + t\mathbf{u}$ and $Q(t) = Q_0 + t\mathbf{v}$ represent the equations of motion for the two aircraft, where aircraft 1 has a constant velocity $\mathbf{u}$ and aircraft 2 has constant velocity $\mathbf{v}$. At any time *t*, the distance between them is

$$d(t) = \left|P(t) - Q(t)\right| = \left|w(t)\right|$$

where $w(t) = w_0 + t(\mathbf{u} - \mathbf{v})$ and $w_0 = P_0 - Q_0$. The distance $d(t)$ is minimum when $D(t) = d(t)^2$ is minimum. Since

$$D(t) = w(t) \cdot w(t) = (\mathbf{u} - \mathbf{v}) \cdot (\mathbf{u} - \mathbf{v})t^2 + 2w_0 \cdot (\mathbf{u} - \mathbf{v})t + w_0 \cdot w_0$$

then

$$0 = \frac{d}{dt}D(t) = 2t\left[(\mathbf{u} - \mathbf{v}) \cdot (\mathbf{u} - \mathbf{v})\right] + 2w_0 \cdot (\mathbf{u} - \mathbf{v})$$

and the time at the closest point of approach is

$$t_c = \frac{-w_0 \cdot (\mathbf{u} - \mathbf{v})}{\left|\mathbf{u} - \mathbf{v}\right|^2}$$

and its corresponding distance is

$$d_c\left(P(t), Q(t)\right) = \left|P(t_c) - Q(t_c)\right|.$$

Since the segments do not start at the same time, for each pair of aircraft, the position of the earlier of the two aircraft must be shifted forward in time to its location at the start time of the later segment so that they both start at the same time before performing the calculation. In addition, only segment pairs that overlap in time (plus look ahead) are computed.

Each segment is treated to exist only during the start and end times of the segment, plus the look-ahead time. In some cases, the CPA might occur beyond the look-aheads of the aircraft, as shown in Figure 47. Let the Minimum Valid Distance (MVD) between two aircraft be the shortest distance between the aircraft during the times that the two segments are both valid. Then, in this case, the MVD at the earlier ending time of the two segments must also be checked if the separation between them is within the separation standards. Likewise, if the CPA is located before the start of the segments, their MVD at time t=0 also must be checked for separation as shown in Figure 48.

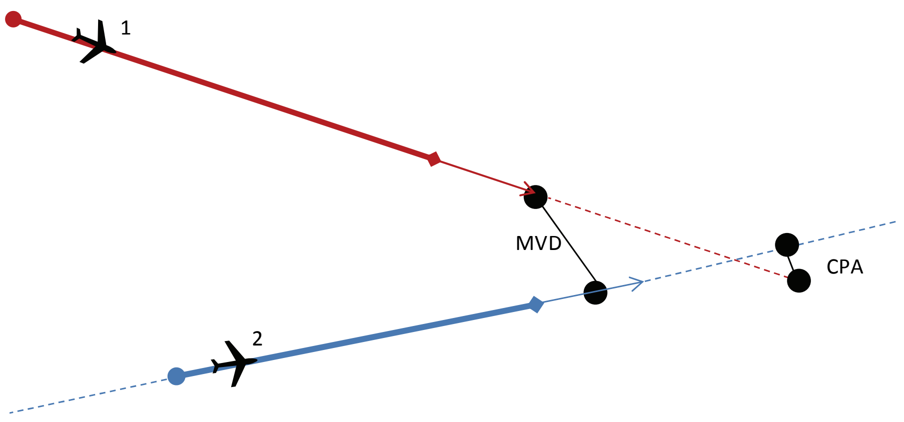


Figure 47. Checking separation distance at end of segment.

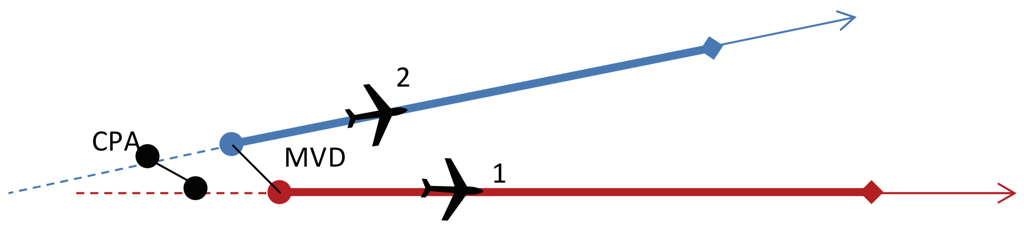


Figure 48. Checking separation distance at beginning of segment.

For a look ahead of 20 minutes, typical results for a full day of traffic for the entire CONUS with 14,160 segments results in about 1,700 conflicts, which averages to about 3-4 conflicts per ARTCC per hour. Figure 49 shows the distribution of the conflicts per hour throughout the day. For *n* segments, the computation is run on $O(n(n-1)/2)$ pairs of segments, but the actual number of computations is significantly less due to the filtering done in advance. The algorithm was implemented in Matlab, and has an extremely fast performance since it is coded without for loops to take advantage of Matlab's efficient matrix operations.

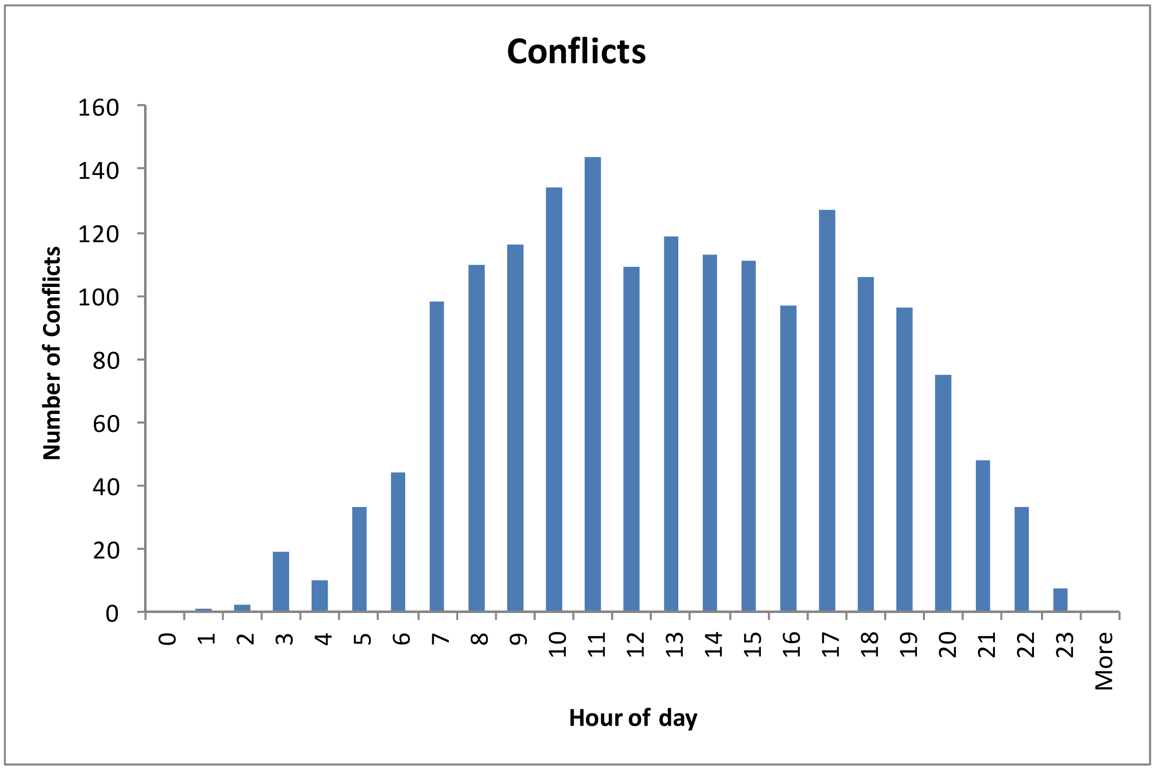


Figure 49. Histograph of conflicts for 1 day.

To determine the changes in conflict points, clustering was performed for the entire CONUS, and the original unclustered segments were moved to their corresponding clusters, and the conflicts computation algorithm was run for the unclustered and clustered scenarios. For each cluster, the ratio of the number of conflicts resulting from using the clustered flows over the number of conflicts resulting from the baseline unclustered traffic was computed. Figure 50 shows a histogram of the ratios, indicating that the largest bins show a 10-15% reduction in number of conflict points after clustering. With traffic increases in the future, the percent reduction in the number of conflict points is expected to be greater with clustering.

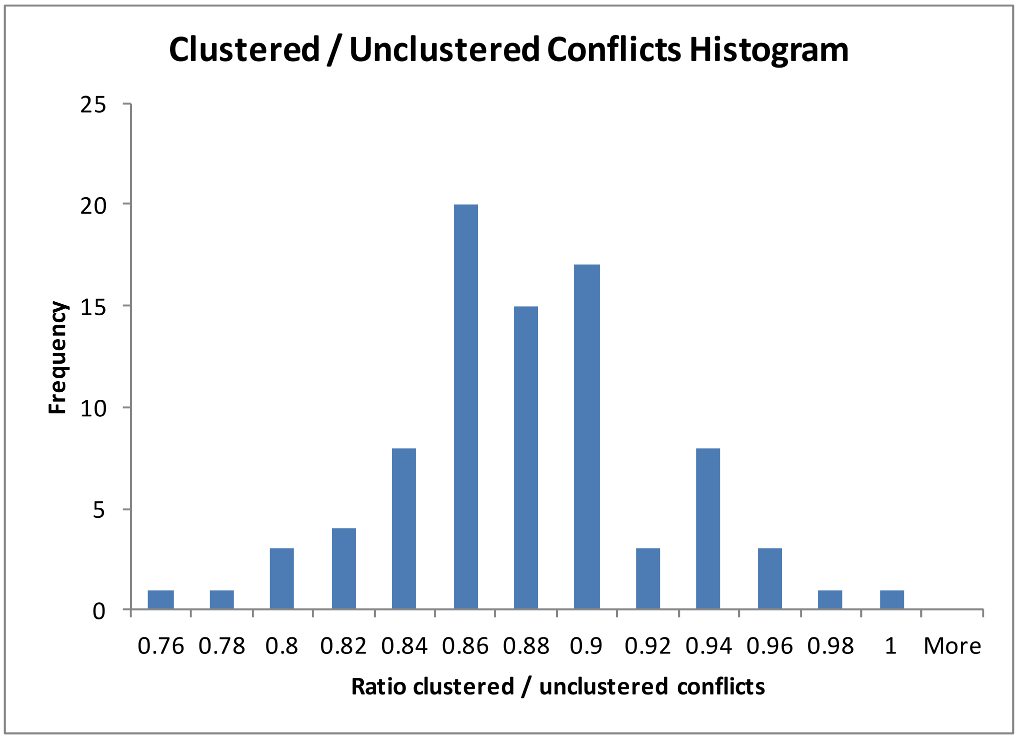


Figure 50. Clustered / Unclustered Conflicts Histogram

Chapter 5. Applications of Traffic Flow Cluster Information

In this Chapter, some possible "online" near-real time uses of the flow segments that are produced from the clustering algorithm are highlighted. In particular, we discuss how the information could be integrated into automation systems in the near-future timeframe to facilitate identification and management of Airspace Flow Programs (AFPs), and possible use in a farther time frame for rerouting traffic at an aggregate level.

5.1 Support for Airspace Flow Programs

In this section, we describe at high level how the clustering information could be useful to traffic managers in initiating and managing Airspace Flow Programs, without changing the fundamental operations of an AFP. Further details on the current concept of operations can be found in [25].

A tool could be developed that continually performs the clustering algorithm described in the previous chapters based on received flight plan information, and output the major traffic flow segments throughout the NAS. The results, updated on a regular basis (perhaps about every 5 minutes) based on the flight plans available, could be integrated into TMA (Traffic Management Advisor) or the Traffic Situation Display (TSD). Initially, as a prototype, this tool would be displayed onto a separate screen available for controllers to use as additional information for awareness, and not for "official" guidance. Similar to other information, the display of flows could be turned on or off at the discretion of the controller / traffic manager.

Ideally, the tool would be used by Traffic Management Specialists (TMS) at the ATCSCC (command center), and also Traffic Management Coordinators (TMC) at the

ATCSCC and ARTCCs. The Airline Operations Centers (AOCs) could also have access to the information for planning their routes and collaboration with the FAA.

