ADAPTIVE STRESS TESTING OF AIRBORNE COLLISION AVOIDANCE SYSTEMS

Ritchie Lee, Carnegie Mellon University Silicon Valley, Moffett Field, CA
Mykel J. Kochenderfer, Stanford University, Stanford, CA
Ole J. Mengshoel, Carnegie Mellon University Silicon Valley, Moffett Field, CA
Guillaume P. Brat, NASA Ames Research Center, Moffett Field, CA
Michael P. Owen, MIT Lincoln Laboratory, Lexington, MA

Abstract

This paper presents a scalable method to efficiently search for the most likely state trajectory leading to an event given only a simulator of a system. Our approach uses a reinforcement learning formulation and solves it using Monte Carlo Tree Search (MCTS). The approach places very few requirements on the underlying system, requiring only that the simulator provide some basic controls, the ability to evaluate certain conditions, and a mechanism to control the stochasticity in the system. Access to the system state is not required, allowing the method to support systems with hidden state. The method is applied to stress test a prototype aircraft collision avoidance system to identify trajectories that are likely to lead to near mid-air collisions. We present results for both single and multi-threat encounters and discuss their relevance. Compared with direct Monte Carlo search, this MCTS method performs significantly better both in finding events and in maximizing their likelihood.

Introduction

Airborne collision avoidance systems are mandated worldwide on large transport and cargo aircraft to help prevent mid-air collision. Since it entered operation in 1993, the Traffic Alert and Collision Avoidance System (TCAS) has played a crucial role in greatly reducing the risk of mid-air collision [1]. With air traffic projected to double in the next 30 years [3], the Federal Aviation Administration (FAA) has decided to develop a new aircraft collision avoidance system capable of addressing the growing needs of the national airspace. The next-generation Airborne Collision Avoidance System (ACAS X) is currently being developed and tested by the FAA and promises a number of potential improvements including further reduction in collision risk while simultaneously reducing the number of unnecessary alerts [4]. ACAS X uses a partially observable Markov decision process to model the problem and dynamic programming to efficiently compute its solution [4].

Prior to certification, an aircraft collision avoidance system must undergo extensive verification and validation. A variety of different metrics are used to evaluate the safety, suitability, and acceptability of the system [19]. One of the primary safety metrics is the likelihood of near mid-air collision (NMAC), defined as being when two aircraft come less than 100 feet vertically and 500 feet horizontally from one another. It is well accepted that the likelihood of NMAC cannot be driven to zero due to surveillance noise, pilot response delay, and the need for an acceptable alert rate. However, it is still important to understand the situations where NMACs can arise, even if they are extremely rare.

There have been a variety of both "white box" and "black box" methods applied to the analysis of NMAC events. White box methods assume that the inner details of the system are available to be inspected and leveraged in the analysis. For example, Von Essen and Giannakopoulou [16] used probabilistic model-checking to analyze a simplified version of ACAS X, and Jeannin, Ghorbal, Kouskoulas, *et al.* [18] used an automated theorem prover. Unfortunately, white box methods generally have to rely on approximate representations of the system because of their difficulty scaling to the many internal variables governing the state of the system.

In contrast with white box methods, black box methods do not use information about the internal details of the system. Consequently, they can scale to more complicated systems like TCAS and ACAS X that have many variables. All that is required in a

black box approach is the ability to simulate the system. Prior analysis of TCAS involved sweeping through a low-dimensional parametric model of head-on encounters and simulating the collision avoidance system [20]. Although this kind of stress testing can find NMACs, it is limited to the low-dimensional parameterization of the encounter space and it can be difficult to assess the likelihood of the various NMACs. An alternative approach is to run simulations drawn from a statistical representation of the airspace [17]. Unfortunately, directly sampling from such a model is very computationally inefficient due to the rarity of NMACs.

This paper presents a black box method for adaptive stress testing that aims to find the most likely scenarios that lead to NMAC. The approach is related to reinforcement learning, an area of machine learning concerned with making decisions in an unknown environment so as to maximize a numerical reward [6]. Our problem differs from a traditional reinforcement learning problem in that we operate on non-Markovian systems and optimize over state transitions rather than an explicit set of actions.

The proposed methodology is quite general and is applicable to the adaptive stress testing of a variety of different systems. This paper first presents the general methodology and then applies it to the analysis of the official binaries encoding the decision logic of an ACAS X prototype.

Problem Description

We describe the general problem of finding the most likely state trajectory that leads to a critical event E where only a generative black box simulator $\mathscr{S}$ is available. We assume the system underlying the simulator to be discrete-time Markovian with stochastic transitions. However, the state of the system is hidden, making the process non-Markovian from the search algorithm's perspective. We use s_t to denote the hidden internal state of the simulator $\mathscr{S}$ at time t.

We specify the inputs to the problem by a pair $(\mathscr{S}, E)$, where $\mathscr{S}$ is a generative black box simulator and E is a subset of the state space where the event of interest occurs. The black box simulator exposes the system of interest as a discrete-time Markov chain with seeded stochastic transitions. Specifically, the simulator steps through time drawing a random next state at each time step and updating its internal state, where the nominal randomness for the sampling is pseudorandomly generated from a provided seed. The *seed* can be the pseudorandom seed or state of the pseudorandom number generator. The simulator exposes three functions for simulation control:

- INITIALIZE($\mathscr{S}$) resets the simulator $\mathscr{S}$ to its initial state s_0.
- STEP($\mathscr{S}, a_t$) updates the hidden state of the simulator $\mathscr{S}$ by pseudorandomly sampling a next state s_{t+1} given the current state s_t according to the system transition probability $P(s_{t+1} \mid s_t)$ and the seed a_t. The function returns the probability of the transition taken and a boolean that indicates whether s_t is in E. Both INITIALIZE and STEP transform $\mathscr{S}$ in place.
- ISTERMINAL($\mathscr{S}$) returns true if the current state of the simulator is terminal, and false otherwise. We define the *terminal time* T to be the first time at which ISTERMINAL returns true.

The problem is defined as follows: Given $(\mathscr{S}, E)$, find the trajectory that contains the occurrence of E and maximizes the likelihood of the trajectory $\prod_{t=0}^{T} P(s_{t+1} \mid s_t)$.

