Differential Adaptive Stress Testing of Airborne Collision Avoidance Systems

Ritchie Lee*, Ole J. Mengshoel†
Carnegie Mellon University Silicon Valley, Moffett Field, CA 94035

Anshu Saksena‡, Ryan Gardner§, Daniel Genin¶, Jeffrey Brush‖
Johns Hopkins University Applied Physics Laboratory, Laurel, MD, 20723

Mykel J. Kochenderfer**
Stanford University, Stanford, CA 94305

The next-generation Airborne Collision Avoidance System (ACAS X) is currently being developed and tested to replace the Traffic Alert and Collision Avoidance System (TCAS) as the next international standard for collision avoidance. To validate the safety of the system, stress testing in simulation is one of several approaches for analyzing near mid-air collisions (NMACs). Understanding how NMACs can occur is important for characterizing risk and informing development of the system. Recently, adaptive stress testing (AST) has been proposed as a way to find the most likely path to a failure event. The simulation-based approach accelerates search by formulating stress testing as a sequential decision process then optimizing it using reinforcement learning. The approach has been successfully applied to stress test a prototype of ACAS X in various simulated aircraft encounters. In some applications, we are not as interested in the system's absolute performance as its performance *relative* to another system. Such situations arise, for example, during regression testing or when deciding whether a new system should replace an existing system. In our collision avoidance application, we are interested in finding cases where ACAS X fails but TCAS succeeds in resolving a conflict. Existing approaches do not provide an efficient means to perform this type of analysis. This paper extends the AST approach to differential analysis by searching two simulators simultaneously and maximizing the difference between their outcomes. We call this approach differential adaptive stress testing (DAST). We apply DAST to compare a prototype of ACAS X against TCAS and show examples of encounters found by the algorithm.

Nomenclature

$\bar{S}$	=	simulator, subscripted with simulator number where appropriate
t	=	time
s	=	internal state of the simulator
a	=	actions, controls the stochastic variables of the environment
$\bar{a}$	=	pseudorandom seed, controls simulator's randomness
r	=	reward
p	=	transition probability, subscripted with simulator number where appropriate
e	=	Boolean indicating whether an event occurred, subscripted with simulator number where appropriate
d	=	miss distance, user-defined measure to a failure, subscripted with simulator number where appropriate
T	=	terminal time
$\dot{z}_{own}$	=	vertical velocity of own aircraft

*Research Staff, Silicon Valley Campus, AIAA Member, ritchie.lee@sv.cmu.edu
†Associate Professor, Department of Electrical and Computer Engineering, ole.mengshoel@sv.cmu.edu
‡Senior Research Scientist, Research and Exploratory Development Department, anshu.saksena@jhuapl.edu
§Senior Staff, Asymmetric Operations Sector, ryan.gardner@jhuapl.edu
¶Senior Staff, Asymmetric Operations Sector, daniel.genin@jhuapl.edu
‖Chief Scientist, Critical Infrastructure Protection Group, jeff.brush@jhuapl.edu
**Assistant Professor, Department of Aeronautics and Astronautics, AIAA Associate Fellow, mykel@stanford.edu

I. Introduction

The Traffic Alert and Collision Avoidance System (TCAS) is mandated on all large transport and cargo aircraft in the United States and many countries around the world to help prevent mid-air collisions. Its operation has played a crucial role in the exceptional level of safety in the national airspace [1]. While TCAS has been very successful in the past, and will continue to operate successfully in the near future, studies have revealed fundamental limitations that prevent TCAS from operating effectively in the next generation airspace [1]. To address the growing needs of the national airspace, the Federal Aviation Administration (FAA) has decided to create a new aircraft collision avoidance system. The next-generation Airborne Collision Avoidance System (ACAS X) is currently being developed and tested and promises a number of potential improvements over TCAS including a reduction in collision risk while simultaneously reducing the number of unnecessary alerts [2].

One of the primary safety metrics for airborne collision avoidance systems is the probability of near mid-air collision (NMAC), defined as two aircraft coming closer than 500 feet horizontally and 100 feet vertically. While studies have shown that NMACs are extremely rare, the risk of NMAC cannot be completely eliminated due to factors such as surveillance noise, pilot response delay, and the need for an acceptable alert rate [2]. Studying the situations where NMACs do arise is vital to assessing system risk and informing development of the system.

Recently, an improved stress testing approach has been proposed for finding the most likely path to a failure event. The method, called adaptive stress testing (AST), formulates stress testing as a sequential decision process then uses reinforcement learning to optimize it [3]. The key idea is to define an appropriate reward function that favors the occurrence of failure events and maximization of path probabilities. By equipping the algorithm with a means for evaluative feedback, the reinforcement learner can explore and learn through interactions with the simulator and automatically discover the most important parts of the state space. AST is a black box simulation-based approach that does not rely on knowledge of the internals of the test system. As a result, the method scales well to large complex systems and applications where the test configuration may vary. AST has been previously applied to stress test a prototype of ACAS X in simulation where it successfully found examples of NMAC encounters [3]. NMAC is considered a failure event in the analysis of collision avoidance systems.

While AST can efficiently find examples of poor behavior of a single system, in some applications, it may also be very valuable to identify scenarios of *relatively* poor behavior when compared to some other baseline system. In other words, we are not so interested in cases where both systems perform poorly as cases where the system under test performs poorly but the baseline system performs well. We call this type of analysis *differential analysis*. Such situations arise, for example, during regression testing where a new version of a system is compared to a previous one to identify areas of comparative weakness. In the case of ACAS X, we compare the performance of ACAS X to TCAS to decide whether an in situ system should be replaced by another next generation system.

