Conflict-Aware Flight Planning for Avoiding Near Mid-Air Collisions

Saswata Paul, Stacy Patterson, and Carlos A. Varela
Department of Computer Science
Rensselaer Polytechnic Institute, Troy, New York, 12180
pauls4@rpi.edu, sep@cs.rpi.edu, cvarela@cs.rpi.edu

***Abstract*—We present a novel conflict-aware flight planning approach that avoids the possibility of near mid-air collisions (NMACs) in the flight planning stage. Our algorithm computes a valid flight-plan for an aircraft (ownship) based on a starting time, a set of discrete way-points in 3D space, discrete values of ground speed, and a set of available flight-plans for traffic aircraft. A valid solution is one that avoids loss of standard separation with available traffic flight-plans. Solutions are restricted to permutations of constant ground speed and constant vertical speed for the ownship between consecutive way-points. Since the course between two consecutive way-points is not changed, this strategy can be used in situations where vertical or lateral constraints due to terrain or weather may restrict deviations from the original flight-plan. This makes our approach particularly suitable for unmanned aerial systems (UAS) integration into urban air traffic management airspace. Our approach has been formally verified using the Athena proof assistant. Our work, therefore, complements the state-of-the-art pairwise tactical conflict resolution approaches by enabling an ownship to generate strategic flight-plans that ensure standard separation with multiple traffic aircraft, while conforming to possible restrictions on deviation from its flight path.**

I. Introduction

Loss of standard separation between aircraft can have hazardous consequences such as mid-air collisions and wake-vortex induced rolls. According to Air Route Traffic Control Center (ARTCC) reports, losses of standard separation rose to 10.6% in 2017 [1]. With the integration of unmanned aircraft systems (UAS) for civilian applications, the density of aircraft in the National Airspace System (NAS) is expected to increase significantly in the near future. This will make it necessary to implement smarter air traffic management (ATM) systems that can ensure required separation between airborne aircraft in the NAS. Under such circumstances, the concept of *free-flight*, which involves a system of dynamic, automated, and distributed air traffic control, may become popular. It will be imperative for pilots and UAS computers to have the ability to independently generate flight-plans that avoid loss of standard separation with other aircraft.

Tactical collision avoidance systems like the Traffic Collision Avoidance System (TCAS) can advise pilots to resolve pairwise conflicts where tactical vertical resolutions are possible [2]. However, TCAS cannot be used for conflict avoidance in scenarios that present vertical constraints on maneuvers, for example, busy reduced vertical separation minima (RVSM) airspaces and terminal areas where aircraft are already flying too close to the ground to perform downward resolutions. Therefore, there is a need for strategic conflict management such as traffic flow management (TFM), which deals with capacity handling, air-traffic flow management and flexible use of the NAS [3]. With the introduction of Automatic Dependent Surveillance-Broadcast (ADS-B) in the NAS, pilots and flight computers will be equipped with better information about traffic aircraft. The data made available by ADS-B will give way to new techniques for efficiently ensuring minimum standard separation with traffic aircraft.

In this paper, we present a formally verified approach for strategic conflict-aware flight planning. We assume that with the advancement of ADS-B technology, aircraft will be able to transmit and receive multi-segment flight-plans. A flight-plan is a collection of way-points in three-dimensional space that an aircraft is expected to follow. Each way-point has an associated configuration that expresses the aircraft's position, velocity, and the expected course of the aircraft's flight to the next way-point. Conflict-detection is based on the assumption that aircraft follow constant-velocity, straight-line segments between consecutive way-points and a set of flight-plans for traffic aircraft is available. Given a set of flight-plans for traffic aircraft, a set of 3D way-points, a set of discrete values of ground speed, and a start time, our algorithm generates, when feasible, a flight-plan for the ownship that will allow the ownship to always maintain standard separation with all traffic aircraft. This is done by assigning a suitable value of ground speed to each straight-line segment between consecutive way-points in the ownship's flight-plan.

Since the 3D heading of the velocity vector between way-points is not changed, the only changes observed in the solution flight-plan are the times of arrival for the way-points. Therefore, our approach can be advantageous in situations where spatial deviation from the original way-points is restricted by lateral and/or vertical constraints. These restrictions may be imposed by terrain, restricted airspace, weather, or airspace capacity. This is usually true for terminal areas where aircraft are appointed standard terminal arrival routes (STAR) just before the final approach. Civilian applications of UAS, such as package delivery and urban transportation, are also important use cases for this approach since urban environments usually provide little room for lateral and vertical maneuvers. Our technique is therefore highly suitable for UAS integration into the urban air-traffic management airspace.

A flight-plan generated for an ownship by our approach can only guarantee that the ownship will maintain standard

separation with the given set of traffic flight-plans. Possible conflicts between flight-plans in the given set of traffic flight-plans cannot be resolved using our approach. Therefore, a practical implementation would require the presence of a centralized or decentralized controller for a designated airspace (outside the scope of this paper). The controller may be elected from aircraft that are already inside the airspace using a leader election protocol. The controller would maintain a dynamic set of traffic flight-plans for the airspace and decide if a new flight-plan proposed by an aircraft can be accommodated in the airspace. If the flight-plan cannot be accommodated, the controller would try to compute a solution flight-plan. However, due to the restrictions on the nature of a solution flight-plan in our approach, a solution may not always exist. If no solution flight-plan can be computed, the controller would delay the entry of the new aircraft in the airspace by delaying its take-off or by assigning it a holding pattern outside the airspace (e.g., see Fig. 1). If a solution can be computed or the proposed flight-plan itself can be accommodated in the airspace, the controller would add the flight-plan to the set of traffic flight-plans and wait for new proposals by other aircraft.

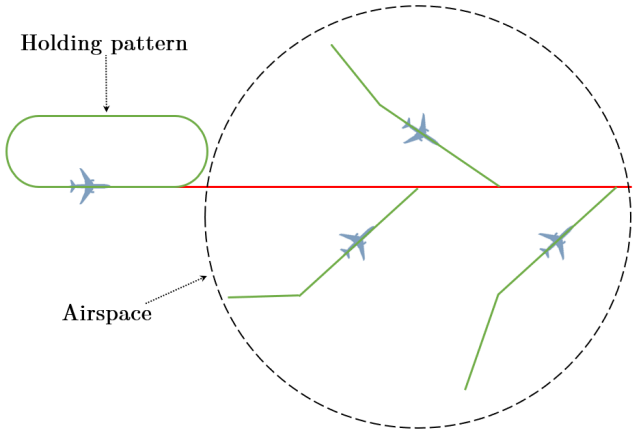

Fig. 1: Top view of a scenario where an aircraft cannot be currently accommodated in an airspace, so it is assigned a holding pattern to delay entry into the airspace (aircraft markers represent the direction of flight-plans and not actual positions).

We have formally verified using the Athena proof language [4]–[6] that our algorithm generates flight-plans for an ownship that guarantee standard separation with a given set of traffic flight-plans at all times. Thus, our work contributes to automated strategic traffic flow management by giving aircraft pilots or UAS flight computers the ability to check if their desired flight-plans are safe and otherwise generate conflict-aware flight-plans if possible.

The rest of the paper is divided as follows: Section II discusses prior work on conflict detection and avoidance techniques and our prior work on aircraft trajectory generation; Section III describes the problem statement of our conflict-aware flight planning approach and our strategy for computing a conflict-free flight-plan; Section IV describes our algorithm for computing a conflict-free flight-plan; Section V formally describes the properties of our approach; Section VI describes the simulations we performed using our approach; and finally Section VII concludes the paper with some future directions of work.

