Airborne Collision Avoidance in Mixed Equipage Environments

by

Dylan M. Asmar

B.S. Astronautical Engineering
B.S. Mathematical Sciences
United States Air Force Academy (2011)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author .....
Department of Aeronautics and Astronautics
May 1, 2013

Certified by .....
Jonathan P. How
Richard C. Maclaurin Professor or Aeronautics and Astronautics
Thesis Supervisor

Certified by .....
Mykel J. Kochenderfer
Technical Staff, MIT Lincoln Laboratory
Thesis Supervisor

Accepted by .....
Eytan H. Modiano
Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Disclaimer: The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

Airborne Collision Avoidance in

Mixed Equipage Environments

by

Dylan M. Asmar

Submitted to the Department of Aeronautics and Astronautics
on May 1, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Over the past few years, research has focused on the use of a computational method known as dynamic programming for producing an optimized decision logic for airborne collision avoidance. There have been a series of technical reports, conference papers, and journal articles summarizing the research, but they have primarily investigated two-aircraft encounters with only one aircraft equipped with a collision avoidance system.

This thesis looks at recent research on coordination, interoperability, and multiple-threat encounters. In situations where an aircraft encounters another aircraft with a collision avoidance system, it is important that the resolution advisories provided to the pilots be coordinated so that both aircraft are not instructed to maneuver in the same direction. Interoperability is a related consideration since new collision avoidance systems will be occupying the same airspace as legacy systems. Resolving encounters with multiple intruders poses computational challenges that will be addressed in this thesis.

The methodology presented in this thesis results in logic that is safer and performs better than the legacy Traffic Alert and Collision Avoidance System (TCAS). To assess the performance of the system, this thesis uses U.S. airspace encounter models. The results indicate that the proposed methodology can bring significant benefit to the current airspace and can support the need for safe, non-disruptive collision protection as the airspace continues to evolve.

Thesis Supervisor: Jonathan P. How
Title: Richard C. Maclaurin Professor or Aeronautics and Astronautics

Thesis Supervisor: Mykel J. Kochenderfer
Title: Technical Staff, MIT Lincoln Laboratory

Acknowledgments

I would like to thank MIT Lincoln Laboratory for supporting me through my two years of education at MIT. I would especially like to thank Col. (ret) John Kuconis, Division 4 head Dr. Israel Soibelman, and Group 42 Leaders Mr. Gregory Hogan and Dr. Gregg Shoults. I am also thankful to Dr. Wesley Olson's leadership and support as manager of the ACAS X program at Lincoln Laboratory. This work is the result of research and development sponsored by the TCAS Program Office at the FAA, and I am grateful for the assistance provided by the TCAS Program Manager, Neal Suchy. I would also like to thank Prof. Jonathan How as my advisor and thesis supervisor.

I am extremely grateful for having Dr. Mykel Kochenderfer as my thesis supervisor at Lincoln Laboratory. His guidance, expertise, opinions, suggestions, and critiques were invaluable. He played a crucial role in my academic development and always provided a source of inspiration. His seemingly endless work ethic continually amazes me. Dr. Kochenderfer always provided an ear for my "crazy ideas" and elegantly dismissed most of them. Without Dr. Kochenderfer, this thesis would not have been possible.

I would like to thank James Chryssanthacopoulos. James played a crucial role in helping me understand ACAS X in the beginning months. He was very patient and was always willing to help. James provided the basis for a lot of the work in this thesis. In addition, James provided great conversations when I wanted to procrastinate.

I would also like to extend my thanks to numerous other members of Group 42. In particular I would like to thank Jessica Holland for a good laugh and great suggestions on hiking adventures and Tom Billingsley for Air Force advice, and replacing James as my outlet for procrastination at Lincoln. I would also like to specifically thank Ann Drumm and her unmatched expertise on TCAS coordination.

I can not express enough gratitude for the love and support from my friends and family. I would like to especially thank my beautiful fiancée, Kendra Miller. Kendra continually showed support for me and would always listen to my ideas.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

The Traffic Alert and Collision Avoidance System (TCAS), currently mandated on all large transport and cargo aircraft, has been shown to significantly reduce the risk of mid-air collision. TCAS uses an on-board surveillance system to monitor the local air traffic. The surveillance information is then provided to the threat resolution logic to determine whether to alert the pilot of a potential collision. TCAS will issue a resolution advisory to the pilot to climb or descend at a particular rate to prevent a collision.

Developing robust collision avoidance logic that reliably prevents collision without excessive alerting is challenging due to sensor error and the uncertain future paths of the aircraft. The current TCAS logic was the result of many years of development and involved the careful engineering of many heuristic rules. Unfortunately, due to the complexity of the logic, it is difficult to revise to accommodate the evolution of the airspace and the introduction of new surveillance technologies and procedures.

Over the past few years, research has focused on the use of a computational method known as dynamic programming for producing an optimized decision logic for airborne collision avoidance. This research has resulted in the establishment of the Airborne Collision Avoidance System X (ACAS X) program and is being targeted at becoming the next international standard for collision avoidance for both manned and unmanned aircraft. Research up to this point has primarily focused on scenarios with a single unequipped intruder.

This thesis focuses on recent research on coordination, interoperability, and multiple threat (multithreat) encounters. In situations where an aircraft encounters another aircraft with a collision avoidance system, it is important that the resolution advisories provided to the pilots be coordinated so that both aircraft are not instructed to maneuver in the same direction. Interoperability is a related consideration since new collision avoidance systems will be occupying the same airspace as legacy systems. Resolving encounters with multiple intruders will become increasingly important as the airspace becomes more dense, but poses some computational challenges.

The collision avoidance problem in this thesis is constrained to last minute collision avoidance. ACAS X for coordinated encounters must use the existing communication architecture and advisory sets used by TCAS. Therefore, there is limited communication between aircraft and the collision avoidance system can only issue vertical resolution advisories.

The contributions of this thesis can be separated in two categories–coordination and multithreat.

- *Coordination.* A comprehensive evaluation of various coordination methods for ACAS X is performed for two aircraft scenarios. The methods are discussed in terms of computational complexity, collision avoidance performance, and robustness to non-compliance.

- *Multithreat.* A simple, yet effective, extension of ACAS X to multiple intruders is presented. The performance of the method is compared against TCAS in simulations of realistic encounters and stressful scenarios. The computational complexity is also discussed.

