Real-time discrete control law synthesis for hybrid systems using MILP: Application to congested airspace[1]

Alexandre M. Bayen[2] and Claire J. Tomlin[3]
Hybrid Systems Laboratory, Stanford University
bayen@stanford.edu, tomlin@stanford.edu

Abstract

We use a mathematical model based on hybrid automata theory to describe air traffic flow in arrival regions near airports. We show that the problem of scheduling the arrival flow subject to airspace and airport metering constraints may be formulated as a *Mixed Integer Linear Program* (MILP). We demonstrate an implementation of this program, which accepts *Air Traffic Control* (ATC) data, performs the scheduling, and generates a set of ATC commands directly executable by the aircraft. Simulations indicate an empirical upper bound on the number of aircraft the program can treat while still resolving the problem in real time.

1 Introduction

1.1 Motivation: physical problem

The *National Airspace System* (NAS) is a large scale, layered, nonlinear dynamic system; its control authority is currently organized hierarchically with a single *Air Traffic Control System Command Center*, in Herndon VA, supervising the overall traffic flow. This is supported by 22 *Air Route Traffic Control Centers* (or simply, centers) organized by geographical region. Each center is sub-divided into about 20 sectors, with at least one air traffic controller responsible for each sector. In centers which have relatively high traffic density close to terminal areas, there exist prescribed routes corresponding to different approaches into airport runways, called *arrivals*. These arrivals are the final portion of the aircraft *flight plans*. A flight plan is a set of *waypoints* (reference points defined precisely in the airspace), which the aircraft are expected to follow. Even though in low traffic density regions, aircraft might fly off these flight plans to benefit from faster routes (because of winds, for example), when this airspace becomes congested, aircraft will follow arrivals for up to 200 nautical miles (nm) from the destination airport. Arrivals aid controllers in the problem of flow *metering*, or delivering a prescribed number of aircraft per unit time to the airport runway, as the routes can be viewed as tracks which the aircraft follow closely with minor deviations until they reach the arrival airport. In the current system, controllers build a *mental model* of this airspace: they know how much time an aircraft takes to fly from one point to another, and how much time an aircraft can lose using minor deviations of their flight plans in order to delay the arrival. A controller can thus regulate the flow by adjusting the flight plans of individual aircraft, according to procedures or *playbooks* which have been established over time to meet the acceptance rates at airports. In this paper, we study the feasibility of automatically generating these flight plan adjustments, in order to meter flow in real time, even when the system is operating at maximal capacity.

We present and demonstrate the implementation of an algorithm which takes as input current positions of aircraft in the airspace (available from radar data through a monitoring system), produces a schedule which attempts to optimize a user-defined cost, and generates a set of commands which are contained within the command set of air traffic controllers, and are thus directly understandable by pilots. These commands are represented as discrete transitions of hybrid automata [18]. We then determine empirically, through simulations, a bound on the number of aircraft the algorithm can treat while still resolving the scheduling problem in real time (meaning that with the currently used data format, the algorithm outputs results faster than the update rate of the current monitoring system).

The main challenge in the design of this algorithm is the non-convexity of the problem, as it precludes the use of efficient convex optimization techniques with guarantees of global optimality. We employ a technique which has been used in [16, 15] to address the inherent non-convexity in conflict avoidance problems. The problem can be transformed by recognizing that non-convex constraints such as $u \in U$ where U is not convex, can sometimes be rewritten as $(u \in V) \vee (u \in W)$ where V and W are convex. More generally, non-convex constraints may be written as the disjunction of several convex constraints. If in addition V and W can be expressed as $V = \{u \in \mathbb{R}^n | \mathbf{A}u \leq b\}$ and $W = \{u \in \mathbb{R}^n | \mathbf{C}u \leq d\}$,

[1]Research supported by NASA under Grant NCC 2-5422, by ONR under MURI contract N00014-02-1-0720, by DARPA under the Software Enabled Control Program (AFRL contract F33615-99-C-3014) and by a Graduate Fellowship of the Délégation Générale pour l'Armement (France).

[2]Ph.D. Student, corresponding author. Aeronautics and Astronautics, Durand 028, Stanford University, Stanford CA, 94305-4035. Tel: (650)498-0530, Fax: (650)723-3738.

[3]Assistant Professor, Aeronautics and Astronautics.

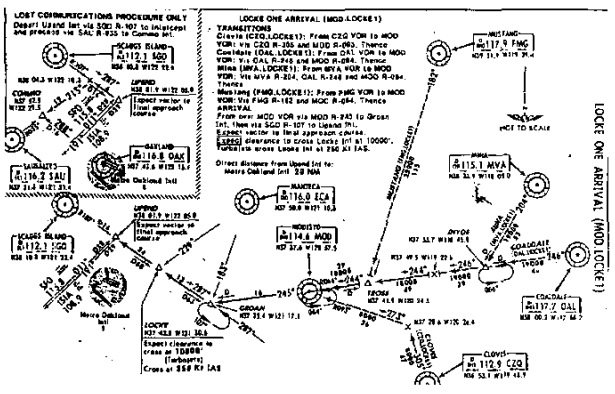

Figure 1: LOCKE 1 arrival into the Oakland (OAK) airport. Aircraft enter this airspace through the waypoints: MUSTANG, MINA, COALDALE, CLOVIS, whose acronyms are FMG, MVA, OAL, CZQ. Note the tracks for holding patterns (shown as loops at various merge points). Source: JEPPESEN[17].

with $\mathbf{A} \in \mathbb{R}^{p \times n}$, $\mathbf{C} \in \mathbb{R}^{q \times n}$, $b \in \mathbb{R}^p$ and $d \in \mathbb{R}^q$ (p and q are integers), and the objective function is linear in u, the problem can be posed as a *Mixed Integer Linear Program* (MILP), where the logical OR ($\vee$) is encoded as an integer variable.

