Determining the Value of Information for Minimizing Controller Taskload: A Graph-Based Approach

Adan Vela, John-Paul Clarke, Eric Feron, Nicolas Durand, William Singhose

***Abstract*—In the future, air traffic controllers will most likely come to rely on decision-support tools and increased levels of automation to help manage and separate aircraft. Conflict detection and conflict resolution are examples of two key areas where increased automation and improved accuracy are considered imperatives to the future efficiency of airspace systems. The inclusion of decision-support tools for conflict-detection and resolution is expected to reduce controller workload by decreasing the mental stress associated with identifying potential conflicts and maintaining aircraft separation. Despite the benefits of such systems, there has been little study into the best methods to implement conflict-detection and resolution algorithms in practice, and what is the resulting controller taskload. In this paper, we examine how the capabilities and implementation strategy of conflict-detection and resolution tools affect controller taskload. Our goal is to understand how conflict-detection and resolution decision-support tools can best be designed and implemented to support human-based control of aircraft.**

I. Introduction

THE projected growth in air transportation demand is likely to result in traffic levels that exceed the capacity of the unaided air traffic controller. Consequently, air navigation service providers are making efforts to improve the capacity and throughput of existing airspaces through airspace redesign, trajectory based operations, incorporation of new traffic flow management tools, and introduction of automated communication and navigation systems [1].

In addition to the above mentioned efforts, there has been significant investment into the study and development of aircraft conflict-resolution algorithms over the past two decades. Early examples include, [2], [3], with a more comprehensive survey of the proposed models presented in [4]. Recently, research and development has focused on the design of automated conflict-resolution algorithms that provide provably safe solutions with realistic aircraft trajectories [5], [6]. As a whole, the automated algorithms neglect the human controller; researchers inherently assume that these algorithms will form the basis for a fully automated air traffic control system. There are key exceptions that provide 'human-centric' algorithms for conflict resolution [7]–[10].

In light of such conflict-resolution algorithms, many air traffic operators and researchers believe that the tactical role of radar air traffic controllers will eventually transition to a more strategic role, an example discussion is provided in [11]. Overall, there still exist limitations and problems associated with most conflict-resolution algorithms, both human-centric and automated, as they are considered to be part of a completely autonomous system, require advanced communication and navigation sub-systems, or fail to state how they are implemented in practice with a human controller. In particular, they do not address fundamental questions like: How far ahead in time should conflicts be considered? How often should trajectory and conflict information be updated? What are the required levels of certainty for best performance? Furthermore, for many automated systems, there remain concerns about the safety and realizability of automated tactical conflict-resolution algorithms in governing traffic. Even the human-centric approaches found in [7], [8] are hindered by the slow uptake of the advanced avionics required to fully support semi-automated tactical air traffic control.

Thus, before the conversion to a semi- or fully-automated air traffic control system, there will be a need for systems to aid air traffic controllers without replacing them. Such a human-in-the-loop framework makes use of integrated conflict-detection and resolution tools to identify conflicts and propose resolution commands for the air traffic controller to verify and issue to aircraft. The inclusion of human-in-the-loop decision-support tools in human-based air traffic control operations requires a fundamentally different approach to the design and implementation of conflict-detection and resolution algorithms. The algorithms must explicitly acknowledge the role of the controller and accommodate their abilities.

The research presented here seeks to understand how the formulation and implementation of conflict-detection and resolution decision-support tools affect controller taskload for conflict resolution. This task is accomplished by the modeling conflict-detection and resolution process through graph-based relationships, similar to [12]. Instead of focusing on the nature of a maneuver applied to an aircraft (i.e. heading, altitude, or speeds changes), the proposed research approach tracks the number of resolution commands required to resolve potential air traffic conflicts. The graph-based representation of conflicts enables this approach, and thereby provides a lower-bound on the amount of effort required to deconflict aircraft. The lower-bound on controller effort defines a performance reference model by which future air traffic conflict-detection and resolution algorithms may be compared against.

In this study, we develop an airspace model to generate uncontrolled aircraft trajectories for multiple traffic intensities. Graph-based conflict-detection and resolution algorithms are then applied to each traffic sample to determine the number of resolution commands required to separate aircraft. Each graph-based conflict-detection and resolution model is parameterized to consider multiple policies (e.g. first-come first-serve, optimal), a range of conflict-detection certainty and horizons (e.g. 5, 10, 20 min look-ahead), and implementations (event-based vs. discrete time). An analysis of the algorithms is presented, focusing on the number of resolution commands required to

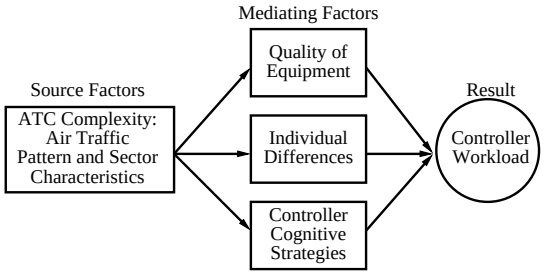


Fig. 1: Representation of factors affecting controller workload. [15]

separate traffic, defined in this paper as *taskload*.

This papers addresses the following research questions:

1) What is the value of information towards reducing controller taskload?
2) How often should information be updated?
3) How does the quality of information affect controller taskload?

The questions listed above are fundamental to the design of human-based decision-support tools for conflict-detection and resolution algorithms. Ultimately, the major contribution in this paper is the introduction and demonstration of methodologies for determining performance reference models for conflict-detection and resolution algorithms. These models provide knowledge into the best practices for the design and implementation of decision-support tools to aid air traffic controllers in limiting taskload under high traffic volumes.

In the next section, a brief review of previous research on the conflict-resolution process and controller workload is presented. Section III describes how graphs are used to model conflict relationships and the conflict-resolution process. Next, Section IV presents two conflict-resolution implementation policies, and Section V introduces the traffic model used for studying the conflict-resolution process. An initial study of conflicts within a sector is provided in Section VI. Afterwards, the graph-based conflict-detection and resolution policies are applied, and analysis results are given in Section VII. Finally, conclusions are presented in Section VIII.

II. Background

In practice, airspace capacity is limited by the ability of air traffic controllers to manage and separate aircraft [13]. The stresses associated with air traffic control correspond to controller workload, "the amount of effort, both physical and psychological, expended in response to system demands (task load) and also in accordance with the operators internal standard of performance [14]." Therefore, if human-in-the-loop conflict-detection and resolution tools are to be used to reduce controller workload, and thereby increase the effective capacity of an airspace, then they must account for the factors that lead to high controller workload in the conflict-detection and resolution decision-making process.

In the review by Mogford et al [15], the authors identify four sets of factors that influence controller workload: air traffic control complexity, quality of equipment, individual differences, and controller cognitive strategies. The relationship between these four factors and controller workload are highlighted in Figure 1.