Prior to deployment, analysis and coordination with the users would have to take place to determine the appropriate settings for the clustering algorithm, and display characteristics. In addition, "recurring" flows that appear repeatedly on a daily basis should be identified based on historical data. The pre-defined Flow Evaluation Areas (FEAs) / Flow Constraint Areas (FCAs) can then be setup or adjusted to coincide with the most likely areas of traffic flow.

The recurring flow information can also be use do show projected flows in the forecasted timeframes of 4-24 hours to facilitate long-term strategic planning, when flight plans are not yet available. As the time nears, clustering information from actual flight plan should replace the forecasted recurring ones.

As the day progresses, another tool can be used to calculate the WAF (Weather Avoidance Field) based on the latest forecasted weather data, and overlay the WAF onto the flows. Areas where the forecasted WAF (or temporary flight restrictions, etc.) intersect the major flows would be highlighted. These alert traffic managers of possible constrained areas and help them select / create FEAs for possible AFP activation.   This provides a more direct visual representation of flows and weather interaction. Currently, without such displays, the traffic management specialists have to mentally form "flow" patterns based on their experience and traffic position information on TSD, and then merge that with weather forecasts.

5.2 Aggregate Traffic Flow Management

In this and subsequent sections, the use of clustering is applied to aggregate traffic flow management. Managing individual aircraft and applying traffic management initiatives on them can help find the best options for the aircraft, but may sacrifice system-level benefits. While objectives can be added to balance the scale in favor of the overall system, they increase dramatically the complexity of the problem. Instead, the knowledge of flows can be used to operate on the entire flow of traffic, rather than individual flights, to provide system benefits and reduce computational complexity.

The high-level concept is reduction in the state space of the problem, and can be viewed as the following steps:

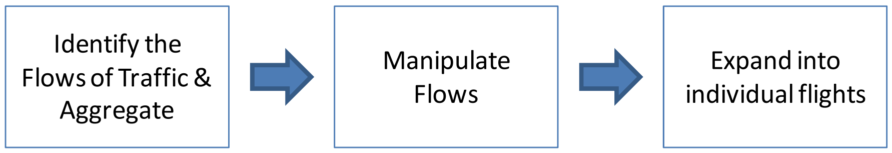

Identify the Flows of Traffic & Aggregate
Manipulate Flows
Expand into individual flights

Figure 51. Aggregate Traffic Flow Management.

Past work in [28] on aggregate traffic flow management has included modeling traffic as a network of interconnected, one-dimensional control volumes, whose inspiration was derived from the work of Lighthill, Whitham, and Richards (LWR) [26][36] on vehicular traffic. Work done in [33] included development of a network cell routing model as an integer program that dynamically routes traffic flows through congested areas while optimizing for minimum delay and maximum throughput. To complement the aggregate traffic flow models that reduce the state space, work has also been done to disaggregate the controls on the group of flights into individual flight-specific actions. [43] In [6], a multi-stage stochastic integer program was presented that minimized the expected average weighted sum of the number of aircraft that were assigned ground delay versus airborne delay based on scenarios of airport capacities. Recently, a multi-commodity flow model was implemented NAS-wide for a limited set of regional markets (e.g. New York), with each treated as a commodity. [32]

While “aggregation” is used often in the literature to describe various models, there are differing levels of aggregation that are implied. Most of the work has focused on higher levels of aggregation, looking at flights between centers or variations in routes between the same origin-destination pairs. In this application, a relatively low level of aggregation is used, one that is essentially only one level up from individual flight segments, and used only at a localized region for the specific flow segments. The models shown in this chapter are intended to show how the flow segment information from the clustering process can be used in such a low-level aggregated form. It is conceivable that other existing models in the literature, while not targeted at this level of aggregation, could be adapted to do so, and the clustering results could be provided as input into such models as well.

We will look in particular at an application of aggregated flow management to rerouting of traffic around weather. There is an extensive body of work that has been performed on traffic rerouting. In one of the seminal works on network flow modeling, a generalized network flow stochastic integer program model was provided in [8] that accounts for various demand scenarios and was shown to have a dual network structure that can be efficiently solved with linear programming techniques. In [34], a single aircraft was rerouted to avoid multiple storms and minimize the expected delay, with the uncertainty of the stationary weather treated as a two-state Markov chain, and solved using dynamic programming. Follow-on work expanded to modeling multiple aircraft with multiple states of weather and considered capacity and separation constraints at the storms [35].

The focus of the subsequent sections in this chapter is not on the rerouting model itself, but rather to show how the output of the clustering can be applied to rerouting and to examine the possible benefits

When weather or other events such as the activation of Special Use Airspace (SUA) occur, the traffic scheduled to traverse them is affected, and often needs to be rerouted. Current and planned air traffic rerouting strategies reroute individual aircraft rather than a collection of aircraft as a whole. Some of the major disadvantages to rerouting individual aircraft include:

- The rerouted path for each aircraft might be different, possibly destroying any major flow patterns. This potentially leads to increased controller workload in managing disparate traffic streams.
- Without further planning, individual aircraft might select paths that are optimal for itself, but might lead to congestion or conflicts if they have routes that fly near each other, causing less efficient use of airspace, introduce more intersections, and uneven distribution of flight density. All of these ultimately result in a reduced throughput potential. In the short run, individual aircraft might benefit at the cost of overall system disadvantages, but as the system gets bogged down, individual aircraft that come later will be impacted with further delays.

- Rerouting individual flights requires more computational energy, especially for large numbers of flights over large regions.

Instead, once a flow is established, the flow can be manipulated and affect all flights that traverse it. In the spirit of Collaborative Decision Making (CDM), flights would not be required to use a flow, but if they chose to join the flow, traffic management initiatives can provide a path that has already been determined to have sufficient capacity and minimize congestion. So, those participating in the stream are incentivized with prioritized treatment.

Because the flow already represents a considerable stream of aircraft, the flows help lower the demand elsewhere in the airspace, and aircraft not within a flow might not need to be moved due to lowered demand. If needed, they can be rerouted using conventional means.

5.3 Flow Rerouting

Moving a flow is similar to moving an individual aircraft with a slight variation. Each flow has a flow rate f in units of number of flights per unit time, which can be computed from the number of flights that are planned to use the flow during the active period. The capacity of the airspace will be equivalently defined in terms of the maximum flow rate that can be supported at a certain node or sector rather than in terms of the number of aircraft in a sector.

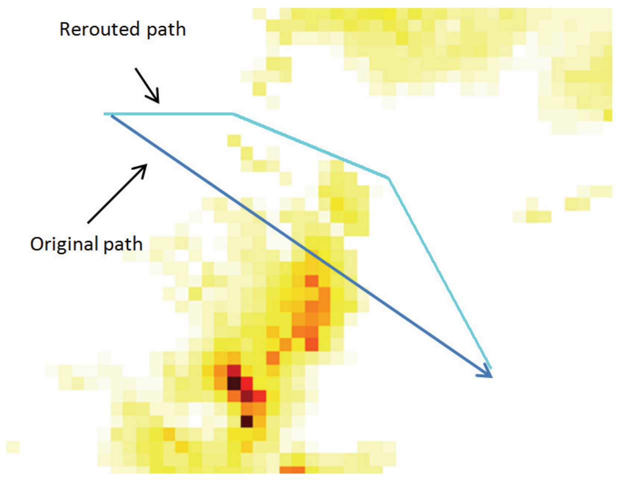


Figure 52. Rerouting path through / around weather.

The approach for rerouting consists of the following steps:

1. Identify the major traffic flows that intersect weather.
2. Meter traffic into the entry point of the flow.
3. Determine the shortest path through or around the flow constrained area by constructing a network of edges between adjacent nodes, and solve as an Integer Program.
4. Smooth the path by finding the longest possible edges between non-adjacent nodes in the shortest path.

The first Step involves running the Incremental Clustering algorithm, selecting only the clusters with the most traffic, and determining if they intersect weather. As will be shown in the model for rerouting flows, Step 2 is necessary in order to establish an uncapacitated flow of aircraft through the corridor. Although important, this step is not part of the focus of this section, and will be discussed later in the Implementation Considerations section. Assuming that the aircraft maintain proper separation and speed controls, the flow rerouting model in Step 3 can be simplified to just determine the correct geometry, without including the temporal interaction of flights within the route. To maximize performance and reduce the problem size, the process of determine the routes is performed in two steps. The result of the reroute in Step 3 is an unsmooth route that conforms to edges on a grid, which is left to be smoothed in Step 4.

5.3.1 Shortest Path IP

For the first phase, we establish a graph $G=(V,E)$ consisting of a grid of nodes *V*, and edges *E* only between adjacent and immediately diagonal nodes. Each node *j* is given a capacity K_j in terms of the number of aircraft it can support. Let x_{ij} be the binary decision variable such that

$$x_{ij}=\begin{cases}1 \text{ if the flow goes from node } i \text{ to node } j\\ 0 \text{ otherwise}\end{cases}$$

Let c_{ij} be the cost, defined in terms of distance, of the flow going from node *i* to node *j*. The optimization is formulated as a capacitated shortest path problem, were the cost function is minimized as follows:

$$\min \sum_j \sum_i c_{ij} x_{ij}$$

To ensure that the capacity is satisfied, the flow is not allowed to go into a node *j* if its flow rate *f* exceeds the node's capacity flow rate K_j:

$$\sum_i \left(x_{ij}+x_{ji}\right)=0 \qquad \forall j : K_j < f$$

Let *r* be the entry point of the flow and *s* be the exit, then the following constraint ensures proper flow in and out of each node:

$$\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1 & \text{if } i=r \\ -1 & \text{if } i=s \\ 0 & \textit{otherwise} \end{cases} \qquad \forall i$$

To simplify the model, the capacity constraints can be removed by simply not defining edges for which the capacity is not sufficient to support the flow. Only edges between adjacent nodes with sufficient capacity need to be defined. With this constraint removed, the only remaining set of constraints is the flow conservation constraints, making this problem purely a shortest path problem, which can be solved by linear relaxation. So, multiple flight trajectories can be rerouted efficiently with a single linear relaxation programming model.

5.3.2 Data

Through various aviation weather programs conducted in collaboration between the FAA, NOAA, NASA, UCAR, and other organizations, a flurry of weather products have been developed to aid pilots and controllers route aircraft around weather activity. [49] The common physical phenomena that these products use as input are the amount and form of precipitation, wind direction and intensity, the top of the precipitation activity (echo top), and lightning strikes. The products vary in the frequency of update and other post-processing performed on the raw data.

The Corridor Integrated Weather System (CIWS) provides 0-2 hour "tactical" weather decision support information in a number of products, including Echo Tops (ET), Vertically Integrated Liquid (VIL), and growth and decay trends. [13][14] The Collaborative Convective Forecast Product (CCFP) is a graphical representation of expected convective occurrence at 2-, 4-, and 6-hours and provides more strategic traffic flow management. [31] The National Convective Weather Forecast Product, Version 2

(NCWF-2) provides one and two-hour probabilistic convective forecasts updated every five minutes. [9] This is the risk that hazardous convection will affect an area at a specific place and time with the airspace. Development is currently underway with the Consolidated Storm Prediction for Aviation (CoSPA) that will combine aviation-related research activities and provide weather products that forecast out to 6 or more hours. Within the 0-2 hour window, the forecast is provided in 5 minute intervals, and updated every 5 minutes with 1km resolution. Within the 2-6 hour window, the forecast is available in 15 minute intervals and updated every hour with 3km resolution. [48]

Efforts are underway to translate weather data from the raw physical properties to more actionable information for pilots and controllers, with inclusion of such information as the probability that a pilot would want to avoid the region, and airspace capacity. In addition, the ultimate goal is for this information to not only be displayed for situational awareness and decision making, but also to integrate the information and decisions resulting from them among systems that support the NAS.

Methods for determining the capacity of a region based on weather conditions are still in development. Work done in [40][41] provides an approximation of sector capacity based on patterns of traffic traversing through each sector and their complexity using graph theory. In [30], a method for determining the probability distribution of airspace throughput capacity based on stochastic weather was presented that used geometric computations to obtain capacities of 2-dimensional regions.