Transposed in the framework of hybrid systems, this idea reveals itself to be very useful. A hybrid system H_i is a system with $K_i \in \mathbb{N}$ different modes $\{q_i^k\}_{k \in \{1,\cdots,K_i\}}$. Each of these modes has a given dynamic $\dot{x}_i = f_{q_i^k}(x_i)$. We want the system to accomplish certain tasks, characterized by an initial state $x_i = x_i^{\text{initial}}$ and a final state $x_i = x_i^{\text{final}}$ (and any trajectory between these two points, defined as a sequence of discrete modes whose trajectories obey the corresponding $\dot{x}_i = f_{q_i^k}(x_i)$). There may be many possible trajectories between x_i^{initial} and x_i^{final}; for each trajectory, the elapsed time may be different, as it depends on the mode *occupancy time*, or the time spent in each mode. The *completion time* is thus a function of the trajectory; we call S_i the set of all possible completion times to go from $x_i = x_i^{\text{initial}}$ to $x_i = x_i^{\text{final}}$ using the set of modes q_i^k:

$$S_i = \left\{ t \;\middle|\; \begin{array}{l} \exists L \in \mathbb{N}, \exists \{\theta_l\}_{l \in \{1,\cdots,L\}} \in \mathbb{R}^L,\ \theta_L = t \\ \exists \mu(\cdot) : \{1,\cdots,L-1\} \to \{1,\cdots,K_i\} \end{array} \text{ such that } x_i^{\text{final}} = x_i^{\text{initial}} + \sum_{l=1}^{L-1} \int_{\theta_l}^{\theta_{l+1}} f_{q_i^{\mu(l)}}(\xi) d\xi \right\} \quad (1)$$

In the previous formula, the function μ gives the mode $q_i^{\mu(l)}$ of the system between θ_l and θ_{l+1}. S_i is in general not a single interval of $\mathbb{R}$: we will assume that it can be represented as a finite closed union of intervals.

We combine multiple automata H_i, where $i \in \{1,\cdots,N\}$, with prescribed initial and final conditions for each. Assume we can compute the set S_i for each H_i.[1] Call $\Theta \in \mathbb{R}^N$ a generic element of the set of possible completion times for this family of automata. $\Theta := \{t_i\}_{i \in \{1,\cdots,N\}} \in \prod_{i=1}^N S_i$. We are interested in computing a discrete control law which achieves a user-fixed linear cost in Θ, and where linear combinations of components of Θ belong to non-convex sets:

$$\begin{array}{ll} \textbf{min:} & c^T \cdot \Theta \\ \textbf{s.t.:} & \Theta \in \bigcup_{p=1}^P \{u | \mathbf{A}_p u \leq b_p\} \\ & w_r^T \cdot \Theta \in \bigcup_{s=1}^S \{v | \mathbf{C}_{s,r} v \leq d_{s,r}\} r \in \{1,\cdots,R\} \end{array} \quad (2)$$

where $c \in \mathbb{R}^N$, $P \in \mathbb{N}$, $u \in \mathbb{R}^N$, $\mathbf{A}_p \in \mathbb{R}^{n_p \times N}$ ($n_p \in \mathbb{N}$, given by the constraints of the physical problem), $b_p \in \mathbb{R}^{n_p}$, $w_r \in \mathbb{R}^N$, $S \in \mathbb{N}$, $R \in \mathbb{N}$, $v \in \mathbb{R}$, $\mathbf{C}_{s,r} \in \mathbb{R}^{n_{s,r} \times 1}$, $d_{s,r} \in \mathbb{R}^{n_{s,r}}$ ($n_{s,r} \in \mathbb{N}$, given by the constraints of the physical problem). This problem in turn can be formulated as a MILP. We can retrieve from its solution the switching history (mode sequences and switching times of the corresponding hybrid automaton) which achieves it. Our paper presents an instantiation of (2) useful for *Air Traffic Control* (ATC).

Our formulation bears some similarities with the formulation of [5], however the physical problem is different. The authors of [5] deal with the global NAS, and optimize a cost including delays on the ground as well as airborne delays. We deal with local airborne flights which we have to adjust in their last flight portion, we do not have the notion of sectors but of flight plan alteration. The single airport problem is posed in [14] and treated as a queue problem but solved using ground hold policies. In [13], optical networks are used to solve spatial conflicts; we use similar concepts for time conflicts.

MILP [6] is a powerful mathematical formulation to extend linear programming to problems with both continuous and integer variables. It appears naturally in various fields where these two types of variables coexist, for example processing or chemical engineering [11]. It enables inclusion of computational logic [19] into optimization problems as (2), which provides an excellent tool for multi-vehicle or conflict avoidance problems [16, 15] and discrete time hybrid systems [4]. The present work enables solving continuous time hybrid systems problems under the assumptions on the continuous dynamics stated above.

This paper is organized as follows. Section 2 defines the mathematical model used for the aircraft, which is based on hybrid automata models. It explains the commands used by the ATC, and gives a model of airspace capacity (in terms of aircraft storage capability), which leads to a mathematical model for the physical constraints of the system. Section 3 shows how to pose

[1] Note that the computation of S_i does not necessary require the knowledge of occupancy time in each of the modes, i.e. the ability to integrate. It only requires the ability to produce tight bounds on the set of occupancy times for each modes, which is much easier. This is what we do here.

the problem as a MILP and how to use its solution to synthesize a set of executions of these hybrid automata which are ATC commands. The method is summarized in the form of an algorithm, whose implementation is demonstrated in section 4. Several examples of simulations are given and analyzed, for up to 60 aircraft, and an empirical bound on the number of aircraft for real time operation is presented.

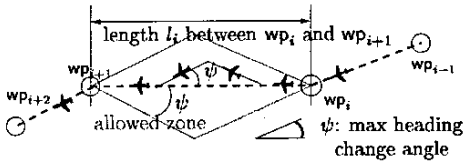

Figure 2: Deviation ψ from flight plan using a *Vector For Spacing* (VFS) available in a given sector [1]. Other models are available in [8]. Here, the range of available ψ is given by the maneuvering availability (for example: between CZQ and MOD in Figure 1, $\psi \in [-45°, 10°]$ because of the jetway TROSE-MOD).