Unaccounted for in the modeling of Mogford is the feedback between source factors and mediating factors. As air traffic controllers issue resolution commands to aircraft they introduce a loop that allows the airspace and controller interaction to be characterized as a dynamical control system. We hypothesize that by considering both source and mediating factors, a better understanding of the relationship between controller workload and the conflict-resolution process will enable the design of decision-support tools that are both technologically sound and consistent with modes of human work-practice. Specifically, we account for both source and mediating factors by studying the number of potential conflicts (source) that are resolved according to set controller policies (Controller Cognitive Strategy) and the ability to identify conflicts (quality of equipment).

The concept that controllers operate within a feedback loop is not new. In human-factors research there is a recognition that "workload is not something imposed upon a passive [air traffic controller] but, rather, is something the [air traffic controller] actively manages [16]." More so, when resolving potential conflicts, air traffic controllers consider a number of factors in determining if, what, and when resolution commands are issued in an effort to manage workload. Noted in [17], during periods of low workload, controllers wait for potential conflicts to develop before deciding to take action. Conversely, during times of high workload, controllers take action immediately, instead of waiting to establish if a conflict will be realized or not. The economy of air traffic controller resources extends to the type of resolution commands used to deconflict aircraft. The authors of [18] state that while diverse in conflict-resolution strategies, during periods of high workload, typically, air traffic controllers economize their time and effort by issuing resolution commands that are clear, simple, safe, and require limited follow-up.

If human-in-the-loop conflict-detection and resolution tools are to replace some of the mental functions of air traffic controllers, then they will be required to complete the same set of tasks: identifying potential conflicts; determining if action is needed; and establishing what and when resolution commands are issued. How to implement these tasks remains a relatively unanswered question. While it is true a number of studies have considered human-in-the-loop control for semi- and fully-automated separation [19], [20], the mathematical formulations behind the algorithms have not been thoroughly explored. In fact, despite the plethora of knowledge concerning human factors, there has been limited integration of this knowledge into strong mathematically-based conflict-detection and resolution tools that are consistent with human work-practice. Instead of designing algorithms to work with controllers, the compatibility between algorithms and controllers appears to be verified as an afterthought, as though all automated conflict-detection and resolution algorithms are the same.

That is not to say that consideration of the mathematical formation of conflict-detection and conflict-resolution tools does not exist. Studies have focused on the accuracy of trajectory prediction in identifying potential conflicts, an example includes [21]. Also, it has been demonstrated that rule-based priority policies for conflict resolution always have shortcomings [22]; there are no simple winning strategies.

We believe that the human controller will continue to play an important role in tactical air traffic control well into the future. Accordingly, conflict-detection and resolution algorithms will need to be designed to support, not replace, human controllers. In this paper, we strive to understand how the design and implementation of conflict-detection and resolution algorithms affect the controller taskload. By answering the three fundamental question in regards to the amount, the update rate, and the quality of information, insights into this relationship aids in the design of decision-support tools.

III. Problem Modeling

Addressing bounds on controller taskload is fundamental to establishing the amount of effort required by air traffic controllers to manage and separate aircraft. The required effort to resolve conflicts remains relatively unanswered according to the three questions surrounding the nature of information posed in Section I. Through a graph-based approach, controller taskload associated with the conflict-detection and resolution process is addressed. Graphs provide a methodology to better understand how implementation parameters, and the availability and quality of information in the conflict-detection and resolution process affects controller taskload.

Consider a set of M aircraft, $\mathcal{A} = \{A_1 \dots A_M\}$, traveling through an airspace, as illustrated in Figure 2a. Aircraft trajectories are assumed to occur in 3D space. According to aircraft trajectories and intents, the aircraft have the potential to be in multiple conflicts if no control action is taken. For the en route environment, aircraft are declared to be in conflict if they come within 5NM laterally and 1,000ft vertically of each other. The aircraft and aircraft conflict relationships are represented by a graph. A possible representation of the conflicts for the example in Figure 2a is given by the undirected graph, $G = (V, E)$, depicted in Figure 2b. Aircraft are represented by nodes in the vertex set $V = \{n_1, \dots, n_M\}$, where node n_i corresponds to aircraft A_i. Any pair of aircraft, (A_i, A_j), that are in potential conflict requires resolution. Potential conflicts are indicated by an undirected edge in the edge set E. That is, $(n_i, n_j) \in E$. In this framework, the conflict-resolution process acts on the graph by removing edges between any two nodes. A conflict-free airspace implies a completely disconnected graph, and vice-versa.

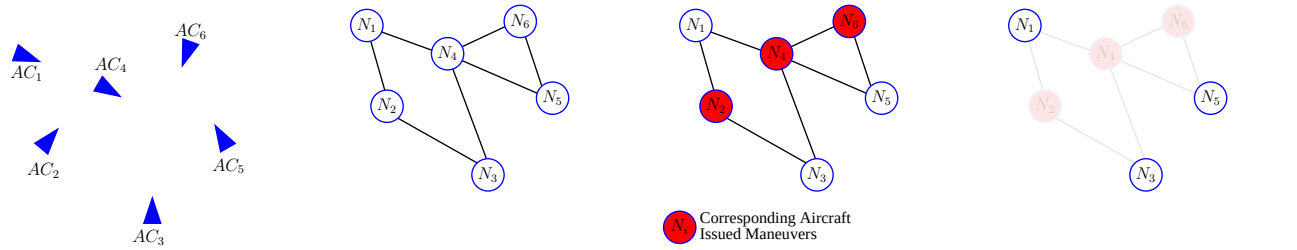
Determining the minimum number of resolution commands required to separate and deconflict aircraft is equivalent to applying the minimum vertex cover problem for graphs. The minimum vertex cover problem asks: 'What is the minimum number of nodes to remove such that the graph is completely disconnected?' The corresponding act of removing a node n_i from the graph is to issue the aircraft A_i a resolution command. At this stage, no assumptions are made concerning the nature of the resolution command. In fact, the command may consist of an arbitrary number of maneuvers (heading, speed, or altitude changes), each of arbitrary magnitude. An example application of the minimum vertex cover problem is shown in Figure 2c; following removal of the nodes, the remaining graph is completely disconnected, as illustrated in Figure 2d. Note, the minimum vertex cover problem may have multiple solutions. For the remainder of the paper, application of the minimum vertex cover problem to the conflict-resolution process is referred to as the minimum taskload problem.

Application of the minimum taskload problem does have drawbacks. Given that it is independent of any conflict-resolution program, the solutions provide no information on the nature of the resolution commands required to deconflict aircraft (heading, speed, altitude changes). Furthermore, there exists an underlying assumption that the aircraft remain conflict-free after being issued resolution commands. These assumptions lead the minimum taskload problem to represent a lower bound on the number of required maneuvers to deconflict aircraft.