The FAA is currently considering various models for defining Weather Avoidance Fields (WAFs) that translate the meteorological data into polygons that can be acted upon. These WAFs alleviate pilots and controllers from having to be meteorologists and define the region that aircraft should avoid based on weather conditions. One such model uses a matrix of VIL and EchoTop values, and assigns a severity index for each combination of VIL and Echo Top ranges. [7] The greater the VIL and Echo Top, the more severe the index, and areas where the index is above a certain threshold (which could vary by region) are avoided.

Figure 53 shows a sample of VIL, Echo Top, and the corresponding Weather Avoidance Field (shown in red).

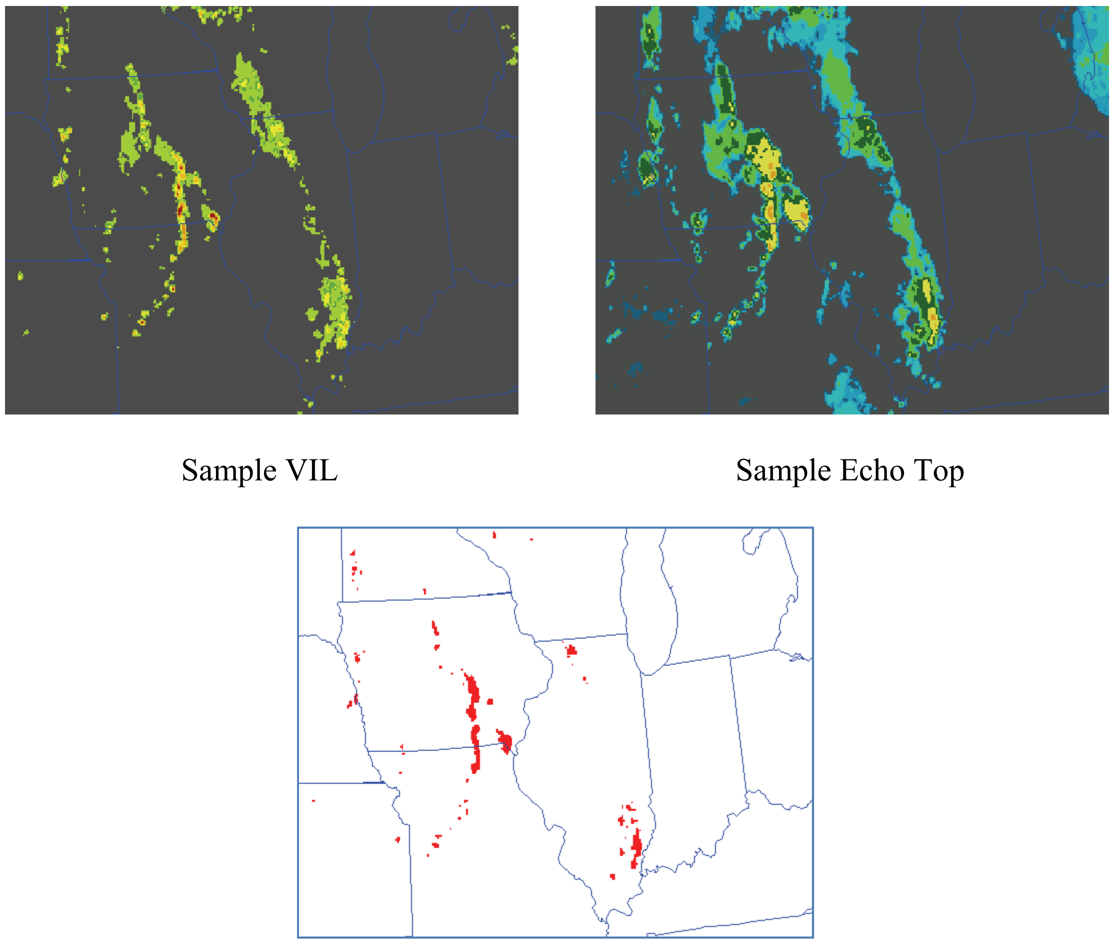


Figure 53. VIL, Echo Top, WAF sample.

For this problem, the capacity values will be determined by using the WAF that is computed from VIL and Echo Top. [7] The WAF value WAF_j at node j takes on a binary value:

$$WAF_j = \begin{cases} 0 \text{ if there is severe weather preventing passage} \\ 1 \text{ otherwise} \end{cases}$$

The shortest path model above is then implemented with $f = 1$ and $K_j = WAF_j \in \{0,1\}$. For the sake of showing comparisons between rerouting based on an entire flow versus individual rerouting, the weather will be assumed to be static during the time frame under consideration.

5.3.3 Sample Rerouting Output

Figure 54 shows the results of the rerouting, with the original path shown as a line intersecting the weather activity, and the rerouted path that circumvents weather. Because of the structure of the constraint matrices, the problem was solved very efficiently using linear relaxation.

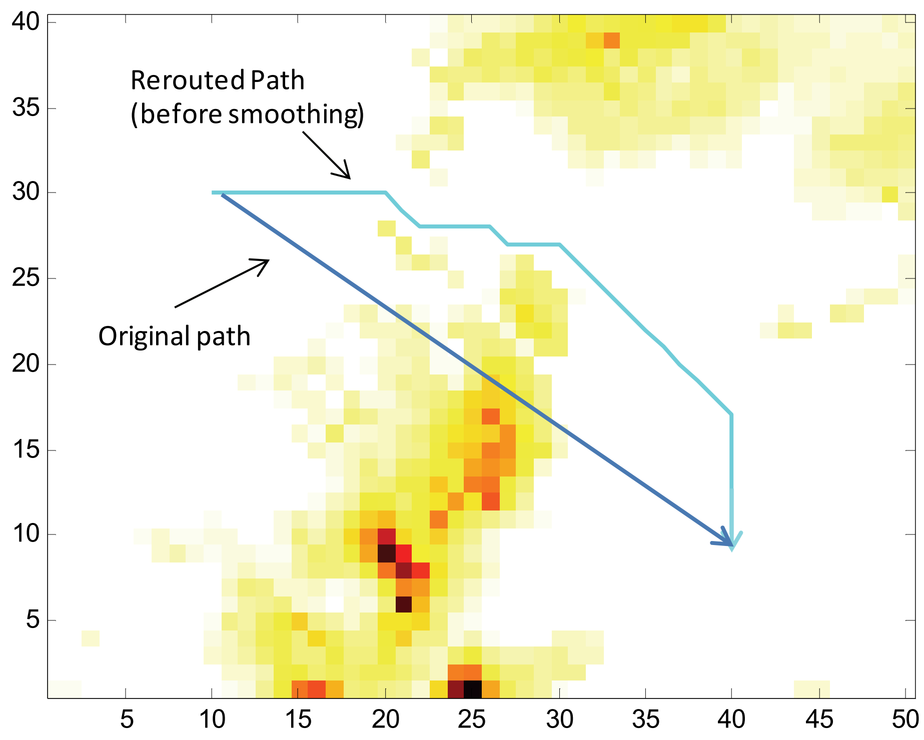


Figure 54. Results of flow corridor rerouting.

5.3.4 Path Smoothing

Once the output from the integer program is complete, the second phase involves smoothing out the path by generating direct segments between points on the path such that they are as long as possible without crossing over capacitated areas.

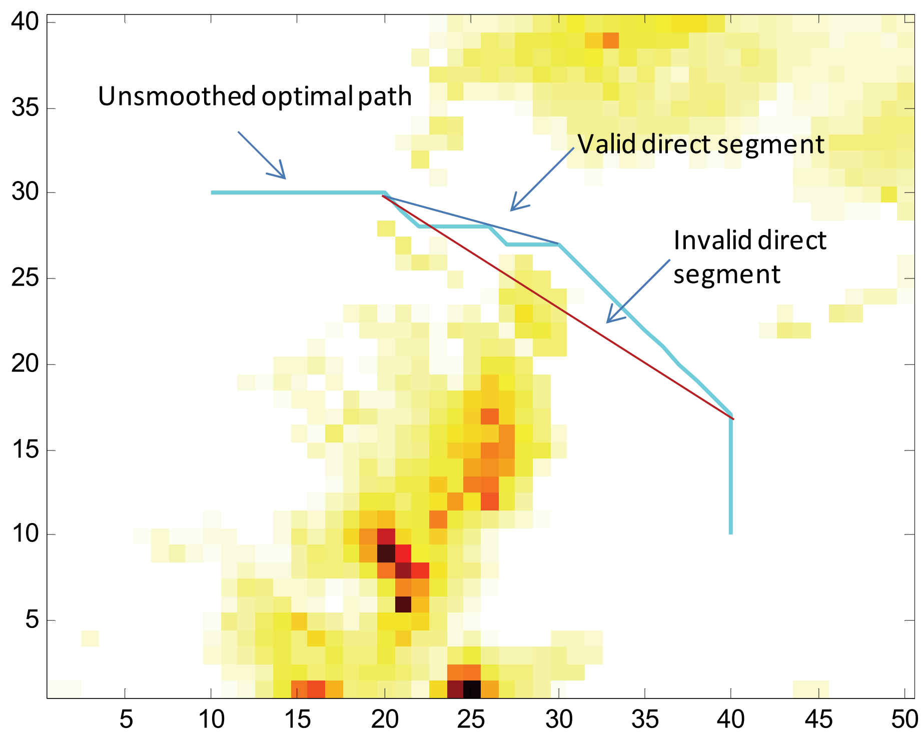


Figure 55. Valid direct segments.

The algorithm for the smoothing process is as follows:

```
1. Initialize
    a. Set $i$ to be the first node in the shortest path
    b. Set $j$ to be $i$+2
2. Find the longest possible edge between non-adjacent nodes:
    a. Check if the Edge($i$, $j$) intersects any weather activity
        i. If the edge intersects weather:
            1. Disregard this edge, but add the previous Edge($i$, $j$-1) to the
               list of smoothed edges
            2. Set $i = j$ and $j = i + 2$ to find subsequent edges
            3. Go to 2)
        ii. If the edge does not intersect weather:
            1. Lengthen the edge: $j = j + 1$
            2. Go to 2a)
3. Add the last remaining edge to the last smoothed edges
```

The result of the smoothed path is shown below.

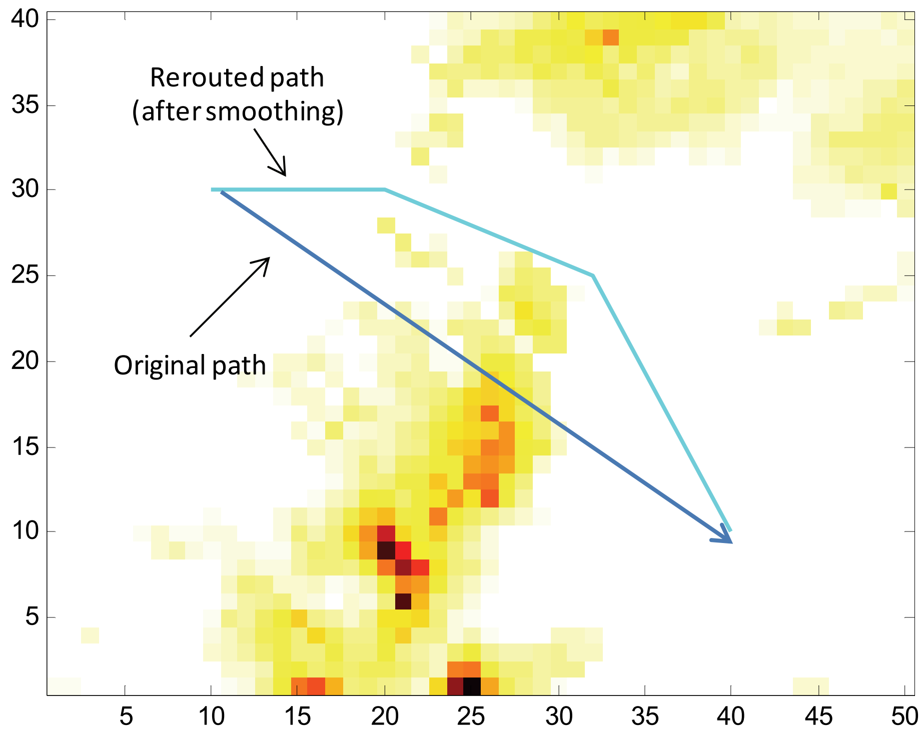


Figure 56. Path after smoothing.