Proposed Method

The overall strategy of our proposed solution is to recast the given problem into a decision-making problem and apply a variation of a tree-based reinforcement learning algorithm called Monte Carlo Tree Search (MCTS). A system diagram is shown in Figure 1. At the center of the method is the system under test modeled as a black box simulator. It takes as input basic simulator controls (i.e., INITIALIZE, STEP, and ISTERMINAL) and a seed, and outputs the likelihood of the current transition and a boolean indicating whether the current state is an event. The outputs are transformed into the reward using the reward function and passed to the MCTS algorithm. Finally, to complete the loop, MCTS uses the reward to choose the seed and control inputs of the simulator. We describe each component of the system in detail.

Reformulation as a Decision Process

We take the stochastic process defined by $\mathscr{S}$ and insert decision points between each time step where each decision is to choose the seed a_t. Recall that a_t controls the stochastic transition of the system by

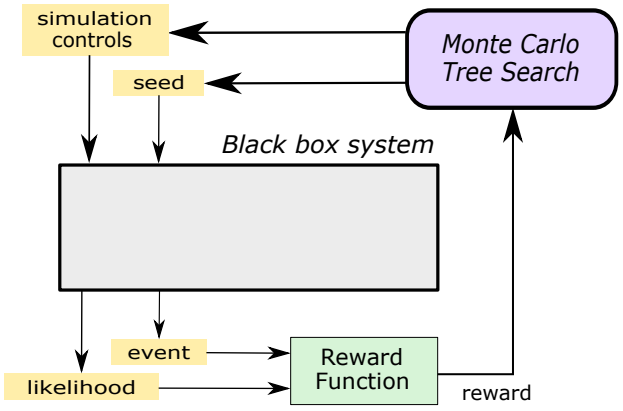

Figure 1. System diagram of stress testing method

specifying a pseudorandomly-generated sample of the next state in STEP. As a result, rather than allowing the system to evolve naturally (and stochastically), the sequence of decisions uniquely determines the state evolution of the system.

Reward Function

We design a reward function for the decision process that is equivalent to the objective in the original problem, which is to find trajectories that contain events and maximize the likelihood of the trajectory $\prod_{t=0}^{T} P(s_{t+1} \mid s_t)$. Since reinforcement learning maximizes the expected sum of rewards (as opposed to product), we choose the following reward function:

$$R(s_t, s_{t+1}) = \begin{cases} 0 & \text{if } s_t \in E, \\ -\infty & \text{if } s_t \notin E, t \geq T \\ \log P(s_{t+1} \mid s_t) & \text{if } s_t \notin E, t < T. \end{cases} \tag{1}$$

Throughout this discussion, we assume, without loss of generality, that terminal states are *absorbing*. An absorbing state is one that transitions to itself with probability 1. After initially collecting the reward for being in that state, subsequent transitions give rewards of 0. Recall that the terminal time T is the first time at which the state becomes terminal. As a result, we can use the shorthand notation $t \geq T$ to indicate a terminal state, and $t < T$ to indicate a non-terminal one. Furthermore, we assume that states in E are terminal, so that $s_t \in E$ implies $t \geq T$.

The first two conditions implement the event occurrence constraint. If the trajectory terminates and E occurs, the first condition awards a maximum reward of 0. However, if the trajectory terminates and E does not occur, then the second condition infinitely penalizes the learner. The third condition in the reward function maximizes the trajectory likelihood by giving a negative reward $\log P(s_{t+1} \mid s_t)$ for each non-terminal transition. By choosing a reward of $\log P(s_{t+1} \mid s_t)$, we maximize $\sum_{t=0}^{T} \log P(s_{t+1} \mid s_t)$ in the reinforcement learning problem, which is in fact equivalent to maximizing $\prod_{t=0}^{T} P(s_{t+1} \mid s_t)$ in the original problem.

We observe that each term in the reward function is non-positive. As a result, the total reward of any trajectory, $\sum_{t=0}^{T} R(s_t, s_{t+1})$, will be non-positive as well. In fact, if E occurs, then the total reward is the log likelihood of the trajectory. Otherwise, the total reward is $-\infty$.

Monte Carlo Tree Search

One of the most successful sampling-based online approaches to reinforcement learning in recent years is Monte Carlo Tree Search (MCTS). The approach has become widely known because of recent successes in the game of computer Go [10]. MCTS incrementally builds a search tree using sampling and forward simulation to inform the search and focus on the most promising areas. Here, we describe a variation of MCTS that uses *double progressive widening*, a technique that controls the branching factor of the search tree [11]. This variation is specifically necessary when the action space is continuous or so large that all actions cannot possibly be explored. The latter applies in our case since the actions are in the space of pseudorandom seeds, which is too large to exhaustively explore. For a detailed review of MCTS, see [10].

The algorithm involves running many simulations from the current state while updating an estimate of the *state-action value function*. The state-action value function $Q(s,a)$ represents the expected sum of rewards resulting from choosing action a in state s. The following is an overview of the three stages of each simulation:

- *Search.* In the search stage, the algorithm starts at the root of the tree and recursively selects a child to follow. At each visited state node, the first progressive widening criterion determines whether to choose amongst existing actions, or

to generate a new one. The criterion limits the number of actions at a state s to be no more than polynomial in the total number of visits to that state. Specifically, a new action is generated according to a user-defined function GETACTION if $\|A(s)\| < kN(s)^{\alpha}$, where k and α are parameters, $\|A(s)\|$ is the number of existing actions at state s, and $N(s)$ is the total number of visits to state s. Otherwise, the existing action that maximizes

$$Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}} \tag{2}$$

is chosen, where c is a parameter that controls the amount of exploration in the search, and $N(s,a)$ is the total number of visits to action a in state s. The second term in eq. 2 is an *exploration bonus* that encourages selecting actions that have not been tried as frequently.

At each visited action node, the second progressive widening criterion determines whether to follow an existing next state node, or to draw a new next state from the simulator. The second criterion has the same form as the first and limits the number of next states to be no more than polynomial in the number of visits to that state-action pair. We draw a new next state if $\|V(s,a)\| < k'N(s,a)^{\alpha'}$, where k' and α' are parameters, $\|V(s,a)\|$ is the number of next states visited from the state-action pair (s,a), and $N(s,a)$ is the total number of visits to (s,a). Otherwise, we randomly select a next state with probability proportional to $N(s,a,s')$, the number of times s' was encountered choosing action a in state s. The search stage continues in this manner until the system transitions to a state that is not in the tree.