2 Physical model

This section presents the mathematical model we use to describe the airspace as well as aircraft motion. It is essentially used to derive a mathematical expression of the physical constraints of the problem.

2.1 Airspace model

Arrival routes pass through different sectors before reaching the final descent in the *Terminal Radar Approach Control* (TRACON). The geometry of these sectors, as well as the flows of aircraft going through them, determine the maximum possible deviations available to the aircraft. In other words, it is possible to look at the geometry of these sectors and find the maximal additional flight plan length available to ATC in order to delay the flight. Besides speed changes when they are available, the most common way to lengthen the flight plan of an aircraft is called *Vector For Spacing* (VFS) (see Figure 2). In the present study, we use the following approximation for this additional length: $l_{\text{additional}} = \sum_i l_i \left(\frac{1}{\cos\psi} - 1\right)$, where $l_i = d(\text{wp}_i, \text{wp}_{i+1})$ is the distance of flight plan between waypoint wp_i and waypoint wp_{i+1} (see Figure 2), and ψ is the maximal available rotation (heading change) angle. The VFS translates into a delay of $t_{\text{additional}} = \sum_i \frac{l_i}{v}\left(\frac{1}{\cos\psi} - 1\right)$, where v is the speed of the aircraft.

The second way for ATC to delay flights with respect to their current schedule is to prescribe *Holding Patterns* (HP). The LOCKE 1 arrival shown in Figure 1 has three possible holding patterns in the airspace, respectively at INYOE, MODESTO and CEDES. These holding patterns generate a prescribed delay for an aircraft (see Figure 3). When such an option is available to a given flight plan, our model will take it into account by adding $t_{\text{additional}} = pT_{\text{hp}}$ to any feasible arrival time at the destination airport, where $p \in \mathbb{N}$ is the number of loops done by the aircraft before recovering its original course. Note that we model the HP for the sake of completeness. However, it should be clear that one of the goals of this study is to alleviate the use of them, since they are in general the least preferred option chosen by the ATC.

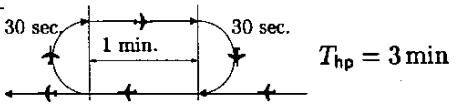

Figure 3: *Holding Pattern* (HP). The prescribed "time to lose" is usually given to the aircraft in minutes: for example one minute in each straight portion and 30 seconds in each half circle.

2.2 Aircraft dynamics

In previous work [1], we have defined and performed a validation of a model for aircraft in the NAS. Our model is based on hybrid automata [18], in which we assume that an aircraft is well modeled using a finite set of modes Q_i with a dynamical system $\dot{x}_i = f_{q_i^k}(x_i)$ associated to each mode. Figure 4 represents the model corresponding to the arrival shown in Figure 1 for a single aircraft; it is defined as a *hybrid automaton* $H_i = (Q_i, \mathbb{X}_i, \Sigma_i, \text{Init}_i, f_{q_i}, \text{Dom}_i, R_i)$.

1 $Q_i \cup \mathbb{X}_i = \{$ OAL_{slow}, MVA_{slow}, CZQ_{slow}, FMG_{slow}, OAL_{fast}, MVA_{fast}, FMG_{fast}, CZQ_{fast}, HP_{fast}, VFS_{fast}, HP_{slow}, VFS_{slow} $\} \cup \mathbb{R}^2$. The first eight modes correspond to the dynamics of the aircraft incoming into the arrival from any entry waypoint (acronym), at a speed (fast or slow). The other modes are HPs and VFSs, respectively, at fast and slow speed. $\mathbb{X}_i = \mathbb{R}^2$ where $x_i \in \mathbb{X}_i$ represents the lateral position of the aircraft. Here, we solve a routing scheduling problem at fixed altitude, as in [1, 8, 14, 5] (assuming that it is possible to separate aircraft vertically if they conflict horizontally).

2 Σ_i, the set of discrete inputs, which indexes the switches from one mode to another. Figure 4 shows the two different types of elements in Σ_i. The first set (σ_{c_p}, where p is an integer) is generated by ATC: for example "slow down on MOD.LOCKE1 from MVA" ($\sigma_{c_{15}}$). The second set (σ_{a_p}, where p is an integer) is generated by the airspace: for example at the end of a slow HP on MOD.LOCKE1 from CZQ, retrieve original course (σ_{a_9}).

3 $\text{Init}_i \subseteq Q_i \times \mathbb{X}_i$: each aircraft is initially in a given mode $q_i \in Q_i$ at a given location $x_i \in \mathbb{X}_i$.

4 $f_{q_i} : Q_i \times \mathbb{X}_i \to T\mathbb{X}_i$ is a vector field $f_\cdot(\cdot)$, where $q_i \in Q_i$ represents the current mode. Note that q_i is denoted as a subscript for brevity (for a given $q_i \in Q_i$ and $x_i \in \mathbb{X}_i$, we write $\dot{x}_i = f_{q_i}(x_i)$). For $q_i \in \{$ HP_{fast}, VFS_{fast}, HP_{slow}, VFS_{slow} $\}$, the dynamics $f_{q_i}(x_i)$ can be derived very easily from Figures 3 and 2. For the other modes, they are a set of successive speed vectors.

Proceedings of the American Control Conference
Denver, Colorado June 4-6, 2003

For example, for FMG_{fast}, (see Figure 1):

$f_{q_i}(x_i) =$

$$\begin{cases} v_1 & \text{if } \mathbf{A}_1 x_i \le b_1 \quad \text{aircraft above the line OAL-GROAN} \\ v_2 & \text{if } \mathbf{A}_2 x_i \le b_2 \quad \text{aircraft east of the line GROAN-UPEND} \\ v_3 & \text{if } \mathbf{A}_3 x_i \le b_3 \quad \text{aircraft east of the line UPEND-SFO(D29)} \\ v_4 & \text{if } \mathbf{A}_4 x_i \le b_4 \quad \text{descent into OAK (not shown in Figure 1)} \end{cases}$$

The matrices $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, and $\mathbf{A}_4$ and the corresponding vectors b_1, b_2, b_3, b_4 encode the geometrical constraints of the problem: $v_i \in \mathbb{R}^2$ have all the same magnitude (corresponding to the fast mode), but point towards different destination waypoints.