The minimum taskload problem can be expressed as a Mixed Integer Linear Program (MILP). For a set of M aircraft with conflict graph, $G = (V, E)$, let R_i be a binary variable indicating if aircraft A_i is to be issued a resolution command. If any two aircraft have nodes within the edge set E, then at least one of the corresponding binary variables must be 1. To minimize the number of maneuvers, the sum of the binary variables, R_i, is minimized. The MILP formulation is:

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{N} R_i \\ \text{s.t.} \quad & R_i + R_j \geq 1 \quad \forall\ (n_i,\ n_j) \in E \\ & R_i \ \in \{0, 1\}\ \forall\ i = 1 \dots M \end{aligned} \tag{1}$$

The minimum taskload problem, or any other algorithm used for conflict resolution, can also be applied in the dynamic case. That is, as aircraft appear in the airspace the minimum taskload problem is solved, resulting in an updated graph for the system. Let $G(k) = (V(k), E(k))$ be the conflict graph representing the visible/identifiable potential conflicts, and the aircraft controllable by the air traffic controller at time-step k. The visibility of potential conflicts (edges) are determined by the certainty associated with trajectory prediction. For example, if a conflict between the aircraft pair (A_i, A_j) occurs in 20 minutes, but, limitations in trajectory prediction only allow forecasting of conflicts in the next 10 minutes, then the conflict edge $(n_i, n_j) \notin E(k)$. Aircraft are visible if they enter the airspace within the next H minutes. Also, aircraft (nodes) are controllable if the aircraft is handed-off to the controller or located within the airspace. Visibility of the potential conflict between an arbitrary aircraft pair (A_i, A_j) at time-step k, implies that the edge $(n_i, n_j) \in E(k)$. Visibility and controllability of aircraft A_i is interpreted to mean that the node $n_i \in V(k)$. Thus, if an aircraft A_i is visible and controllable, and also part of a visible conflict, then application of the minimum taskload problem can issue a resolution



(a) Aircraft traversing airspace (b) Graph representation (c) Possible solution (d) Remaining graph is completely disconnected

Fig. 2: Represent and solve air traffic conflicts through a graph representation.

command to aircraft A_i; otherwise, no command is issued to aircraft A_i.

In the dynamic case, the conflict graph grows/shrinks incrementally with the appearance/exit of aircraft and the resolution of potential conflicts. Beginning with the conflict graph $G(k) = (V(k), E(k))$, $V^+(k)$ is defined to be the set of aircraft arriving into the airspace at time-step k. Any new conflicts within the same time-step are described by the edge set $E^+(k)$. Note, $E^+(k)$ consists of new conflicts generated by the arrival of new aircraft, and also any potential conflicts that have just became visible within the airspace. $E^+(k)$ may contain nodes (aircraft) that were already in the graph (airspace) prior to time-step k. Let $V^-(k)$ be the set of nodes corresponding to the aircraft exiting the airspace. If resolution commands are not yet issued at time-step k, then the temporary graph at k, just prior to time-step $k+1$, is given by $G^T(k) = (V^T(k), E^T(k))$, such that

$$\begin{aligned} V^T(k) &= (V(k) \cup V^+(k)) \setminus V^-(k) \\ E^T(k) &= (E(k) \cup E^+(k)). \end{aligned} \tag{2}$$

Assume that conflict-resolution control actions act on the temporary graph $G^T(k)$. Let CRP represent an arbitrary conflict-resolution policy (CRP) function. The function takes as an input $G^T(k)$ and returns $G(k+1)$. That is, $G(k+1) = CRP(G^T(k))$. The conflict-resolution policy issues resolution commands to aircraft, and in doing so removes edges from the graph. Let $E^r(k)$ correspond to the edges removed through the conflict-resolution process. For example, if the potential conflict between aircraft A_i and A_j is resolved by CRP at time-step k, then $(n_i, n_j) \in E^r(k)$. The updated edge set $E(k+1)$ at time-step $k+1$ is $E(k+1) = E^T(k) \setminus E^r(k)$.

For the remainder of the paper, it is assumed aircraft pairs only have the possibility of conflicting once within a sector. If the conflict between A_i and A_j is resolved at time-step k, then there is no potential for future conflicts between the same aircraft. As such, if $(n_i, n_j) \in E^r(K)$, then for all $K > k$, $(n_i, n_j) \notin E(K)$.

If aircraft A_i is issued a resolution command at time-step k, then the aircraft is guaranteed to be conflict-free for length of the look-ahead time of the trajectory predictor. For example, if the controller and computer systems are capable of predicting aircraft trajectories for the next H minutes, then the generated resolution command for aircraft A_i should ensure conflict-free travel over the same time period. After H minutes, aircraft A_i may be be involved in other new potential conflicts.

In future sections, the minimum taskload problem is applied as a control policy under a variety of implementation parameters to discern the value of look-ahead times, increased information update rates, improved accuracy of conflict-detection, and resolution strategies. Hence forth, application of the minimum taskload problem is referred to as the MTP policy.

IV. Implementation and Strategy Models

As stated, the goal of our research is to understand how controller taskload relates to a number of implementation and strategy models that are employed in the conflict-detection and resolution process. In doing so, we address questions concerning the value of information introduced in Section I. Study and comparison of the implementations provides reference models to assess controller taskload. Specifically, a first-come first-serve rule-based policy that consider issuing resolution commands to newer aircraft is compared to the MTP policy implemented under receding-horizon control. Both conflict-resolution policies act on the dynamic graph representation of the traffic as described in Section III.

Considering that conflict-resolution algorithms can only function under the guidance of a conflict-detection algorithm, the application of both must be considered in parallel. The manner in which conflicts are identified is critical; a conflict-resolution algorithm takes as its input a set of identified potential conflicts. Each conflict is typically characterized by its time of occurrence and the estimated minimum separation between the aircraft. Which potential conflicts are passed as constraints is also important. Potential conflicts that are too far into the future, or that have too much uncertainty are temporarily ignored or go unresolved until more accurate information is present. In addressing the three fundamental questions about information, three key parameters are considered: H, refers to time range over which conflicts can be identified; δt refers to how often new potential conflicts are identified and how often the conflict-resolution problem is solved; and finally, D_{sep} is defined to be the minimum distance between aircraft

at which a potential conflict is identified to require resolution. Aircraft trajectories that come within D_{sep} of each other require resolution commands. Typically, air traffic controllers resolve conflicts to ensure approximately 8NM separation [10], however, it has been noted that aircraft controllers take action (or are concerned) about aircraft pairs that come within 15NM of each other [7].

For the study, the receding-horizon control process used by the conflict-resolution algorithms is first discussed, along with the pertinent implementation parameters. Afterwards, the two controller policies (first-come first-serve and MTP) are discussed.