1.1 Simulation Overview

In order to evaluate the success of the various concepts proposed in this thesis, simulations were conducted which attempted to model real encounters as accurately as possible. Encounters were developed using a high fidelity encounter model. The

aircraft were simulated based on approximate aircraft dynamics. Unless otherwise stated, the pilot response to an advisory was modeled as a $1/4\,\mathrm{g}$ acceleration applied $5\,\mathrm{s}$ after the advisory until the minimum commanded vertical rate is achieved. Subsequent advisories are modeled with a $1/3\,\mathrm{g}$ acceleration applied $3\,\mathrm{s}$ after the advisory is issued [1]. The on-board surveillance sensors used were based on the standard TCAS sensors currently used on TCAS equipped aircraft [2]. Sensor models are discussed in Section 3.4.

1.1.1 Encounter Sets

Multiple encounter sets were constructed from a dynamic Bayesian network generated from collected radar data [3]. All equipped aircraft in the encounters were required to be "discrete-code" aircraft and have a minimum airspeed of $100\,\mathrm{kt}$. There were no restrictions on tracks for unequipped aircraft. The multithreat encounter sets were generated from a similar model [4].

The simulations were conducted with aircraft that are either equipped or unequipped. Equipped aircraft were simulated with a variation of ACAS X or TCAS. Unequipped aircraft were simulated with no collision avoidance system and were either Mode S or Mode C aircraft. Mode S aircraft generally broadcast their altitude with $25\,\mathrm{ft}$ quantization while Mode C aircraft broadcast their altitude with $100\,\mathrm{ft}$ quantization. For this thesis, every equipped and Mode S aircraft are assumed to use $25\,\mathrm{ft}$ quantization. Each aircraft in the simulations was assigned a unique Mode S address. The first aircraft in the encounter was always assigned the lower address.

The coordination scheme used by TCAS and some proposed schemes for ACAS X use Mode S addresses. Therefore, the encounter sets created were permuted to ensure no biasing for position of higher Mode S aircraft. For example, if there was an equipped vs. equipped encounter, then it would appear twice in the set. The first time the encounter would be unchanged and the second time the tracks exchanged.

Importance sampling was used when generating the encounter sets. Proposal distributions for horizontal and vertical miss distances were created that resulted in acceptable variance of the metric estimates with reasonably sized encounter sets.

The encounter sets used were the following:

- Equipped vs. Equipped: 1.33×10^6 encounters,
- Equipped vs. Unequipped: 1.59×10^6 encounters,
- Equipped vs. Equipped vs. Equipped: 1.39×10^6 encounters,
- Equipped vs. Equipped vs. Unequipped: 1.77×10^6 encounters, and
- Equipped vs. Unequipped vs. Unequipped: 2.29×10^6 encounters.

1.1.2 Metrics

The performance of a collision avoidance system (CAS) can be separated into safety performance and operational performance. The primary objective of a collision avoidance system is to increase safety. However, a collision avoidance system should not interfere with normal, safe flight operations. Excessive alerts and changes in the advisories affect the efficiency of a collision avoidance system. The metrics used to evaluate collision avoidance systems in this report include:

- *Near Mid-Air Collision.* A near mid-air collision (NMAC) occurs when two aircraft come within 500 ft horizontally and 100 ft vertically. NMACs can be broken into two categories: induced and unresolved. An induced NMAC is an NMAC that occurs with a CAS but does not occur without a CAS. An unresolved NMAC is an NMAC that occurs both with and without a CAS.
- *Risk Ratio.* The risk ratio is defined as the probability of an NMAC given the aircraft is equipped with a CAS divided by the probability of an NMAC given no aircraft is equipped:

$$\text{Risk Ratio} = \frac{\Pr(\text{NMAC} \mid \text{aircraft with CAS})}{\Pr(\text{NMAC} \mid \text{aircraft without CAS})}. \tag{1.1}$$

- *Induced Risk Ratio.* The induced risk ratio (induced RR) is calculated by dividing the probability of an induced NMAC by the probability of NMAC without a collision avoidance system:

$$\text{Induced RR} = \frac{\Pr(\text{Induced NMAC})}{\Pr(\text{NMAC without a CAS})}. \tag{1.2}$$

- *Alert.* An alert is defined as when a CAS issues an advisory during an encounter.

- *Strengthening.* A strengthening is any change in a commanded vertical rate to a greater vertical rate in the same direction of the previous advisory.

- *Reversal.* A reversal is any advisory that changes the sense of a previous advisory (e.g., climb to descend).

- *Restart.* A restart is when a CAS terminates an advisory and then issues a new advisory within 20 s.

1.2 Thesis Outline

Chapter 2 summarizes the major approaches to collision avoidance and in particular, coordinated aircraft collision avoidance. An overview of Markov decision processes (MDPs) and partially observable Markov decision processes is provided.

Chapter 3 provides an overview of ACAS X. The model used for ACAS X along with the optimization processes is discussed. Chapter 3 concludes with example policies and an encounter for ACAS X and an unequipped Mode S aircraft.

Chapter 4 discusses multiple methodologies to extend ACAS X to handle encounters with a single intruder equipped with a collision avoidance system (CAS). A scheme similar to TCAS, called forced cooperation, is explored in detail. Simulation results are presented where each aircraft is equipped with ACAS X and encounters where the intruder is equipped with TCAS. The robustness of different coordination approaches to delayed pilot responses and non-compliant intruders is analyzed.

Chapter 5 generalizes ACAS X to handle encounters with multiple intruders. Two variations of a cost fusion approach is explored. Simulation results are presented for various equipage scenarios and a stress testing analysis is performed.

Chapter 6 concludes the report and outlines directions for future work.

Chapter 2

Background

This chapter provides a summary of the major approaches to collision avoidance and coordination. To develop a basis for the understanding and extension of ACAS X, a review of Markov decision processes (MDPs) and partially observable Markov decision processes (POMDPs) is presented. Finally, we will discuss variations of MDPs and POMDPs for multiple agents.

2.1 Previous Work

Collision avoidance appears in a plethora of fields due to its importance in motion planning. As a result, collision avoidance techniques have been widely researched. In 2000, Kuchar and Yang produced a survey of over 60 different methods that have been proposed to address conflict detection and resolution and many more important approaches have been published since [5]. This section will summarize the major approaches that appear in the air traffic domain.

2.1.1 Potential Field Methods

The idea behind potential field methods can be found in the air traffic domain in 1969 and the methods have been a popular approach to collision avoidance since their introduction [6]–[8]. The problem is generally modeled as virtual forces acting

on the aircraft. Waypoints or goal locations act as attractive forces, while other aircraft and other obstacles act as repelling forces. This approach is very simple, easy to understand and implement, and scales well computationally to many aircraft, but it has some disadvantages [9]. One problem is that potential field methods are prone to local minima. An example is when agents get "stuck" due to the canceling out of forces.