II. Related Work

Pairwise conflict detection and avoidance have been previously investigated. Pritchett *et al.* [7] have presented a decentralized algorithm for aircraft conflict resolution that is based on negotiated bargaining. The resolutions proposed by them are multi-dimensional, i.e., an aircraft can either move up, down, right or left, or increase or decrease its velocity. They associate a cost to each type of resolution and find the solution that entails the least cost. They do this by using the game theory concept of bargaining. Since both aircraft maneuver in this approach, the total cost of maneuvers is divided between the ownship and the traffic aircraft, reducing the cost involved for each individual aircraft. Dowek *et al.* [8] have proposed a formally verified pairwise coordinated technique for conflict avoidance in which only one component of the velocity vector is modified. Their solutions are coordinated such that the ownship and the traffic aircraft independently choose different directions for their conflict avoidance maneuvers. This makes sure that if either one or both the aircraft maneuver, then the potential conflict is avoided. Galdino *et al.* [9] have proposed an optimization over of Dowek *et al.*'s work by considering maneuvers in the horizontal plane that include combined modifications of the ground speed and heading of the ownship. In both these approaches, the ownship's course is modified so that it deviates from its original path. Balachandran *et al.* [10] have presented an approach for scheduling multiple unmanned aerial vehicles (UAV) moving towards an intersection in an urban environment. In this approach, UAVs can adjust their time of arrival by modifying their speed and using lateral deviations. They use the RAFT consensus algorithm [11] to synchronize information across all vehicles and a scheduling algorithm to compute a schedule for the UAVs. Alejo *et al.* [12] have proposed a technique for finding collision-free trajectories for autonomous quadrotors or helicopters whose minimum speed can be zero. Their approach involves adding new way-points to a proposed trajectory and changing the speeds of the UAVs. They initially find a non-optimal solution and then use particle swarm optimization [13] to find better solutions. In contrast to the above work, the solutions generated by our approach do not include lateral or vertical deviations in order to resolve conflicts. Therefore, making it suitable in situations where additional way-points cannot be added due to restrictions imposed by terrain, weather, or airspace capacity.

Work has been done by NASA on formally verified sense and avoid systems for integration of UAS in the NAS [14], [15]. Muñoz *et al.* [16] have presented an enhancement of the TCAS conflict detection and resolution system. They provide a formally verified algorithm for predicting the issuance of TCAS resolution advisories so that pilots and UAS flight computers can take preventive actions to avoid issuance of TCAS resolution advisories. Given a look-ahead interval, they detect resolution advisories instead of conflicts.

NASA's DAIDALUS [17] suite of algorithms are aimed at increasing situational awareness of UAS pilots. DAIDALUS contains algorithms for detecting possible violations of well-clear volumes and alerting the pilots. It also includes algorithms for providing maneuvering guidance to successfully avoid conflicts. These algorithms are integral components of NASA's ICAROUS [18] and DANTi [19] systems. Our work improves upon the state of the art *see-and-avoid* tactical conflict resolution systems by extending the existing conflict detection and resolution approaches to detect and resolve conflicts between multi-segment flight-plans.

Our previous work on aircraft trajectory planning includes trajectory planning for fixed-wing aircraft in total or partial loss of thrust emergencies. In [20], we have presented a dynamic data-driven approach for computing trajectories to reachable runways in loss of thrust situations. Our approach takes into consideration dynamic factors like partial power and aircraft surface damage that can not be predicted in advance. We have further extended our work by considering wind in the trajectory generation phase [21], [22]. Our wind-model considers the baseline glide ratio of the accident aircraft and the horizontal wind conditions. This model can be used to generate high fidelity wind-aware trajectories that are feasible in the presence of a steady, horizontal wind.

III. Conflict-Aware Flight Planning

In this section, we present the mathematical foundation of our conflict-aware flight planning approach. We introduce important definitions and terms, the problem statement, and the strategy for computing valid conflict-aware flight-plans. Our geometrical model of the conflict detection problem is based on a flat-Earth assumption with a global frame of reference. We also assume that aircraft are capable of sending and receiving multi-segment flight-plans with configuration information for each way-point.

Type	Specification	Set	Domain
Configuration	$\langle s_x, s_y, s_z, v_g, \lambda, v_z \rangle$	$\mathbb{C}$	$\mathbb{R}^3 \times \mathbb{R} \geq 0 \times [0, 2\pi) \times \mathbb{R}$
State	$\langle s_x, s_y, s_z, v_x, v_y, v_z \rangle$	$\mathbb{S}$	$\mathbb{R}^6$
Flight-plan	$\langle t, \mathbb{C}^n \rangle$	$\mathbb{F}$	$\mathbb{R} \geq 0 \times \mathbb{C}^{n \in \mathbb{N}}$

TABLE I: Semantic domains.

Definitions:

- The *configuration* of an aircraft is a vector $\langle s_x, s_y, s_z, v_g, \lambda, v_z \rangle$ where s_x, s_y, and s_z are the coordinates in the x, y, and z dimensions, v_g is the ground speed, λ is the course or actual direction of motion of the aircraft with respect to the ground, and v_z is the vertical speed. The set $\mathbb{C}$ is the set of all configurations (Table. I).
- The *state* of an aircraft is a vector $\langle s_x, s_y, s_z, v_x, v_y, v_z \rangle$ where s_x, s_y, and s_z are the coordinates and v_x v_y, and v_z are the components of the velocity vector in the x, y, and z dimensions. The set $\mathbb{S}$ is the set of all states (Table. I).
- A *way-point* is a position in 3D space and is represented by $\langle s_x, s_y, s_z \rangle$ where s_x, s_y, and s_z are the coordinates in the x, y, and z dimensions.
- A *segment* is the 3D straight-line flight segment between two consecutive way-points. A segment is represented by its initial and final way-points.
- A *flight-plan* is a pair $\langle t, \mathbb{C}^n \rangle$ containing a start time t and a vector of configurations $\mathbb{C}^n$ that represent n way-points in the flight-plan. Two consecutive way-points in a flight-plan are connected by a segment. In a flight-plan with n way-points, there are always $n-1$ segments. The set $\mathbb{F}$ is the set of all flight-plans (Table. I).