One way to compare the behavior of two collision avoidance systems is to evaluate them against a common set of encounters. These encounters, for example, can be randomly generated according to a stochastic model. However, the undirected nature of this approach can be very inefficient due to the size and complexity of the state space, and the rarity of failure events. Moreover, the method does not find the most likely path to a failure, which is very valuable in the analysis of NMAC.

We propose a novel stress testing method, called differential adaptive stress testing (DAST), for finding the most likely path to a failure event that occurs in the system under test, but not in the baseline system. DAST extends the AST framework to perform comparative analysis retaining its desirable properties, including scalability, efficiency, and support for black box systems. DAST searches two simulators simultaneously and maximizes the difference in the outcomes of the simulators. Optimization is performed using reinforcement learning. We apply DAST to stress test a prototype of ACAS X where behavior is evaluated relative to a TCAS baseline. We find examples of encounters where an NMAC occurs when using one system, but not when using the other.

The remaining sections are organized as follows. Section II reviews related studies of ACAS X and NMACs. Section III provides background information on ACAS X and AST. Section IV presents the general DAST framework. Section V presents the experiments and results of applying DAST to analyze NMACs.

II. Related Work

A variety of approaches have been applied to the analysis of aircraft collision avoidance systems. Prior studies have modeled aircraft encounters as discrete-time discrete-state Markov chains then used probabilistic model checking (PMC) to evaluate safety properties on the model [4][5]. PMC provides complete evaluations of the properties and their

probabilities of occurrence. Other studies have used hybrid systems theorem proving (HSTP), which models encounters as hybrid systems with idealized advisories and continuous dynamics then formally proves whether the system always avoids collision or whether a collision is possible [6][7]. The advice of core components of ACAS X is then compared to that of the idealized system in different states to identify cases where the advice given by core ACAS X components may not avoid NMAC. PMC and HSTP are formal methods. They evaluate properties over the entire state space and thus provide completeness guarantees. However, they have difficulty scaling to the size and complexity of systems like ACAS X and the aircraft encounter environment. Consequently, the studies focus on core components of ACAS X and generally use simpler models of aircraft dynamics and pilot behavior.

Simulation-based methods use a simulation model and evaluate the system at a finite number of simulation paths. These methods treat the simulator as a black box and thus do not require the internal details of the models used. As a result, these methods can handle larger and higher-fidelity models. Prior work has analyzed TCAS by sweeping through a low-dimensional parametric model of head-on encounters and simulating the collision avoidance system [8]. An alternative approach is to run simulations drawn from a statistical representation of the airspace [9]. Holland et al. simulated large numbers of encounters, including tracks from recorded flight data, to evaluate the performance of ACAS X [10]. Simulation-based methods are generally easier to implement than formal methods because simulation models of real systems are usually simpler to build than the models used in formal methods.

Simulation-based methods can be used to compare two collision avoidance systems by simulating the two systems with a common set of encounters then comparing their behaviors. The set of encounters can, for example, be generated by Monte Carlo sampling a stochastic model of the airspace [9]. While this method can find significant differences in behaviors, it can be very inefficient due to the size and complexity of the state space, rarity of NMAC, and undirected nature of Monte Carlo sampling. Furthermore, the method does not maximize the probability of reaching a failure event.

In contrast, our approach formulates the problem as a sequential decision process and uses reinforcement learning to explicitly optimize the difference in behaviors and the path probabilities. The result is a focused and adaptive search. DAST offers a unique set of trade-offs in analysis capability and is part of an arsenal of tools the FAA uses in the safety validation of ACAS X.

III. Background

A. Airborne Collision Avoidance Systems

There are several versions of ACAS X under development. In this paper, we refer to ACAS Xa version 0.10.3, which is designed to be a direct replacement to TCAS. Despite the internal logics of ACAS X and TCAS being derived completely differently, the input and output interfaces of these two systems are identical. Consequently, the following description of aircraft collision avoidance systems applies to both ACAS X and TCAS.

Airborne collision avoidance systems monitor the airspace around an aircraft and issue alerts to the pilot if a conflict with another aircraft is detected. These alerts, called resolution advisorys (RAs), instruct the pilot to maneuver the aircraft to a certain target vertical velocity and maintain it. The advisories are typically issued when the aircraft are within approximately 20-40 seconds to a potential collision. Table 1 lists the possible primary RAs. We use $\dot{z}_{own}$ to denote the current vertical velocity of own aircraft.

Table 1 Primary ACAS X Advisories

Abbreviation	Description	Rate to Maintain (ft/min)
COC	clear of conflict	N/A
DND	do not descend	0
DNC	do not climb	0
MAINTAIN	maintain current rate	$\dot{z}_{own}$
DS1500	descend at 1,500 ft/min	−1500
CL1500	climb at 1,500 ft/min	+1500
DS2500	descend at 2,500 ft/min	−2500
CL2500	climb at 2,500 ft/min	+2500

The COC advisory stands for "clear of conflict" and is equivalent to no advisory. The pilot is free to choose how to control the aircraft. The DND and DNC advisories stand for "do not descend" and "do not climb", respectively. They restrict the pilot from flying in a certain direction. The MAINTAIN advisory is preventative and instructs the pilot to maintain the current vertical rate of the aircraft. The advisories DS1500 and CL1500 instruct the pilot to descend or climb at 1,500 feet per minute. The pilot is expected to maneuver the aircraft at $\frac{1}{4}g$ acceleration until the target vertical rate is reached then maintain that vertical rate. The DS2500 and CL2500 advisories instruct the pilot to descend or climb at an increased rate of 2,500 feet per minute. These advisories are strengthened advisories and expect a stronger response from the pilot. For these strengthened RAs, the pilot is expected to maneuver at $\frac{1}{3}g$ acceleration until the target vertical rate is reached then maintain that vertical rate. Strengthened RAs must follow a weaker RA of the same vertical direction. They cannot be issued directly. For example, a CL1500 advisory must precede a CL2500 advisory. Advisories issued by collision avoidance systems on different aircraft are not completely independent. When an RA is issued, a coordination message is broadcasted to other nearby aircraft to prevent other collision avoidance systems from accidentally issuing an RA in the same vertical direction.