Most of the work with potential fields in the air traffic domain does not account for uncertainty in control or observation [10]–[12]. Despite not considering uncertainty in observations or control, Kelly and Eby showed that potential field methods may be used to resolve complex, random, multi-aircraft conflicts without the use of intent information [13]. Prandini *et al.* took a potential field approach and generalized the idea to work for probabilistic dynamic environments [14].

2.1.2 Geometric Optimization

This category of collision avoidance technique utilizes the geometric characteristics of the environment and agents involved (i.e. aircraft trajectories). Often a closed-form analytical solution for minimal changes in the velocity vector are obtained. Bilimoria introduced a 2-D conflict resolution algorithm given only current positions and velocity vectors of aircraft [15]. The resolutions are combinations of heading or speed changes and are generated from a closed-form analytic solution. Multiple aircraft encounters are handled in a pair-wise "most immediate" conflict ordering. KB3D is a 3-D extension of this algorithm where a cylinder defines the protected zones of intruders [16]. Additional modifications to KB3D only requires one aircraft to maneuver for successful collision avoidance, and implicit coordination is guaranteed with perfect state information [17].

Luongo *et al.* developed an analytical solution based on a cylindrical safety area surrounding the aircraft that minimizes the aircraft's deviation from the nominal trajectory [18]. The algorithm was developed for non-cooperative aircraft and multiple aircraft encounters are handled in a pair-wise fashion. The authors perform numerical simulations taking into account aircraft dynamics and on-board noisy sensors.

To account for the unmodeled uncertainty, the protected zone was increased prior to simulation to a static size. However, realistic navigation sensor errors led to a deterioration of the proposed collision avoidance algorithm [18].

Chamlou developed a model-based geometric approach that did not require a fixed look-ahead time to predict a loss of separation [19]. With this approach the protected zone around the aircraft could be varied real-time to account for uncertainties. Provided no state uncertainty, independent implicitly coordinated solutions were guaranteed.

Geometric approaches provide very fast analytic solutions that can be easily implemented real-time. However, all of the geometric approaches discussed rely on a linear propagation of the state vector and do not consider state uncertainty in the algorithm. For last minute collision avoidance, the protected zone surrounding the aircraft would have to be superficially enlarged to ensure system robustness which might lead to unnecessary alerts and degrading operational performance.

2.1.3 Mathematical Programming

Mathematical programming has gained popularity since the Kuchar and Yang survey was published as the computational power available to agents has increased. Frazzoli *et al.* proposed a semidefinite programming approach that resulted in centralized conflict resolution while minimizing deviations from a desired path [20]. The proposed approach does not account for state uncertainty and relies on a centralized planning algorithm in which the desired headings of the aircraft are shared.

Using models where the agents' dynamics can be approximated using only linear constraints enables the use of mixed integer linear programming (MILP) [21], [22]. Schouwenaars *et al.* use a MILP approach to develop a decentralized planning structure in which each aircraft updates its trajectory one at a time [23]. The authors were able to guarantee safety by developing a safe "loiter pattern" that the aircraft would default to if a safe trajectory could not be found. Other MILP approaches have been introduced that also claim real-time implementation from fast solutions using commercial software [24], [25].

Oishi *et al.* proposes the use of a mixed integer nonlinear program [26]. The authors focus on guaranteeing safety while meeting certain performance criteria. The optimization assumes cooperative aircraft and the solvers used were very sensitive to the initial values of the state and control trajectories. Christodoulou and Kontogeorgou used nonlinear programming (NLP) to solve for minimum velocity changes to avoid conflicts and then attempted to create a neural network that can predict the velocity changes [27]. Borrelli *et al.* compare a nonlinear programming approach using an interior point algorithm to a MILP approach [28]. The authors determined that the MILP was always faster then the NLP and as the problem size grew, the gap between the MILP solution time and the NLP solution time increased. They also determined that the MILP and the NLP always provide similar results in terms of performance.

The mathematical programming methods mentioned thus far do not factor in uncertainty of dynamics or sensor measurements. The Robust Safe but Knowledgeable (RSBK) algorithm is a MILP planning algorithm which uses robust model predictive control, which is robust to external disturbances and modeling errors [29]. RSBK solves a centralized problem when it is applied to multiple agents. Luders developed Efficient RSBK which is a MILP formulation with non-uniform time steps between target waypoints, and plans a detailed short-term trajectory and a coarse long-term trajectory [30]. A decentralized version of RSBK was developed in which each vehicle only computes its own trajectory, which is obtained by solving a sub-problem of reduced size [31]. The drawback of the RSBK family is the requirement of sharing plans. That is, when an agent is solving for its trajectory, it requires the plans (e.g. waypoints) of the agents in its neighborhood. To apply this idea to last minute aircraft collision avoidance, unequipped intruders would have to be modeled as obstacles or the plans of the intruders would have to be extrapolated from the current state information.

2.1.4 MDP/POMDP Approaches

Most of the methods previously discussed are some form of open-loop planning. Often, new plans are regenerated as new observations are made and this is called open-loop feedback control. Closed-loop planning computes a strategy for selecting actions from a given state and accounts for selecting future actions based on new information using a probabilistic model. Chryssanthacopoulos and Kochenderfer demonstrate some performance gains when using closed-loop planning strategies in highly stochastic environments [32]. Recent work on collision avoidance algorithms in closed-loop planning strategies are based on solutions to Markov decision processes (MDPs).

Winder formed the aircraft collision avoidance problem as an MDP with simple dynamics where he assumed Gaussian intruder process noise [33]. The MDP approach assumed that all necessary state variables were known at all times. Winder extends the problem and used a partially observable Markov decision process (POMDP) formulation where the aircraft is in an uncertain mode. The modes correspond to different aircraft dynamics (e.g. level-off mode and a descent mode). The POMDP formulation demonstrated that using this approach, safety can be maintained or improved with a reduction in unnecessary alerts when compared to other methods.

Kaelbling and Lozano-Pérez assume highly accurate state measurements and uncertainty in the pilot response to advisories. Under these assumptions, they model the aircraft collision avoidance problem as an MDP [34]. The authors investigate control policies derived using a policy gradient method and a policy search by dynamic programming approach. These methods avoid having to discretize the state space and thus have a weaker dependence on the size of the state space.

Wolf and Kochenderfer use an online POMDP approach to collision avoidance using a Monte Carlo real-time belief space search [35]. Their approach assumed noisy sensors that provided aircraft position, orientation, rates, and accelerations from both the own aircraft and the intruder. The formulation uses a finite action space, continuous state and observation spaces, and use sample-based representations of state uncertainty.