5 $\text{Dom}_i \subseteq Q_i \times \mathbb{X}_i \times \Sigma_i$ defines for each discrete mode $q_i \in Q_i$ the subset of $\mathbb{X}_i \times \Sigma_i$ for which continuous evolution is allowed. Dom can be expressed as $\text{VFS}_{\text{fast}} \times P_{\text{VFS}_{\text{fast}}} \cup \text{HP}_{\text{fast}} \times P_{\text{HP}_{\text{fast}}}$, etc. , where $P_{\text{VFS}_{\text{fast}}}$ and $P_{\text{HP}_{\text{fast}}}$, etc. are subsets of $\mathbb{R}^2$ where these maneuvers are allowed (see for example the allowed VFS zone in Figure 2 for the VFS, and the three HP available in LOCKE 1 in Figure 1 for the HP).

6 $R_i : Q_i \times \mathbb{X}_i \times \Sigma_i \to 2^{Q_i \times \mathbb{X}_i}$ is a reset relation which maps a state to the sets of its possible successors. For example, if the aircraft has to be put on hold by ATC from its current flight plan at location x_i at low speed arriving from OAL, $(\text{HP}_{\text{slow}}, x_i) \in R_i(\text{OAL}_{\text{slow}}, x_i, \sigma_c)$, where σ_c is the corresponding discrete controller action.

If one ignores for a moment the continuous part of the problem and views H_i as a finite state machine, the grammar [12, 18] accepted by H_i is easy to compute. Experimental and statistical studies of a similar grammar have been realized for another airspace in [10]. In the present work, we are interested in synthesizing a set of "timed sentences" accepted by this grammar, which represent the ATC commands delivered to the aircraft (i.e. which we can use to generate a hybrid automaton execution). For this, we can use the definition of *hybrid time trajectory* $\tau(i)$ of aircraft i (see for example [18]) $\tau(i) = \{[\tau_l, \tau_l']\}_{l=0}^{L_i}$ (where $L_i \in \mathbb{N}$) which times the execution of H_i. The intervals $[\tau_l, \tau_l']$ represent occupancy time of a mode. The mode switches are given by $(q_i(\tau_{l+1}), x_i(\tau_{l+1})) \in R_i(q_i(\tau_l'), x_i(\tau_l'), \sigma(\tau_l'))$. The timed grammar that we are interested in synthesizing is therefore of the following form for each aircraft: $([\tau_0, \tau_0'], \sigma_0, [\tau_1, \tau_1'], \sigma_1, [\tau_2, \tau_2'], \sigma_2, \cdots)$, where in addition to the σ_p (and modes) of a usual grammar, we have the corresponding time intervals $[\tau_p, \tau_p']$. Note that equation (1) directly follows from this grammar.

3 Hybrid controller synthesis

We can now restate in mathematical terms the problem described in the introduction, and present a resolution algorithm for the problem above, for a set of aircraft (indexed by i), given a set of possible arrivals (indexed by j).

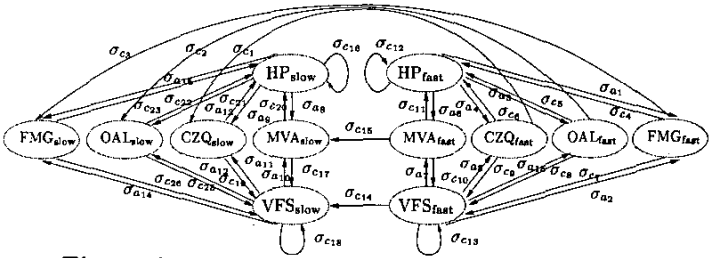

Figure 4: Hybrid automaton for each aircraft for the arrival of Figure 1. The transition between modes is given either by airspace use (σ_{a_p} switches, $p \in \mathbb{N}$) and ATC commands (σ_{c_p} switches $p \in \mathbb{N}$, which we are interested in synthesizing). The duplication of modes fast/slow is done to ensure that aircraft only decelerate through the execution of the hybrid automaton, as for a normal arrival. This model is a subset of our model [1], for this specific scenario of arrivals.

Generate a procedure which takes N airborne aircraft (with the following information: current position x_i, requested entrance waypoint into the arrival airspace, range of achievable speeds), and provides:
(1) *An optimal schedule with arrival assignment for each aircraft; where optimal means either the result of minimizing the sum $\sum_{i=1}^{N} t_i$ of arrival times $\{t_i\}_{i\in\{1,\cdots,N\}}$ of all aircraft, or the result of maximizing the minimal separation $\min_{i<j,(i,j)\in\{1,\cdots,N\}^2} |t_i - t_j|$ of two successive aircraft at the destination airport without putting any aircraft on hold.*
(2) *The set of mode switches $\{\sigma_k\}_{k\in\{1,\cdots,K_i\}}$ to achieve this schedule with corresponding hybrid time trajectory for each aircraft i: $([\tau_0, \tau_0'], \sigma_0, [\tau_1, \tau_1'], \sigma_1, [\tau_2, \tau_2'], \sigma_2, \cdots)$.*

Scheduling / routing resolution algorithm

```
    for all aircraft i=1:N
1     wp_i := entrance wp into airspace
      for all arrivals j starting from wp_i
2       Compute shortest distance s_ij to destination
3       Compute buffered distance b_ij to destination
4       p_ij=number of HP available for aircraft i in arrival j
5       Feas_ij := ⋃_{k=1}^{p_ij} [s_ij/(max speed i) + kT_hp, b_ij/(min speed i) + kT_hp]
        (feasible arrival times for aircraft i in arrival j)
      end
6     S_i = ⋃_j Feas_ij (set of possible arrival times of aircraft i)
7     Rewrite S_i as disjoint union ⋃_{k=1}^{n_i} [a_k^i, b_k^i], s.t. b_k^i < a_{k+1}^i
8     if MILP(ΔT,{S_i}) is not feasible
        Return not feasible
      else
9       For every t^i, identify arrival route and number of HP
10      Compute switching time sequence {τ_1^i, ··· ,τ_N^i, t^i}
      end
    end
```

Step 1 At time t, get the requested entrance waypoint wp_i of aircraft i into the arrival area, as well as predicted arrival time $t_i^{\text{predicted}}$ at this way point.