- *Expansion.* Once we have reached a state that is not in the tree, we create a new node for the state and add it. The set of actions taken from this state, $A(s)$, is initialized to empty.
- *Rollout.* Starting from the state created in the expansion stage, we perform a *rollout* that repeatedly samples state transitions until the desired depth (or termination) is reached. State transitions are drawn from the simulator with actions chosen according to a rollout (or default) policy π_0. The total reward of the sampled trajectory is returned and used to update the value for $Q(s,a)$ used by the search phase.

Simulations are run until some stopping criterion is met, often simply a fixed number of iterations. We then execute the action that maximizes $Q(s,a)$. Once that action has been executed, we can rerun the MCTS to select the next action. The process is repeated until all actions are executed.

We now describe the specific choices of parameters and modifications we have made to tailor it for our purposes. Since we have a deterministic transition given the state and seed, there is no need to consider multiple samples of the next state s'. As a result, we disable the second progressive widening by setting $k' = 1$ and $\alpha' = 0$. The choice of these parameters makes $\|V(s,a)\| < k'N(s,a)^{\alpha'}$ true only once where a sample s' is drawn, then false thereafter independent of the value of $N(s,a)$. Since there is no reason to distinguish among pseudorandom seeds, we choose $a \sim \pi_0$ and GETACTION to sample uniformly from all available seed values. The choice of a uniform π_0 generates unbiased samples of the next state from our simulator. We initialize all state-action values to $Q_0(s,a) = 0$.

MCTS is designed for stochastic Markovian systems, and thus does not generally support systems that are non-Markovian. The problem is that the algorithm needs the ability to set the system state back to any visited state of the tree, and there is no general way to do that in the stochastic setting without an explicit state representation. Fortunately, since each transition from s_t to s_{t+1} is deterministic given action a_t, we can return to any visited state s_t by remembering the sequence of actions $a_{0:t-1} = a_0, \ldots, a_{t-1}$ taken since s_0. To revisit s_t, we first call INITIALIZE to return to s_0, then repeatedly call STEP with the sequence of actions $a_{0:t-1}$. When an explicit representation of s_t is used, we implicitly make the substitution $s_t = a_{0:t-1}$ reusing (and abusing) the notation for s_t. This key modification to the algorithm enables our method to support Markovian systems with hidden state. The process is outlined in Algorithms 1 and 2.

Figure 2 shows a search tree for one decision in a single threat encounter. One thousand iterations are shown. The figure shows how MCTS explores broadly, but also focuses deeply on a number of potentially high-reward areas.

Algorithm 1 Tailored Monte Carlo tree search with double progressive widening

function MONTECARLOTREESEARCH($\mathscr{S}, s, d$)
  loop
    GOTOSTATE($\mathscr{S}, s$)
    SIMULATE($\mathscr{S}, s, d$)
  return $\arg\max_a Q(s,a)$
function SIMULATE($\mathscr{S}, s, d$)
  if $d = 0$ **then**
    return 0
  if $s \notin \mathscr{T}$ **then**
    $\mathscr{T} \leftarrow \mathscr{T} \cup \{s\}$
    $(N(s), A(s)) \leftarrow (0, \emptyset)$
    return ROLLOUT($\mathscr{S}, s, d$)
  $N(s) \leftarrow N(s) + 1$
  if $\|N(s,a)\| < kN(s)^{\alpha}$ **then**
    $a \leftarrow$ GETACTION()
    $(N(s,a), V(s,a)) \leftarrow (0, \emptyset)$
    $Q(s,a) \leftarrow Q_0(s,a)$
    $A(s) \leftarrow A(s) \cup \{a\}$
  $a \leftarrow \arg\max_a Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}$
  if $\|V(s,a)\| < k'N(s,a)^{\alpha'}$ **then**
    $(P, E) \leftarrow$ STEP($\mathscr{S}, a$)
    $r \leftarrow$ GETREWARD(P, E)
    $s' \leftarrow [s, a]$
    if $s' \notin V(s,a)$ **then**
      $V(s,a) \leftarrow V(s,a) \cup \{s'\}$
      $R(s,a,s') \leftarrow r$
      $N(s,a,s') \leftarrow 0$
    else
      $N(s,a,s') \leftarrow N(s,a,s') + 1$
  else
    $n \leftarrow \sum_{s'} N(s,a,s')$
    $s' \leftarrow$ SAMPLE($V(s,a), N(s,a,\cdot)/n$)
    $r \leftarrow R(s,a,s')$
    $N(s,a,s') \leftarrow N(s,a,s') + 1$
  $q \leftarrow r + \gamma$SIMULATE($\mathscr{S}, s', d-1$)
  $N(s,a) \leftarrow N(s,a) + 1$
  $Q(s,a) \leftarrow Q(s,a) + \frac{q - Q(s,a)}{N(s,a)}$
  return q
function ROLLOUT($\mathscr{S}, s, d$)
  if $d = 0$ **then**
    return 0
  $a \sim \pi_0$
  $(P, E) \leftarrow$ STEP($\mathscr{S}, a$)
  $r \leftarrow$ GETREWARD(P, E)
  $s' \leftarrow [s, a]$
  return $r + \gamma$ROLLOUT($\mathscr{S}, s', d-1$)

Algorithm 2 MCTS auxiliary function

function GOTOSTATE($\mathscr{S}, s$)
  $a_{0:t-1} \leftarrow s$
  INITIALIZE($\mathscr{S}$)
  for each a in $a_{0:t-1}$ **do**
    STEP($\mathscr{S}, a$)

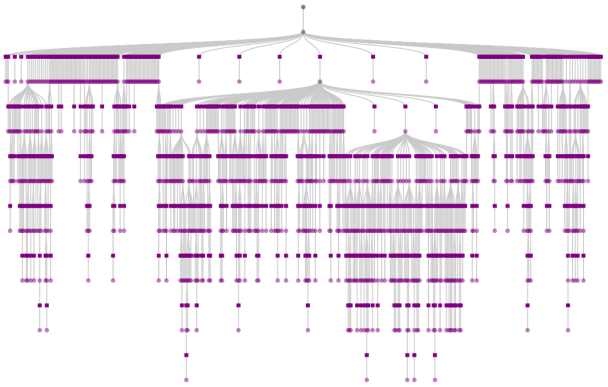

Figure 2. Visualization of the MCTS search tree for one decision point (1000 iterations shown)

Aircraft Collision Avoidance

ACAS X is targeted to replace TCAS as the standard collision avoidance system for transport aircraft worldwide. Before widespread acceptance, the safety of the system must be demonstrated. Monte Carlo evaluations of the system in realistic encounters play an essential role in system assessments. Safety evaluations are performed using validated noise models and incorporate altimetry bias effects. One important safety metric is the *risk ratio*, defined as the probability of an NMAC with a collision avoidance system divided by the probability of an NMAC without one. Figure 3 shows the estimated risk ratios of NMAC for TCAS and ACAS X based on 1.5 million encounters generated by the Lincoln Laboratory Correlated Aircraft Encounter Model (LLCEM) [12] [13] with both aircraft equipped with a collision avoidance system.