The performance of the smoothing portion is of $O(nr)$, where n is the number of nodes in the path determined from the shortest path IP problem and r is the number of grid intervals. Performing the smoothing step as a secondary phase to the shortest path algorithm significantly improves performance over trying to find the smoothed path within one step. To find the smoothed path as part of the initial shortest path algorithm, the network would have to be established such that there are edges not only between adjacent nodes, but every other node in the network, resulting in up to V^2 edges.

5.4 Individual Aircraft Rerouting

In this section we include a simplified model for rerouting of individual aircraft to illustrate a comparison with the case of rerouting the entire flow.

5.4.1 Shortest Path IP

For the problem of rerouting multiple individual aircraft, we model it as a multicommodity flow problem. Because there are multiple aircraft paths, the time at which each aircraft occupies a certain portion of space must be taken into account, requiring that the decision variable be expanded to include both flights and time. Let x_{ijtf} be the binary decision variable such that

$$x_{ijtf} = \begin{cases} 1 \text{ if the flight } f \text{ goes from node } i \text{ to node } j \text{ at time } t \\ 0 \text{ otherwise} \end{cases}$$

For the cost c_{ijt} of a flight going from node *i* to node *j* at time *t*, a higher cost is given for times beyond the original scheduled ending time to encourage the best possible solution. The optimization can be formulated as a capacitated shortest path problem, were the cost function is minimized as follows:

$$\min \sum_f \sum_t \sum_j \sum_i c_{ijt} x_{ijtf}$$

To ensure that the capacity is satisfied, the total number of flights present going to node *j* in any time instance cannot be greater than the capacity K_j:

$$\sum_i \sum_f x_{ijtf} \le K_j \qquad \forall j, t$$

If *r* is the start of the flow and *s* is the end, then the following constraint ensures proper flow in and out of each node:

$$\sum_j x_{ijt_2f} - \sum_j x_{jit_1f} = \begin{cases} 1 & \text{if } i{=}r \\ -1 & \text{if } i{=}s \\ 0 & \textit{otherwise} \end{cases} \qquad \forall i, f, t$$

Since this is a multicommodity flow problem, it does not have a totally unimodular constraint matrix. The result for rerouting the individual aircraft that would have been part of the flow are shown below for the first phase.

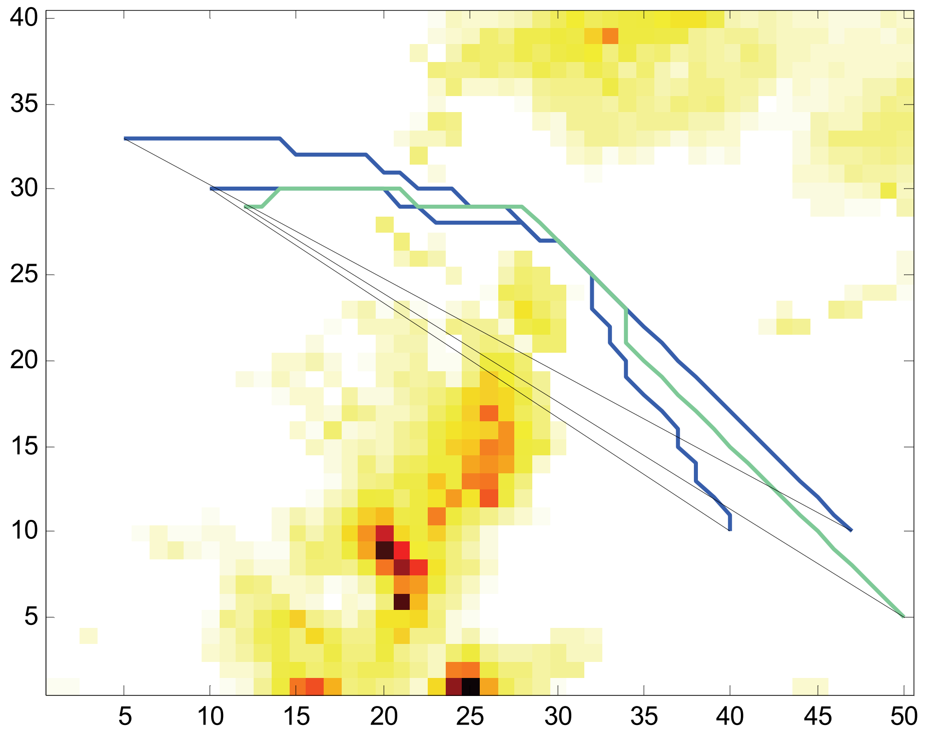


Figure 57. Individual aircraft reroute after first phase.

5.4.2 Smoothing

When rerouting multiple aircraft paths at once, the smoothing phase must take into account the possibility that other aircraft might be present in the path of the direct routes generated by the smoothing process. To take this into consideration, the algorithm is modified to perform smoothing for earlier aircraft first, and the path for subsequent aircraft will take into account the previous aircraft's presence in the form of reduced capacity in the respective areas and times.

The algorithm for the smoothing process for individual aircraft is as follows:

```
1. Sort aircraft in order of scheduled arrival time
2. Perform smoothing for each aircraft in order of time
    a. Initialize
        i. Set $i$ to be the first node in the shortest path
        ii. Set $j$ to be $i$+2
    b. Find longest possible edge between non-adjacent nodes
        i. Check if Edge($i$, $j$) intersects any weather activity
            1. If the edge intersects weather:
                a. Disregard this edge, but add the previous Edge($i$, $j$-1) to the list of smoothed edges
                b. Set $i=j$ and $j=i+2$ to find subsequent edges
                c. Go to 2(b)
            2. If the edge does not intersect weather:
                a. Lengthen the edge: $j=j+1$
                b. Go to 2(b)(i)
    c. Add the last remaining edge to the last smoothed edges
    d. Add the smoothed aircraft's path as reduced capacity for subsequent aircraft for the respective locations and times
3. Repeat Step 2) for each aircraft
```

The figure below shows the results of path smoothing for the individual aircraft.

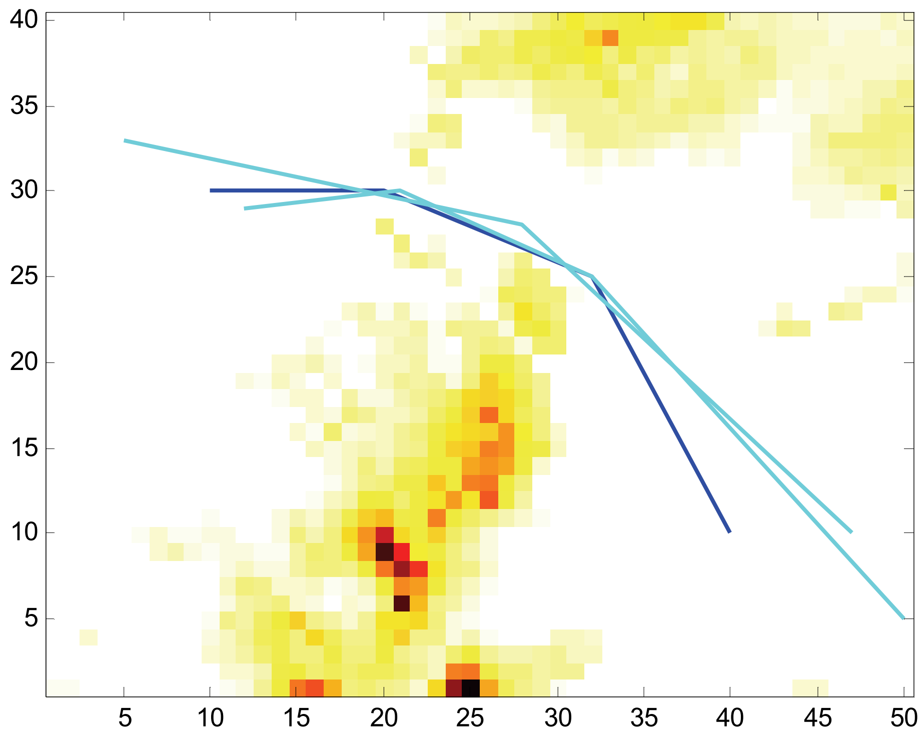


Figure 58. Individual aircraft rerouting after the smoothing phase.

By comparison to routing the single flow corridor, the airspace can be seen to be more complex than that for just one path for the corresponding flow corridor shown below:

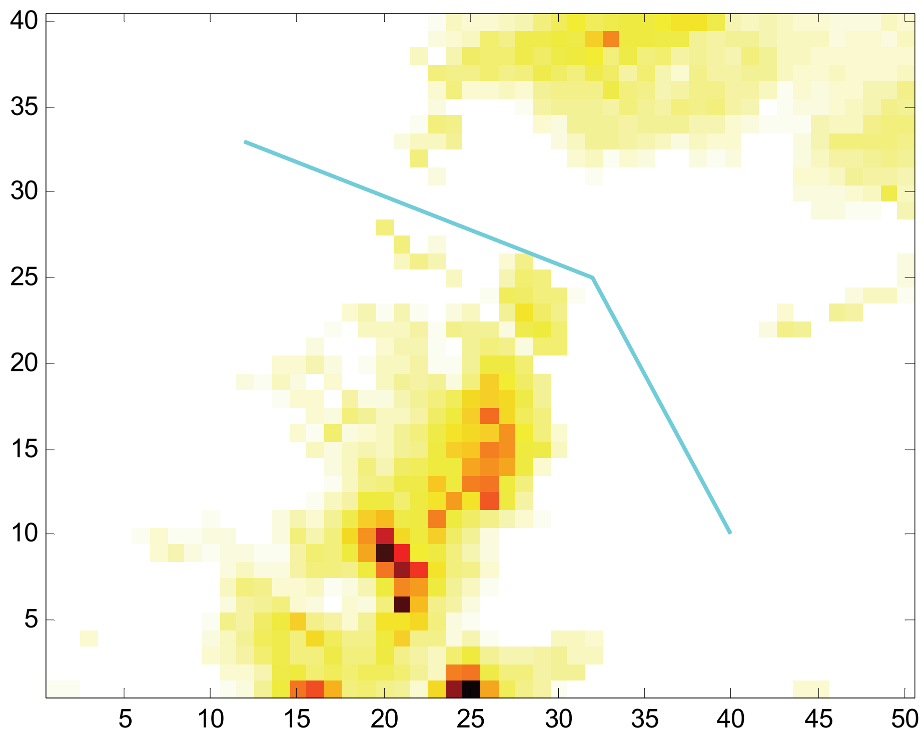
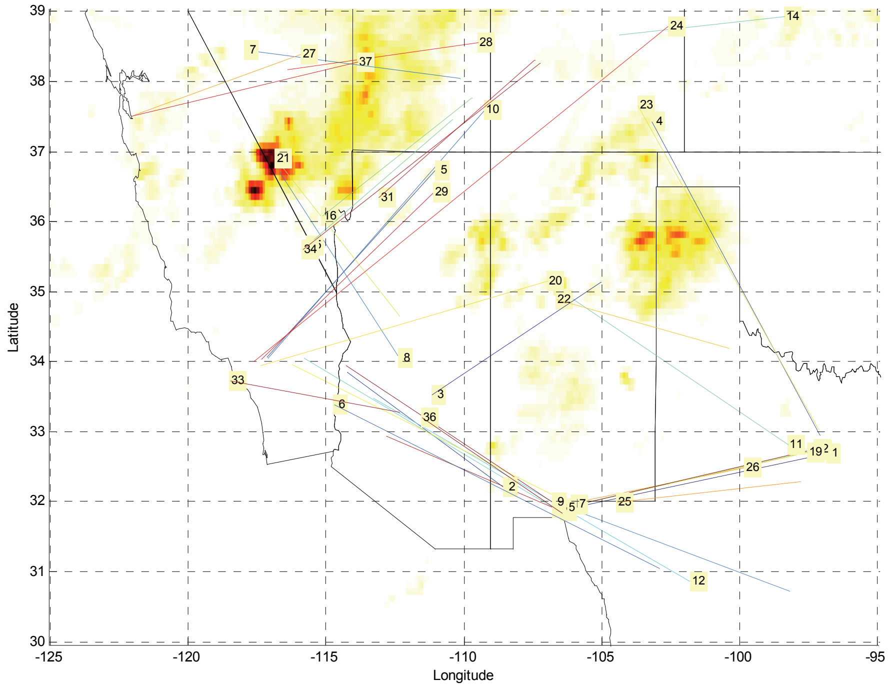


Figure 59. Rerouted flow after smoothing.