Temizer *et al.* model the aircraft collision avoidance problem as a POMDP and investigate the use of difference sensors while solving the problem offline [36]. The authors use discretized state, action, and observation spaces. They showed that the POMDP approach can work for various sensors including TCAS-like sensors where bearing, range, and altitude of the intruder are measured as well as limited field-of-view sensors where elevation of the intruder is measured with respect to the own aircraft instead of altitude. Near-optimal policies were computed using a solver that uses Successive Approximations of the Reachable Space under Optimal Policies (SARSOP) algorithm [37]. The approach only considered a single unequipped intruder and the state spaces were highly discretized. With approximately 3400 states, the SARSOP solver was able to generate policies in 3 to 5 hours.

Kochenderfer *et al.* also formed the collision avoidance problem as a POMDP and showed the formulation to be robust to modeling errors [38]–[42]. The problem allows for the maneuvers in the vertical plane while still accounting for stochastic horizontal aircraft dynamics. This method was shown to be viable for multiple intruders and for coordinated encounters [42]. This thesis extends this approach to improve the handling of coordinated encounters and multiple equipped and unequipped intruders. More detail on this POMDP formulation will be provided in Chapter 3.

2.1.5 Other Approaches

Other approaches to collision avoidance have been proposed. TCAS uses linear propagation and complex heuristic rules to determine proper resolutions [43]. Gariel *et al.* propose a conflict detection algorithm that uses trajectory prediction different from the linear velocity propagation that is currently used by TCAS [44]. Their approach relies on constant turn rate propagation. The authors include two protected areas– one that is fixed during simulation and can be inflated to account for errors in state estimates, and a second zone that varies during simulation based on the closure rate between aircraft.

Velocity obstacle (VO) methods are a geometric approach that rely on velocity information to determine a collision avoidance maneuver [45]. A VO is the set of

velocities for an agent that would result in a collision with another object assuming the other object maintains its current velocity. A collision avoidance maneuver is selected from the intersection of the reachable set of velocities based on dynamic and kinematic constraints and the complement of the VO. The original approach is based on a linear approximation of the obstacle's trajectory and does not consider uncertainty [45]. Shiller *et al.* extended the approach to incorporate known trajectories and obstacles with non-linear velocities [46]. Kluge and Prassler developed probabilistic VOs by considering uncertainty in the shape and velocity of the objects [47]. A probabilistic VO maps the velocities of an agent to the probability of colliding with an object. With probabilistic VOs, selecting an appropriate collision avoidance maneuver requires a utility function which considers whether the velocity is reachable, the probability of collision, and some desired goals [47]. Berg *et al.* extended the basic VO approach to multiple agent systems [48]. The extension to multiple agents assumes that other agents make collision avoidance decisions similarly. The basic idea is for the two agents to share the burden of a collision avoidance maneuver by choosing a velocity that is a convex combination of the agent's current velocity and a desired velocity (i.e., a velocity that is reachable and outside of the VO) [48].

Viebahn and Schiefele developed a conflict detection algorithm that discretizes the state space and derives a threat probability for each element in the space [49]. The position of the own aircraft is also represented using a probability distribution. A conflict is determined when the joint probability is higher than some threshold. Prandini *et al.* also develop a conflict detection algorithm that factors in uncertainty [50]. The authors use probabilistic models for the aircraft dynamics and use randomized algorithms to overcome the computational burden of solving for the probability of conflict.

Kantas *et al.* constructed a simulation based Bayesian approach to determine optimal trajectories in the presence of wind disturbances [51]. The approach attempts to minimize the expected time of arrival for all aircraft by determining optimal maneuvers using sequential Monte Carlo sampling. The planning is handled in a centralized

fashion and the authors suggest that short execution times are attainable when the algorithms are parallelised and implemented on graphics processor units.

An algorithm developed by Hoffmann and Tomlin uses a rule-based approach derived using optimal control and reachability analysis [52]. The algorithm is decentralized and very fast with a large number of vehicles. However, it requires aircraft to share state information and does not factor in uncertainty.

Two cooperative conflict resolution algorithms are presented by Šišlák *et al.* [53], [54]. One algorithm is an iterative approach that resolves conflicts by pairwise negotiations and the second resolves the conflicts based on multi-party negotiation. Both approaches are decentralized, rely on shared partial flight plans, work with imprecise flight plan execution, and provide maneuvers based on variations to the initial flight path.

Goode and Roan present a collision avoidance strategy using a differential game theoretic approach [55]. The agent decides on an action assuming a worse-case scenario. The assumption of an intruder acting as a pursuer provides robustness to the control; however, that assumption can result in a conservative strategy (i.e. many false alerts in the air traffic domain). Another game theoretic approach was developed by Archibald *et al.* The authors developed a decentralized and cooperative algorithm based on satisficing game theory [56]. The idea is that satisficing agents are able to condition their own preferences on the preferences of others, allowing agents to compromise in order to achieve both individual and group goals.

2.2 Review of MDPs and POMDPs

As discussed in Section 2.1.4, collision avoidance problems can be modeled as a Markov decision process. When there is uncertainty in the agent's state, the problem is more appropriately modeled as a partially observable Markov decision process. This section discusses the general framework for MDPs and POMDPs.

2.2.1 MDP

An MDP is a general framework for sequential decision making in stochastic environments when the state of the system is fully observable. The process satisfies the Markov property. That is, the conditional probability distribution of future states only depends on the present state and action taken. An MDP is defined by the tuple $\langle S, A, T, R \rangle$, where S is a finite set of states, A is a set of all possible actions the agent may take, T is a transition probability function, and R is an immediate reward function. T is often called the state-transition function. The probability of a state transitioning from s to s' by action a is denoted $T(s, a, s')$. The reward function, R, dictates the immediate reward received for taking action a in state s and is denoted $R(s, a)$.

Solving an MDP involves searching for a policy that maximizes the expected sum of the instantaneous reward. An MDP policy is a strategy for selecting actions and maps states to an action. Under certain assumptions regarding the structure of the reward function, it is sufficient to only consider policies that deterministically depend only on the current state without losing optimality [57]. Given a policy π, the action to execute from state s is denoted $\pi(s)$. The expected value of executing policy π from state s is denoted $V^{\pi}(s)$. An optimal policy, $\pi^*(s)$ is a policy that maximizes the expected value

$$\pi^*(s) = \arg\max_{\pi} V^{\pi}(s), \forall s \in S. \tag{2.1}$$

Dynamic programming (DP) may be used to compute the value of policy π for a finite number of steps [57]. For the first step, the value is the expected reward of step one, $V_1^{\pi}(s) = R(s, \pi(s))$. If we know the value of executing the policy π for the first $N-1$ steps, then we can compute the the value at step N as

$$V_N^{\pi}(s) = R(s, \pi(s)) + \sum_{s'} T(s, \pi(s), s') V_{N-1}^{\pi}(s'). \tag{2.2}$$

The most common dynamic programming algorithms used to compute optimal policies are policy iteration and value iteration.