Conflict Between Two Flight-Plans:

Given the states of two aircraft a and b at time t_0 as $S_{a,t_0} = \langle s_{x,a,t_0}, s_{y,a,t_0}, s_{z,a,t_0}, v_{x,a,t_0}, v_{y,a,t_0}, v_{z,a,t_0} \rangle$ and $S_{b,t_0} = \langle s_{x,b,t_0}, s_{y,b,t_0}, s_{z,b,t_0}, v_{x,b,t_0}, v_{y,b,t_0}, v_{z,b,t_0} \rangle$, their 2D horizontal position vectors and 2D horizontal velocity vectors are written as:

$$\mathbf{s}_{xy,a,t_0} = \langle s_{x,a,t_0}, s_{y,a,t_0} \rangle$$
$$\mathbf{s}_{xy,b,t_0} = \langle s_{x,b,t_0}, s_{y,b,t_0} \rangle$$
$$\mathbf{v}_{xy,a,t_0} = \langle v_{x,a,t_0}, v_{y,a,t_0} \rangle$$
$$\mathbf{v}_{xy,b,t_0} = \langle v_{x,b,t_0}, v_{y,b,t_0} \rangle$$

Assuming both aircraft maintain constant velocity flight, the relative position and relative velocity of the two aircraft in the horizontal (xy) and vertical dimensions (z) at time t_0 can now be obtained by the following equations:

$$\mathbf{s}_{xy,t_0} = \mathbf{s}_{xy,a,t_0} - \mathbf{s}_{xy,b,t_0}$$
$$\mathbf{v}_{xy,t_0} = \mathbf{v}_{xy,a,t_0} - \mathbf{v}_{xy,b,t_0}$$
$$s_{z,t_0} = s_{z,a,t_0} - s_{z,b,t_0}$$
$$v_{z,t_0} = v_{z,a,t_0} - v_{z,b,t_0}$$

Their relative horizontal and vertical positions at any time $t \geq t_0$ can be computed as:

$$\mathbf{s}_{xy,t} = \mathbf{s}_{xy,t_0} + (t - t_0)\mathbf{v}_{xy,t_0}$$
$$s_{z,t} = s_{z,t_0} + (t - t_0)v_{z,t_0}$$

We will be using $\mathbf{s}$ and $\mathbf{v}$ to denote $\mathbf{s}_{xy}$ and $\mathbf{v}_{xy}$ in the rest of the paper.

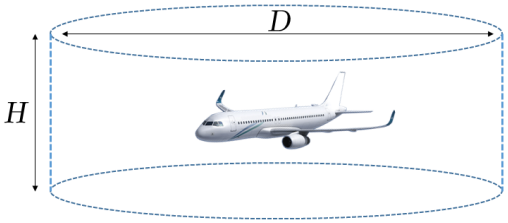

Fig. 2: Well-clear volume around an aircraft.

The well-clear volume of an aircraft is defined as a cylinder of diameter D and height H around the center of the aircraft,

where D is the horizontal separation threshold and H is the vertical separation threshold [8]. Two aircraft a and b are said to be at safe distance from each other if their well-clear volumes do not intersect. They are said to be in *conflict* at time t if their well-clear volumes intersect, i.e., their horizontal distance is less than D and vertical distance is less than H simultaneously at time t. Therefore, if there is a conflict at time t, both of the following equations are satisfied:

$$\|\mathbf{s_t}\| < D \tag{1}$$

$$|s_{z,t}| < H \tag{2}$$

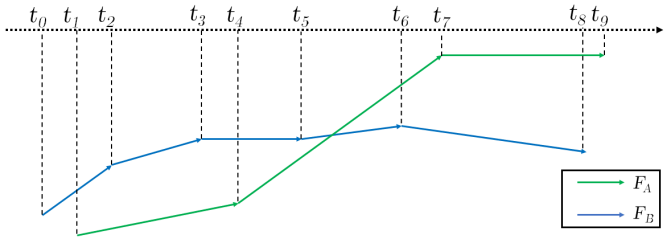

Fig. 3: Periods of interest for two flight-plans.

Conflict detection by using equations (1) and (2) is applicable only when the states of two aircraft a and b at a time t_0 are available as S_{a,t_0} and S_{b,t_0}, along with a known *period of interest* during which both aircraft are expected to maintain their constant-velocity flight. Given two flight-plans F_a and F_b, it is, therefore, necessary to discretize the temporal dimension into discrete periods of interest. This can be done by creating a non-decreasing vector T (which ranges over $\mathbb{R}^n$) of the times corresponding to the way-points comprising the flight-plans. The time between consecutive temporal points in T can then be used as discrete periods of interest for conflict detection. For example, in Figure 3, which shows the top-view of two flight-plans, the times corresponding to the way-points in F_a are t_1, t_4, t_7, and t_9, and the times corresponding to the way-points in F_b are t_0, t_2, t_3, t_5, t_6, and t_8. The vector of times $T = \langle t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9 \rangle$. Each pair of consecutive times in T has been used to discretize the temporal dimension (represented by the dotted line) into periods of interest. It should, however, be noted that periods of interest are only valid if two flight-plans have temporal overlap during that period. For example, in Figure 3, the periods t_0 to t_1 and t_8 to t_9 are not valid periods of interest.

Given two flight-plans F_a and F_b, a horizontal threshold D, and a vertical threshold H the `conflict`(F_a, F_b, D, H) function can check if a possible conflict exists between the two flight-plans. It first uses the function `get-all-times`(F_a, F_b) to generate a non-decreasing vector of times corresponding to the way-points in F_A and F_B as T. `conflict` returns `True` if `T-violation`(T, F_a, F_b, D, H) returns `True`.

```
conflict(F_a, F_b, D, H):
  Let
    T = get-all-times(F_a, F_b)
  in
    if T-violation(T, F_a, F_b, D, H)
      return True
    else
      return False
    endif
```

Id	Input	Output
check-safety	$\{0,1\}^{n\times m} \times \mathbb{F} \times \mathbb{F}^k \times \mathbb{R}^m \times \mathbb{R}^2$	Bool
complete	$\{0,1\}^{n\times m}$	Bool
conflict	$\mathbb{F}^2 \times \mathbb{R}^2$	Bool
exists-violation	$\mathbb{R}^2 \times \mathbb{F}^2 \times \mathbb{R}^2$	Bool
get-all-times	$\mathbb{F}^2$	$\mathbb{R}^n$
get-config	$\mathbb{R} \times \mathbb{F}$	$\mathbb{C}$
get-index	$\mathbb{R}^m \times \mathbb{R}$	$\mathbb{N}$
plan	$\{0,1\}^{n\times m} \times \mathbb{F} \times \mathbb{R}^m$	$\mathbb{F}$
safe	$\mathbb{F}^k \times \mathbb{R}^2$	Bool
set	$\{0,1\}^{n\times m} \times \mathbb{N}^2$	$\{0,1\}^{n\times m}$
solve	$\{0,1\}^{n\times m} \times \mathbb{N} \times \mathbb{F} \times \mathbb{F}^k \times \mathbb{R}^m \times \mathbb{R}^2$	$\mathbb{F}$
T-violation	$\mathbb{R}^n \times \mathbb{F}^2 \times \mathbb{R}^2$	Bool
unassigned	$\{0,1\}^{n\times m}$	$\mathbb{N}$
valid	$\{0,1\}^{n\times m} \times \mathbb{N} \times \mathbb{F} \times \mathbb{F}^k \times \mathbb{R}^m \times \mathbb{R}^3$	Bool

TABLE II: Function signatures.