5.5 Rerouting analysis and results

To examine further the rerouting of flows, we first start with a sample clustering that was performed for a selected day when there was very little impact of weather activity onto NAS traffic, September 20, 2010. The Figure 60a shows the results of clustering for a southwest portion of CONUS during the late afternoon hours, with weather data from another day applied. Figure 60b shows those that have at least 5 flights.



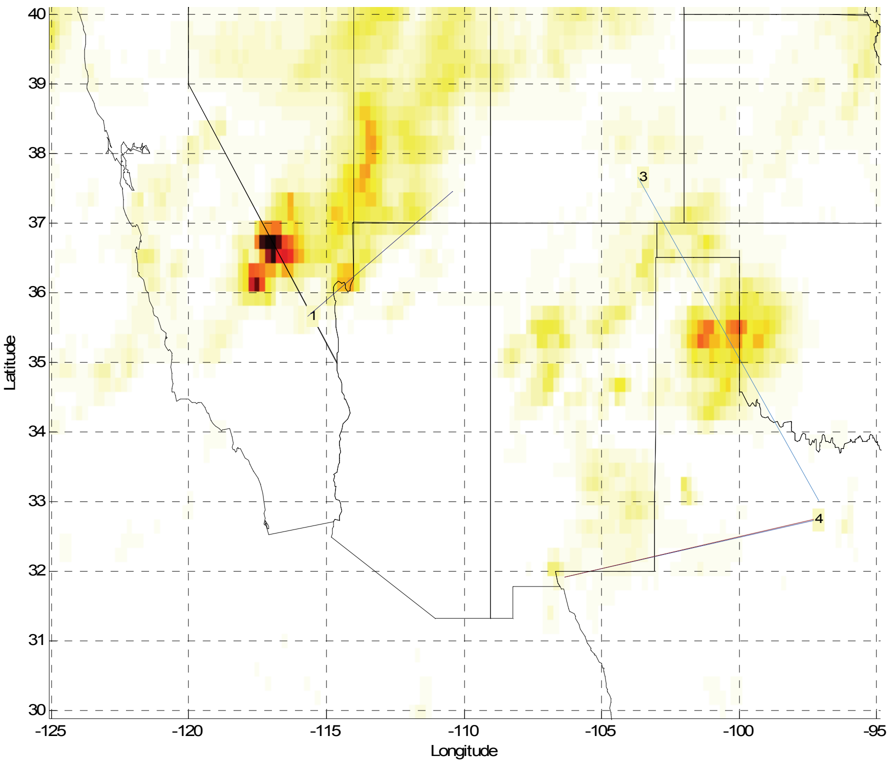


Figure 60: Output SW Flows. a) All flows, b) Flows with more than 5 segments.

The first cluster passes through a capacitated region of weather activity, and the results of rerouting from the first phase is shown in Figure 61.

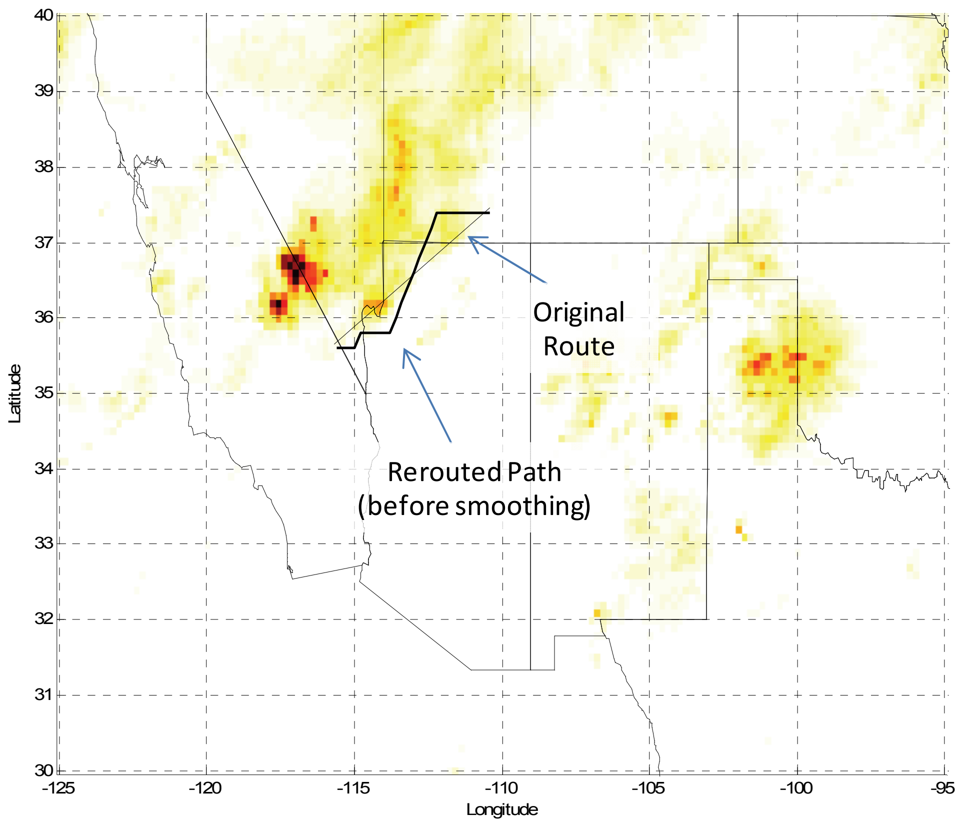


Figure 61. Reroute of first flow before smoothing.

After the second phase, the path is smoothed to have only 2 segments, causing a minor deviation from the weather as shown in Figure 62.

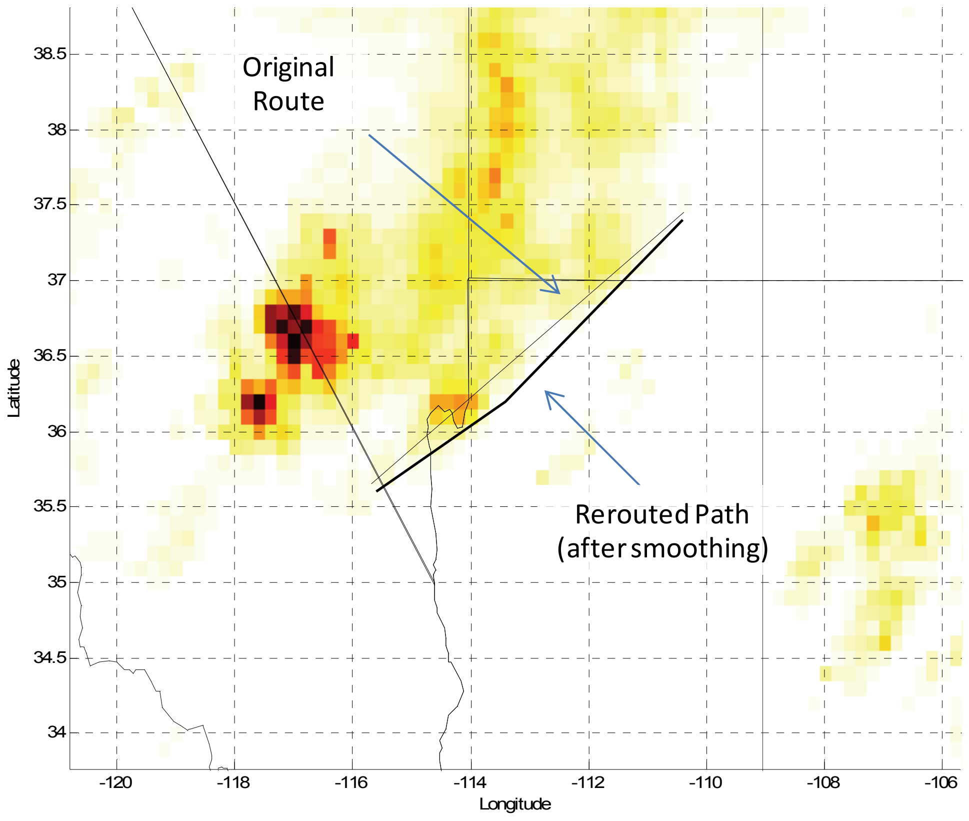
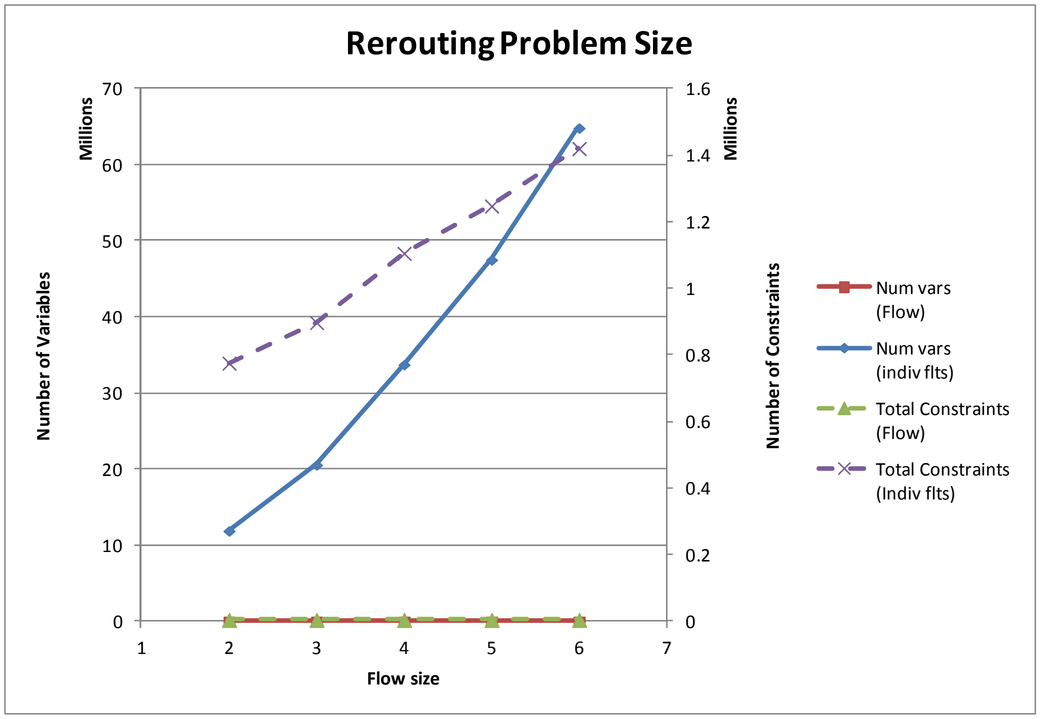


Figure 62. Rerouted flow after smoothing.

Figure 63 shows a comparison between rerouting individual flights versus rerouting entire flow of traffic for both the problem size and computational performance. The grid is setup with a 0.1 degree resolution (approximately 6 nmi), and the Weather Avoidance Fields were determined based on VIL and EchoTop data down-sampled from the original 1km resolution to match the grid resolution. A separate grid is setup for each flow that is to be rerouted. As expected, rerouting individual flights has an exponential increase in problem size and computational performance, while rerouting the flow that those flights belong to has a relatively very little performance impact. The preprocessing additional steps that are taken to generate the flows, including clustering, have a marginal increase on computation that is negligible compared to the much larger state space for individual flight rerouting.