Policy iteration is done in two steps starting with any policy:

- Policy evaluation. Given policy π_n, compute V^{π_n}.
- Policy improvement. Improve the policy π_n to get a new policy π_{n+1}.

This process continues until there is no more improvements to be made (i.e. $\pi_n = \pi_{n+1}$). Algorithm 1 is an outline of policy iteration [57].

Algorithm 1 Policy Iteration

```
1: function PolicyIteration(π_0)
2:     n ← 0
3:     repeat
4:         Calculate V^{π_n}
5:         for s ∈ S do
6:             π_{n+1}(s) ← arg max_a {R(s,a) + Σ_{s'} T(s,a,s') V^{π_n}(s')}
7:         n ← n + 1
8:     until π_n = π_{n+1}
9:     return V^{π_n}, π_{n+1}
```

An alternate and more common method used to solve MDPs is value iteration. The value iteration algorithm relies on the Bellman optimality equation $V^*(s) = \max_a [R(s,a) + \sum_{s'} T(s,a,s')U^*(s')]$ [58]. Algorithm 2 provides an outline of value iteration [57]. The $||\cdot||$ operator denotes the max norm (i.e. $||V|| = \max_s |V(s)|$) and ϵ is a predefined tolerance threshold. The stopping criteria presented in Algorithm 2 is just one of several that can be considered.

Algorithm 2 Value Iteration

```
1: function ValueIteration
2:     n ← 0
3:     V_n(s) ← 0, ∀s ∈ S
4:     repeat
5:         for s ∈ S do
6:             V_{n+1}(s) ← max_a {R(s,a) + Σ_{s'} T(s,a,s') V_n(s')}
7:         n ← n + 1
8:     until ||V_{n+1} − V_n|| < ε
9:     for s ∈ S do
10:        π(s) ← arg max_a {R(s,a) + Σ_{s'} T(s,a,s') V_n(s')}
11:    return V_n, π
```

One benefit of value iteration is that the state-action values are calculated for each state. It is often beneficial to store these values when using approximate DP for continuous state spaces. There are variations on the algorithms presented that result in less computational complexity, but the concept is the same. The formulation presented assumes that there is not a discount factor at each time step. The addition of this parameter is straightforward and does not add any complexity.

2.2.2 POMDP

MDPs require full knowledge of the current state. Often this assumption is not valid in real world problems. POMDPs provide a general framework that factors in the uncertainty of the state. A POMDP is an MDP in which the real state is unknown, but the agent is able to make partial observations of the true system state [57]. A POMDP is defined by the tuple $\langle S, A, \Omega, T, O, R \rangle$, where $S, A, T,$ and R are the same as the MDP. Ω is a set of possible observations the agent can receive. O is the observation function that returns the probability of the agent receiving observation o after taking action a and ending in state s' and is denoted $O(s', a, o)$.

A belief state is a distribution over the state space and is a sufficient statistic for the history of observations received [57]. A policy of a POMDP is a mapping from the observations received to actions. It can be proven that a policy can be represented as a mapping from belief states to actions [59]. At each step an agent chooses an action based on the current belief state and the provided policy. The system transitions based on the chosen action, and an observation and reward are received. The belief state is then updated and the cycle continues. After choosing action a, transitioning to state s', and receiving observation o, the belief state, b is updated by

$$b'(s') = P(s'|o, a, b) \propto O(s', a, o) \sum_{s} T(s, a, s') b(s). \tag{2.3}$$

A POMDP can be thought of as an MDP where the states are belief states. The state space of this belief MDP is the set of all possible beliefs, B. The state transition

function $\tau(b, a, b')$ and the immediate reward $R(b, a)$ are given by

$$\tau(b, a, b') = \sum_{o} P(b'|b, a, o) \sum_{s'} O(s', a, o) \sum_{s} T(s, a, s'), \text{ and} \tag{2.4}$$

$$R(b, a) = \sum_{s} b(s) R(s, a). \tag{2.5}$$

Solving for an exact solution in the worst case is PSPACE-complete [57]. However, solutions can often be approximated well. Both offline and online approximations exist.

One offline approximation is called QMDP. The QMDP approach assumes all state uncertainty disappears at the next time step. This assumption allows for the state-action values, $Q(s, a)$, to be calculated assuming full observability. If the current belief state is b, then the action to be selected is given by

$$\pi(b) = \arg\max_{a} \sum_{s} b(s) Q(s, a). \tag{2.6}$$

This approximation performs well in many real world scenarios when the actions do not reduce state uncertainty. Therefore, when actions are information gathering and can significantly reduce the state uncertainty, the QMDP is a poor approximation [60], [61].

Other offline approximations include the fast informed bound method [61], point-based value iteration (PBVI) [62], Heuristic Search Value Iteration (HSVI) [63], and Successive Approximations of the Reachable Space under Optimal Policies (SARSOP) [37], and randomized point-based value iteration algorithm [64].

Online methods are popular for high dimensional problems. One method uses an approximate value function computed offline and performs a one-step lookahead online to improve upon the policy. This one-step lookahead idea can be extended to an arbitrary depth and is called forward search. Partially Observable Monte Carlo Planning (POMCP) is a Monte Carlo tree search algorithm for POMDPs [65]. Ross et al. survey several online planning algorithms [66].

2.3 Expanding MDPs and POMDPs to Multiple Agents

MDPs and POMDPs are great methods for determining actions for a single agent in stochastic environments. These methods can be extended to cooperative multiagent domains as well. This section discusses the extension of both MDPs and POMDPs to cooperative multiagent domains. In particular the framework for multiagent Markov decision processes (MMDP), decentralized Markov decision processes (Dec-MDP), and decentralized partially observable Markov decision processes (Dec-POMDP) will be discussed.

2.3.1 MMDP

MMDPs extend MDPs and allow for sequential decision making in a cooperative multiagent system. Since the agents are cooperative, there is a single reward function that the agents must maximize [57]. An MMDP can be viewed as a large MDP. An MMDP can be defined by the tuple $\langle S, A, T, R \rangle$, where S is a finite set of states, A is a set actions, T is a transition probability function, and R is an immediate reward function. Since more than one agent is considered, the action space is made up of joint actions. That is, each element of A represents an action for each agent. Then $T(s, a, s')$ is the transition function which gives the probability of transitioning to state s' from state s when the agents execute the joint action a. Similarly, $R(s, a)$ is the immediate reward for taking joint action a in state s.