Step 2 For arrival j, compute the shortest distance s_{ij} from wp_i to the airport: $s_{ij} = \sum d(\text{wp}_k, \text{wp}_{k+1})$ for all waypoints wp_k between the entrance waypoint wp_i and

the destination airport in arrival j.

Step 3 For the arrival j, compute $b_{ij} = \sum_{\text{buffered}} d(\mathsf{wp}_k, \mathsf{wp}_{k+1})\,(1/\cos\psi - 1)$, the buffered distance from wp_i to the airport. The sum ranges over the set of segments for which it is possible to do VFS.

Step 4 p_{ij} is the number of HP allowed by ATC for aircraft i in arrival j. It could thus range from zero (HP locations are totally saturated) to a large number, limited by the fuel autonomy. One of the applications of this study is to compute the best available spacing of the aircraft without HP, in order to alleviate their use.

Step 5 $\mathsf{Feas}_{ij} = \bigcup_{k=1}^{p_{ij}} [\frac{s_{ij}}{\text{max speed}i} + kT_{\mathsf{hp}}, \frac{b_{ij}}{\text{min speed}i} + kT_{\mathsf{hp}}]$ is the set of possible arrival times for aircraft i using arrival j with the allowed number of HP in that arrival.

Step 6 $S_i = \bigcup_j \mathsf{Feas}_{ij}$ represents the set of all possible arrival times for aircraft i using any of the available arrival routes. It is generally not convex.

Step 7 The set $S_i \subset \mathbb{R}$ is a union of intervals, possibly overlapping. This set can be rewritten as $S_i := \bigcup_{k=1}^{n_i} [a_k^i, b_k^i]$, where $b_k^i < a_{k+1}^i$, eliminating any redundancy in the expression of the constraints.

Step 8 We want to minimize the sum of arrival of all aircraft, such that the aircraft arrive in their achievable time intervals and are separated by at least ΔT:

$$\begin{array}{lll} \textbf{min:} & \sum_{i=1}^N t_i & \\ \textbf{s.t.:} & t_i \in S_i = \bigcup_{k=1}^{n_i} [a_k^i, b_k^i] & 1 \le i \le N \\ & |t_i - t_j| \ge \Delta T & 1 \le i,j \le N,\ i > j \end{array} \tag{3}$$

We pick $D \ge \max_{i=1}^N \max\{a_{n_i}^i - a_1^i, b_{n_i}^i - b_1^i\}$, and $C \ge 2(\max_{j=1}^N b_{n_j}^j - \min_{i=1}^N a_1^i)$, and introduce decision variables c_{ij} and d_{ij}. We rewrite the non-convex problem (3) as a MILP, which we denote MILP($\Delta T, \{S_i\}$):

$$\begin{array}{lll} \textbf{min:} & \sum_{i=1}^N t_i & \\ \textbf{s.t.:} & t_i \ge a_1^i & 1 \le i \le N \\ & t_i \le b_{n_i}^i & 1 \le i \le N \\ & t_i \ge a_{k+1}^i - D d_{ik} & 1 \le i \le N,\ 1 \le k \le n_i - 1 \\ & t_i \le b_k^i + D(1 - d_{ik}) & 1 \le i \le N,\ 1 \le k \le n_i - 1 \\ & d_{ik} \in \{0,1\} & 1 \le i \le N,\ 1 \le k \le n_i - 1 \\ & t_i - t_j \ge \Delta T - C c_{ij} & 1 \le i,j \le N,\ i > j \\ & t_i - t_j \le C(1 - c_{ij}) - \Delta T & 1 \le i,j \le N,\ i > j \\ & c_{ij} \in \{0,1\} & 1 \le i,j \le N,\ i > j \end{array} \tag{4}$$

The handiness of MILP formulations for non-convex optimization problems such as (3) has been extensively explored and used successfully in [16, 15]. In this work, we use the same formulation to express non-convex constraints of two types. The first type is "time collision avoidance": $|t_i - t_j| \ge \Delta T$, which enables metering of the flow. The second type is non-convex feasibility constraints $t_i \in \bigcup_{k=1}^{n_i} [a_k^i, b_k^i]$, which encapsulates the airspace structure and the aircraft capacities.

Step 9 For every t_i, $i \in \{1, \cdots, N\}$, $\exists! \nu \in \{1, \cdots, n_i\}$, such that $t_i \in [a_\nu^i, b_\nu^i]$. For this $[a_\nu^i, b_\nu^i]$, we can reconstruct at least one $(\mathsf{wp}_1, \cdots, \mathsf{wp}_{n(\nu, a_\nu^i, b_\nu^i)})$ (choice of arrival and waypoint sequence) and retrieve the number p_{ij} of HP which achieves this t_i. The arrival j, and the number p_{ij} of holding patterns can be retrieved from steps 2-3-4, by labeling the feasible arrival time intervals (i.e. by storing for each arrival interval portion, the arrival routes which achieve it, as well as the number of HP and VFS). Sometimes, more than one solution is available (if for example the achievable arrival times intervals using two different arrivals overlap).