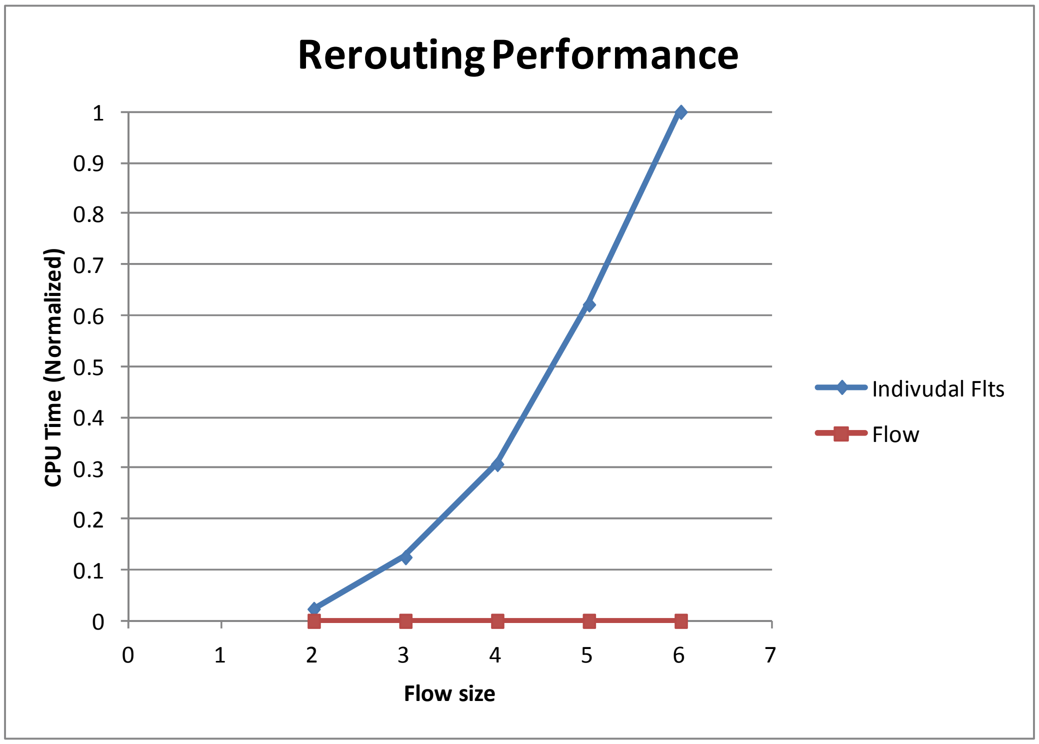


Figure 63. Comparison of a) problem size and b) computational performance.

Figure 64 shows a comparison of the amount of reduction in conflict points that can be achieved when flights use the rerouted flow corridors versus rerouting individually. The number of conflict points can be reduced by as much as 20% by having flights join tubes and rerouting the entire flow of traffic rather than rerouting individual flights. Since the reduction scales with the number of segments, with future traffic expected to grow even further, it is an anticipated that there could be slightly greater reductions in conflict points.

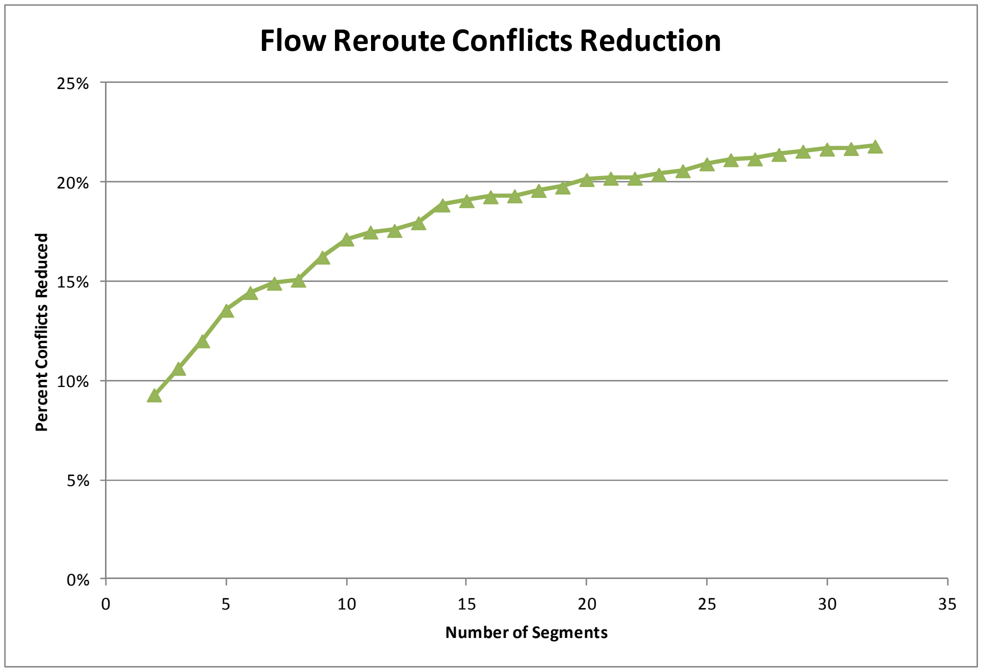


Figure 64. Conflict Reductions from Rerouting Flows.

5.6 Implementation Considerations

In this section, we examine some aspects of managing traffic as sets of clustered flows that should be considered when implementing in practice.

5.6.1 Flow Management & Prioritization

Many strategies can be used to manage the interaction of traffic between those in the flow corridor and those in the surrounding airspace. Within NextGen, there is a strong emphasis on Collaborative Decision Making (CDM), whereby the airlines have a stake in the decision making process for traffic flow and the trajectory that a flight will take rather than having the controller and traffic management specialists unilaterally making decisions. [47] In keeping with CDM principles, aircraft that use the flow would not be required to take the route prescribed, but instead incentivized to participate in the flow. The flow corridor acts as a controlled volume of airspace that establishes a steady flow of aircraft that are conflict-free and already resolved to be uncapacitated with a route that circumvents potential weather activity.

Flights inside the corridor would get priority treatment over those outside. However, there is the potential for multiple major flows of traffic within the same region. In this scenario, a prioritization scheme can be implemented where the corridors with the highest flow rate can be processed first before those with lower flow rates in order to ensure that the most throughput is achieved overall. The route of a corridor processed earlier would act as a capacitated region to those corridors that are rerouted through the same region of space.

5.6.2 Metering of Flights at Flow Entry Point

As flights originate from different airports, they will reach the entry point of the flow corridor at different times. In order to ensure an efficient flow through the corridor and satisfy the flow capacity, the arrival times of the flights at the entry point must be metered.

Much work has been completed, and others still underway, on metering traffic into an airport or certain point in the air space, typically by assigning time slots to flights. A large body of work has been completed on allocation of arrival time slots to flights while they are still on the ground and part of a Ground Delay Program (GDP). The slot assignment process consists of a Ration-by-Schedule algorithm to shift flights to satisfy capacity, and a Compression algorithm to fill empty time slots vacated by cancelled or delayed flights. The process has also been modeled with inter-airline slot exchanges as a bartering process, in which each “round” of bartering is performed by solving an optimization problem by lexicographically minimizing the maximum delay. [46]

In this application, we are interested in metering the traffic coming into a flow corridor to ensure a steady capacity-resolved stream of flights, similar to what would be done while metering traffic on approach to a runway, although without the constraints and complexities of terminal airspace.[44] This step is a precursor to the flow rerouting model discussed in the earlier sections. While these target arrival times could be assigned when the aircraft is still on the ground in a manner similar to that used for GDPs, there would still be a need to fine-tune the time near the entry point due to variations in the aircraft’s path (such as from vectors or reroutes) along the way. Thus, the intent is for this function to be performed about 20 minutes or so prior to the aircraft reaching the entry point. Because the aircraft is already in the air about to enter, there is no chance of cancellation, which alleviates the need for the Compression-type algorithm as in GDPs, leading to a more simplified model. In addition, because the aircraft is assumed to have already planned to use the flow corridor (or the flow corridor is close enough to the aircraft’s original path should it chose to amend its path to join it; otherwise it wouldn’t chose it), the amount of delay that an aircraft is expected to absorb to join the stream is assumed to be small and manageable by the aircraft. The approach consists of two steps:

- Based on the aircraft’s current flight plan, determine when the aircraft would be scheduled to arrive at the entry point of the flow tube.
- Adjust the flights’ arrival times to smooth out congestion, and assign each flight a Required Time of Arrival (RTA) at the entry point.

The first step is assumed to be completed separately, and can be determined using any flight trajectory generation tool. We will focus here on the second step of determining the RTA for the flights, shown in Figure 65.

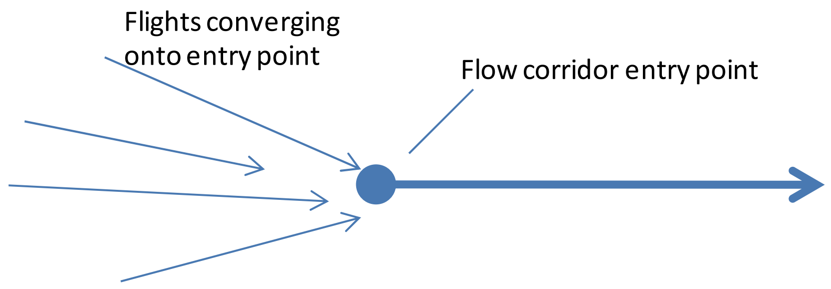

Flights converging onto entry point
Flow corridor entry point

Figure 65. Multiple flights arriving at the entry point.

5.6.2.1 General Formulation

For this problem, we wish to minimize the difference in time (delay) of each flight's original arrival time and its new arrival time while adhering to capacity / flow rate constraints. Let c_{tf} be the cost of assigning flight *f* to time slot *t*. The objective function is then

$$\min \sum_{t,f} c_{tf} x_{tf}$$

where $x_{tf} = \begin{cases} 1 \text{ if flight } f \text{ is assigned to time slot } t \\ 0 \text{ otherwise} \end{cases}$

For the costs, a super-linear function was used in [46] for the assignment of arrival slots to flights under a GDP to encourage giving a moderate amount of delay to two flights rather than a large delay to one and a smaller delay to another. For this application, a similar form of the constraint can be used:

$$c_{tf} = \left(t - s_f\right)^{1+\varepsilon}$$

where s_f is the flight's original scheduled time of arrival, and $0 < \varepsilon < 1$. To ensure that the number of flights assigned to each time slot is within capacity K_t, we define the constraint

$$\sum_f x_{tf} \le K_t \qquad \forall t$$

To ensure that a flight is not assigned an arrival time that is earlier than when it could have arrived initially, let t_0^f be the time that flight *f* was originally scheduled to arrive at the entry point. Then,

$$x_{tf} = 0 \qquad \forall t \in \{1, \ldots, t_0^f\}$$

Finally, all flights must be assigned a time:

$$\sum_t x_{tf} = 1 \qquad \forall f$$

5.6.2.2 Problem Structure & Results

The problem can be simplified with the elimination of the constraints that ensure arrival times no earlier than the scheduled times by simply not defining those variables. This reduces the number of constraints and variables significantly, especially for long durations. For the remaining two constrains, we note that a set of capacity constraints exist for each time slot, whereas a total assignment constraint exists for each flight. The combination of these two sets of constraints mean that there are only at most two 1s in each column, and the rows can be arranged so as to separate the two sets of constraints, thus yielding a Totally Unimodular (TU) structure. Thus, the problem can be solved via Linear Relaxation methods.

Figure 66 shows a sample result. The white bars indicate the capacity at each time slot. Red bars show the original traffic scheduled to arrive at the entry point. The blue bars show some of the traffic delayed in order to keep below the flow capacity.

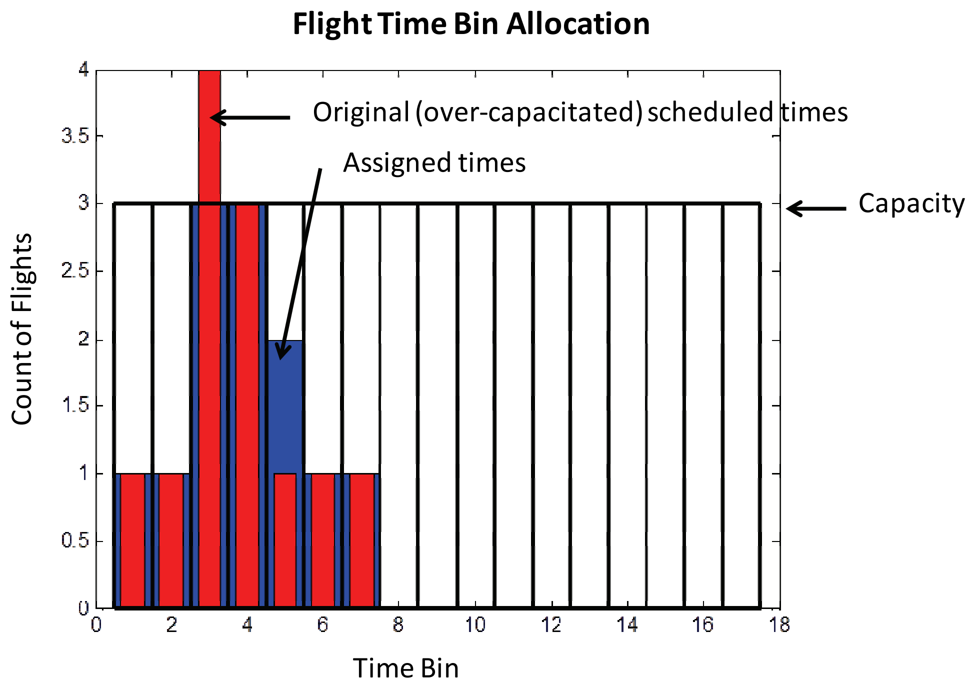


Figure 66. RTA slot allocation.