B. Adaptive Stress Testing

Adaptive stress testing (AST) is a recently proposed approach to stress testing that formulates it as a sequential decision process then uses reinforcement learning to search for the most likely path to a failure event [3]. The key idea is to define a reward function to give evaluative feedback to the learner, so that it can automatically discover important parts of the state space on its own. Figure 1 shows a conceptual overview of the AST framework. The system under test is placed in a simulator where it interacts with a simulated environment. The reinforcement learner interacts with the simulator over multiple time steps to maximize the reward it receives. To do so, the learner manipulates the stochastic elements of the environment to trigger a failure event and also maximize its path probability. In other words, the reinforcement learner is searching for a sequence of the environment variables that is most adversarial to the system under test. The result of the optimization is the most likely sequence of states that results in a failure event.

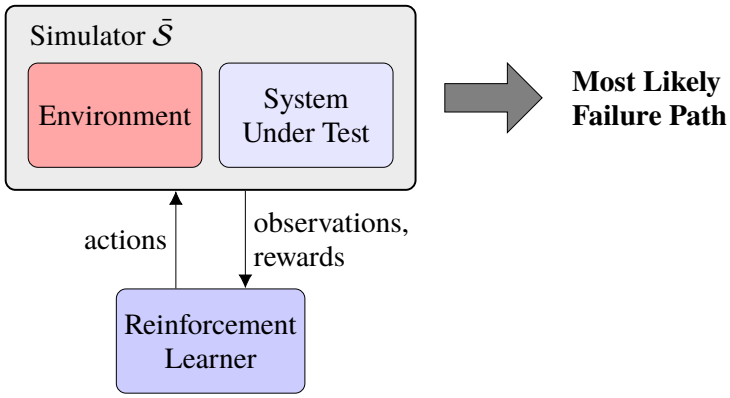

Fig. 1 Adaptive Stress Testing

In cases where the state of the system cannot be fully observed, a modified Monte Carlo tree search (MCTS) algorithm can be used [3]. The algorithm does not require the state of the system to be explicitly represented. Instead, it assumes control of the pseudorandom seed of the simulator, which allows it to be able to deterministically return the simulator to a previously-visited state by replaying a sequence of seeds from the initial state.

Figure 2 illustrates the AST framework, where pseudorandom seeds are used as the simulator input. The simulator $\bar{\mathcal{S}}$ maintains state internally but does not expose it. The seed $\bar{a}$ is used to seed the pseudorandom number generation process of the simulator, where, internally, a sample of the environment's stochastic variables and state transition is drawn. The state of the simulator is updated in-place with the new state. The seed renders the entire sampling process deterministic, which allows deterministic replay of the simulator. The simulator outputs the transition probability p, which is the product of the probability of sampling a set of environment variables $P(a \mid s)$ and the probability of the state transition $P(s' \mid s, a)$; a Boolean indicating whether an event occurred e; and the miss distance d, which is some measure defined by the user that is indicative of how close the simulation came to a failure. If such a measure is not available, then a constant may be used. However, providing a good miss distance can greatly accelerate the search by giving the reinforcement learner the ability to distinguish the desirability of two paths that do not contain failure events.

The reward function translates the simulator outputs into a reward. Finally, the MCTS algorithm maximizes the rewards received by controlling the seed $\bar{a}$.

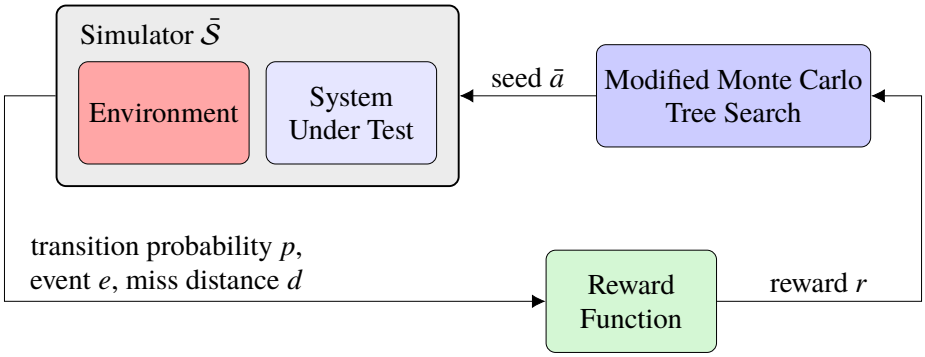

Fig. 2 Adaptive Stress Testing of a Partially Observable System

The reward function is designed to find failure events as the primary objective and to maximize the path probability as a secondary objective. To this end, we define three terms in the reward function as shown in Equation 1.

$$R(s, a, s') = \begin{cases} R_E & \text{if } s \in E \\ -d & \text{if } s \notin E,\ t \geq T \\ \log(P(s' \mid s, a)P(a \mid s)) & \text{if } s \notin E,\ t < T \end{cases} \tag{1}$$

The first term assigns a non-negative reward R_E if the path terminates and a failure event occurs. If the path terminates and an event does not occur, the second term penalizes the learner by assigning the negative of the miss distance d to the learner. We use the shorthand notation $s \notin E$ to denote that no event has occurred and $t \geq T$ to denote that the simulation has terminated. The third term maximizes the overall path probability by awarding the log probability of each transition. The transition probability p is given by the product of the probability of sampling the environment variables $P(a \mid s)$ and the probability of transitioning to the new state $P(s' \mid s, a)$. The probability of choosing the environment variables a, in general, depends on the current state s. Recall that reinforcement learning maximizes the sum of expected rewards. By choosing a reward of the log probability at each step, the reinforcement learning algorithm maximizes the sum of the log probabilities, which is equivalent to maximizing the product of the probabilities.