A. Receding-Horizon Control

Receding-horizon control provides an implementation framework in which to resolve potential conflicts. When conflict-detection and resolution occur by means of a decision-support tool, the tool can only handle limited amounts of information, both certain and uncertain. To overcome information overload, decision-support tools are required to parse information. As such, receding-horizon control is a reasonable representation for computer assisted conflict resolution.

Implementation of a conflict-resolution policy under receding-horizon control is depicted in Figure 3. Every δt, the conflict-resolution problem is solved while looking H time into the future. All potential conflicts between an arbitrary aircraft pair, (A_i, A_j), are assigned a time of conflict given by $t^c_{i,j}$. The potential conflict times $t^c_{i,j}$ within H time are visible and of concern to the controller. Those potential conflicts that occur within δt of the current time must be resolved or else they will become realized. Therefore, the solution of the conflict-resolution problem, while considering all available information, is only applied to potential conflicts occurring within the next δt minutes.

The receding-horizon control process for conflict resolution operating on the conflict graph is illustrated in Figure 4. First, as shown in Figure 4a, some of the aircraft in the airspace are in potential conflict, of which a fraction become realized at the end of the δt time period if no action is taken. Additionally, aircraft soon entering the sector within H time are visible to the controller. However, the aircraft has yet to be handed-off to the controller, and hence it is not yet controllable.

Within the δt time window, new aircraft become visible to the controller. These aircraft approaching the boundary may be in conflict within H time, as depicted in Figure 4b. New potential conflicts within the airspace may be identified. Note though, that all newly identified potential conflicts do not occur within the next δt. As a requirement $\delta t < H$, otherwise, conflicts are not identified in sufficient time to resolve them.

In Figure 4c, the conflict-resolution problem is solved for the current graph, however, only the aircraft corresponding to conflicts occurring within the next δt are issued resolution commands. Figure 4d depicts the resulting graph following the arrival of aircraft, and the identification and resolution of conflicts within the δt time window. At this point, new initial conditions are present for the next δt time period, and the process is repeated. Since trajectory information is available for all aircraft for the H minutes, any well-designed conflict-resolution algorithm should guarantee resolution commands issued to an aircraft are conflict-free for a least H minutes.

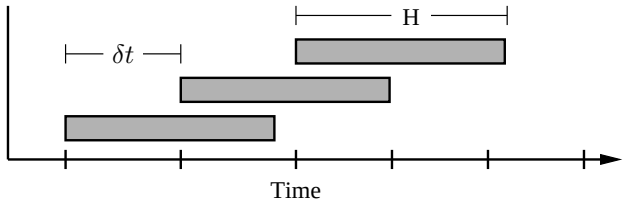


Fig. 3: Conflict-resolution problem solved at δt time-steps, looking ahead H.

Adjusting the parameters H, δt and D_{sep} varies implementation conditions for the conflict-detection and resolution process. For example, as $H \to \infty$ and $D_{sep} \to$ 5NM, simultaneously, the solution to the receding-horizon control implementation is similar to having complete knowledge of all conflicts with complete certainty. And when the minimum taskload problem is applied as the conflict-resolution algorithm, the solution corresponds to the globally optimal solution with the least number of resolution commands. When the conflict-resolution problem is implemented in practice as a decision-support tool, then every δt, the air traffic controller may receive a new set of resolution commands.

Converse to the previous example, when $\delta t \to 0$ and $H \to 0$, the implementation approaches an event-based implementation. Because as $\delta t \to 0$ and $H \to 0$, for each infinitely small time-step only a single event can occur. In this case, the event corresponds to identification of an upcoming conflict or a future aircraft arrival into the airspace. In the event-based case, the implementation degenerates into a simplified case where conflicts are resolved as they appear. Such an implementation is the worst possible implementation, as information is not collected in an effort to reduce controller taskload.

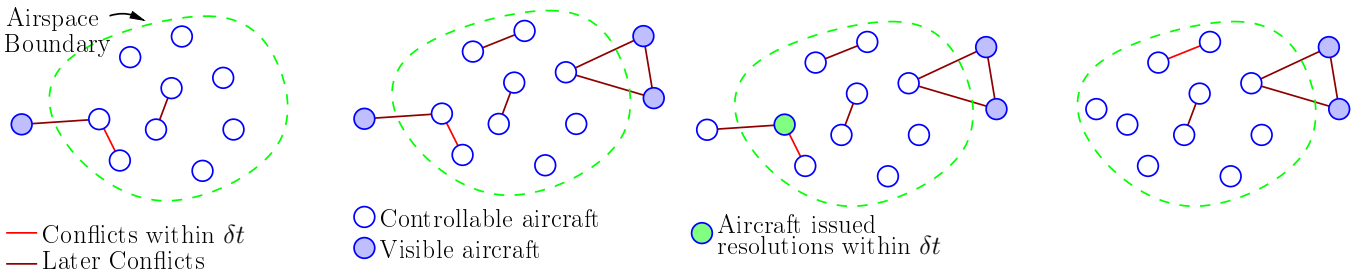
In summary, H corresponds to a measure of the amount of information available to the conflict-resolution algorithm, while D_{sep} is a measure of the certainty/quality in the prediction of future aircraft trajectories and potential conflicts. Adjusting δt varies how often potential conflicts are identified and resolution solutions are provided to the controller.

B. MTP Policy

For MTP policy, as part of the receding-horizon control framework, the MTP is solved at each time-step to resolve any conflicts. By minimizing the number of resolution commands over any particular time period, the conflict-resolution taskload is implicitly minimized over longer periods of time. Again, the MTP policy is based on the minimum taskload formulation described in Section III.

C. Rule-Based Policies

As a basis of comparison, a first-come first-serve (FCFS) rule-based policy is introduced. The FCFS policy represents a



(a) Initial graph.

(b) New aircraft and conflicts become visible.

(c) Apply conflict resolution.

(d) Resulting graph. Conflicts become actionable, aircraft become controllable.

Fig. 4: Receding-horizon conflict-resolution process applied to a graph representation.

simple strategy commonly found in conflict-resolution implementations. The policy issues resolution commands to aircraft in accordance to their arrival into the airspace. Similar to the MTP policy, the rule-based policy makes use of δt, H, and D_{sep}, to describe its implementation and information availability, as well as the identification and inclusion of potential conflicts. The FCFS policy requires the use of $\tilde{T}^a$, the set of aircraft arrival times into the sector.(Note the arrival and conflict time, t_i^a and $t_{i,j}^c$ are in absolute terms.)

When implementing the FCFS policy, aircraft already in the airspace have priority over new aircraft; if a potential conflict is identified between an existing aircraft and a newer aircraft, then the newer aircraft is issued a resolution command.

V. Simulation Data

Based on the implementation models described in Section IV, simulations are completed in accordance with mock flight plans generated through selective sampling of 28 days of historical high-altitude (≥FL200) air traffic passing through Minneapolis Center (ZMP)[1]. Historical flight radar data cannot be used directly to generate potential conflicts, as the associated aircraft are under the control of air traffic controllers, and hence should be conflict-free throughout their trajectories. Therefore, a new traffic model is required in which flight trajectories are not yet deconflicted with each other.