As shown in Figure 3, ACAS X provides a substantial safety benefit relative to TCAS while simultaneously reducing the alert rate. Although there are many other operational considerations that are important to system acceptance, this paper focuses on better understanding the remaining NMAC risk associated with ACAS X. While 1.5 million encounters provide

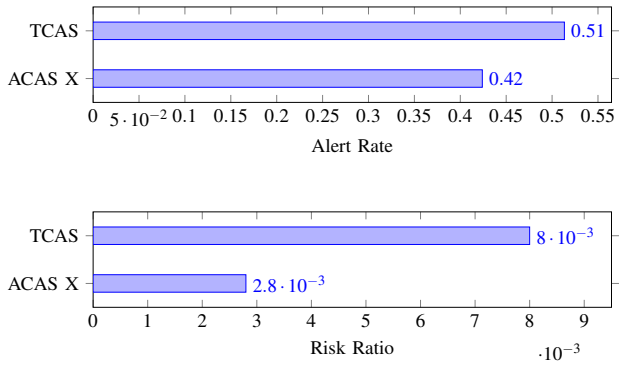

Figure 3. ACAS X and TCAS metrics.

insights into their relative safety, characterizing encounters that are still risk-bearing is important to system development. Using conventional methods to produce a meaningful set of NMAC examples would require an extremely large number of simulations. This work shows how the search can be performed much more effectively using a reinforcement learning approach. Insight gained from this research will inform the development process.

We stress test a prototype of ACAS X following the method described earlier by constructing a simulation environment, recasting it as a decision process, and solving it using MCTS. Insights gained from this work will inform design decisions for future iterations of the collision avoidance system. An overall diagram of our approach is shown in Figure 4.

Encounter Simulation

At the core of the analysis is a simulation that models the various components of a mid-air encounter. We implement the components using the SISLES.jl framework [14]. Our simulation includes an encounter model, sensor model, collision avoidance system, pilot model, and an aircraft dynamics model. We describe each component in detail.

Encounter Model: Encounter models are used to initialize encounters in such a way to be both realistic and likely to lead to NMACs. Initial states of aircraft, including positions, velocities, and headings are set in this manner. Pairwise (two-aircraft) encounters use the *Lincoln Laboratory Correlated Aircraft Encounter Model (LLCEM)*, and multi-threat (three-aircraft) encounters use the *Star Encounter Model*.

LLCEM is comprised of two parts. The first is a Bayesian network that models the geometry of two

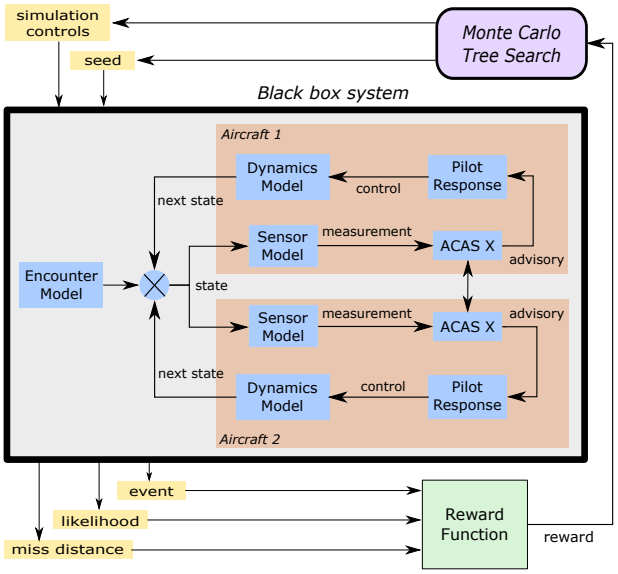

Figure 4. System diagram for stress testing pairwise ACAS X encounters

aircraft at point of closest approach, and the second is a dynamic Bayesian network that models how pilot commands transition over time. Both of these models are learned from a large body of radar data over the entire national airspace [12]. To obtain a set of encounter initial conditions from the two models, we follow the simulation and transformation procedure in the paper.

The Star Encounter Model initializes aircraft on a circle heading towards the origin and at equal angles apart. Initial airspeed, altitude, and vertical rate are sampled from a uniform distribution. The horizontal distance from the origin is set such that without intervention, the aircraft crosses the origin at a preset time. For our experiments, we set the crossing time to 40 seconds.

Sensor Model: The sensor model captures how the collision avoidance system perceives the world. We assume active, beacon-based sensor capability with no noise. The main sensor inputs from own aircraft are $O_0 = \{\dot{z}, z, \psi, h\}$, where

- $\dot{z}$ is vertical rate
- z is barometric altitude
- ψ is the heading
- h is height above ground

For each intruding aircraft i, the main sensor inputs are $O_i = \{r_s, \chi, z\}$, where

- r_s is slant range (relative distance to intruder)
- χ is bearing (relative angle to intruder)
- z is altitude

ACAS X System: ACAS X is the system under test. We use an official binary of a prototype obtained from the FAA. The binary exposes the functions INITIALIZE and STEP, but is otherwise a black box. It is known that ACAS X maintains internal state, but the state is not exposed. Although there are many output variables of the ACAS X system, the pilot is mostly concerned with the issued resolution advisory (RA), which instructs the pilots to climb or descend at a particular rate.

ACAS X has a coordination mechanism to ensure that issued RAs do not conflict with each other, such as to recommend two aircraft to maneuver in the same vertical direction. The messages are communicated to all nearby aircraft through coordination codes.

Pilot Model: There are two aspects to modeling the pilot. First is the intended commands, which is what the pilot would have done if there were no conflicts. Second is the pilot response model, which is how the pilot responds to an RA.