IV. Differential Adaptive Stress Testing

We introduce differential adaptive stress testing (DAST), a stress testing method for efficiently finding the most likely path to a failure event that occurs in the system under test, but not in the baseline. The key idea behind DAST is to drive two simulation systems simultaneously and maximize the difference in their outcomes. DAST extends the AST framework to differential analysis. Following the AST approach, DAST formulates stress testing as a sequential decision process and optimizes it using reinforcement learning. We use the same modified MCTS algorithm for optimization, which is appropriate for partially observable systems [3].

At a high level, we create two instances of the simulator $\bar{S}_1$ and $\bar{S}_2$. The instances are identical except that $\bar{S}_1$ contains the system under test, while $\bar{S}_2$ contains the baseline system. In particular, they contain identical models of the environment with which the test systems interface. The simulators are driven by the same pseudorandom seed input, which leads to the same sequence of stochastic variables being drawn from the environment when the behaviors of the test and baseline systems match. When the behavior of the two systems diverge, the seed automatically allows different environment variables to be drawn from each simulator following their diverging states.

Figure 3 illustrates the DAST framework. We define a combined simulator that contains the two parallel simulators $\bar{S}_1$ and $\bar{S}_2$. They are both driven by the same input seed $\bar{a}$. Each simulator produces its own set of outputs, which include the transition probability p, an indicator of whether an event occurred e, and the miss distance d. These variables are combined in a reward function, where a single reward is provided to the reinforcement learner. Finally, a modified MCTS algorithm chooses the seed to maximize the reward it receives.

The reward function combines the output from the two individual simulators to produce a single reward for the learner. The primary objective of the reward function is to maximize the difference in outcomes of the simulators

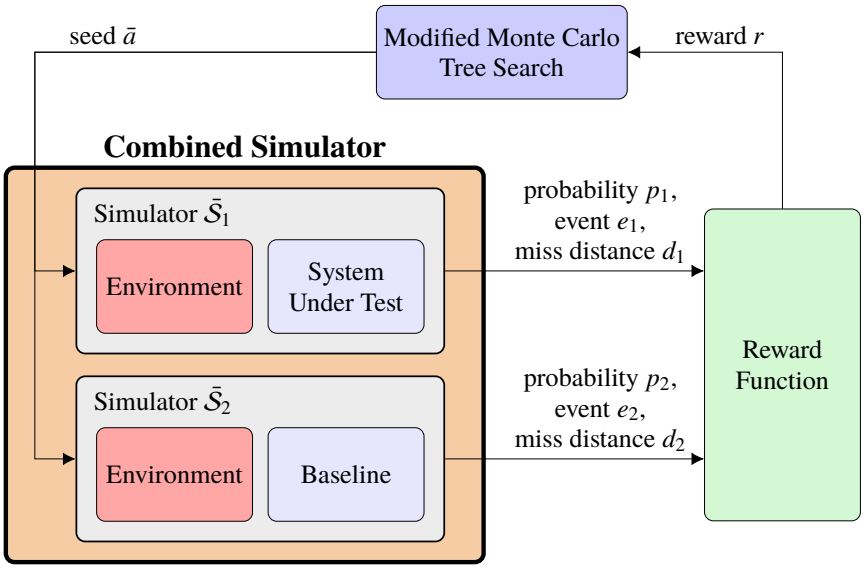

Fig. 3 Differential Adaptive Stress Testing Framework

driving the first simulator to a failure event, while keeping the second simulator away from one. The secondary objective is to maximize the path probabilities of the two simulators to produce the most likely paths. The DAST reward function is given by Equation 2. The indicator function $\mathbb{1}\{x\}$ returns 1 if x is true and 0 otherwise.

$$
\begin{aligned}
R(p_1, e_1, d_1, p_2, e_2, d_2) = & \\
& + R_E && \cdot \mathbb{1}\{e_1\} \\
& - d_1 && \cdot \mathbb{1}\{\neg e_1 \wedge (t \geq T)\} \\
& - R_E && \cdot \mathbb{1}\{e_2\} \\
& + d_2 && \cdot \mathbb{1}\{\neg e_2 \wedge (t \geq T)\} \\
& + (\log p_1 + \log p_2) && \cdot \mathbb{1}\{\neg e_1 \wedge \neg e_2 \wedge (t < T)\}
\end{aligned} \tag{2}
$$

The DAST reward function extends the AST reward function to the differential setting and has a similar structure. The first term gives a non-negative reward R_E to the learner if the first simulator $\bar{\mathcal{S}}_1$ terminates in an event. If $\bar{\mathcal{S}}_1$ terminates and an event did not occur, then the second term penalizes the learner by giving the negative miss distance $-d_1$. The third and fourth terms are the negations of the first and second terms, respectively, applied to the second simulator $\bar{\mathcal{S}}_2$. The third term gives $-R_E$ if $\bar{\mathcal{S}}_2$ terminates in an event and the fourth term gives d_2 if $\bar{\mathcal{S}}_2$ terminates and an event did not occur. To maximize the probabilities of the paths, the fifth term gives the sum of the log transition probabilities of both simulators. The terminal state of the simulators are treated as *absorbing*. That is, once a simulator enters a terminal state, it stays there for all subsequent transitions and collects zero reward for these transitions. We use $t \geq T$ to denote that the combined simulator has terminated.

We optimize the reward function using the same modified MCTS algorithm used in AST, which applies to partially observable systems [3]. The pseudorandom seed interface used in the approach conveniently abstracts the internal details of the simulator from the reinforcement learner. As a result, no modifications to the algorithm are necessary.