Since MMDPs can be thought of as large MDPs, the same solution approaches can be used. Methods like policy iteration (Algorithm 1) and value iteration (Algorithm 2) can be used to solve MMDPs.

The model of an MMDP does not consider individual observations. Therefore each agent must have full observability of the entire system, a centralized planner is used that has access to the system state, or the agents communicate their observations

free and instantaneously. The last idea of communicating observations requires the system to be jointly fully observable.

2.3.2 Dec-MDP/Dec-POMDP

The requirement of individual complete observability of the system state is often not realistic. A decentralized control where the system is jointly fully observable can be modeled as a Dec-MDP and modeled as a Dec-POMDP if the system is jointly partially observable. Dec-MDPs are a special case of Dec-POMDPs where the system state is jointly observable.

A Dec-POMDP is defined as a tuple $\langle S, A, \Omega, T, O, R \rangle$, where $S, A, T,$ and R are the same as the MMDP. Ω is a set of joint observations the agents can receive. O is the observation function that returns the probability of the agents receiving observations $\langle o_1, \ldots, o_n \rangle$ after taking joint action a and ending in state s' and is denoted $O(s', a, \langle o_1, \ldots, o_n \rangle)$.

Solutions to Dec-MDPs and Dec-POMDPs are joint policies. A joint policy π for a Dec-POMDP with n agents is a set of individual policies $\langle \pi_1, \ldots, \pi_n \rangle$, where π_i is the individual policy for agent i. An optimal policy is one that maximizes the expected total reward of the system [57].

Finding optimal solutions to Dec-POMDPs is difficult. The number of joint policies for a system is doubly exponential in the horizon of the problem. The problem of finding the optimal solution for a finite horizon Dec-POMDP with more than one agent is NEXP-complete [67]. Even finding a bounded approximation is NEXP-hard [68]. The same complexity holds for Dec-MDPs [57]. Oliehoek provides an overview of various approaches to find approximate solutions to Dec-POMDPs [69].

Chapter 3

ACAS X Overview

The Airborne Collision Avoidance System X (ACAS X) is under development and is intended to replace the existing Traffic Alert and Collision Avoidance System (TCAS). To minimize the amount of retraining for aircrews, ACAS X is limited to the same resolution advisories as TCAS. Therefore, only vertical maneuvers are considered and ACAS X assumes that the aircraft dynamics in the horizontal plane are unaffected by advisories.

ACAS X models the collision avoidance problem as a POMDP. An approximate solution to the POMDP is found using the QMDP approach discussed in Section 2.2.2. The state-action values are found using value iteration (Algorithm 2).

This chapter describes the action space, state space, the dynamic model, and how state estimation is performed. The cost function used in the optimization and the online costs used to improve performance are discussed. The chapter will conclude by showing an example policy and and example encounter with an unequipped intruder.

3.1 Action Space

The current version of TCAS issues advisories to the pilot through an aural annunciation, such as "climb, climb," and through a visual display. The visual display varies, but it is typically implemented on a vertical speed indicator, a vertical speed tape, or a pitch cue on the primary flight display. The set of advisories issued by TCAS

Table 3.1: Advisory set

	Vertical rate (ft/min)			
Name	Minimum	Maximum	Strength (g)	Available from
COC	$-\infty$	∞	0	All
DNC	$-\infty$	0	1/4	COC, DES1500, SDES1500, SDES2500, MDES
DND	0	∞	1/4	COC, CL1500, SCL1500, SCL2500, MCL
MDES	$-\infty$	$\dot{h}_{\text{curr}}$	1/4	COC, DND, CL1500, SCL1500, SCL2500
MCL	$\dot{h}_{\text{curr}}$	∞	1/4	COC, DNC, DES1500, SDES1500, SDES2500
DES1500	$-\infty$	-1500	1/4	COC
CL1500	1500	∞	1/4	COC
SDES1500	$-\infty$	-1500	1/3	DNC, DND, MCL, CL1500, SCL1500, SCL2500
SCL1500	1500	∞	1/3	DNC, DND, MDES, DES1500, SDES1500, SDES2500
SDES2500	$-\infty$	-2500	1/3	MDES, DES1500, SDES1500
SCL2500	2500	∞	1/3	MCL, CL1500, SCL1500

can be interpreted as target vertical rate ranges. If the current vertical rate is outside the target vertical rate range, the pilot should maneuver to come within the required range. If the current vertical rate is within the target range, a corrective maneuver is not required, but the pilot should be careful not to maneuver outside the range.

The set of advisories used by ACAS X is summarized in Table 3.1. In the table, COC stands for "clear of conflict," which means that no advisory has been issued or there is no longer a threat. DNC and DND stand for "do not climb" and "do not descend," respectively. The sense of all other advisories are labeled either CL or DES, for either climb or descend, respectively. The prefix "M" stands for maintain. The maintain advisories are issued only when the magnitude of the current vertical rate of the own aircraft ($\dot{h}_{\text{curr}}$) is greater than 1500 ft/min. The maintain advisory is issued with the current rate as the upper or lower bound on the commanded rate. The prefix "S" indicates that a stronger response is assumed. Only the minimum and maximum rates are displayed to the pilot, and not the "strength" of the response, so the eleven advisories in the table only correspond to nine different advisories to be displayed to the pilot (including COC). However, it is useful to distinguish the advisories according to the assumed strength of the maneuver when developing the MDP model.

Table 3.1 also indicates the availability of each advisory given the current advisory on display. For example, COC can be issued at any time. However, because DES1500 and CL1500 are initial advisories, they can only be issued if COC is on display to

the pilot. The advisory SDES1500 can be issued following DND, MCL, CL1500, SCL1500, and SCL2500, in which case it acts as a reversal, or following DNC, in which case it acts as a strengthening. Because SDES1500 is a subsequent advisory, it cannot be issued following COC. It also cannot be issued following DES1500 because they are fundamentally the same advisory, differing only in strength. The allowed transitions are modeled after those of TCAS.

The advisories in Table 3.1 are a subset of the advisories available in the current version of TCAS. Although it captures most of the advisories issued by TCAS, it does not contain certain rate limit preventive advisories. One advisory that is incorporated into the set for mulithreat purposes is a multithreat level-off (MTLO). The MTLO is not included in the action state but is still allowed to be issued, as discussed in Section 5.3. The computational and storage requirements of the MDP approach scale linearly with the addition of new actions.

3.2 State Space

The state is represented using five variables:

- h: altitude of the intruder relative to the own aircraft,
- $\dot{h}_0$: vertical rate of the own aircraft,
- $\dot{h}_1$: vertical rate of the intruder aircraft,
- τ: time to potential NMAC, and
- s_{RA}: the state of the resolution advisory.