Step 10 Construction of the timed grammar $([\tau_0, \tau_0'], \sigma_0, [\tau_1, \tau_1'], \sigma_1, [\tau_2, \tau_2'], \sigma_2, \cdots)$. We restrict, $k \in \mathbb{N}$, $\tau_k' - \tau_k > 0$ (the system has to stay in each mode for a nonzero amount of time).We thus have $\tau_k' = \tau_{k+1}$ for all relevant indices k. Therefore, we only need to compute the τ_k. There are two cases:

Case 1: If $t_i - t_i^{\text{predicted}} - p_{ij}T_{\mathsf{hp}} \in [s_{ij}/\overline{v}_i, s_{ij}/\underline{v}_i]$, there is no need for use of VFS. The inclusion above means that a simple solution where aircraft i switches from upper speed $\overline{v}_i$ to lower speed $\underline{v}_i$ between wp_1 and $\mathsf{wp}_{n(\nu,a_\nu^i,b_\nu^i)}$ works. The aircraft trajectory should include a portion at upper speed of duration $T_{\text{fast}} = [s_{ij} - \underline{v}_i(t_i - t_i^{\text{predicted}} - p_{ij}T_{\mathsf{hp}})]/(\overline{v}_i - \underline{v}_i)$, eventually interrupted by k of the p_{ij} holding patterns (for a cumulated duration of kT_{hp}), and a portion at lower speed of duration $T_{\text{slow}} = [-s_{ij} + \overline{v}_i(t_i - t_i^{\text{predicted}} - p_{ij}T_{\mathsf{hp}})]/(\overline{v}_i - \underline{v}_i)$, eventually interrupted by the rest of the holding patterns (for a cumulated duration of $(p_{ij} - k)T_{\mathsf{hp}}$). Note again that the current speed does not matter for the holding patterns, since they are prescribed in "time to lose" (losing 3 minutes at $\underline{v}_i$ is the same as losing 3 minutes at $\overline{v}_i$). The hybrid trajectory of aircraft i is then computed according to the diagram in Figure 5: the interval $[t_i^{\text{predicted}}, t_i]$ is divided in two: $[t_i^{\text{predicted}}, t_i^{\text{slow down}}]$ and $[t_i^{\text{slow down}}, t_i]$, such that $t_i^{\text{slow down}} - t_i^{\text{predicted}} = T_{\text{fast}} + kT_{\mathsf{hp}}$ and $t_i - t_i^{\text{slow down}} = T_{\text{slow}} + (p_{ij} - k)T_{\mathsf{hp}}$. Here, $k \le p_{ij}$ is the number of high speed HPs, and $p_{ij} - k$ is the number of low speed HPs.

Case 2: If $t_i - t_i^{\text{predicted}} - p_{ij}T_{\mathsf{hp}} \in [s_{ij}/\underline{v}_i, b_{ij}/\underline{v}_i]$, flying at minimal speed $\underline{v}_i$ for the complete arrival is not sufficient. The aircraft has to switch from upper speed $\overline{v}_i$ to lower speed $\underline{v}_i$ upon entrance in the arrival area, and will then have to use VFS for a portion of the path, given by: $T_{\text{VFS slow}} = t_i - t_i^{\text{predicted}}/(b_{ij} - s_{ij})[-s_{ij} + \underline{v}_i(t_i - t_i^{\text{predicted}} - p_{ij}T_{\mathsf{hp}})]$, and be at low speed for the remaining portion $T_{\text{slow}} = t_i - t_i^{\text{predicted}}/(b_{ij} - s_{ij})[b_{ij} - \underline{v}_i(t_i - t_i^{\text{predicted}} - p_{ij}T_{\mathsf{hp}})]$. As in the previous case, both portions can be interrupted by HPs, if $p_{ij} > 0$. The hybrid trajectory can then again be computed using the diagram of Figure 5.

4 Implementation and simulations

We now describe a possible implementation of the algorithm derived in the previous section, and assess its performance through various simulations which enable us to draw conclusions about its limitations.

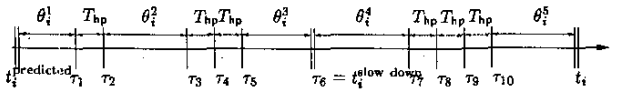

Figure 5: Example of hybrid trajectory $\{t_i^{predicted}, \tau_1, \tau_2, \cdots \tau_9, \tau_{10}, t_i\}$ of the hybrid automaton for aircraft i. τ_1, τ_3, τ_4, τ_7, τ_8, τ_9 are determined by geometry (i.e. the switch from F to HP occurs upon entry on the HP zone: see Figure 1). θ_i^1, θ_i^2, θ_i^3 are determined by the τ_i and the equation $T_{fast} = \theta_i^1 + \theta_i^2 + \theta_i^3 = (\overline{v}_i - \underline{v}_i)[s_{ij} - \underline{v}_i(t_i - t_i^{predicted} - p_{ij}T_{hp})]/(\overline{v}_i - \underline{v}_i)$. Here $p_{ij} = 3$ since there are three HPs. This enables the computation of τ_6, the switching time from upper speed $\overline{v}_i$ to lower speed $\underline{v}_i$. For τ_i, $i \geq 7$, the same construction applies. For the case of VFS, the construction is identical, except that the $\underline{v}_i$ portions and the VFS portions might alternate depending on spatial availability of VFS along certain routes.

4.1 Implementation and computational time

The interface of the code is written in MATLAB, which reads ETMS [2] data and translates it into AMPL code [9] (steps 1-7). The MILP is coded and solved in CPLEX (step 8), interfaced by the modeling language AMPL. The result is read in MATLAB and transformed into ATC commands and ETMS format data (steps 9-10), readable directly by FACET [7]. MATLAB is run under UNIX on a SOLARIS 8 workstation (1GB of RAM), used for the CPU computations presented here. This implementation works in real time for a small number of aircraft in most of the realistic cases (flow almost metered). It is important to show the limits of this approach as well, i.e. to identify a threshold number of aircraft above which the CPU time required by the method becomes too large to be realizable online. We show that when partial order is not available as a decision heuristic, the computational time can become extremely large: we generate scenarios where the S_i of different aircraft overlap on a significant portion of their length. Figure 8 right shows an example for 10 aircraft. For these scenarios, we solve the following MILP, which is a slightly modified version of (4): instead of giving the best sum of arrival times as (4), it provides the best spacing Δ^* of the aircraft:

$$
\begin{array}{lll}
\textbf{max:} & \Delta & \\
\textbf{s.t.:} & t_i \geq a_1^i & 1 \leq i \leq N \\
& t_i \leq b_{n_i}^i & 1 \leq i \leq N \\
& t_i \geq a_{k+1}^i - Dd_{ik} & 1 \leq i \leq N,\ 1 \leq k \leq n_i - 1 \\
& t_i \leq b_k^i + D(1 - d_{ik}) & 1 \leq i \leq N,\ 1 \leq k \leq n_i - 1 \\
& d_{ik} \in \{0, 1\} & 1 \leq i \leq N,\ 1 \leq k \leq n_i - 1 \\
& t_i - t_j \geq \Delta - Cc_{ij} & 1 \leq i, j \leq N,\ i > j \\
& t_i - t_j \leq C(1 - c_{ij}) - \Delta & 1 \leq i, j \leq N,\ i > j \\
& c_{ij} \in \{0, 1\} & 1 \leq i, j \leq N,\ i > j
\end{array}
\tag{5}
$$

We compute the CPU time required by the algorithm.

[2] The *Enhanced Traffic Management System* database contains all flight plan information for flights in the NAS. Data are collected from the entire population of flights in the NAS with filed flight plans. ETMS data is sent from the Volpe National Transportation System Center to registered participants via the *Aircraft Situation Display to Industry* electronic file server.

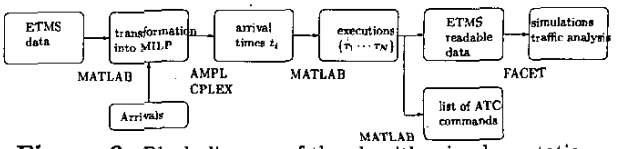

Figure 6: Block diagram of the algorithm implementation.

The relevant part to time is the CPLEX computation: we want to quantify the dependence of the MILP computation time on the number N of aircraft. Figure 8 shows the result of 420 simulations (30 for each N, N ranging from 3 to 16), with initial conditions given and perturbed by noise. The user CPU time grows exponentially with the number of aircraft, making $N = 20$ aircraft impossible to schedule online.

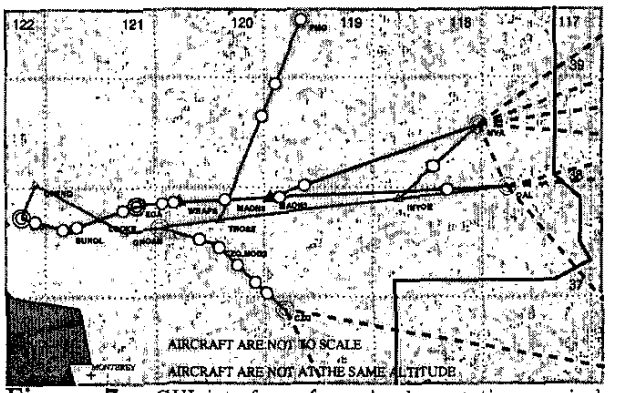

Figure 7: GUI interface of our implementation: arrivals LOCKE1 and MADWIN3 to the Oakland airport.

4.2 Merging flows robustness

We show a possible use of this algorithm to characterize flow stability of merging traffic. A common phenomenon in congested areas is delay of short flights because of incoming long flights. For example, in the Oakland Center, in the presence of incoming long flights (Europe, Asia), ATC tend to delay short flights (L.A., Seattle, Vancouver) if the capacity of the arrival airports is limited: the long flights are already airborne and might not have as much freedom in delays (for fuel reasons). We would like to show that our algorithm enables quantification of the influence of additional traffic into merging traffic: *given a sequence of m incoming aircraft into a single airport, by how much does an additional set of n aircraft reduce the available spacing of the original sequence ?* Thus, we want to compute the Δ^* resulting from an additional n aircraft when solving (5). We generate the following scenario (Figure 9 left): m aircraft are scheduled to come into Oakland through the two arrival LOCKE 1 and MADWIN 3, entering through the waypoints OAL, MVA and FMG. An additional n aircraft are fed into LOCKE 1 through CZQ, with very little available maneuvering freedom (short single intervals $[a_1^i, b_1^i]$). Figure 8 shows a numerical example of achievable schedule for this scenario, with $m = n = 10$. Figure 9 (right) shows the averaged results of 300 simulations of perturbed merging traffic.

We perturb $m = 10$ aircraft by n aircraft, where n ranges from 1 to 10. We compute the maximal available spacing Δ^* for the set of $m + n$ aircraft. We see that the additional 10 aircraft reduce the spacing by almost three. In this case, we thus see that HPs will be necessary, since in general, one aircraft a minute is too high a frequency for a single track airport.

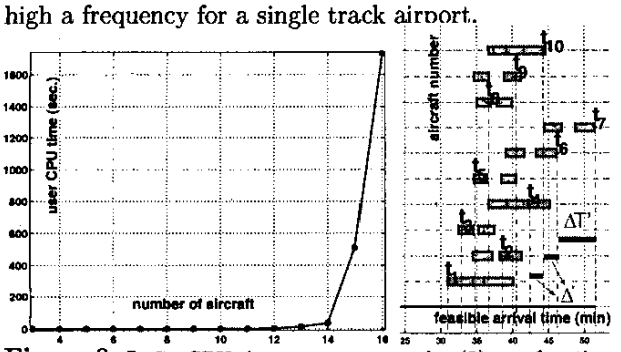

Figure 8: Left: CPU time necessary to solve (5) as a function of N. Each point is the average of 30 runs (identical initial conditions perturbed enough to swap aircraft order). Clearly 20 aircraft becomes impossible to manage in real time. **Right:** Example of solution of (5) for 10 aircraft, with two arrivals to the Oakland airport (LOCKE1 and MADWIN3, without HPs). For aircraft i, the two horizontal segments represent the feasible arrival times: $[a_1^i, b_1^i]$ and $[a_2^i, b_2^i]$. In some cases, they overlap ($i = 1, 4, 10$) or have empty intersection ($i = 2, 3, \cdots$).