To generate potential conflicts based on realistic air traffic, radar data for the busiest traffic day in the 28 day set[2] is used as a seed to create new mock flight plans. Matching the origin-destination pairs found in the seed day, trajectories from all days are sampled and reinserted to create 50 new sets of mock flight plans. For each set of mock flight plans, the traffic intensity is increased by reducing the timespan over which the flights occur.

The first step in generating the traffic model is to record all origin-destination pairs for the 107,671 high-altitude flights over the 28 days of traffic. A visual representation of the traffic routes is provided in Figure 5; the densest traffic regions are located in the southern half of the center, which correspond to

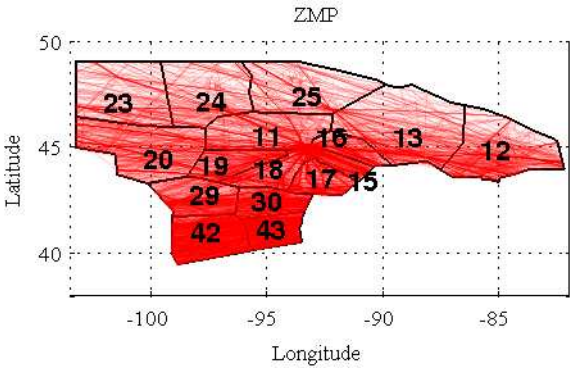


Fig. 5: Traffic density in ZMP.

the darker regions in the figure. The origins and destinations are described according to fixes (e.g. NASAL), airports (e.g. KATL), and latitude-longitude data (e.g. 4510N/10354W). Origin and destination pairs that are described by fixes and airports are indexed directly. For approximately 4% of the data (4329 trajectories), the origin data is described by latitude and longitude. In these cases, the origins are spatial clustered by hand, and indexed accordingly. Any latitude-longitude origins that are not clustered, form an 'outlier' cluster.

Following clustering and categorization, a total of 11,014 origin-destination pairs are identified. Ranking the origin-destination pairs according to the number of occurrences, the most common origin-destination pair, ROBBY to KMSP, occurs 2442 times. The fix ROBBY is located outside and to the south of ZMP, near the border with Chicago Center; KMSP corresponds to Minneapolis St.Paul International Airport. According to the distribution, 82% of all flights correspond to approximately 1300 origin-destination pairs that repeat at least 10 times over the 28 days.

The original seed day contains 4,654 flights, and is described by the aircraft arrival times into the center (i.e. 0-24hr)[3] by $\mathbf{T}^a = [T_1^a, \ldots, T_{4654}^a]$, and the corresponding origin-destination pairs, $\mathbf{P}^{od} = [P_1^{od}, \ldots, P_{4654}^{od}]$, where each origin-destination pair is a couple, e.g. $P_{18}^{od} = (\text{KMSP, KDFW})$.

To mimic the traffic pattern of the seed day, for each origin-

[1]PDARS data consists of radar tracks for the following days: May 21, 2007 - June 17, 2007

[2]Busiest traffic day corresponds to June 14, 2007

[3]Some flights appear in the 24th hour.

destination pair P_i^{od}, a new aircraft trajectory is selected from all other flights with the same origin-destination pair over the 28 days. An origin-destination pair is defined to be rare if over the 28 days there are less than 10 flights with the same origin and destination. In the case when P_i^{od} from the seed day is rare, a new mock flight plane is selected from the set of all rare origin-destination pairs within 15 minutes of the arrival time of the seed aircraft, T_i^a. By matching the arrival times within a 30 minute window (± 15 min), the overall traffic pattern during the time of day is approximately preserved.

The new aircraft arrivals times into the center, $\tilde{T}_i^a$, for the new mock flight plans are given by $\tilde{T}_i^a = (T_i^a + 5\mathcal{N})/I$, where $\mathcal{N}$ is a standard normal random variable, and I is a scaling factor based on the traffic intensity. Adding the random normal variable perturbs the initial aircraft arrivals to minimize the likelihood of replicating overly similar traffic cases. The value I scales the timespan over which the simulation occurs. When $I = 1$, the new mock flight plans and traffic arrivals have the same traffic intensity of the original seed data over the center. When $I = 3$, the traffic intensity is declared to be 3.0X.

Once the 50 sets of mock flight plans (each with multiple intensity levels) are generated for the complete center (each set containing 4654 mock flight plans), they are further parsed down to the sector level. The center definition, along with sector boundaries, used for ZMP are illustrated in Figure 5. Based on the mock flight plans, the arrival times into each sector are recalculated to provide t_i^a, which are used by the MTP and the rule-based policy FCFS.

For each sector, conflict graphs are generated by checking which pairs of mock flight plans pass within 5NM and 1000 vertical feet of each other. Interpolating each mock flight plan, conflicts are checked every second. Additionally, because this study focuses on en route aircraft, only potential conflicts that occur at or above FL200 are considered. The pair of aircraft and time corresponding to each conflict is documented. The conflict-time, $t_{i,j}^c$, for aircraft A_i and A_j, is assigned the value of the time the minimum separation is first violated. The same process is repeated to determine the aircraft pairs that come within 6, 7, 8, 10, 12, and 15NM of each other (according to the parameter value of D_{sep}) for all traffic intensities.

VI. Sector Conflict Analysis

To better understand the conflict event-process in a sector, an initial analysis of the mock flight plans is provided.

For the purpose of this paper, a key measure is the number of potential conflicts within a sector. Here, a potential conflict is defined to occur between two planes when the flight plans break the minimum separation requirements, according to the parameter D_{sep}. Sets of aircraft that have potential conflicts with one another are considered to form a conflict cluster. In relation to the graph representation, the number of conflicts is given by the number of edges in the edge set, i.e Number of potential conflicts = $card(E)$. The number of conflict clusters is defined by the number of disconnected subgraphs within the graph G, which is given by the number of 0 valued eigenvalues of the Laplacian of the the graph G.

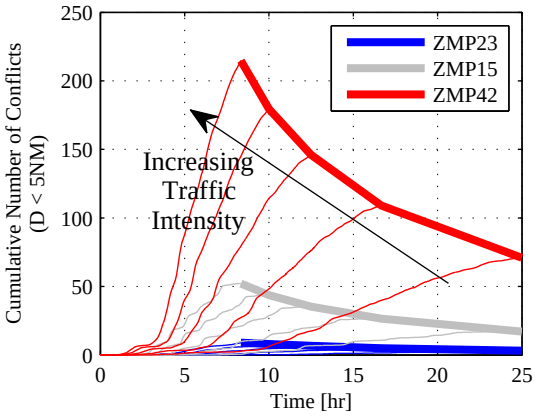


Fig. 6: Cumulative number of potential conflicts for 1-3X traffic intensity.