For the intended commands, we use the probabilistic model given by the dynamic Bayesian network in LLCEM. Samples are drawn according to the transition probabilities given. For the response model, we use the deterministic pilot response model described in [15]. The model assumes that pilots respond to initial RAs with a 5-second delay then accelerate towards the recommended target rate $\dot{h}_{target}$. Pilots respond to subsequent RAs (i.e., strengthenings and reversals) in the same manner except with a 3-second delay. During the delay period, the pilot continues to execute actions associated with the previous RA. That is, the pilot continues to fly the intended trajectory during initial RA delays, and continues responding to the previous RA during subsequent RA delays. Multiple RAs issued successively are queued so that both their order and timing are maintained. In the case where a subsequent RA is issued within 2 seconds or less of an initial RA, the timing of the subsequent RA is used and the initial RA is skipped. The output of the pilot model is a command given by $\{a, h_d, \psi_d\}$, where

- a is commanded airspeed acceleration
- h_d is commanded vertical rate
- ψ_d is commanded turn rate

Aircraft Dynamics Model: The aircraft dynamics model determines how the state of the aircraft propagates given the pilot commands. The aircraft state is given by $x = \{v, N, E, z, \psi, \theta, \phi\}$, where

- v is airspeed
- N is position north
- E is position east
- z is altitude
- ψ is heading angle
- θ is pitch angle
- ϕ is roll angle

We use forward Euler integration at 1 Hz to propagate the aircraft state.

Reward Function Modification

When using the reward function in eq. 1 to search, all trajectories not meeting the event constraint return a reward of $-\infty$ and so the algorithm cannot distinguish amongst them. As a result, the algorithm is largely just using Monte Carlo to search for event trajectories. In reality, some of these non-event trajectories are much closer to an event occurrence than others. If the closeness can be quantified, we can greatly speed up the algorithm by modifying the reward function to incorporate this information. In particular, instead of returning $-\infty$ when the event does not occur, we return a large negative reward correlated with how "close" the trajectory came to reaching an event. This modification has the effect of making the reward function more gradual and guiding the search to focus on areas near events.

In the case of mid-air encounters and NMACs, an obvious closeness metric is the miss distance D_{miss}, defined as the Euclidean distance between aircraft at their closest point. The miss distance is a good metric because it is monotonically decreasing as trajectories get closer to the event E and is minimum at E. We use this modified reward function for all our ACAS X experiments.

Single Threat Encounters

We performed studies on two-aircraft encounters with initial conditions sampled from the LLCEM. The algorithm searches the space of trajectories starting from the encounter's initial conditions and returns the trajectory with the highest reward found. Although we have crafted the reward function to find NMACs, the

algorithm may or may not have been successful at finding one. A returned trajectory where an NMAC does not occur could mean either that the number of samples used is insufficient, or that one cannot be reached from the initial conditions. With the configuration shown in Table I, NMACs were found in 18% of analyzed encounters. We manually clustered the resulting trajectories and present our findings.

Table I. Single Threat Study Configuration

Simulation	
Number of aircraft	2
Encounter Model	LLCEM
Sensors	Active, Beacon-Based, Noiseless
Collision Avoidance System	ACAS X 0.8.5
Pilot Response Model	ICAO 5s–3s
MCTS	
depth	50
iterations	2000
exploration constant	100.0
k	0.5
α	0.85
k'	1.0
α'	0.0

Crossing Time: A number of NMACs resulted from well-timed vertical maneuvers. In particular, aircraft crossing in altitude during the delay period of an initial RA tends to be problematic. Figure 5 shows one such encounter that eventually ends in an NMAC at 36 seconds. The probability density of this trajectory evaluated using LLCEM is $5.3 \cdot 10^{-18}$. This quantity, which we will refer to as the *likelihood metric*, can be used as a relative measure of how likely is a trajectory to occur. In this encounter, the aircraft cross in altitude during pilot 1's delay period. This results in aircraft 1 starting the climb from below aircraft 2. Unfortunately after this has occurred, there is not enough time to recover using a reversal due to subsequent pilot response delays.

High Turn Rates: Turns, especially those at higher rates, tend to complicate the conflict resolution process by quickly shortening the time to closest approach. ACAS X does not have full state information about its intruder and must estimate it by tracking relative distance, relative angle, and the intruder altitude. Figure 6 shows an example of an encounter that has similar crossing behavior as Figure 5 but exacerbated by the high turn rate of aircraft 2 (approximately

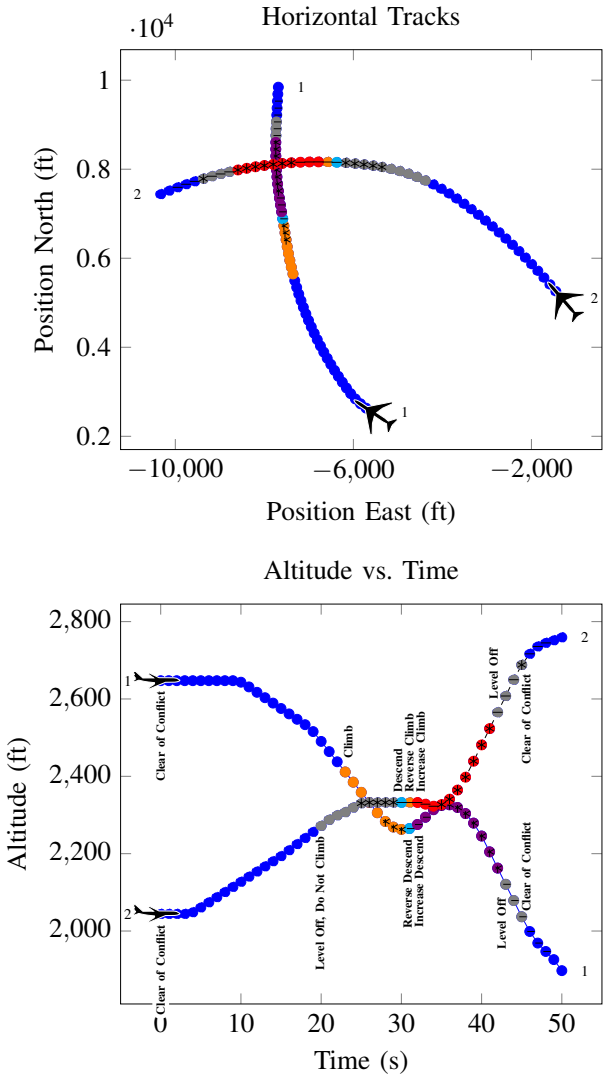

Figure 5. NMAC trajectory where altitude crossing occurs during pilot's initial response delay.