The function `T-violation`(T, F_a, F_b, D, H) recursively checks for possible conflicts in the period of interest between every pair of consecutive times in T and returns `True` if a conflict is detected.

```
T-violation(T, F_a, F_b, D, H):
  Let
    T = ⟨t_1, t_2, T'⟩
  in
    if T == Null
      return False
    else
      if exists-violation(t_1, t_2, F_a, F_b, D, H)
        return True
      else
        return T-violation(⟨t_2, T'⟩, F_a, F_b, D, H)
      endif
    endif
```

The `exists-violation`$(t_1, t_2, F_a, F_b, D, H)$ function returns `True` if there exists a time $t : t_1 < t \leq t_2$ & $\|\mathbf{s}(t)\| < D$ & $|s_z(t)| < H$. It uses the `get-config`(t, F) function to get the configurations of aircraft a and b at time t_1 as $C_{a,t_1} = \langle s_{x,a,t_1}, s_{y,a,t_1}, s_{z,a,t_1}, v_{g,a,t_1}, \lambda_{a,t_1}, v_{z,a,t_1} \rangle$ and $C_{b,t_1} = \langle s_{x,b,t_1}, s_{y,b,t_1}, s_{z,b,t_1}, v_{g,b,t_1}, \lambda_{b,t_1}, v_{z,b,t_1} \rangle$. Then the x and y components of the velocity vectors of a and b at time t_1 are computed as:

$$v_{x,a,t_1} = v_{g,a,t_1} \times \cos \lambda_{a,t_1}$$

$$v_{y,a,t_1} = v_{g,a,t_1} \times \sin \lambda_{a,t_1}$$

$$v_{x,b,t_1} = v_{g,b,t_1} \times \cos \lambda_{b,t_1}$$

$$v_{y,b,t_1} = v_{g,b,t_1} \times \sin \lambda_{b,t_1}$$

Therefore, their initial states for the period of interest t_1 to t_2, are given by:

$$S_{a,t_1} = \langle s_{x,a,t_1}, s_{y,a,t_1}, s_{z,a,t_1}, v_{x,a,t_1}, v_{y,a,t_1}, v_{z,a,t_1} \rangle$$

$$S_{b,t_1} = \langle s_{x,b,t_1}, s_{y,b,t_1}, s_{z,b,t_1}, v_{x,b,t_1}, v_{y,b,t_1}, v_{z,b,t_1} \rangle$$

Now equations (1) and (2) can be used to determine if a violation of the well-clear volumes of the two aircraft is

possible between t_1 and t_2. `exists-violation` returns `True` if a possible violation is detected.

```
exists-violation(t1, t2, Fa, Fb, D, H):
  Let
    ⟨s_{x,a,t1}, s_{y,a,t1}, s_{z,a,t1}, v_{g,a,t1}, λ_{a,t1}, v_{z,a,t1}⟩
                                        = get-config(t1, Fa)
    ⟨s_{x,b,t1}, s_{y,b,t1}, s_{z,b,t1}, v_{g,b,t1}, λ_{b,t1}, v_{z,b,t1}⟩
                                        = get-config(t1, Fb)
    v_{x,a,t1} = v_{g,a,t1} × cos λ_{a,t1}
    v_{y,a,t1} = v_{g,a,t1} × sin λ_{a,t1}
    v_{x,b,t1} = v_{g,b,t1} × cos λ_{b,t1}
    v_{y,b,t1} = v_{g,b,t1} × sin λ_{b,t1}
    S_{a,t1} = ⟨s_{x,a,t1}, s_{y,a,t1}, s_{z,a,t1}, v_{x,a,t1}, v_{y,a,t1}, v_{z,a,t1}⟩
    S_{b,t1} = ⟨s_{x,b,t1}, s_{y,b,t1}, s_{z,b,t1}, v_{x,b,t1}, v_{y,b,t1}, v_{z,b,t1}⟩
    s_{a,t1} = ⟨s_{x,a,t1}, s_{y,a,t1}⟩
    s_{b,t1} = ⟨s_{x,b,t1}, s_{y,b,t1}⟩
    v_{a,t1} = ⟨v_{x,a,t1}, v_{y,a,t1}⟩
    v_{b,t1} = ⟨v_{x,b,t1}, v_{y,b,t1}⟩
    s_{t1} = s_{a,t1} − s_{b,t1}
    v_{t1} = v_{a,t1} − v_{b,t1}
    s_t = s_{t1} + (t − t1)v_{t1}
    s_{z,t1} = s_{z,a,t1} − s_{z,b,t1}
    v_{z,t1} = v_{z,a,t1} − v_{z,b,t1}
    s_{z,t} = s_{z,t1} + (t − t1)v_{z,t1}
  in
    if ∃ t . ||s_t|| < D & |s_{z,t}| < H & t1 < t ≤ t2
      return True
    else
      return False
    endif
```

The Problem

Given a set of flight-plans Φ (which ranges over $\mathbb{F}^k, k \in \mathbb{N}$), the set is considered to be *safe* if there exists no conflict between any pair of elements in Φ. The `safe`(Φ, D, H) function returns `True` if for all $F_i, F_j \in \Phi$, `conflict`(F_i, F_j, D, H) returns `False`. Thus, `safe`(Φ, D, H), implies that the set Φ is safe.

```
safe(Φ, D, H):
  if ∀Fi, Fj ∈ Φ : ¬conflict(Fi, Fj, D, H)
    return True
  else
    return False
  endif
```

The problem statement can now be expressed as: *Given a set of immutable traffic flight-plans Φ, a horizontal threshold D, a vertical threshold H, and a proposal flight-plan F_a such that* `safe`(Φ, D, H) *but* $\neg$`safe`$(\Phi \cup \{F_a\}, D, H)$*, the goal is to find a valid solution flight-plan $\bar{F}_a$ such that* `safe`$(\Phi \cup \{\bar{F}_a\}, D, H)$.

The Strategy for Computing $\bar{F}_a$:

We have shown how conflicts between two given flight-plans can be detected using the `conflict` function and how the `safe` function can determine if a set of flight-plans is safe. As expressed in the problem statement, the goal is to find a solution flight-plan $\bar{F}_a$ given a safe set of immutable flight-plans Φ and a proposal F_a. Since the flight-plans in Φ are immutable, only the proposal flight-plan F_a may be adjusted to find a solution. Let us consider an aircraft a that proposes F_a to be the ownship and another traffic aircraft b such that $F_b \in \Phi$. For the well-clear volumes of a and b to intersect, there needs to exist a time t such that equations (1) and (2) are simultaneously satisfied. Therefore, our strategy for avoiding conflicts between a and b is to assign suitable values of ground speed (v_g) to the different segments of F_a, such that intersection of the well-clear volumes of a and b can be avoided. Our algorithm takes as input a vector ξ_{air} (which ranges over $\mathbb{R}^n, n \in \mathbb{N}$) of discrete values of airspeed that the ownship can fly with. Given the horizontal wind vector $\mathbf{w}$, a corresponding vector $\xi = \xi_{air}(\mathbf{w})$ (which ranges over $\mathbb{R}^n, n \in \mathbb{N}$) of values of ground speed for the ownship can be computed. The algorithm assigns a value of ground speed $v_g \in \xi$ to each segment in the ownship's flight-plan to create a solution $\bar{F}_a$ such that $\neg$`conflict`$(\bar{F}_a, F_b, D, H)$. The vertical speed for each segment is then adjusted accordingly to ensure that the 3D profile of $\bar{F}_a$ is similar to that of the original flight-plan F_a. We assume that aircraft can change their ground speed and vertical speed instantaneously. This pairwise strategy for conflict avoidance is extended to the set Φ so that a valid $\bar{F}_a$ will avoid conflict with every $F_b \in \Phi$, i.e., $\forall F_b \in \Phi : \neg$`conflict`$(\bar{F}_a, F_b, D, H)$. Our solution space is limited to different permutations of $v_g \in \xi$ in the segments comprising F_a. Therefore, it should be noted that this conservative approach may not always be able to compute a solution flight-plan.