According to the generated model, the center ZMP exhibits approximately three levels of conflict complexity across all sectors. For the 1X traffic intensity level, Sectors 42 and 43 have the greatest number of potential conflicts over the nominal day. The least number of potential conflicts are found in Sectors 23, 24, and 25. The remaining sectors have a similar number of potential conflicts. As the traffic intensity is increased from 1X to 3X, these groupings remain the same. The relative ranking of the number of potential conflicts in each sector is expected given the traffic distribution illustrated in Figure 5; Sectors 42 and 43, the busiest sectors, are located in the southern portion of the center, while Sectors 23, 24, and 25 are located in the less busy northern portion of the center.

As the traffic intensity increases, the number of potential conflicts in the sectors increases super-linearly. Figure 6 illustrates the cumulative potential conflict totals for Sectors 15, 23, and 42 over multiple traffic intensities (1.0X, 1.5X, 2.0X, 2.5X, 3.0X) with $D_{sep} = 5$NM. Over a 1 day time period, at 1X traffic intensity, Sector 42 averages 70 uncontrolled potential conflicts; at 3X intensity, there are approximately 220 conflicts over 8 hours. The growth rate in the cumulative number of conflicts grows fastest for Sector 42, in comparison to Sector 15 and 23. Interesting to note, that even at 3X traffic intensity, the number of conflicts in Sector 15 is less than that for Sector 42 at 1X traffic intensity.

The conflict complexity is also measured by the fraction of conflicts that are pairwise. Conflict resolution of pairwise conflicts, excluding the possibility of generating secondary conflicts, requires only a single aircraft be issued a resolution command. However, for multi-aircraft conflict clusters, the conflict-resolution process becomes more difficult as more aircraft require maneuvers. Thus, more advanced algorithms are required for sectors with greater conflict complexity. For all the sectors in ZMP, as the traffic intensity increases from 1X to 3X, the fraction of pairwise conflicts decreases linearly. Sector 42 has the greatest conflict complexity: approximately 88% of all conflicts are pairwise at 1X traffic intensity; at 3X intensity, 66% of all potential conflicts are pairwise when

$D_{sep} = 5$NM. Sectors 23, 24, and 25, show low levels of complexity. Even at 3X traffic intensity, more than 90% of all conflicts are pairwise.

Because most potential conflicts are pairwise in Sectors 23, 24, and 25, over all traffic intensities, the differences in taskload between the FCFS and MTP policies will be relatively negligible. Near equal performance of the two policies is also hypothesized for Sectors 42 and 43 at low traffic intensities.

For purposes of interest, due to its high conflict totals and large percentage of multi-aircraft conflicts, Sector 42 is isolated for further study.

VII. Simulation Results

According to the two policies described in Section IV (MTP and FCFS), simulations are run based on the traffic model for Sector 42 of ZMP for a range of traffic intensities (1.0X, 1.5X, 2.0X, 2.5X, 3.0X). By adjusting the parameters H, δt, and D_{sep}, the three questions posed in the introduction are addressed. The value of the look-ahead time is tested by increasing the magnitude of H, and studying the number of resolution commands required to deconflict traffic. Decreasing the value of δt, represents increasing the update rate of information provided to the conflict-resolution algorithm and how often the controller issues resolution commands. And finally, to study the affect of the quality of information on controller taskload, the required separation distance is adjusted. Larger values of D_{sep} represent increased uncertainty in radar systems; this often leads to resolution commands that promote safely conservative actions by air traffic controllers. As a reference standard, the policy MTP is compared against the FCFS policy. Additionally, because air traffic controllers traditionally separate aircraft at distance greater than 5NM, the basis of comparison assumes that aircraft are spaced at 8NM.

Applying the two policies with $D_{sep} = 8$NM, the average taskload counts for the 5 traffic intensities are depicted in Figure 7 and Figure 8 for FCFS and MTP. As expected for the policies, as the traffic intensity increases, so does the average taskload counts across all values of H and δt. The value of the look-ahead time, H, is vital to reducing controller taskload for both policies. However, the benefit of increasing H, to reduce controller taskload, has diminished returns for $H > 20$ minutes. Considering the distribution of aircraft transit times within the sector, such a result should be of no surprise. For Sector 42, 65% of all aircraft take 20 minutes or less to pass through the airspace, and as shown in Figure 9, virtually all take less than 25 minutes.

While H has a meaningful effect on the taskload of the controller, the update rate δt, is less influential under more realistic operating conditions. For both the MTP and FCFS policies, given sufficiently large values of $H \geq 20$, the update rate, has little effect on the the taskload. But, when $H < 20$, larger differences in taskload are expected when varying the update rate. For the case when $\delta t/H \rightarrow 1$, the influence grows. Therefore, for the purposes of minimizing controller taskload, if the range of the look-ahead time is limited to 10 minutes, then efforts should focus on generating rapid updates

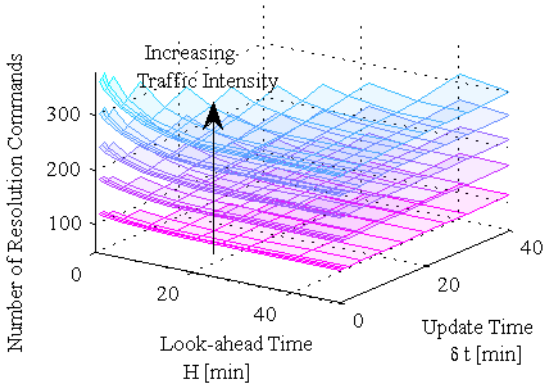


Fig. 7: Average number of resolution commands for the FCFS policy ($D_{sep} = 8$NM).

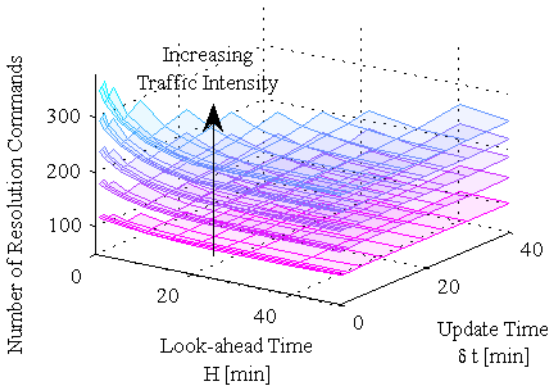


Fig. 8: Average number of resolution commands for the MTP policy ($D_{sep} = 8$NM).

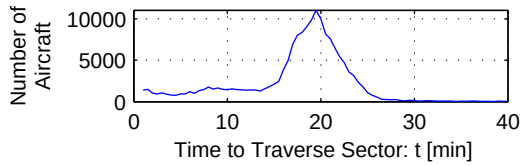


Fig. 9: Histogram of aircraft transit times through the sector.

to both the conflict-detection and conflict-resolution processes. It should be noted that this analysis is based on a lower-bound. Actual implementation of conflict-resolution algorithms may demonstrate that δt has greater importance in practice.