5.6.3 Route modification

After establishing a route for traffic to circumvent the capacitated area, the weather situation may have changed, possibly requiring the route to be modified, and causing the affected aircraft to transition to a new route. Much work has also been done in examining the dynamics of changes in a route structure, but not so much specifically for the purposes of tubes and corridors in this situation.

In the terminal domain, research had been conducted by NASA as part of the Distributed Air-Ground Traffic Management (DAG-TM) on the notion of Self-Spacing for merging and in-trail separation for use in generating a stream of aircraft coming in to a final approach fix for landing. [2][4] As part of the Advanced Terminal-Area Approach Spacing (ATAAS) flight evaluation and demonstration, a control system was developed for the aircraft Flight Management System (FMS) that tracked aircraft, and was successfully demonstrated with live flights at Chicago O'Hare International Airport. While this concept was targeted at the final approach phase of flight, some of the principles can be applied.

Once a flow corridor is defined and the arrival times and speeds are assigned to each aircraft, the stream of aircraft can be directed through without much interaction from one aircraft to another. As shown in Figure 67, if the route needs to change, the aircraft that first approaches a point where the change occurs is assigned as the lead aircraft. This lead aircraft would be assigned a new route and be the first aircraft through the new trajectory. The aircraft behind the lead would be instructed to follow the lead aircraft and maintain proper speed and separation as instructed by the controller or automation. This strategy would require that the aircraft be properly equipped with a Flight Management System (FMS) capable of tracking a lead aircraft and maintain a minimum separation.

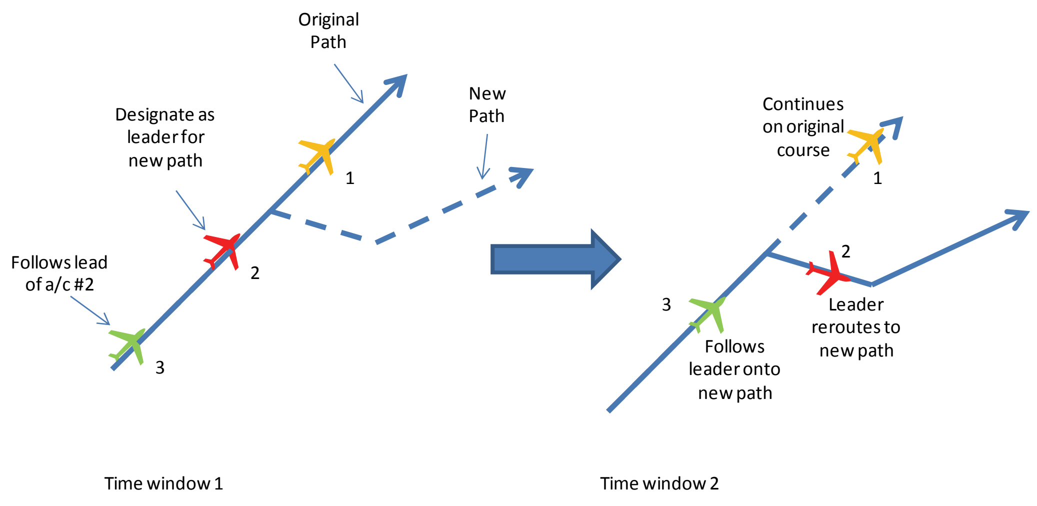

Original Path
New Path
Designate as leader for new path
Follows lead of a/c #2
1
2
3
Time window 1
Continues on original course
1
2
3
Leader reroutes to new path
Follows leader onto new path
Time window 2

Figure 67. Moving aircraft onto modified route.

5.6.4 Re-evaluation

This overall approach of aggregating flights using clustering, rerouting, and assigning individual flight arrival times could be used repeatedly for a sliding time window. While weather products are being developed with ever longer-term forecasts, they will not be as accurate as the nearer term forecasts. Longer-term forecasts will be less accurate by nature, thus relying more on the need for probabilistic information, resulting in more complex automation and concepts of operation to make use of the information.

The more tactical weather products within the 0-2 hour period will continue to provide more accurate and reliable predictions. Thus, the aggregate traffic flow approach can be repeated on regular time intervals (of less than about 2 hours) as follows:

1. Perform for first time window:
    a. Obtain great circle segments for individual flights in time window
    b. Perform the Incremental Clustering algorithm
    c. As needed, reroute flows around flow constrained areas
    d. Assign required time of arrivals at entry points
2. Repeat for each new time window
    a. Obtain great circle segments for flights in time window
    b. Perform the Incremental Clustering algorithm
    c. As needed, reroute flows around flow constrained areas
        i. If route moves, identify the lead aircraft
        ii. Assign others to follow the lead
    d. Assign required time of arrivals at entry points

The main difference between what's done in the subsequent time windows and the initial window mainly consists of incremental clustering using the updated flight plan information and adjusting the routes based on updated weather information. As shown in Figure 68, once the process has been performed for an initial time period, the time window of consideration can be advanced, and the process can be performed again, but this time incrementally clustering based on the previous state and incorporating in the new flight information.

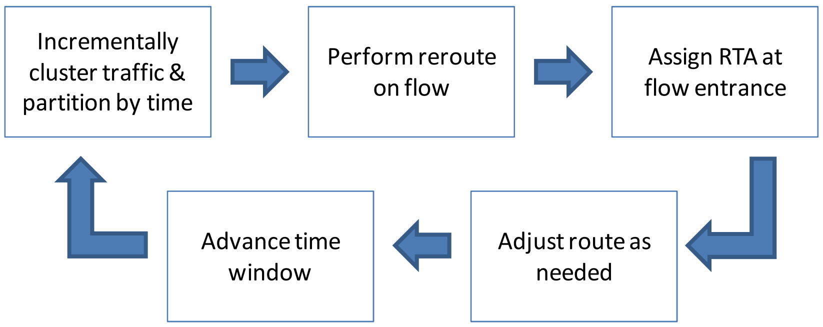

Incrementally cluster traffic & partition by time
Perform reroute on flow
Assign RTA at flow entrance
Adjust route as needed
Advance time window

Figure 68. Cycle of Incremental Clustering & re-evaluating flows.

Chapter 6. Conclusions and Future Work

6.1 Summary & Discussion

A novel approach to determining the major flows of air traffic using an incremental clustering technique was presented. An agglomeration approach to Deterministic Annealing was used that was shown to perform in about 4/100th of the time of the traditional Deterministic Annealing cooling method while still producing results that are nearly identical. As a further improvement, a subsetting algorithm was developed to cluster only those segments that are near the segments of interest which further reduced the processing time by another 75%, performing at just 1/100th of the time of the cooling algorithm, while producing clustering results that have a similarity index of over 0.92.

In addition to clustering by geometry, a method to partition the clusters by time so as to obtain sections of clusters that occupy the same space in an ongoing flow was also shown. Metrics for evaluation of the quality of the clusters in terms of cluster compactness and isolation were also presented and used for selection of the two key parameters ($l_{\max}$ as the stopping condition for geometric clustering and $\tau_{\max}$ for time partitioning).

Together, the clustering and time partitioning strategy provide intuition on the location and time of the major flows of air traffic that are crucial for pilots to avoid congestion, for controllers to target their traffic management initiatives to the proper locations, and for airspace designers to determine the best layout of airways that closely match actual routes. If the individual flight segments were grouped into their respective clusters in the form of flow corridors, the number of conflict points can be reduced by as much as 10-15%.

An application of the traffic flows was also shown for aggregate traffic flow management, where operations on traffic management are performed on entire flows of traffic rather than individual flights. In particular, a framework for rerouting flows through weather activity in an aggregate manner was shown, and compared with the case of rerouting individual flights. With these two cases modeled as Integer Programs, rerouting individual flights was shown to have a problem size and performance that are both second-order polynomial relative to rerouting entire flows. In addition, rerouting flows provides as much as 20% reduction in the number of conflict points as compared rerouting individual flights.

Considerations for implementing these algorithms for real-time air traffic management were also discussed. The concept and framework was further expanded to periodically re-cluster the traffic at sliding time windows to make use of new flight and weather information. The ability to repeatedly re-cluster and re-assess the situation reduces the need for longer-term weather forecasts that usually require complex probabilistic models. Instead, the simpler, faster models presented can be solved periodically based on the latest, most accurate weather information available. Finally, strategies for handling changes in the flow corridor’s trajectory were discussed with the use of leader-based in-trail self-separation techniques.

6.2 Future Improvements & Analyses

Although the Agglomerate Deterministic Annealing clustering algorithm is already rather efficient, its performance could be further increased if the temperature of the next merge between clusters can be predicted, which will allow for intermediate temperatures to be skipped. A potential area of focus is the total entropy vs total distortion throughout the clustering run, and determining when there are knees in the curve that would indicate a potential point for cluster merges.

In addition, the angular distortion between segments within the same cluster is implicitly incorporated to a certain extent in the Euclidean distance measure used in the Deterministic Annealing algorithm. For very short segments or large $l_{\max}$, this could result in large angular differences. Enhancements to the model could include an additional constraint on the angular difference between each segment and the cluster center to help combat the possibility.

For the reroute problem, the formulation can be enhanced by limiting the number of turns in the trajectory. Rather than adding a constraint, the preferred approach would be to add this to the objective function as another cost factor.

Comparisons can be performed on how successful traffic management controls can be when applied to operations inside a flow corridor versus individual flights. Work done in [18] has provided a rate control index as a single performance value for the actual flow of traffic into a certain region of airspace as compared to the planned flow, and could be applied for examining traffic in the clusters.

Simulations can be setup to run the scenarios with aircraft trajectories as input, the clustering performed, and the flights adjusted to fly on the clustered trajectories. This could be integrated into NAS simulation tools such as NASA's Airspace Concept Evaluation System (ACES) or Future ATM Concepts Evaluation Tool (FACET). The dynamics of the flights can be observed, including potential conflicts and flow volumes.

6.3 Further Applications of Clustering

This section describes some potential future applications of the clustering algorithm. The clustering algorithm can also be used to facilitate dynamic resectorization, where the goal is to optimally partition the airspace to minimize workload and airspace complexity. The result of clustering provides the location and times of the major flows in the airspace and a workload measure can be developed that is used to derive the cost functions for objective function.

6.3.1 Structure of Tubes

Work done in [19][42] has provided some insight into some principles for designing tubes for Dynamic Airspace Configuration. Further work could continue on tube structures. Increased separation may be needed upon exit of the tube for additional safety as flights begin to diverge towards their various destinations, resulting in a larger exit. The increased separation may also be needed because of worsening weather conditions.

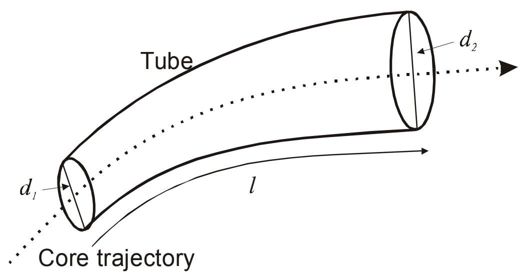


Figure 69. Great-circle tube with varying diameters for entry and exit.

In addition, one or more tubes could be connected into longer segments of varying shape, as shown in Figure 70. Each tube portion throughout the NAS could have varying diameters depending on the level of traffic that it needs to support over that zone or region of airspace. In the example below, the middle tube may have a lot of traffic entering and leaving that tube in addition to traffic that is transitioning / traversing through the tube from the first to the third tube, and thus requires a larger diameter.