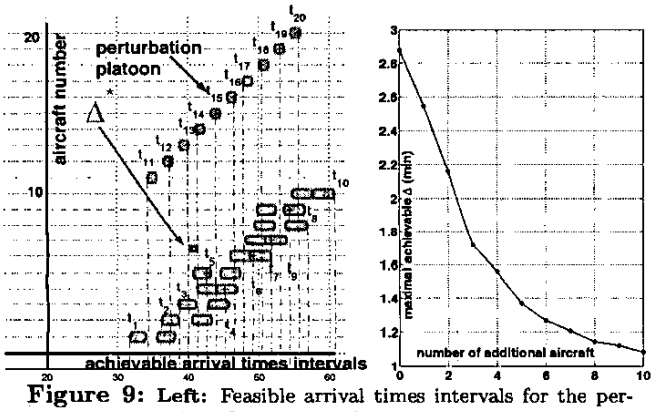

Figure 9: Left: Feasible arrival times intervals for the perturbed flow. Aircraft 1 to 10 are incoming into LOCKE 1 and MADWIN 3 through OAL, MVA and FMG. Aircraft 11 to k (where $k \in \{11, \cdots, 20\}$ are incoming into LOCKE 1 through CZQ. their maneuvering availability is very small (short intervals) The Δ^* for this run is shown as well. **Right:** Variation of Δ^* with k. As expected, the more the flow is perturbed (large k), the smaller Δ^* becomes.

5 Conclusion and current work

We have shown an algorithm capable of generating a set of ATC commands to achieve scheduling of merging traffic. Our implementation takes inputs in the form of ETMS data and outputs ATC commands as well as the corresponding prescribed flight plans. The computational time bottleneck of the algorithm is the solution of the MILP. We have shown that a CPLEX implementation cannot guarantee an upper bound on the computational time. Our current research is focused on exact or approximation algorithms for solving this problem. For $n_i = 1$, we derived a polynomial-time algorithm [3] to solve the problem exactly. We showed numerical evidence of our guaranteed upper bound on running time: it can solve (5) for 100 aircraft in a few seconds. We believe that the general case ($n_i > 1$) is NP-complete, but were not able to prove it. When S_i is periodic in space, we derived a 5-approximation algorithm [2] to solve (4) in polynomial time.

Acknowledgments: We are grateful to Tom Schouwenaars for his help on CPLEX, to Shon Grabbe for his help on ETMS data and Francis Carr for his suggestions and interest in this work.

References

[1] A. M. BAYEN, P. GRIEDER, and C. J. TOMLIN. A control theoretic predictive model for sector-based air traffic flow. In *AIAA Conf. on GNC.*, Monterey, CA, August 2002.

[2] A. M. BAYEN, J. ZHANG, C. J. TOMLIN, and Y. YE. A 5 approximation algorithm for periodic aircraft scheduling. In preparation for the *2003 European Symposium on Algorithms.*

[3] A. M. BAYEN, J. ZHANG, C. J. TOMLIN, and Y. YE. Milp formulation and polynomial time algorithm for an aircraft scheduling problem. Submitted to the *2003 CDC.*

[4] A. BEMPORAD, F. BORRELLI, and M. MORARI. Piecewise linear optimal controllers for hybrid systems. In *2000 ACC*, pages 1190–1194, Chicago, IL, June 2000.

[5] D. BERTSIMAS and S. STOCK. The air traffic flow management problem with enroute capacities. *Operations Research*, 46(3):406–422, 1997.

[6] D. BERTSIMAS and J. N. TSITSIKLIS. *Introduction to linear optimization.* Athena Scientific, Belmont, MA, 1997.

[7] K. BILIMORIA, B. SRIDHAR, G. CHATTERJI, K. SETH, and S.GRABBE. FACET: Future ATM concepts evaluation tool. In *3rd USA/Europe ATM R&D Seminar*, Naples, Italy, June 2001.

[8] D. DUGAIL, E. FERON, and K. BILIMORIA. Conflict-free conformance to en route flow-rate constraints. In *AIAA Conf. on GNC.*, Monterey, CA, August 2002.

[9] R. FOURER, D. M. GAY, and B. W. KERNIGHAN. *AMPL: a modeling language for mathematical programming.* Boyd and Fraser, Danvers, MA, 1999.

[10] J. HISTON and R. J. HANSMAN. The impact of structure on cognitive complexity in air traffic control. Technical report, MIT, June 2002.

[11] V. JAIN and I. E. GROSSMANN. Algorithms for hybrid MILP / CP models for a class of optimization problems. *INFORMS Journal on Computing*, 13:258–276, 2001.

[12] H. LEWIS and C. PAPADIMITRIOU. *Elements of the theory of computation.* Prentice Hall, Upper Saddle River, NJ, 1982.

[13] P. MISHRA and G. PAPPAS. Flying hot potatoes. In *2002 ACC*, pages 754–759, Anchorage, AK, May 2002.

[14] C. PANAYIOTOU and C. CASSANDRAS. A sample path approach for solving the ground holding policy problem in air traffic control. *IEEE Trans. on Control Sys. Tech.*, 9(3):510–523, 2001.

[15] A. RICHARDS, J. BELLINGHAM, M. TILLERSON, and J. HOW. Co-ordination and control of multiple UAVs. In *AIAA Conf. on GNC.*, Monterey, CA, August 2002.

[16] A. RICHARDS, T. SCHOUWENAARS, J. HOW, and E. FERON. Spacecraft trajectory planning with collision and plume avoidance using mixed-integer linear programming. *AIAA Journal of Guidance, Control and Dynamics*, 25(4):755–764, 2002.

[17] JEPPESEN SANDERSON, INC, http://www.jeppesen.com.

[18] C. J. TOMLIN, J. LYGEROS, and S. SASTRY. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970, 2000.

[19] H. P. WILLIAMS and S. C. BRAILSFORD. Computational logic and integer programming. In J. Beasley, editor, *Advances in Linear and Integer Programming*, pages 249–281. Oxford University Press, 1996.