V. Aircraft Collision Avoidance Application

We apply DAST to stress test a prototype of ACAS X in an aircraft encounter simulator using TCAS as a baseline. In particular, we search for the most likely scenarios where ACAS X results in an NMAC, but TCAS does not.

A. Experimental Setup

We use the DAST framework shown in Figure 3. The details of the components are described below.

1. Simulation Model

We create a combined simulator that contains two instances of the same aircraft encounter simulator except that one simulator uses ACAS X and the other uses TCAS. Figure 4 shows a system diagram of the base simulator, which contains models of a mid-air encounter involving two aircraft. The simulator is organized into two loops, one for each aircraft. The states of the aircraft are initialized according to a probabilistic model. The sensor models how the collision avoidance system perceives the world. The sensor produces (potentially noisy) sensor measurements to the collision avoidance system. Based on these measurements, the collision avoidance system may issue an RA to the pilot. When there is no advisory, the pilot commands the aircraft based on his intended flight trajectory. When there is an advisory, the pilot responds to it according to a response model. The pilot commands, together with the dynamics of the aircraft, determine the state of the aircraft at the next time step.

Initial State. The initial state of the encounter includes initial positions, velocities, and headings of the aircraft. The initial state is drawn from a distribution that gives realistic initial configurations of aircraft that are likely to lead to NMAC. Our experiments initialize encounters using samples from the Lincoln Laboratory Correlated Aircraft Encounter Model (LLCEM) [11]. LLCEM is a statistical model of the airspace learned from a large body of radar data of the entire national airspace. We follow the encounter generation procedure described in the paper [11].

Sensor Model. The sensor model captures how the collision avoidance system perceives the world. We assume active, beacon-based radar capability with no noise. For own aircraft, the sensor measures the vertical rate, barometric altitude, heading, and height above ground. For each intruding aircraft, the sensor measures slant range (relative distance to intruder), bearing (relative horizontal angle to intruder), and relative altitude.

Collision Avoidance System. The collision avoidance system is the main component of interest. The system under test is a prototype of ACAS X that is a binary library obtained from the FAA. The binary has a minimal interface that allows initializing and stepping the state forward in time. The system maintains internal state, but does not expose it. The primary output of the ACAS X system is the RA. ACAS X has a coordination mechanism to ensure that issued RAs from different aircraft are compatible with one another, i.e., that two aircraft are not instructed to maneuver in the same vertical direction. The messages are communicated to all nearby aircraft through coordination messages. The baseline system is a software implementation of TCAS that is also a binary library obtained from the FAA. Both binaries have identical input and output interfaces making them interchangeable in the simulator.

Pilot Model. The pilot model consists of a model for the pilot's intent and a model for how the pilot responds to an RA. The pilot's intent is how the pilot would fly the aircraft if there were no RAs. To model intended commands, we use the pilot command model in LLCEM, which gives a realistic stochastic model of aircraft commands in the airspace [11]. The pilot response model defines how pilots respond to an issued RA. Pilots are assumed to respond deterministically to an RA with perfect compliance after a fixed delay [12]. Pilots respond to initial RAs with a five-second delay and subsequent RAs (i.e., strengthenings and reversals) with a three-second delay. During the initial delay period, the pilot continues to fly intended trajectory. During response delays from subsequent RAs, the pilot continues responding to the previous RA. Multiple RAs issued successively are queued so that both their order and timing are preserved. 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 pilot command includes commanded airspeed acceleration, commanded vertical rate, and commanded turn rate.

Aircraft Dynamics Model. The aircraft dynamics model determines how the state of the aircraft propagates with time. The aircraft state includes the airspeed, position north, position east, altitude, roll angle, pitch angle, and heading angle. The aircraft state is propagated forward at 1 Hz using forward Euler integration.

2. Reward Function

We use the reward function given by Equation 2 with one exception. We include an additional term that strongly penalizes encounter paths where a pilot accelerates heavily against an issued advisory. The ACAS X team is aware of this degenerate NMAC case and is seeking other, more interesting scenarios. As a result, we include the penalty term to discourage the reinforcement learner from considering these scenarios. The event e_i indicates whether an NMAC has occurred in simulator $\bar{\mathcal{S}}_i$. An NMAC occurs when two aircraft are closer than 100 feet vertically and 500 feet horizontally. The miss distance d_i is the distance of closest approach in simulator $\bar{\mathcal{S}}_i$, which is the Euclidean distance of the aircraft at their closest point in the encounter. The distance of closest approach is a good metric because it is monotonically decreasing as trajectories get closer and reach a minimum at an NMAC. Because the models in the simulator use continuous distributions, we use the transition probability density instead of the transition probability in the reward function and denote it by p_i for simulator $\bar{\mathcal{S}}_i$.

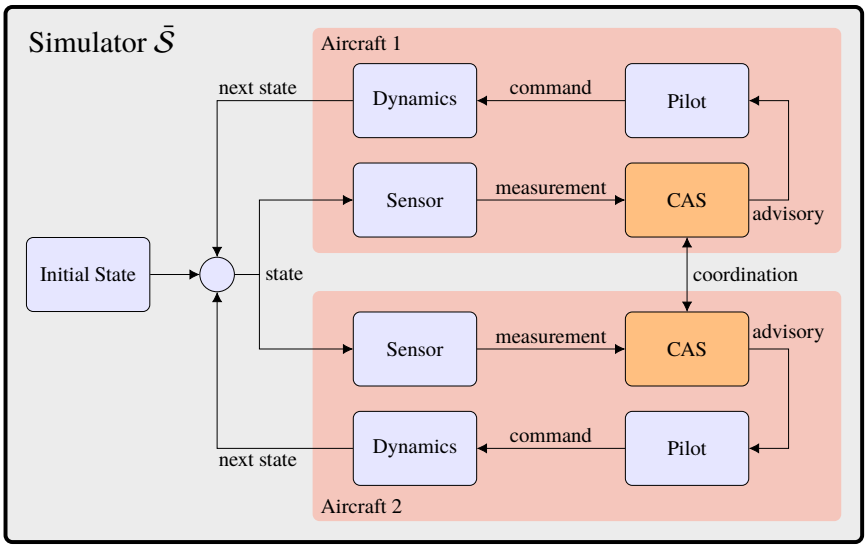

Fig. 4 Aircraft Encounter Simulator

3. Optimization

We use the same modified MCTS algorithm in AST to search our combined simulator [3]. Table 2 shows the parameters used for the study.