Since the heading of the velocity is not changed in a segment, only two components of the velocity vector are altered – the ground speed and the vertical speed. This restricts the introduction of new way-points in the solution flight-plan. Thus, for a flight-plan $F_a = \langle t, \langle C_1, C_2, ..., C_n \rangle \rangle$, where $C_i = \langle s_x, s_y, s_z, v_g, \lambda, v_z \rangle$, a valid solution $\bar{F}_a = \langle t, \langle C'_1, C'_2, ..., C'_n \rangle \rangle$ satisfies the condition that $C'_i = \langle s_x, s_y, s_z, {v_g}', \lambda, v'_z \rangle$.

There are two steps involved in the computation of a solution flight-plan:

1) Finding a value of ground speed $v_g \in \xi$ for every segment $L \in F_a$.
2) Adjusting the vertical speed to maintain the 3D profile of the flight-plan

In the next section, we will present the algorithm for computing a valid assignment of ground speed for the segments, that has been formally verified to result in a conflict-free flight-plan if a solution is found.

When the ground speed in a flight segment between two consecutive way-points is changed from v_g to ${v_g}'$, in order to maintain the 3D profile of the flight segment, the vertical speed v_z for that flight segment needs to be adjusted.

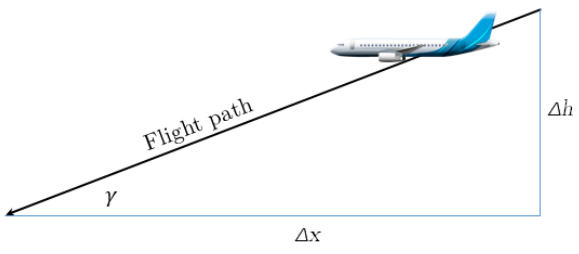

Fig. 4: Vertical flight path angle.

The angle that the straight-line flight path makes with the horizontal is given by the following equation:

$$\gamma = \tan^{-1} \frac{\Delta h}{\Delta x} = \frac{v_z}{v_g}$$

For the vertical profile to remain same, the angle γ needs to remain unchanged. Therefore, the required vertical speed $v_z^{'}$ for a segment can be computed by using equation 3.

$$v_z^{'} = v_g{}^{'} \times \frac{v_z}{v_g} \tag{3}$$

IV. Algorithm for computing the ground speed assignments in $\bar{F_a}$:

In this section, we present an algorithm for assigning a discrete value of ground speed to each segment of a flight-plan F_a to create a valid solution flight-plan $\bar{F_a}$. We also describe certain properties of the algorithm that are later formally specified.

The assignment of ground speed to the segments can be arranged as an assignment matrix M of dimension $n \times m$ where $n \in \mathbb{N}$ is the number of segments in F_a and $m \in \mathbb{N}$ is the number of discrete values of ground speed in ξ (M ranges over $\{0,1\}^{n \times m}$). Each row r represents a segment of $\bar{F_a}$ and each column c represents a value of ground speed in ξ. Initially, all the cells of M are marked with 0's. A matrix comprised of only 0's is represented by M_0. When a value of ground speed corresponding to column c is assigned to a segment corresponding to row r, the cell $M[r,c]$ is marked with a 1. An example assignment matrix is given in Fig. 5. It represents a flight-plan with 5 segments that have been assigned ground speeds of 140, 200, 120, 140, and 240 *kts* respectively.

	240 kts	200 kts	180 kts	160 kts	140 kts	120 kts
1	0	0	0	0	**1**	0
2	0	**1**	0	0	0	0
3	0	0	0	0	0	**1**
4	0	0	0	0	**1**	0
5	**1**	0	0	0	0	0

Fig. 5: Example assignment matrix for a solution flight-plan.

An algorithm that solves the above matrix assignment problem should satisfy the following properties:

- *Safety* - The algorithm should return an assignment matrix M if and only if adding the corresponding flight-plan F to the given set of traffic flight-plans Φ creates a safe set of flight-plans $\Phi \cup \{F\}$.

  This property ensures that if a solution flight-plan is found, the resulting set of flight-plans in the airspace is safe.
- *Completeness* - If there is a value of ground speed that can be assigned to a segment such that it leads to a valid solution, then the algorithm will not return M_0.

  This property ensures that if a valid permutation of values of ground speed exists, the algorithm will find the corresponding assignment matrix and not incorrectly terminate by returning M_0.

We propose a backtracking algorithm for finding the assignment matrix M corresponding to a valid $\bar{F_a}$. It returns a solution only if it can find an assignment matrix such that the corresponding flight-plan can be safely added to the set of traffic flight-plans. It returns M_0 if no such solution exists.

```
complete(M):
 if ∀ row ∈ M : assigned(row)
  return True
 else
  return False
```

```
check-safety(M, F_a, Φ, ξ, D, H):
  Let
    F = plan(M, F_a, ξ)
  in
    if safe(Φ ∪ {F}, D, H)
      return True
    else
      return False
    endif
```

```
valid(v, M, F_a, Φ, ξ, D, H):
 Let
  M' = set(M, unassigned(M), getindex(ξ, v))
  safety = check-safety(M', F_a, Φ, ξ, D, H)
  succeeding = solve(M', F_a, Φ, ξ, D, H)
 in
  if (safety & succeeding ≠ M_0)
    return True
  else
    return False
  endif
```

```
solve(M, F_a, Φ, ξ, D, H):
 if complete(M) & check-safety(M, F_a, Φ, ξ, D, H)
  return M
 endif
 if complete(M) & ¬check-safety(M, F_a, Φ, ξ, D, H)
  return M_0
 endif
 if ¬complete(M) &∃v : v ∈ ξ&valid(v, M, F_a, Φ, ξ, D, H)
  Let
   M' = set(M, unassigned(M), getindex(ξ, v))
  in
   return solve(M', F_a, Φ, ξ, D, H)
 endif
 if ¬complete(M) &¬(∃v : v ∈ ξ&valid(v, M, F_a, Φ, ξ, D, H))
  return M_0
 endif
```

Our algorithm comprises of four primary functions: `solve`, `check-safety`, `complete`, and `valid`. The `solve` function takes as input an assignment matrix M, the proposal flight-plan F_a, a set of traffic flight-plans Φ, a vector ξ of discrete values of ground speed for aircraft a, and the thresholds D and H. It tries to assign a value of ground speed to the first unassigned row of M (given by `unassigned(M)`).

The `complete` function takes as input an assignment matrix M and returns `True` only if all the rows of M have a ground speed assignment. An assignment matrix M is *complete* if `complete(M)` returns `True` and *incomplete* otherwise.

The check-safety function can determine if the flight-plan corresponding to an assignment matrix M can be safely added to Φ. It takes as input an assignment matrix M, the proposal flight-plan F_a, a set of traffic flight-plans Φ, a vector ξ of discrete values of ground speed for aircraft a, and the thresholds D and H. It uses the plan function to create a flight-plan F corresponding to M and returns True only if the set $\Phi \cup \{F\}$ is safe.

solve returns the assignment matrix M if it is complete and check-safety returns True for M. If M is incomplete, then solve checks if a valid assignment of ground speed is possible for the first unassigned row.

An assignment of ground speed for a row is valid only if the following conditions are satisfied:

- The corresponding intermediate flight-plan can be safely added to the set Φ.
- The assignment allows assignments of ground speed for succeeding rows.