1.5 times the standard turn rate). In this scenario, the aircraft become almost head-on at time of closest approach and a reversal is not attempted. An NMAC with a likelihood metric of $6.5 \cdot 10^{-17}$ occurs at 48 seconds.

Initially Moving Against RA: We found that a number of NMACs were caused by the pilot initially moving against the issued RA before proceeding to fully comply with it. Recall that the pilot is following the intended trajectory during the first 5 seconds of an initial RA, and this intended trajectory may disagree with the RA. In most cases, the disagree-

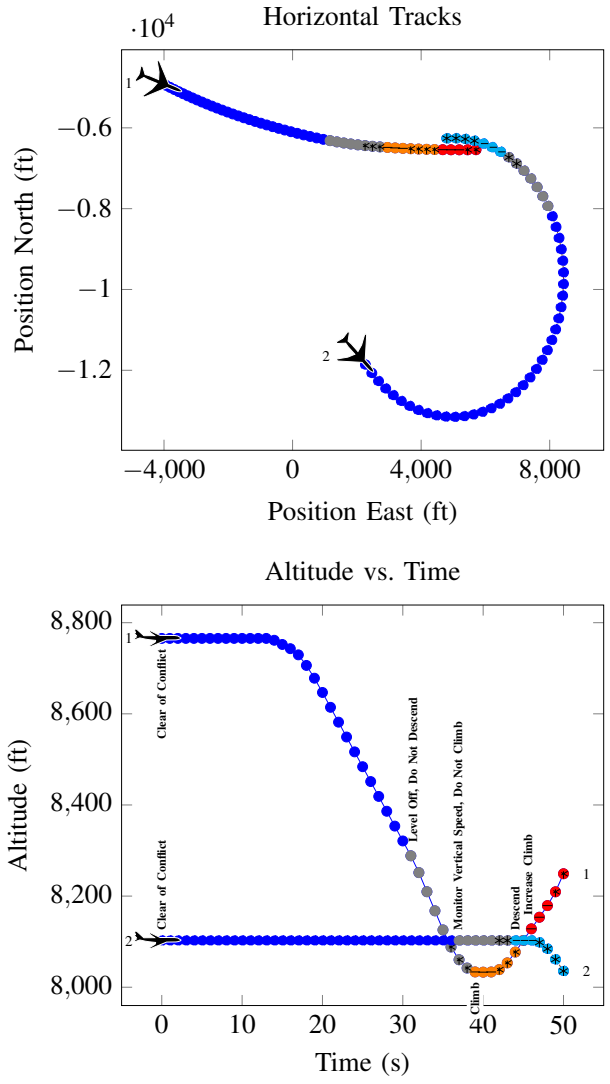

Figure 6. NMAC trajectory where high turn rate plays a leading factor.

ment must be severe to cause an NMAC, where the pilot aggressively accelerates against the RA. Perhaps unsurprisingly, this makes it extremely difficult for the collision avoidance system to resolve the conflict despite full compliance later on.

Sudden Aggressive Accelerations: Sudden maneuvers can lead to NMACs when they are sufficiently aggressive and occurring at just the right time. In particular, we observed some encounters where two aircraft are approaching one another separated in altitude and flying level, then one aircraft suddenly accelerates vertically towards the other aircraft as they are about to pass. Under these circumstances, given the pilot response delays and dynamic limits of the aircraft, the collision avoidance system often does not have enough time to resolve the conflict.

Fortunately, the chances of such maneuvers are exaggerated in our modeling since ACAS X issues traffic alerts (TAs) to alert pilots to nearby traffic. These warnings typically occur well in advance of issued RAs to help pilots from entering into situations where RAs are needed. The pilot response model used in this work does not capture the effect of the TAs.

Combination of Factors: In our experiments, it is very rare for an NMAC to occur due to a single cause. Typically a combination of factors contribute to the eventual NMAC as shown in the example encounter in Figure 5. Although crossing time played a crucial role, there are a number of other factors that are important. The horizontal behavior where they are turning into each other is significant as it reduces the time to closest approach. The two vertical maneuvers of aircraft 1 before receiving an RA are also important. Similar observations can be made on nearly all NMAC encounters found.

Multi-Threat Encounters

We performed studies on three-aircraft encounters with initial conditions sampled from the Star Encounter Model. With the configuration shown in Table II, NMACs were found in 25% of analyzed encounters.

Table II. Multi-Threat Study Configuration

Simulation	
Number of aircraft	3
Encounter Model	Star Model
Sensors	Active, Beacon-Based, Noiseless
Collision Avoidance System	ACAS X 0.8.5
Pilot Response Model	ICAO 5s–3s
MCTS	
depth	50
iterations	1000
exploration constant	100.0
k	0.5
α	0.85
k'	1.0
α'	0.0

Pairwise Coordination in Multi-Threat: Our algorithm discovered a number of NMAC encounters where all aircraft are issued a "multi-threat" RA and asked to follow an identical climb rate. Unfortunately, compliance with the RA results in the aircraft eventually closing horizontally without gaining vertical separation. We show an example of such an encounter in Figure 7 where an NMAC with likelihood metric of $5.8 \cdot 10^{-7}$ occurs at 38 seconds.

In discussing these results with the ACAS X development team, we learned that this behavior is a known issue that can arise from performing multi-aircraft coordination using a pairwise coordination mechanism. The pairwise coordination messages in essence determine which aircraft will climb and which will descend in an encounter. Since coordination messaging occurs pairwise, under rare circumstances it is possible for each aircraft to receive conflicting coordination messages from the other aircraft in the scenario. In normal encounters, the aircraft that receives conflicting coordination messages from two aircraft remains level and lets the other aircraft climb or descend around it. Although uncommon, this is an important case that is being addressed by both TCAS and ACAS X development teams.

Maneuverable Space: In general, multi-threat encounters are more difficult to resolve than pairwise encounters because there is typically less open space for the aircraft to maneuver. Figure 8 shows an example of an NMAC trajectory where aircraft 1 (the aircraft in the middle altitude between 10 and 36 seconds) needs to simultaneously avoid an aircraft below and a vertically closing aircraft from above. An NMAC with a likelihood metric of $1.0 \cdot 10^{-16}$ occurs at 39 seconds. Aircraft 2's downward maneuver greatly reduces aircraft 1's maneuverable airspace. To prevent aircraft 2 from maneuvering downwards would likely require preemptive action by the collision avoidance system occurring much earlier in the encounter. Admittedly, this is an extremely challenging scenario that is somewhat unfair as a test case. Nevertheless, insight can be gained by looking closely at how the collision avoidance system handled the problem. In this case, ACAS X on aircraft 1 decides to issue a crossing RA rather than to remain sandwiched.