Table 2 MCTS Parameters

Parameter	Value
maximum steps	50
iterations	3000
exploration constant	100.0
k	0.5
α	0.85
k'	1
α'	0

We make one modification to the algorithm so that additional NMAC examples can be returned. Instead of returning only the single path with the highest reward from each search, we return the top 10 paths. Then, we check whether these paths contain an NMAC and report the ones that do.

B. Results

We searched 2700 pairwise encounters initialized with samples from LLCEM. Each encounter took three hours to search on a single processor, which is double the computation time of the original AST due to running two simulators in parallel. The top 10 highest reward paths were returned for each encounter initialization producing a total of 27,000 paths. Of these paths, a total of 28 contained NMACs, which originated from 10 encounter initializations. We analyzed the scenarios and found that they represented operationally rare scenarios. We present examples of NMAC found by the algorithm and discuss their properties.

ACAS X issues RA, but TCAS does not. We found some NMAC cases where ACAS X issued an RA but TCAS did not. An example is shown in Figure 5 where an NMAC occurs at 40 seconds in the ACAS X simulation but no NMAC occurs in the TCAS simulation. No RA was issued in the TCAS simulation. In the example, both aircraft are traveling at a high absolute vertical rate exceeding 70 feet per second toward each other. Both aircraft receive an RA to level-off but the aircraft cannot respond in time due to the high vertical rates and the pilot response delay. The aircraft

proceed to cross in altitude. After crossing, ACAS X increases the strength of the advisory in the same direction as the previous RA. Since the aircraft have crossed in altitude, responding to the RA results in a loss of vertical separation and an NMAC.

High vertical rates are known to make conflict resolution more difficult, especially for vertical-only collision avoidance systems like TCAS and this version of ACAS X. Aircraft with high vertical rates take longer to reverse direction vertically and more vertical distance is traveled during the pilot's response delay. As a result, advisories take longer to take full effect. Moreover, in cases where both aircraft are maneuvering in the same vertical direction, an aircraft may lose the ability to "outrun" the other aircraft. For example, even if a maximal climb advisory is issued, it may not be sufficient for the aircraft to stay above a second aircraft climbing at an even higher rate. Another interesting feature of this encounter is the horizontal behavior. Aircraft 2 is initially turning away from the other aircraft before turning towards it at 8 seconds into the encounter. The initial RA is issued shortly after that maneuver. Large rapid changes in turn rate around the time of an RA can make it difficult for a collision avoidance system to accurately estimate the time to horizontal intersection.

Overall, ACAS X is much safer and more operationally suitable than TCAS. However, there are some trade-offs between the two systems. ACAS X's late altering characteristic, which reduces the number of unnecessary alerts, can sometimes hurt encounters with higher vertical rates, such as seen in this example. In deciding the trade-off, a designer must weigh the relative likelihood of these encounters versus the effect on other more frequently observed trajectories.

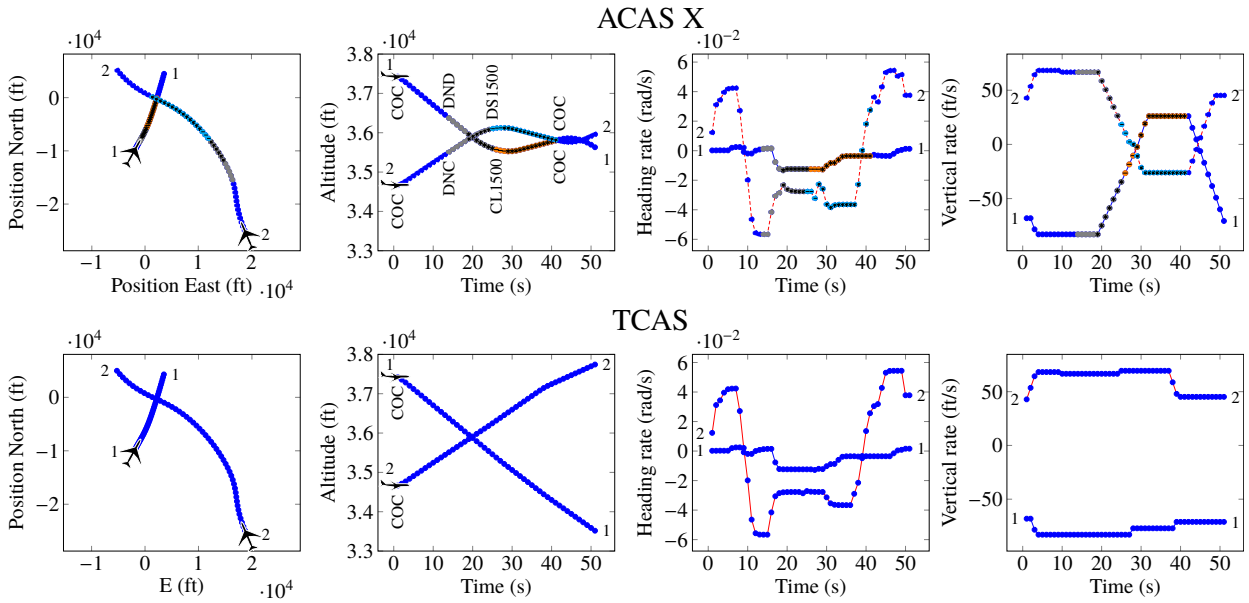

Fig. 5 An NMAC where ACAS X issues an RA but TCAS does not

Simultaneous horizontal and vertical maneuvering. Many of the NMAC encounters found involve an aircraft turning while simultaneously climbing or descending very rapidly. Figure 6 shows an example where an NMAC occurs at 45 seconds in the ACAS X simulation but no NMAC occurs in the TCAS simulation. In this example, aircraft 1 is flying generally straight and level while aircraft 2 is simultaneously turning and climbing at a vertical rate exceeding 80 feet per second. Aircraft 2 receives an RA to level-off but is unable to maneuver in time before losing vertical separation resulting in an NMAC. In this example, TCAS is able to resolve the conflict by issuing RAs to both aircraft earlier in the encounter.