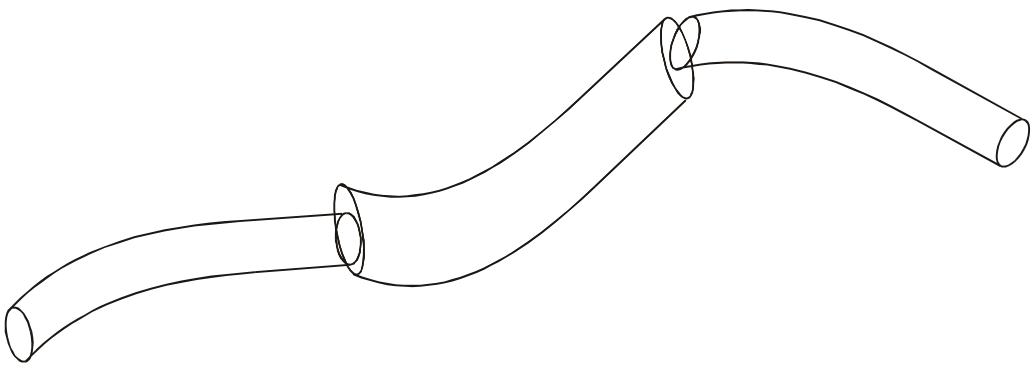


Figure 70. Connection of multiple great-circle tubes to form longer a longer tube.

Nominally, there will be a single path of flights traversing through the tube. However, if there is a high volume of traffic within a certain tube, parallel routes can be dynamically generated either within the tube, as shown in Figure 71, or in a separate tube to be managed by another controller. In this case, the width of the tube would have to be made wider to accommodate the additional tracks.

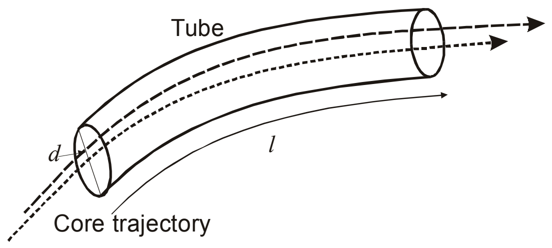


Figure 71. Tube with two parallel tracks.

With these tube structures, the underlying network of sectors would still need to be in place to manage traffic outside the tube. As shown in the figure below, the tube structures would overlap and pass through several of these underlying sectors. The shaded area may require coordination between both controllers if flights will enter or exit. Methods and procedures for coordination between the underlying sectors and tube structures will need to be developed.

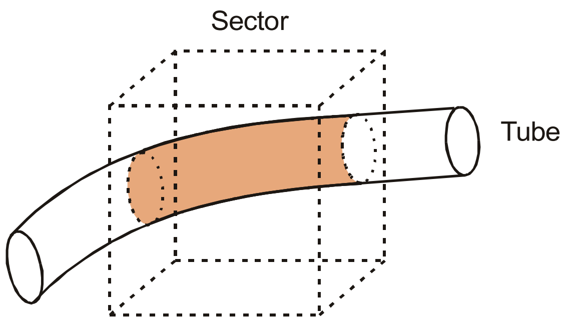


Figure 72. Intersection of traditional airspace sector with tube structure.

References

[1] ___, “Concept of Operations for the Next Generation Air Transportation System,” Version 3.0, Joint Planning and Development Office, 2009.

[2] ___, “DAG-TM Concept Element 11 Terminal Arrival: Self-Spacing for Merging and In-Trail Separation Operational Concept Description,” NASA Ames Research Center, Dec. 12, 2002.

[3] ___, “NextGen Implementation Plan,” Federal Aviation Administration, 2012.

[4] Abbott, T., “Speed Control Law for Precision Terminal Area In-Trail Self Spacing,” NASA Langley Research Center, NASA/TM-2002-211742, July 2002.

[5] Abrahamsen, T.R., Bowden, B.D., DeArmon, J.S., and McLaughlin, M.P., “Identifying coherent flows in air traffic” *IEEE Digital Avionics Systems Conference (DASC)*, 2003.

[6] Andreatta , G., Dell’Olmob, P., and Lulli, G., “An Aggregate Stochastic Programming Model for Air Traffic Flow Management,” *European Journal of Operational Research*. Vol. 215, No. 3, 2011, pp. 697–704.

[7] Avjian, R., Dehn, J., and Stobie, J., "NextGen Trajectory-Based Integration of Grid-Based Weather Avoidance Fields," *American Meteorological Society Annual Meeting*, Jan. 2011.

[8] Ball, M.O., Hoffman, R., Odoni, A., and Rifkin, R., “A Stochastic Integer Program with Dual Network Structure and Its Application to the Ground-Holding Problem,” *Operations Research*, Vol. 51, 2003, pp. 167–171.

[9] Benner, W., and Carty, T., “National Convective Weather Forecast (NCWF) Product Demonstration Final Report”, Federal Aviation Administration, 2000.

[10] Bilimoria, K., and Jastrzebski, M., “Aircraft Clustering Based on Airspace Complexity,” *AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, Sept. 2007.

[11] Bilimoria, K., and Lee, H., “Analysis of Aircraft Clusters to Measure Sector-Independent Airspace Congestion,” *AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, Sept. 2005.

[12] Ester, M., Kriegel, H.P., Sander, J., and Xu, X., “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. 2nd Int'l Conf. on Knowledge Discovery and Data Mining*, Aug. 1996.

[13] Evans, J. E., Carusone, K. M., Wolfson, M. M., Crowe, B. A., Meyer, D. R., and Klingle-Wilson, D., “The corridor integrated weather system (CIWS),” *Conference on Aviation, Range, and Aerospace Meteorology (ARAM)*, 2002.

[14] Evans, J.E., and Ducot, E., “Corridor Integrated Weather System”, *Lincoln Laboratory Journal*, Vol. 16, No. 1, 2006.

[15] Ganji, M., Lovell, D., Ball, M., and Nguyen, A., “Resource Allocation in Flow-Constrained Areas With Stochastic Termination Times,” *Transportation Research Record: Journal of the Transportation Research Board*, Vol. 2106, 2009, pp. 90-99.

[16] Hackwood, S., and Beni, G., “Self-organizing Sensors by Deterministic Annealing,” *IEEE/RSJ international Workshop on Intelligent Robots and Systems IROS*, 1991, pp. 1177–1183.

[17] Han, J., and Kamber, M., *Data Mining Concepts and Techniques, 2nd Edition*, Elsevier, Boston, 2006.

[18] Hoffman, R. and Ball, M., “The Rate Control Index for Traffic Flow Management,” *IEEE Transactions on Intelligent Transportation Systems*, Vol. 2, No. 2, 2001, pp. 55-62.

[19] Hoffmann, R. and Prete, J., “Principles of Airspace Tube Design for Dynamic Airspace Configuration,” *Congress of International Council of the Aeronautical Sciences*, Sept 2008.

[20] Hoffmann, T. and Buhmann, J., “Pairwise Data Clustering by Deterministic Annealing,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 1, Jan. 1997, pp. 1–14.

[21] Horng, J., and Li, J., "A Dynamic Programming Approach for Fitting Digital Planar Curves with Line Segments and Circular Arcs," *Pattern Recognition Letters*, Vol. 22, No. 2, Feb. 2001, pp. 183–197.

[22] Jonk, A., and Smeulders, A., “An Axiomatic Approach to Clustering Line-segments,” *International Conference on Document Analysis and Recognition*, 1995, Volume 1, p. 386.

[23] Lee, J., Han, J., and Whang, K., “Trajectory Clustering: A Partition-and-Group Framework,” *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2007.

[24] Li, X., Hu, W., and Hu, W., “A Coarse-to-Fine Strategy for Vehicle Motion Trajectory Clustering,” *International Conference on Pattern Recognition*, 2006.

[25] Libby, M., Buckner, J., and Brennan, M., “Operational Concept for Airspace Flow Program (AFP),” Version 1, FAA Air Traffic Organization, 2005.

[26] Lighthill, M., and Whitham, J., “On Kinematic Waves. I Flow movement in long rivers. II A theory of Traffic Flow on Long Crowded Roads,” *Proc. Royal Society*, Vol. 229, No. 1178, 1955, pp. 281-345.

[27] Manousakis, K., and Baras, J. “Dynamic Clustering of Self Configured Adhoc Networks Based on Mobility,” *Proceedings of Conference on Information Sciences and Systems*, March 2004.

[28] Menon, P., Sweriduk, 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, Sept-Oct 2004.

[29] Milenova, B., and Campos, M., “O-Cluster: Scalable Clustering of Large High Dimensional Data Sets”, *IEEE International Conference on Data Mining*, 2002.

[30] Mitchell, J., Polishchuky, V., Krozel, J., “Airspace Throughput Analysis Considering Stochastic Weather”, *AIAA Guidance, Navigation, and Control Conference*, 2006.

[31] Mullen, K., and Hayoz, F. (Editors), "Statement of Needs, Collaborative Convective Forecast Product (CCFP-2005)," Weather Applications Workgroup (WAWG), 2005.

[32] Myers, T., Kierstead, D., and Wagner, D., "A fully-dynamic network flow model of the NAS," *Integrated Communications, Navigation and Surveilance Conference (ICNS)*, 2011.

[33] Nguyen, A., and Baras, J., "Network Cell Routing Model For Control Of Throughput And Delay of Air Traffic", *Proceedings of The 28th Digital Avionics Systems Conference (DASC)*, 2009.

[34] Nilim, A., El-Ghaoui, L., Duong, V., and Hansen, M., “Trajectory-Based Air Traffic Management (TB-ATM) Under Weather Uncertainty,” *Proceedings of the USA/Europe Air Traffic Management R&D Seminar*, Dec. 2001.

[35] Nilim, A., El Ghaoui, L., and Duong, V., “Multi-Aircraft Routing and Traffic Flow Management under Uncertainty,” *Proceedings of the USA/Europe Air Traffic Management R&D Seminar*, 2003.

[36] Richards, P., “Shockwaves on the Highway,” *Operations Research*, Vol. 4, No. 1, 1956, pp. 42-51.

[37] Rose, K. “Deterministic Annealing for Clustering, Compression, Classification, Regression and Related Optimization Problems,” *Proceedings of the IEEE*, Vol. 86, No. 11, Nov. 1998, pp. 2,210-2239.

[38] Sabhnani, G., Yousefi, A., Kierstead, D., Kostitsyna, I., Mitchell, J., and Polishchuk, V., "Algorithmic traffic abstraction and its application to NextGen

generic airspace," *AIAA Aviation Technology, Integration, and Operations Conference*, 2010.

[39] Sander, J., Ester, M., Kriegel, H., and Xu, X., “Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications,” *Data Mining and Knowledge Discovery*, Vol. 2, No. 2, 1998. pp 169-194.

[40] Song, L., Wanke, C., and Greenbaum, D., “Predicting Sector Capacity for TFM Decision Support,” *AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, Sept. 2006.

[41] Song, L., Wanke, C., Greenbaum, D., and Callner, D., “Predicting Sector Capacity under Severe Weather Impact for Traffic Flow Management,” *AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, Sept. 2007.

[42] Sridhar, B., Grabbe, S., Sheth, K., and Bilimoria, K., “Initial Study of Tube Networks for Flexible Airspace Utilization,” *AIAA Guidance Navigation and Control Conference*, 2006.

[43] Sun, D., Clinet, A., and Bayen, A., “Disaggregation Method for an Aggregate Traffic Flow Management Model,” *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 3, May-June 2010.

[44] Swenson, H. N., et al, “Design and Operational Evaluation of the Traffic Management Advisor at the Fort Worth Air Route Traffic Control Center,” *Air Traffic Management R&D Seminar*, June. 1997.

[45] Vlachos, M., Kollios, G., and Gunopulos, D., “Discovering Similar Multidimensional Trajectories,” *International Conference on Data Engineering, San Jose*, 2002.

[46] Vossen, T., Ball M., “Optimization and mediated bartering models for ground delay programs,” *Naval Research Logistics*, Vol. 53, No. 1, 2005, pp75–90.

[47] Wambsganns, M., "Collaborative decision making through dynamic information transfer," *Air Traffic Control Quarterly*, Vol. 4, 1996, pp. 107–123.

[48] Wolfson, M., et al, “Consolidated Storm Prediction for Aviation (CoSPA),” *Integrated Communications, Navigation, Surveillance Conference (ICNS)*, 2008.

[49] Wolfson, M., and Clark, D., “Advanced aviation weather forecasts,” *Lincoln Laboratory Journal*, Vol. 16, No. 1, 2006, pp. 31-58.

[50] Yousefi, A., Zadeh, A., and Tafazzoli, A., “Dynamic Allocation and Benefit Assessment of NextGen Flow Corridors,” *International Conference on Research in Air Transportation (ICRAT)*, June 2010.