The valid function can be used to check the validity of an assignment. It takes as input a value of ground speed v, an assignment matrix M, the proposal flight-plan F_a, a set of traffic flight-plans Φ, a vector ξ of discrete values of ground speed for aircraft a, and the thresholds D and H. It returns True if the assignment of v to the first unassigned row satisfies the two conditions for a valid assignment and False otherwise.

If a valid assignment of ground speed v can be found for the first unassigned row, the current assignment matrix M is updated to a new assignment matrix M' using the set function. The set function assigns 1 to the cell (unassigned(M),getindex(ξ, v)), where the function getindex(ξ, v) returns the index of v in ξ. solve then proceeds to make an assignment to the first unassigned row in the updated assignment matrix M'. If no valid assignment of ground speed can be found for the first unassigned row of M, then solve exits by returning M_0.

In the next section, we will formalize some important properties of our approach, including the safety and completeness properties.

V. Formal Verification of the Flight-Planning Algorithm

In this section, we present some important correctness properties of the specifications introduced in Section III. We also present the formal specifications of safety and completeness properties.

Theorem 1. $\forall D, H, t_1, t_2 : \mathbb{R}$ *and* $F_a, F_b : \mathbb{F}$, exists-violation$(t_1, t_2, F_a, F_b, D, H) \iff \exists\ t : \|\mathbf{s}_{xy,t}\| < D\ \&\ |s_{z,t}| < H\ \&\ t_1 < t \leq t_2$.

Theorem 1 asserts that, the exists-violation function correctly returns True if and only if a violation of minimum horizontal separation D and minimum vertical separation H between time t_1 and t_2 exists.

Theorem 2. $\forall\ D, H, t_1, t_2 : \mathbb{R}$, $n : \mathbb{N}$, $T : \mathbb{R}^n$, *and* $F_a,\ F_b : \mathbb{F}$, $\big($exists-violation$(t_1, t_2, F_a, F_b, D, H) \implies$ T-violation$(\langle t_1, t_2, T\rangle, F_a, F_b, D, H)\big)\ \&\ \big(\neg$exists-violation$(t_1, t_2, F_a, F_b, D, H) \implies$ T-violation$(\langle t_1, t_2, T\rangle, F_a, F_b, D, H) = $T-violation$(\langle t_2, T\rangle, F_a, F_b, D, H)\big)$.

Theorem 2 asserts that the function T-violation recursively checks for possible conflicts between pair of consecutive times in a vector of times $\langle t_1, t_2, T\rangle$ and correctly returns True when a conflict is detected between two consecutive times t_1 and t_2.

Theorem 3. $\forall D, H : \mathbb{R}$ *and* $F_a, F_b : \mathbb{F}$, T-violation$($get-all-times$(F_a, F_b), F_a, F_b, D, H) \iff$ conflict(F_a, F_b, D, H).

Theorem 3 asserts that the conflict function correctly returns True if a conflict is detected at any time between two given flight-plans and returns False otherwise.

Theorem 4. $\forall\ D, H : \mathbb{R}$, $k : \mathbb{N}$, $\Phi : \mathbb{F}^k$ *and* $F_i, F_j : \mathbb{F}$, $\neg\big(\exists\ F_i,\ F_j \in \Phi$: conflict$(F_i, F_j, D, H)\big) \iff$ safe(Φ, D, H).

Theorem 4 asserts that the safe function returns True if and only if there is no conflict between any two flight-plans in the given set of flight-plans. Therefore, safe(Φ, D, H)=True guarantees that a set of flight-plans Φ is safe when the horizontal and vertical thresholds D and H are given.

Theorem 5. $\forall\ D, H : \mathbb{R}$, $n, m, k : \mathbb{N}$, $M : \{0,1\}^{n\times m}$, $F :\ \mathbb{F}$, *and* $\Phi : \mathbb{F}^k$, $\xi : \mathbb{R}^m$, safe$(\Phi \cup$plan$(M, F, \xi), D, H) \iff$ check-safety(M, F, Φ, ξ, D, H).

Theorem 5 asserts that the check-safety function correctly returns True for an assignment matrix M if and only if adding the corresponding flight-plan to the set of traffic flight-plans Φ results in a safe set of flight-plans.

Theorem 6. $\forall\ n,\ m\ :\ \mathbb{N},\ M : \{0,1\}^{n\times m}, (\forall\ r \in M$: assigned$(r)\big) \iff$ complete(M).

Theorem 6 asserts that the complete function returns True if and only if every row in the matrix M has been assigned a value of ground speed. Therefore, it can correctly detect if an assignment matrix is complete.

Theorem 7. $\forall\ D, H, v : \mathbb{R}$, $n, m, k : \mathbb{N}$, $M :\ \{0,1\}^{n\times m}$, $F_a :\ \mathbb{F}$, $\Phi :\ \mathbb{F}^k$, *and* $\xi :\ \mathbb{R}^m$, valid$(v, M, F, \Phi, \xi, D, H) \iff \big($check-safety$(set(M,$unassigned$(M)),$getindex$(\xi, v)), F, \Phi, \xi, D, H)\ \&\ \neg($solve$(M, F, \Phi, \xi, D, H) = M_0)\big)$

Theorem 7 asserts that the valid function returns True if and only if the assignment of v to the first unassigned row satisfies the two conditions for a valid assignment.

The Safety Property

Theorem 8. $\forall\ D, H : \mathbb{R}$, $n, m, k : \mathbb{N}$, $M : \{0,1\}^{n\times m}$, $F_a :\ \mathbb{F}$, $\Phi : \mathbb{F}^k$, *and* $\xi : \mathbb{R}^m$, complete$(M) \implies \big($solve$(M, F, \Phi, \xi, D, H) = M) \iff$ safe$(\Phi \cup$plan$(M, F, \xi), D, H)\big)$.

Theorem 8 asserts that if an assignment matrix M is complete, solve will return M if and only if adding the corresponding flight-plan $\bar{F}$ to the set of traffic flight-plans

Φ creates a safe set of flight-plans. Therefore, Theorem 8 guarantees that the safety property is satisfied by the algorithm specifications.

The Completeness Property

Theorem 9. $\forall\ D, H : \mathbb{R},\ n, m, k : \mathbb{N},\ M : \{0,1\}^{n\times m},\ F_a : \mathbb{F},\ \Phi : \mathbb{F}^k,$ *and* $\xi : \mathbb{R}^m,\ \neg$`complete(`$M$`)` $\implies \big($`(`$\exists\ v : v \in \xi$ `&` `valid(`v, M, F, Φ, ξ, D, H`))` $\iff \neg$ `(solve(`M, F, Φ, ξ, D, H`)=`M_0`))`$\big)$

Theorem 9 states that when the assignment matrix M is incomplete, if there exists a valid assignment of ground speed v for the first unassigned row, then solve does not return M_0. Therefore, Theorem 9 summarizes the completeness property of our algorithm.

All the lemmas and theorems introduced in this section have been verified using Athena[1]. Therefore, our algorithm has been mechanically verified to satisfy the safety and completeness properties mentioned in Section IV.