The state space is discretized using a multidimensional grid. The discretization used is shown in Table 3.2, which results in approximately 15 million grid vertices that correspond to discrete states. The discretization can be made finer to improve the quality of the discrete model approximation, but it would be at the expense of additional computation and storage. Previous studies involving ACAS X have found this level of discretization acceptable [42].

Table 3.2: Discretization scheme

Variable	Minimum	Maximum	Number of values
h	$-4000\,\text{ft}$	$4000\,\text{ft}$	31
$\dot{h}_0$	$-10000\,\text{ft/min}$	$10000\,\text{ft/min}$	25
$\dot{h}_1$	$-10000\,\text{ft/min}$	$10000\,\text{ft/min}$	25
τ	$0\,\text{s}$	$40\,\text{s}$	41
s_{RA}	N/A	N/A	19

The state variable s_{RA} dictates the state of the resolution advisory. For each action besides COC, the aircraft can either be in a responding state or a non-responding state. The responding and non-responding states for each action allow for a probabilistic pilot response model [42].

3.3 Dynamic Model

The dynamics of the aircraft involved in the encounter are governed by sequences of accelerations. These accelerations are used to update the vertical rates of the aircraft and, consequently, their positions. The maximum vertical rate of both aircraft is assumed to be within $\pm$10,000 ft/min, which is the same as TCAS. The limits can be adjusted to meet the performance constraints for particular aircraft. The dynamics are transformed into a discrete state MDP using the multilinear-interpolation and sigma-point sampling scheme [42].

When pilots are not following a resolution advisory, the aircraft follow a white-noise acceleration model. At each step, the aircraft selects an acceleration from a zero-mean Gaussian distribution with $\sigma_{\ddot{h}}$ standard deviation. We assume $\sigma_{\ddot{h}} = 3\,\text{ft/s}^2$, but this parameter may be estimated from radar data. When the system is in a responding s_{RA}, it is assumed that the pilot maneuvers with exactly the prescribed acceleration to come within the target vertical rate range. Once within the target vertical rate range, the aircraft resumes white-noise vertical accelerations.

For the transition between s_{RA} states, a linear model is used for transitioning between a non-responding state and a responding state. For example, if a DES1500 advisory is issued, the system has a 1/6 probability of transitioning to the responding DES1500 s_{RA} and a 5/6 probability of transitioning to a non-responding DES1500

s_{RA}. Transitions to reversal advisories are modeled the same as the transition for the initial advisory. Transitions to strengthening or weakening advisories are modeled with a 1/4 probability of transitioning to a responding s_{RA}. When the advisory is terminated, the system transitions to the COC s_{RA} with probability 1. Using this approach, the response to an advisory is a geometric random variable. The success probability of 1/6 and 1/4 were used so on average the modeled pilot response will occur in 5 s and 3 s, respectively. These delays come from the ICAO recommended practices for responding to resolution advisories [1].

3.4 State Estimation

State estimation is performed differently for each variable. The sensor model used to detect intruder aircraft is based on the current TCAS sensor. The TCAS sensor measures the slant range and bearing of all nearby intruder aircraft. The slant range error is modeled as a zero-mean Gaussian with 50 ft standard deviation. The bearing error is modeled as a zero-mean Gaussian with 10° standard deviation. This section describes how the belief state over each state variable is computed.

- $h, \dot{h}_0, \dot{h}_1$. The own aircraft's vertical state is assumed to be known. In reality, the ownship's quantized altitude will be supplied. The quantization will be relatively small and an accurate vertical state estimate can be made. The intruder's vertical state is estimated using a modified Kalman filter designed for quantized measurements [70], [71]. The intruder's quantized altitude is obtained from the intruder's transponder. The altitude is either quantized by 25 ft or 100 ft.

- τ. A time distribution which factors in horizontal aircraft dynamics is computed offline using dynamic programming [42]. The dynamic programming uses horizontal range to the intruder, relative horizontal speed, and the difference in the direction of the relative horizontal velocity and the bearing of the intruder as states to determine the time distribution. During execution, two unscented

Kalman filters (UKF) are used to estimate these values. One UKF is used to estimate the range and range rate of the intruder. The second UKF is used to estimate the bearing and the cross range rate. These estimates are then used to look up a distribution over τ from the table generated offline.

- s_{RA}. The belief distribution over s_{RA} is updated recursively using standard model-based filtering techniques [42]. The belief state over s_{RA} is initialized as being in the COC state with probability 1. Then at each subsequent time step t, the belief state is updated as follows:

$$b_t(s'_{\text{RA}}) \propto \sum_{s_{\text{RA}}} p(\dot{h}^0_t | \dot{h}^0_{t-1}, s_{\text{RA}}, a) T(s_{\text{RA}}, a, s'_{\text{RA}}) b_{t-1}(s_{\text{RA}}), \tag{3.1}$$

where $p(\dot{h}^0_t | \dot{h}^0_{t-1}, s_{\text{RA}}, a)$ is the probability density of the own aircraft vertical rate at time t, $\dot{h}^0_t$, conditioned on the previously observed vertical rate, $\dot{h}^0_{t-1}$, the action taken, a, and the previous advisory state, s_{RA}.

3.5 Cost Function

The cost of executing action a from state s is denoted $C(s, a)$ and costs of various events are summarized in Table 3.3. The costs were chosen after several iterations of tuning the logic based on operational and safety analysis [72].

Historically, the primary safety metric for evaluating TCAS has been Pr(NMAC), and so NMACs are assigned high cost. The rewards associated with COC are given at every time step the system is not alerting to provide some incentive to discontinue alerting after an encounter has been resolved. The rewards for the advisories DND, DNC, and maintain provide incentives to issue less aggressive advisories.

If the state space does not require expanding, the computation required to construct the expected cost table grows linearly with the number of cost factors and the storage required remains constant. Online execution of the logic also remains constant.