Horizontally, aircraft 2's turn quickly shortens the time to closest approach between the two aircraft. Vertically, aircraft 2 receives a level-off advisory but the high climb rate and pilot response delay limit how quickly the aircraft can be brought to compliance. Scenarios that involve turning and simultaneously climbing or descending at high rate are operationally very rare.

Horizontal maneuvering. NMACs can also result from horizontal maneuvering alone. An example is shown in Figure 7 where an NMAC occurs at 40 seconds in the ACAS X simulation but no NMAC occurs in the TCAS simulation. The aircraft are initially headed away from each other but they are also turning towards each other. At 25 seconds into the encounter, the aircraft turn more tightly towards each other rapidly reducing the time to closest approach. Crossing

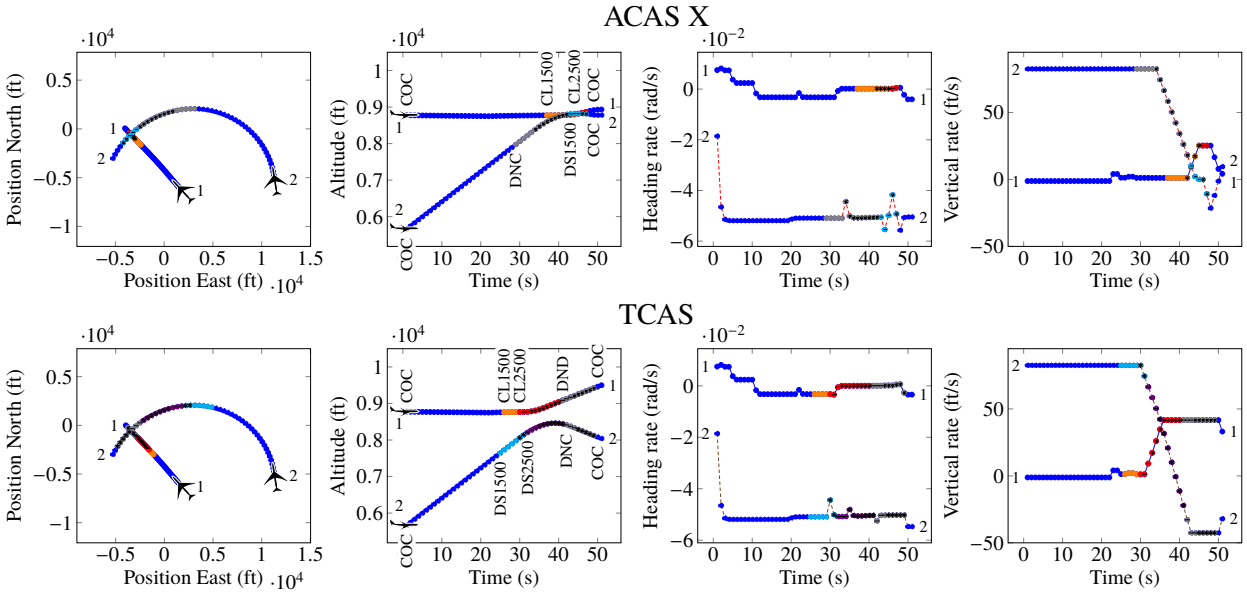

Fig. 6 An NMAC where an aircraft turns while simultaneously climbing very rapidly

advisories are issued to the aircraft 9 seconds prior to NMAC. However, there is not enough time remaining to cross safely and an NMAC occurs. In this example, TCAS is able to resolve the conflict by issuing RAs to both aircraft earlier in the encounter. Operationally, it is unclear how the aircraft got to their initial positions in the encounter.