The comparison between the MTP policy and the FCFS policy demonstrate that the MTP policy is better able to manage the number of resolution commands. As illustrated in Figure 10, except at $H \sim 0$ and $\delta t \sim 0$, the MTP policy requires less resolution commands over a range of traffic levels and implementations. The reduction in the average expected taskload reaches 10% for many cases when $H = 15$ minutes. While not large, such a reduction is without doubt beneficial for reducing the workload of a controller. As hypothesized

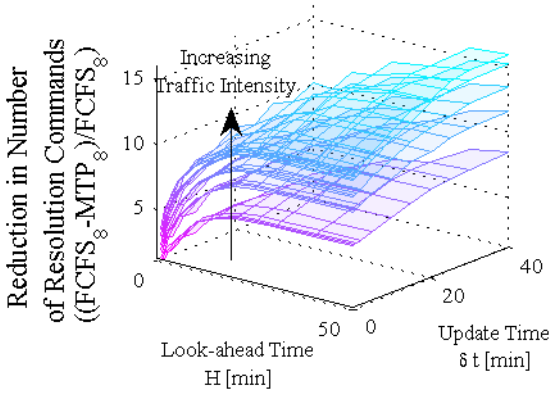


Fig. 10: Percent difference in taskload between the FCFS and MTP policies when $D_{sep} = 8$NM.

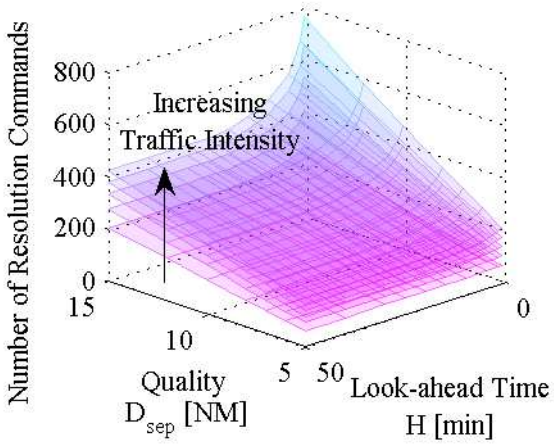


Fig. 11: Average number of resolution commands for MTP for $\delta t = 1$ minute.

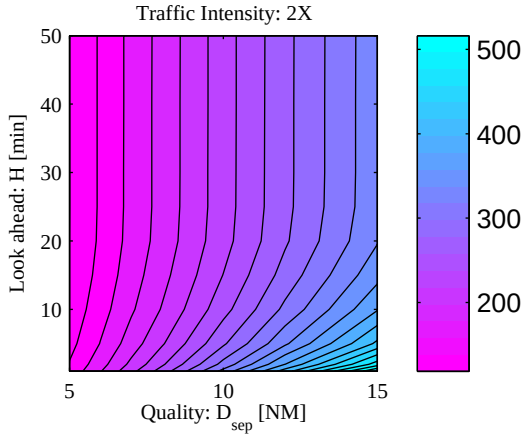


Fig. 12: Average number of resolution commands for MTP for $\delta t = 1$ minute at 2X traffic intensity.

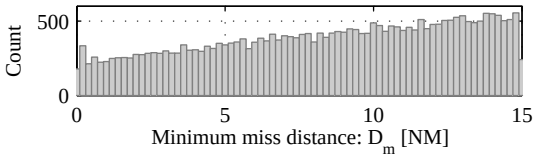


Fig. 13: Histogram of minimum miss distances between uncontrolled aircraft trajectories.

earlier, the benefit of the MTP policy over the FCFS policy is more pronounced as the traffic intensity increases. Again, this results is most likely related to the fraction of pairwise clusters. When cluster size increases, a greater optimality gap between the MTP and FCFS policy exists for controller taskload.

Because the update rate has limited effect on the taskload, a more pertinent study considers the amount of information available to the quality of the information. Figure 11 illustrates the number of resolution commands for the MTP policy as a function of H and D_{sep}. Extracting results for the 2X traffic intensity, a contour plot of the same information is provided in Figure 12. The results demonstrate that when quality of information is poor, ie D_{sep} is large, the most effective measure to reduce controller taskload is to increase the range of the look-ahead time. However, as previously demonstrated, such actions have diminishing returns. Once conflict-detection is able to provide information up to 20 minutes, additional information is not as useful as increasing the accuracy of the information. Figure 11 and Figure 12 highlight the trade-off between information versus quality. When designing conflict-detection and resolution tools, a balance must be struck. Instead of providing large amounts of information with poor quality into conflict-resolution algorithm, less information is sometimes preferred if the quality can be guaranteed. Except at smaller values of $H < 10$, the relationship between controller taskload and D_{sep} is virtually linear. Such a result is best explained by the distribution of minimum miss distances between aircraft, depicted in Figure 13. Even as traffic intensity increases, the distribution of miss distances between uncontrolled aircraft increases linearly.

VIII. Conclusion

Engineers, researchers, and designers of conflict resolution algorithms for air traffic systems have largely and implicitly ignored human factors issues by designing their algorithms to replace rather than support air traffic controllers. And while human factors and cognitive engineering researchers have highlighted key aspects and requirements for the successful design decision-support tools, there has been little actualization of these concepts into mathematically rigorous models. Along these lines, we have demonstrated that the design and the implementation of conflict-detection and resolution algorithms has an effect on controller taskload. Therefore, in order to improve the design human-in-the-loop conflict-detection and resolution decision-support tools, the amount of information (H), the update rate of information (δt), and the quality of information (D_{sep}), as well as the implementation strategy must be considered.

Future research will consider how controller taskload is

temporally distributed according to the implementation parameters. Furthermore, greater fidelity is required in distinguishing between the look-ahead time for conflict detection and conflict resolution. We suggest that similar experiments introduce additional terms to parameterize the algorithms; H_D and H_R to replace H. H_D considers how far into the future conflicts can be identified, while H_R is a measure of how long resolution commands can insure conflict-free travel for an aircraft.

It goes without saying that many of the parameters discussed resemble parameters that describe the accuracy of trajectory prediction algorithms. In many ways, the results presented here can be recast to demonstrate the importance of trajectory prediction for reducing controller taskload. Regardless, with the inclusion of decision support tools, it is the conflict-detection and resolution tool that determines which aircraft are issued resolution commands. In doing so, the algorithm must consider a fixed set of information concerning the conflicts. By resolving conflicts too far in advance, the quality of information cannot be guaranteed, and can result in superfluous resolutions commands. Conversely, waiting for improved accuracy in trajectory information can limit options in reducing taskload. Fundamentally, the amount and quality of information must be balanced to ensure manageable levels of controller taskload.