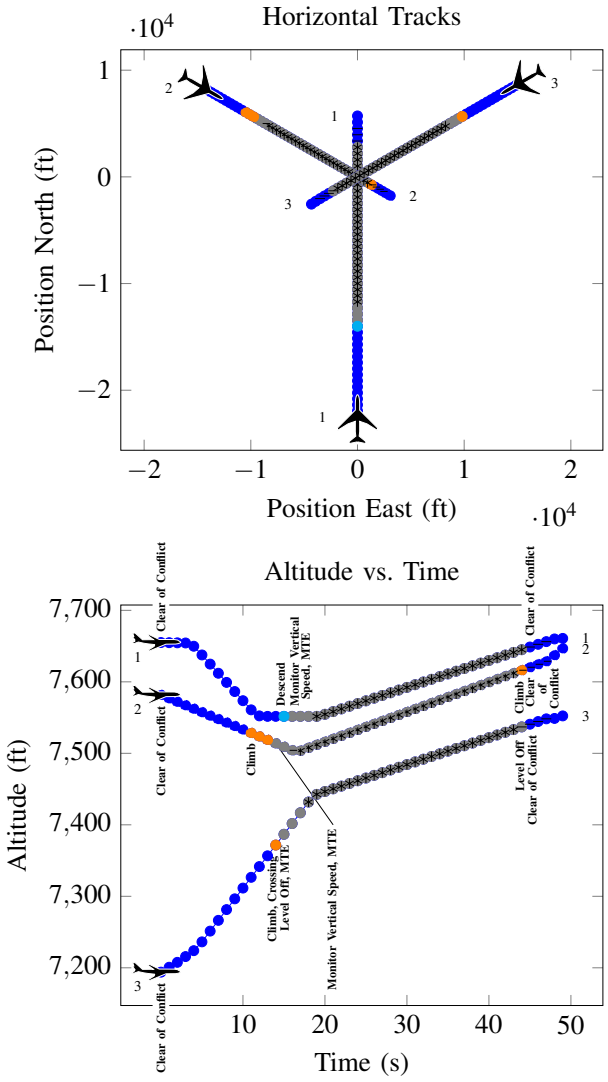

Figure 7. NMAC trajectory where aircraft are in a coordination deadlock.

Pairwise Phenomenon: As one would expect, phenomena that appear in pairwise encounters also appear in multi-threat encounters, albeit typically exacerbated by the presence of the third aircraft. In our multi-threat analysis, we noted similar phenomena related to crossing time, initially moving against the RA, and sudden aggressive accelerations discussed previously. We did not observe the impact of high turn rates in the multi-threat case due to our use of the Star Encounter Model.

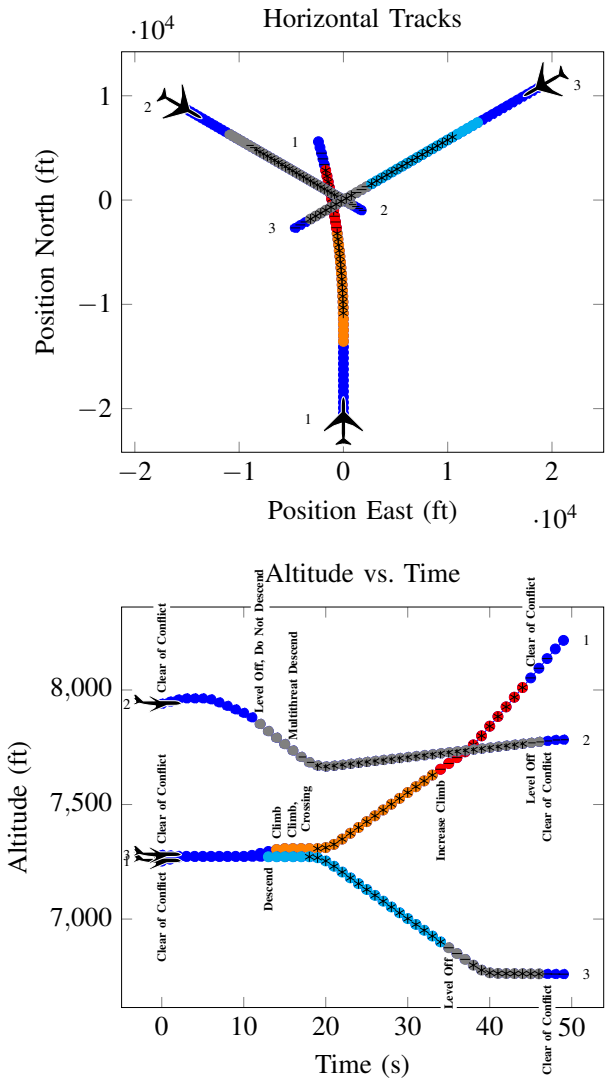

Figure 8. NMAC trajectory where an aircraft must avoid intruders from above and below.