VI. Experimentation and Results

We developed a reference implementation of our approach in Python that closely follows the specifications that were used to verify the properties in Athena. We created a proposal flight-plan F_a for an ownship a and a set of traffic flight-plans Φ. Initially, Φ contained only one traffic flight-plan F_b that had a conflict with F_a. We then incrementally added two more traffic flight-plans F_c and F_d that were designed such that a solution flight-plan $\bar{F_a}$ would not exist for the proposal. All our experiments assume a Euclidean 3D coordinate system (with 100 feet = 1 unit on both axes). We used a value of 1 nautical mile for the horizontal threshold (D) and 1000 feet for the vertical threshold (H). All the flight-plans were at a constant altitude of 5000 feet. The details of the flight-plans are given below:

- F_a had a starting time of 0.00 seconds and was comprised of the way-points $\langle -60, 100, 50\rangle$, $\langle -55, 65, 50\rangle$, $\langle 5, 5, 50\rangle$, and $\langle 125, 5, 50\rangle$ with a proposed ground speed of 240 *kts* in every segment.
- F_b had a starting time of 0.00 seconds and was comprised of the way-points $\langle 125, 5, 50\rangle$, $\langle 5, 5, 50\rangle$, and $\langle -55, -55, 50\rangle$ with a ground speed of 240 *kts* in every segment.
- F_c had a starting time of 61.23 seconds and was comprised of the way-points $\langle 69, -85, 50\rangle$, $\langle 71, -55, 50\rangle$, and $\langle 121, 5, 50\rangle$ with a ground speed of 200 *kts* in every segment.
- F_d had a starting time of 63.66 seconds and was comprised of the way-points $\langle 3, -55, 50\rangle$, $\langle 123, 65, 50\rangle$, and $\langle 160, 70, 50\rangle$ with a ground speed of 174 *kts* in every segment.
- The vector of ground speeds for aircraft a was $\xi = \langle 100, 120, 140, 160, 180, 200, 220, 240\rangle$ (all in *kts*).

[1]Complete Athena code available at: http://wcl.cs.rpi.edu/pilots/fvcafp

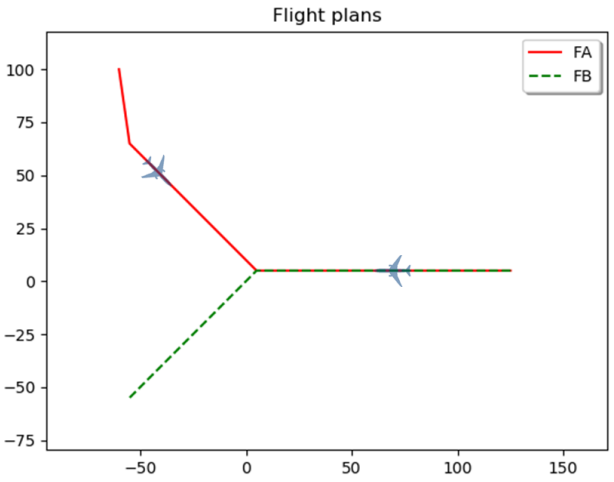

Fig. 6: Top view of a scenario where an aircraft a is in conflict with a traffic aircraft b (aircraft markers represent the direction of flight-plans and not actual positions).

	240 kts	220 kts	200 kts	180 kts	160 kts	140 kts	120 kts	100 kts
Segment 1	0	0	0	0	0	0	0	**1**
Segment 2	0	0	0	0	0	0	0	**1**
Segment 3	**1**	0	0	0	0	0	0	0

Fig. 7: An assignment matrix for the scenario in Fig. 6.

Fig. 6 depicts the first scenario where Φ contains one traffic flight-plan F_b. a has a possible conflict with aircraft b in the third segment of F_a between the way-points $\langle 5, 5, 50\rangle$ and $\langle 125, 5, 50\rangle$. Our algorithm computed the assignment matrix given in Fig. 7 that corresponded to a solution flight-plan $\bar{F_a}$.

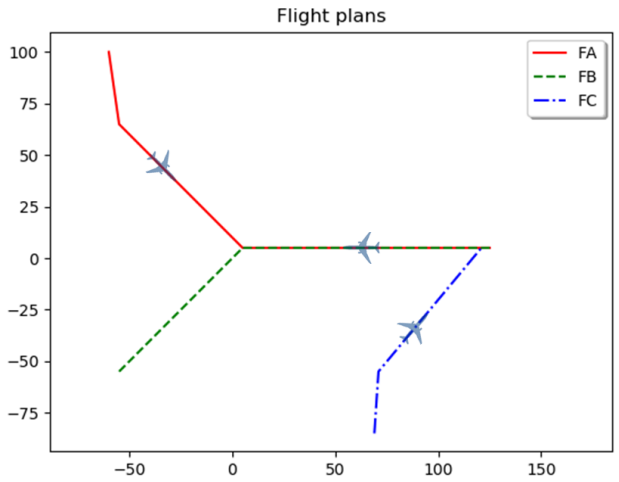

Fig. 8: Top view of a scenario where an aircraft a is in conflict with two traffic aircraft b and c (aircraft markers represent the direction of flight-plans and not actual positions).

	240 kts	220 kts	200 kts	180 kts	160 kts	140 kts	120 kts	100 kts
Segment 1	0	0	0	0	0	0	0	**1**
Segment 2	0	0	0	0	0	0	0	**1**
Segment 3	0	0	0	0	0	**1**	0	0

Fig. 9: An assignment matrix for the scenario in Fig. 8.

In the second scenario depicted in Fig. 8, we added a new traffic flight-plan F_c to Φ ($\Phi^{'} = \Phi \cup \{F_c\}$) such that aircraft a has a conflict with aircraft c if it follows the solution flight-plan $\bar{F_a}$ created from the assignment matrix in Fig. 7. In this case, our algorithm computed the assignment matrix given in Fig. 9 that can be used to create a solution flight-plan $\bar{F_a}^{'}$ that maintains standard separation from both aircraft b and c.

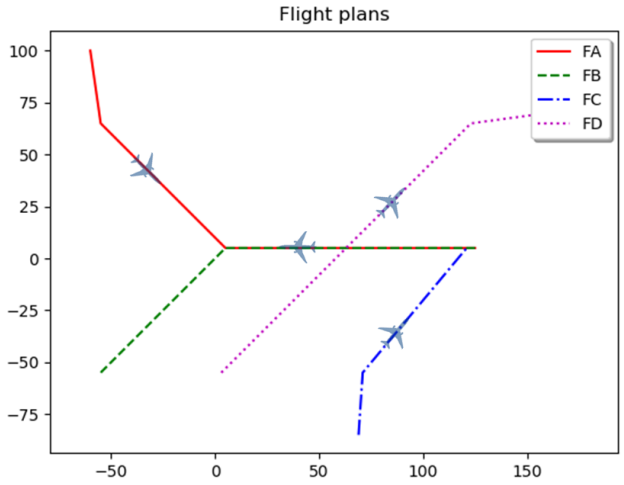

Fig. 10: Top view of a scenario where an aircraft a is in conflict with three traffic aircraft b, c, and d (aircraft markers represent the direction of flight-plans and not actual positions).

	240 kts	220 kts	200 kts	180 kts	160 kts	140 kts	120 kts	100 kts	60 kts
Segment 1	0	0	0	0	0	0	0	**1**	0
Segment 2	0	0	0	0	0	0	0	0	**1**
Segment 3	**1**	0	0	0	0	0	0	0	0

Fig. 11: An assignment matrix for the scenario in Fig. 10.

For the third scenario, we added a fourth traffic aircraft d to the airspace (Fig. 10). Adding the new flight-plan F_d to $\Phi^{'}$ ($\Phi^{''} = \Phi \cup \{F_d\}$) created a situation where no permutation of ground speed in the three segments of F_a could generate a solution that is conflict free from all three traffic aircraft. However, on introducing an additional value of 60 *kts* ground speed in ξ, our algorithm could compute the assignment matrix for a solution flight-plan $\bar{F_a}^{''}$ as given in Fig. 11.