The major advancement of this work is to introduce a framework in which to study the design and implementation of conflict-detection and resolution algorithms in relationship to controller taskload. And ultimately, we have established a performance reference model by which future conflict-detection and resolution algorithms can be compared against.

IX. Acknowledgments

Work funded by NASA Grant NNX08AY52A; FAA Award No.: 07-C-NE-GIT, Amendment Nos. 005, 010, and 020; Air Force contract FA9550-08-1-0375.

References

[1] Joint Planning and Development Office, "Next Generation Air Transportation System Integrated Work Plan: A Functional Outline," Joint Planning and Development Office, Washington, DC, Tech. Rep., Sept. 2008.

[2] W. Niedringhaus, "Maneuver Option Manager: Automated Simplification of Complex Air Traffic Control Problems," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 5, pp. 1047–1057, 1991.

[3] N. Durand and J.-M. Alliot, "Optimal Resolution of En Route Conflicts," in *USA/Europe Air Traffic Management R&D Seminar*, 2001.

[4] J. Kuchar and L. Yang, "A Review of Conflict Detection and Resolution Modeling Methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, pp. 179–189, Dec. 2000.

[5] G. Chaloulos, G. Roussos, J. Lygeros, and K. Kyriakopoulos, "Ground Assisted Conflict Resolution in Self-Separation Airspace," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[6] N. Dougui, D. Delahaye, S. Puechmorel, and M. Mongeau, "A New Method for Generating Optimal Conflict Free 4D Trajectory," in *International Conference on Research in Air Transportation*, 2010.

[7] G. Gawinowski, J.-L. Garcia, R. Guerreau, R. Weber, and M. Brochard, "Erasmus: A new path for 4d trajecotry-based enablers to reduce the traffic complexity," in *Digital Avionics Systems Conference*, 2007.

[8] E. Cruck and J. Lygeros, "Subliminal air traffic control: Human friendly control of a multi-agent system," in *American Control Conference*, 2007.

[9] W. Hylkema and H. Visser, "Aircraft Conflict Resolution Taking into Account Controller Workload using Mixed Integer Linear Programming," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003.

[10] A. Vela, S. Solak, E. Feron, K. Feigh, W. Singhose, and J. Clarke, "A fuel optimal and reduced controller workload optimization model for conflict resolution," in *Digital Avionics Systems Conference*, 2009.

[11] H. Erzberger and R. Paielli, "Concept for next generation air traffic control system." *Air Traffic Control Quarterly*, vol. 10, no. 4, pp. 355–378, 2002.

[12] G. Granger and N. Durand, "A traffic complexity approach through cluster analysis," in *USA/Europe Air Traffic Management R&D Seminar*, 2003.

[13] P. Brooker, "Controller workload, airspace capacity and future systems." *Human Factors and Aerospace Safety*, vol. 3, no. 1, pp. 1–23, 2003.

[14] E. Stein, "Human Operator Workload in Air Traffic Control," *Human Factors in Air Traffic Control*, pp. 155–184, 1998.

[15] R. Mogford, J. Guttman, S. Morrow, and P. Kopardekar, "The Complexity Construct in Air Traffic Control: A Review and Synthesis of the Literature." Federal Aviation Administration, Tech. Rep. DOT/FAA/CT-TN95/22, 1995.

[16] S. Loft, P. Sanderson, A. Neal, and M. Mooij, "Modeling and predicting mental workload in en route air traffic control: Critical review and broader implications," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 49, no. 3, 2007.

[17] K. W. Kallus, D. Van Damme, and A. Dittmann, "Integrated Task and Job Analysis of Air Traffic Controllers - Phase 2: Task Analysis of En-route Controllers," EUROCONTROL, Tech. Rep. HUM.ET1.ST01.1000-REP-04, 1999.

[18] B. Kirwan and M. Flynn, "Towards a controller-based conflict resolution tool - a literature review," EUROCONTROL, Tech. Rep. ASA.01.CORA.2.DEL04-A.LIT, 2002.

[19] P. U. Lee, T. Sheridan, J. L. Poage, L. Martin, K. Jobe, and C. Cabrall, "Identification and Characterization of Key Human Performance Issues and Research in the Next Generation Air Transportation System," National Aeronautics and Space Administration, Ames Research Center, Tech. Rep., 2010.

[20] T. Prevot, J. Mercer, L. Martin, J. Homola, C. Cabrall, and C. Brasil, "Evaluation of NextGen Air Traffic Control Operations with Automated Separation Assurance," in *AIAA Modeling and Simulation Technologies Conference*, 2010.

[21] J. Alliot, N. Durand, and G. Granger, "A statistical analysis of the influence of vertical and ground speed errors on conflict probe," *USA/Europe Air Traffic Management R&D Seminar*, 2001.

[22] N. Archambault and N. Durand, "Scheduling heuristics for on-board sequential air conflict solving," in *Digital Avionics Systems Conference*, 2005.

Adan E. Vela received the B.S. degree from the University of California, Berkeley in 2003, and the M.S. degree from Stanford University in Mechanical Engineering in 2006. He is currently a Mechanical Engineering Ph.D. student at Georgia Institute of Technology. His research focuses on the application of optimization and control theory for air traffic systems.

John-Paul B. Clarke received the S.B. degree in 1991, S.M. degree in 1992 and Sc.D. degree in 1997, all from the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology. He is currently an Associate Professor in the School of Aerospace Engineering and director of the Air Transportation Laboratory at Georgia Institute of Technology. His research addresses issues in optimization and robustness in aircraft and airline operations, air traffic management and the environmental impact of aviation.

Nicolas Durand graduated from the École Polytechnique de Paris in 1990 and from the Ecole Nationale de l'Aviation Civile (ENAC) in 1992. He has been a design engineer at the Centre d'Etudes de la Navigation Aérienne (CENA) from 1992 to 2007 and holds a Ph.D. in computer Science (1996). He is currently in charge of the air traffic Planning, Optimization and Modeling team of the R&D department of DSNA.

Eric Feron holds his B.S. (1989) from the École Polytechnique, his M.S. (1990) in Computer Science from the École Normale Supérieure, and his Ph.D. (1994) in Aerospace Engineering from Stanford University. Currently, he is the Dutton-Ducoffe Professor of Aerospace Software Engineering at the Georgia Institute of Technology, and previously he was a faculty member at MIT. He is interested in using control, optimization and computer science for problems such as air traffic control systems and aerospace software certification.

William E. Singhose received the B.S. degree from Massachusetts Institute of Technology in 1990, the M.S. degree from Stanford University in 1992, and the Ph.D. degree from Massachusetts Institute of Technology in Mechanical Engineering in 1997. He is currently an Associate Professor in the School of Mechanical Engineering at the Georgia Institute of Technology. Dr. Singhose's research focuses on the dynamics and control of flexible machines.