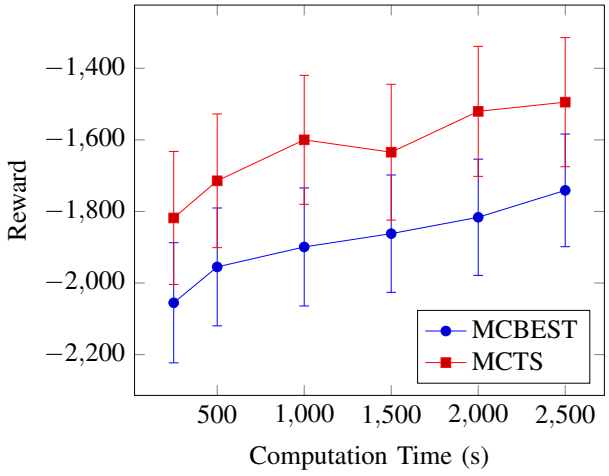

Figure 9. A study of reward vs. computation time

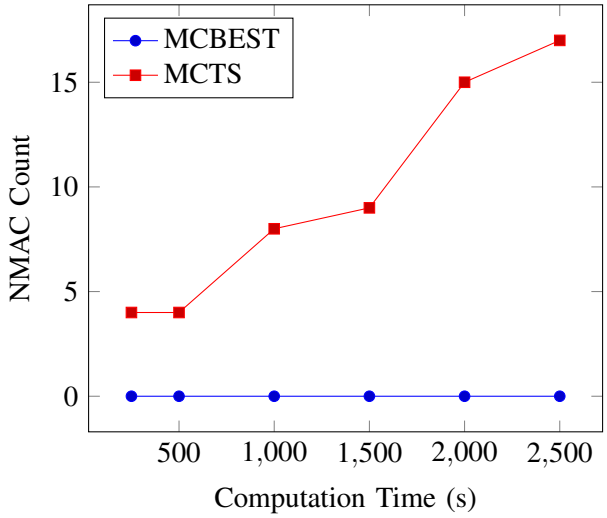

Figure 10. Number of NMACs found vs. computation time

Performance

We compared MCTS against a direct Monte Carlo search algorithm, "MCBEST," given a fixed computational budget. To do this, we make a small modification to the MCTS algorithm to enable precise control of the computation time used. Instead of terminating based on number of iterations as we had previously done, we iterate until the allotted computation time is spent. For MCBEST, we repeatedly draw Monte Carlo samples until the allotted computation time is spent, then return the trajectory with the highest total reward.

Figure 9 compares the two algorithms by the total reward of the resulting trajectory. Each data point reflects 100 pairwise encounters and the mean and standard error of the mean are shown. Figure 10 shows the number of NMACs found out of the 100 encounters searched. In both cases, MCTS clearly performs better than the baseline Monte Carlo search. The effectiveness of MCTS in finding NMACs is of particular importance, and we see that MCTS greatly outperforms direct Monte Carlo search in this regard.

Conclusion and Future Work

Air travel is one of the safest forms of transportation available today due to safety systems like TCAS. ACAS X seeks to make air travel even safer by further reducing collision risk while simultaneously reducing the number of unnecessary alerts. To achieve this, the design, development, and testing of ACAS X all use state-of-the-art techniques.

In this paper, we have proposed a novel approach to stress testing black box systems and demonstrated its utility in stress testing ACAS X. Since the reinforcement learning method is very general, we hope to apply it to a broad range of domains. To facilitate this, we plan to release an open source Julia package of our implementation in the near future.

There are several areas of further work. We would like to leverage information in the problem that is already available but currently not utilized to try to speed up the search. Examples of such information include the distribution of rewards collected from rollouts as well as the state-action value estimates of one action relative to another. Another extension is to leverage the existing search tree to estimate other useful information such as the overall probability of an event. A further interest is in leveraging the search tree to cluster trajectories so that a compact set of representative NMAC examples can be produced.

Acknowledgments

We wish to thank Neal Suchy at the FAA as well as the ACAS X teams at MIT Lincoln Laboratory and Johns Hopkins University Applied Physics Laboratory for their feedback and continued support of the project. We gratefully thank Youngjun Kim and Jonathan Cox for sharing their code with us, and Ryan Gardner for providing valuable feedback on the paper. This work was supported by the Safe and Autonomous Systems Operations (SASO) Project under NASA ARMD's Airspace Operations and Safety Program (AOSP).

Email Addresses

Ritchie Lee, ritchie.lee@sv.cmu.edu
Mykel J. Kochenderfer, mykel@stanford.edu
Ole J. Mengshoel, ole.mengshoel@sv.cmu.edu
Guillaume P. Brat, guillaume.p.brat@nasa.gov
Michael P. Owen, michael.owen@ll.mit.edu

References

[1] J. K. Kuchar and A. C. Drumm, "The traffic alert and collision avoidance system," *Lincoln Laboratory Journal*, vol. 16, no. 2, pp. 277–296, 2007.

[2] A. Barnett, "Cross-national differences in aviation safety records," *Transportation science*, vol. 44, no. 3, pp. 322–332, 2010.

[3] "FAA aerospace forecast fiscal years 2015–2031," Federal Aviation Administration, Tech. Rep., 2014.

[4] M. J. Kochenderfer, J. E. Holland, and J. P. Chryssanthacopoulos, "Next-generation airborne collision avoidance system," *Lincoln Laboratory Journal*, vol. 19, no. 1, pp. 17–33, 2012.

[5] S. D. Whitehead and L.-J. Lin, "Reinforcement learning of non-markov decision processes," *Artificial Intelligence*, vol. 73, no. 1, pp. 271–306, 1995.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[7] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[8] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental Robotics IX*, Springer, 2006, pp. 363–372.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," *ArXiv preprint arXiv:1312.5602*, 2013.

[10] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012. DOI: 10.1109/TCIAIG.2012.2186810.

[11] A. Couëtoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard, "Continuous upper confidence trees," in *Learning and Intelligent Optimization (LION)*, 2011.

[12] M. J. Kochenderfer, L. P. Espindle, J. K. Kuchar, and J. D. Griffith, "Correlated encounter model for cooperative aircraft in the national airspace system," Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-344, 2008.

[13] M. J. Kochenderfer, M. W. M. Edwards, L. P. Espindle, J. K. Kuchar, and J. D. Griffith, "Airspace

encounter models for estimating collision risk," *AIAA Journal on Guidance, Control, and Dynamics*, vol. 33, no. 2, pp. 487–499, 2010. DOI: 10.2514/1.44867.
[14] Y. Kim and M. J. Kochenderfer. (). SISL Encounter Simulator (SISLES.jl), [Online]. Available: http://www.bitbucket.org/sisl/sisles.jl.
[15] International Civil Aviation Organization, "Surveillance, radar and collision avoidance," in *International Standards and Recommended Practices*, 4th, vol. IV, annex 10, Jul. 2007.
[16] C. Von Essen and D. Giannakopoulou, "Analyzing the next generation airborne collision avoidance system," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2014.
[17] M. J. Kochenderfer and J. P. Chryssanthacopoulos, "A decision-theoretic approach to developing robust collision avoidance logic," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2010.
[18] J.-B. Jeannin, K. Ghorbal, Y. Kouskoulas, R. Gardner, A. Schmidt, E. Zawadzki, and A. Platzer, "A formally verified hybrid system for the next-generation airborne collision avoidance system," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2015.
[19] J. E. Holland, M. J. Kochenderfer, and W. A. Olson, "Optimizing the next generation collision avoidance system for safe, suitable, and acceptable operational performance," *Air Traffic Control Quarterly*, vol. 21, no. 3, pp. 275–297, 2013.
[20] B. J. Chludzinski, "Evaluation of TCAS II version 7.1 using the FAA fast-time encounter generator model," Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-346, 2009.