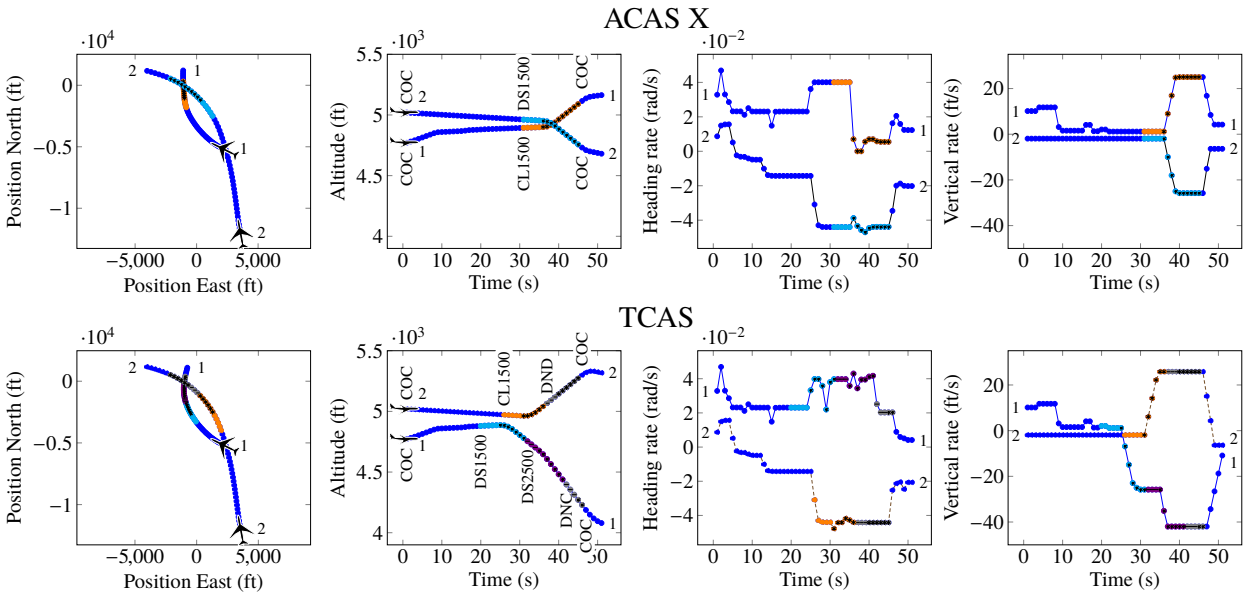

Fig. 7 An NMAC where aircraft maneuver horizontally only

VI. Conclusion

This paper presents an approach, called differential adaptive stress testing (DAST), for finding scenarios where a system under test performs poorly compared to a baseline. In particular, the method finds the most likely path to a failure event that occurs in the system under test, but not in the baseline. Two simulators, one containing the system under test and one containing the baseline, are searched in parallel, using reinforcement learning to maximize the difference in their

outcomes. We applied DAST to stress test a prototype of ACAS X in a medium-fidelity aircraft encounter simulator using TCAS as a baseline. DAST successfully found a number of interesting NMAC encounters. Our implementation of DAST uses the AST Julia package, which is available at `http://github.com/sisl/AdaptiveStressTesting.jl`.

Acknowledgments

We thank Neal Suchy at the Federal Aviation Administration; Michael Owen, Robert Klaus, and Cindy McLain at MIT Lincoln Laboratory; Joshua Silbermann and Rachel Szczesiul at Johns Hopkins Applied Physics Laboratory; and others in the ACAS X team. We thank Guillaume Brat at NASA and Corina Pasareanu at Carnegie Mellon University for their invaluable feedback. This work was supported by the Safe and Autonomous Systems Operations (SASO) Project under NASA Aeronautics Research Mission Dictoriate (ARMD) Airspace Operations and Safety Program (AOSP); and also supported by the Federal Aviation Administration (FAA) Traffic-Alert & Collision Avoidance System (TCAS) Program Office (PO) AJM-233, Volpe National Transportation Systems Center Contract No. DTRT5715D30011.

References

[1] Kuchar, J. K., and Drumm, A. C., "The Traffic Alert and Collision Avoidance System," *Lincoln Laboratory Journal*, Vol. 16, No. 2, 2007, pp. 277–296.

[2] Kochenderfer, M. J., Holland, J. E., and Chryssanthacopoulos, J. P., "Next-Generation Airborne Collision Avoidance System," *Lincoln Laboratory Journal*, Vol. 19, No. 1, 2012, pp. 17–33.

[3] Lee, R., Kochenderfer, M. J., Mengshoel, O. J., Brat, G. P., and Owen, M. P., "Adaptive Stress Testing of Airborne Collision Avoidance Systems," *Digital Avionics Systems Conference (DASC)*, Prague, Czech Republic, 2015.

[4] von Essen, C., and Giannakopoulou, D., "Probabilistic verification and synthesis of the next generation airborne collision avoidance system," *STTT*, Vol. 18, No. 2, 2016, pp. 227–243.

[5] Gardner, R. W., Genin, D., McDowell, R., Rouff, C., Saksena, A., and Schmidt, A., "Probabilistic Model Checking of the Next-Generation Airborne Collision Avoidance System," *Digital Avionics Systems Conference (DASC)*, Sacremento, CA, 2016.

[6] Jeannin, J.-B., Ghorbal, K., Kouskoulas, Y., Gardner, R., Schmidt, A., Zawadzki, E., and Platzer, A., "A Formally Verified Hybrid System for the Next-Generation Airborne Collision Avoidance System," *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2015.

[7] Kouskoulas, Y., Genin, D., Schmidt, A., and Jeannin, J., "Formally Verified Safe Vertical Maneuvers for Non-deterministic, Accelerating Aircraft Dynamics," *Interactive Theorem Proving - 8th International Conference, ITP 2017, Brasília, Brazil, September 26-29, 2017, Proceedings*, 2017, pp. 336–353.

[8] Chludzinski, B. J., "Evaluation of TCAS II Version 7.1 Using the FAA Fast-Time Encounter Generator Model," Project Report ATC-346, Massachusetts Institute of Technology, Lincoln Laboratory, 2009.

[9] Kochenderfer, M. J., and Chryssanthacopoulos, J. P., "A decision-theoretic approach to developing robust collision avoidance logic," *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2010, pp. 1837–1842.

[10] Holland, J. E., Kochenderfer, M. J., and Olson, W. A., "Optimizing the Next Generation Collision Avoidance System for Safe, Suitable, and Acceptable Operational Performance," *Air Traffic Control Quarterly*, Vol. 21, No. 3, 2013, pp. 275–297.

[11] Kochenderfer, M. J., Espindle, L. P., Kuchar, J. K., and Griffith, J. D., "Correlated Encounter Model for Cooperative Aircraft in the National Airspace System," Project Report ATC-344, Massachusetts Institute of Technology, Lincoln Laboratory, 2008.

[12] International Civil Aviation Organization, "Surveillance, radar and collision avoidance," *International Standards and Recommended Practices*, Vol. IV, annex 10, 2007, 4^{th} ed.