It is evident from the experiments that our approach can be effectively used for generating conflict-free flight-plans. The third scenario where a flight-plan was not initially possible but could be generated after a new discrete value of ground speed was allowed clearly shows that given a start time, ξ is an important variable in determining the possibility of a solution. Since ξ is a function of the allowed values of airspeed for an aircraft and the wind vector $\mathbf{w}$, it is obvious that the wind conditions can significantly impact solutions.

VII. Conclusion and Future Work

The future of the National Airspace System, with the integration of unmanned aerial systems for civilian use, will face significant challenges in terms of air-traffic management. With an increase in the density of aircraft in the airspace, it will become necessary to update the current state-of-the-art in traffic flow management to accurately maintain standard separation between aircraft in highly congested airspace. Future advancements in ADS-B technology will give way to the development of smarter and more efficient systems for tracking aircraft in the airspace and providing both tactical and strategic guidance to ensure standard separation between them. The approach for conflict-aware flight planning presented in this paper is one of the many possible techniques that can be implemented with the rich data that can be provided by ADS-B in the near future. Since our approach does not introduce additional way-points in order to maintain standard separation, it can be used in situations where airspace capacity, restricted airspace, weather or terrain may prevent deviations between way-points. Our work, therefore, is particularly suitable for UAS integration in the national airspace since trajectory deviations in urban environments are typically restricted by traffic or terrain. It may also be implemented in terminal areas where systems like TCAS are not feasible due to the low flight altitudes of aircraft.

Our approach can only generate a flight-plan for an ownship given a set of immutable traffic flight-plans. If a solution cannot be found, no changes are made to the traffic flight-plans in order to accommodate the ownship. A cooperative protocol in which all aircraft adjust to accommodate each other can be implemented by using consensus algorithms such as RAFT. However, consensus is too strict a condition, especially if an aircraft needs to propose an emergency flight-plan. In real-world ground transportation systems, emergency vehicles simply broadcast their intent and predefined protocols are followed by the traffic vehicles in order to accommodate them. Ground traffic management systems usually do not require any form of consensus from the vehicles for ensuring conflict-free flow of traffic. Therefore, our future directions of work include the development of intent broadcast-based protocols which can be cooperatively used by aircraft in an airspace for conflict-aware flight-planning. Such protocols will make it possible to better utilize the full capacity of an airspace while ensuring minimum standard separation between aircraft.

Acknowledgment

This research is partially supported by the National Science Foundation (NSF), Grant No. - CNS 1816307 and the Air Force Office of Scientific Research (AFOSR), Grant No. - FA9550-19-1-0054.

REFERENCES

[1] Federal Aviation Administration, "Air Traffic By the Numbers," 2018.

[2] Federal Aviation Administration, "Introduction to TCAS-II Version 7.1," 2011.

[3] Federal Aviation Administration, "Traffic Flow Management in the National Airspace System," 2009.

[4] K. Arkoudas, "Athena." http://proofcentral.org/athena.

[5] K. Arkoudas and D. Musser, "Athena Libraries." http://proofcentral.org/athena/lib.

[6] K. Arkoudas and D. Musser, *Fundamental Proof Methods in Computer Science: A Computer-Based Approach*. MIT Press, 2017.

[7] A. R. Pritchett and A. Genton, "Negotiated decentralized aircraft conflict resolution," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 81–91, 2018.

[8] G. Dowek, C. Muñoz, and V. Carreño, "Provably safe coordinated strategy for distributed conflict resolution," in *AIAA Guidance, Navigation, and Cont. (GNC) Conf.*, (San Francisco, CA), p. 6047, Aug. 2005.

[9] A. L. Galdino, C. Muñoz, and M. Ayala-Rincón, "Formal verification of an optimal air traffic conflict resolution and recovery algorithm," in *Int. Workshop on Logic, Lang., Info., and Comput.*, pp. 177–188, Springer, 2007.

[10] S. Balachandran, C. Muñoz, and M. Consiglio, "Distributed consensus to enable merging and spacing of UAS in an urban environment," in *2018 Int. Conf. on Unmanned Aircraft Syst. (ICUAS)*, pp. 670–675, IEEE, 2018.

[11] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX Annual Tech. Conf.*, pp. 305–319, 2014.

[12] D. Alejo, J. A. Cobano, G. Heredia, and A. Ollero, "Collision-free 4D trajectory planning in unmanned aerial vehicles for assembly and structure construction," *J. Intell. & Robotic Syst.*, vol. 73, no. 1-4, pp. 783–795, 2014.

[13] M. Pontani and B. A. Conway, "Particle swarm optimization applied to space trajectories," *J. Guidance, Cont., and Dynamics*, vol. 33, no. 5, pp. 1429–1441, 2010.

[14] C. Munoz, A. Narkawicz, J. Chamberlain, M. C. Consiglio, and J. M. Upchurch, "A family of well-clear boundary models for the integration of uas in the nas," in *14th AIAA Aviation Technol., Integr., and Operat. Conf.*, p. 2412, 2014.

[15] A. Narkawicz, C. Munoz, and A. Dutle, "Coordination logic for repulsive resolution maneuvers," in *16th AIAA Aviation Technol., Integr., and Operat. Conf.*, p. 3156, 2016.

[16] C. Muñoz, A. Narkawicz, and J. Chamberlain, "A TCAS-II resolution advisory detection algorithm," in *AIAA Guidance, Navigation, and Cont. (GNC) Conf.*, (Boston, MA), p. 4622, Aug. 2013.

[17] C. Muñoz, A. Narkawicz, G. Hagen, J. Upchurch, A. Dutle, M. Consiglio, and J. Chamberlain, "DAIDALUS: Detect and avoid alerting logic for unmanned systems," in *Proc. 34th AIAA/IEEE Digit. Avionics Syst. Conf.*, pp. 5A1–1, IEEE, 2015.

[18] S. Balachandran, C. A. Muñoz, M. C. Consiglio, M. A. Feliú, and A. V. Patel, "Independent configurable architecture for reliable operation of unmanned systems with distributed onboard services," in *Proc. 37th AIAA/IEEE Digit. Avionics Syst. Conf.*, pp. 1–6, IEEE, 2018.

[19] J. P. Chamberlain, M. C. Consiglio, and C. Muñoz, "DANTi: Detect and avoid in the cockpit," in *17th AIAA Aviation Technol., Integr., and Operat. Conf.*, p. 4491, 2017.

[20] S. Paul, F. Hole, A. Zytek, and C. Varela, "Flight trajectory planning for fixed wing aircraft in loss of thrust emergencies," tech. rep., Rensselaer Polytechnic Institute, Troy, NY, USA, Oct. 2017.

[21] S. Paul, F. Hole, A. Zytek, and C. Varela, "Wind-aware trajectory planning for fixed-wing aircraft in loss of thrust emergencies," in *Proc. 37th AIAA/IEEE Digit. Avionics Syst. Conf.*, (London, England, UK), pp. 558–567, Sep. 2018.

[22] S. Paul, "Emergency trajectory generation for fixed-wing aircraft," Master's thesis, Rensselaer Polytechnic Institute, Dec. 2018.