Table 3.3: Event costs

Description	Cost
NMAC	0.65
Alert for DNC and DND when the vertical closure rate is $> 3000\,\text{ft/min}$	0.0005
Alert for advisories other than DNC and DND when the vertical closure rate is $> 3000\,\text{ft/min}$	0.0015
Alert for when the vertical closure rate is $< 3000\,\text{ft/min}$	0.0023
Corrective advisory	1×10^{-5}
Reversal	0.008
Strengthening	0.005
Weakening	0.001
Change in $\dot{h}$	3×10^{-5}
Change in $\dot{h}$ during a crossing	4×10^{-4}
Corrective advisories when relative altitude is $> 650\,\text{ft}$ and the vertical closure rate is $< 2000\,\text{ft/min}$	0.1
Non-corrective advisories when relative altitude is $> 650\,\text{ft}$ and the vertical closure rate is $< 2000\,\text{ft/min}$	0.01
Corrective advisories when relative altitude is $> 1000\,\text{ft}$ and the vertical closure rate is $< 4000\,\text{ft/min}$	0.03
Maintain advisories $< 1500\,\text{ft/min}$	1
Crossing encounters when relative altitude is $> 500\,\text{ft}$	0.01
Prohibited advisory transitions	1
Issue COC	-1×10^{-9}
Issue DNC	-0.0001
Issue DND	-0.0001
Issue a Maintain	-0.0004
Preventive advisories during crossing scenarios	1

3.6 Online Costs

The cost function was designed to be based on only the current state and action, $C(s, a)$. Therefore, any cost that requires "memory" would have to be implemented by introducing new state variables. Online costs were introduced to penalize actions in real-time without introducing new state variables. During execution, the expected costs calculated from the offline optimization are added to the online cost associated with that action. Table 3.4 summarizes the parameters and costs.

- *Altitude Inhibit Cost.* The altitude inhibit cost penalizes advisories below certain altitudes. This online cost was modeled from TCAS and hysteresis is implemented to prevent chatter. Advisories are prohibited if the aircraft starts below the upper threshold and remains prohibited until it crosses above that threshold. From the other direction, advisories are allowed until the aircraft flies below the lower threshold.
- *Advisory Switch Cost.* The advisory switch cost penalizes actions that represent a change from the current advisory within a certain number of seconds after it is issued. Switching to different advisories such as opposite sense advisories are penalized differently. This online cost requires memory of the previous advisory issued and the duration it has been active.

Table 3.4: Online cost parameters

Description	Cost / Value
Altitude Inhibit Cost	
All advisories lower hysteresis bound	900 ft
All advisories upper hysteresis bound	1100 ft
All advisories cost	∞
SDES2500, MDES ($\dot{h}_{\text{curr}} < -2500\,\text{ft/min}$) lower hysteresis bound	1450 ft
SDES2500, MDES ($\dot{h}_{\text{curr}} < -2500\,\text{ft/min}$) upper hysteresis bound	1650 ft
SDES2500, MDES ($\dot{h}_{\text{curr}} < -2500\,\text{ft/min}$) cost	∞
DES1500, SDES1500, MDES ($\dot{h}_{\text{curr}} < -1500\,\text{ft/min}$) lower hysteresis bound	1000 ft
DES1500, SDES1500, MDES ($\dot{h}_{\text{curr}} < -1500\,\text{ft/min}$) upper hysteresis bound	1200 ft
DES1500, SDES1500, MDES ($\dot{h}_{\text{curr}} < -1500\,\text{ft/min}$) cost	∞
DNC lower hysteresis bound	1050 ft
DNC upper hysteresis bound	1150 ft
DNC cost	0.005
Advisory Switch Cost	
Time the online cost is active after an advisory is issued	10 s
Switch to a reversal	0.05
Switch to COC	0.05
Switch to any advisory except for COC or a reversal	0.025
Advisory Restart Cost	
Time the online cost is active after an advisory is terminated	10 s
Restart cost	0.05
Initialization Cost	
Time to incur online cost starting from the start of the track	3 s
Initialization cost for all advisories except COC	∞
Forced Cooperation Cost	
Slave cost for non-cooperative advisory	∞

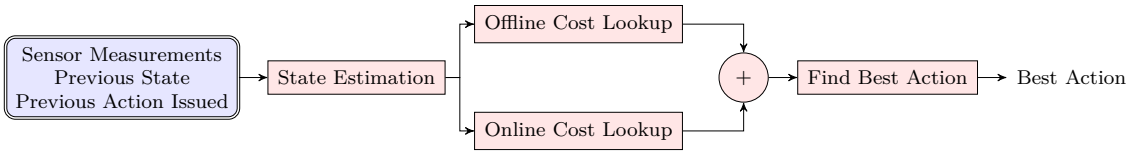


Figure 3-1: Block diagram of ACAS X execution.

- *Advisory Restart Cost.* The advisory restart cost penalizes advisories whenever a prior advisory has been terminated for fewer than a certain number of processing cycles. This cost requires memory of whether an advisory has been issued, whether an advisory has terminated, and the amount of time for which an advisory has been terminated.

- *Initialization Cost.* The initialization cost prohibits advisories from being issued for some number of processing cycles. The initialization period allows for the trackers to stabilize before issuing an advisory.

- *Forced Cooperation Cost.* The forced cooperation cost penalizes actions that are incompatible with the sense of an intruder VRC message. This online cost is the basic implementation of the online forced cooperation scheme discussed in Section 4.1.2.

3.7 ACAS X Execution

A high level block diagram describing how ACAS X is executed in real-time is shown in Figure 3-1. Sensor measurements, the previous state estimate, and the previous action issued are used to calculate the current state estimation. The state estimate is used to determine the offline costs for the actions. The online costs are also calculated from the current state estimate along with various other inputs such as the intruder's Mode S address. The offline and online costs are summed and the best action is determined. In a real system, the action is displayed to a pilot for execution and this process repeats once every second.

3.8 Optimal Policy

Since the collision avoidance logic is critical to safety, it is important for humans to understand and anticipate the behavior of the system. Because the logic makes decisions based on values in an expected cost table, which is not directly informative to a human, it is necessary to develop ways to visualize the logic. Visualization is also important in building confidence that the logic produced through computer optimization is sensible.

The policy plots generated in this thesis are based on simulations with different initial altitudes. The results are discretized into altitude bins for each time step. The action for every track at each time step is deposited in its respective bin. The policy is then represented as the most likely action for each bin. If there is a bin that no trajectory falls in, no action is displayed. The horizontal axis is time and the vertical axis is altitude. All simulations were conducted with standard TCAS sensor uncertainty [1]. The encounters are simulated with no pilot response.

Figure 3-2 shows a policy plot for ACAS X and TCAS. The encounter is between two aircraft with only the own aircraft equipped. Both aircraft are flying level and directly at each other horizontally. The time of closest horizontal approach (TCA) occurs at 40 s. For this scenario, the policies are very similar; however, the ACAS X alerting region is much smaller than that of TCAS, delaying alerting by about 5 s.

3.9 Example Encounter

Figure 3-3 is an example of an encounter with an unequipped Mode S aircraft that is not resolved by TCAS. The aircraft have a horizontal time of closest approach (TCA) of 40 s and the minimum horizontal distance is 438 ft. TCAS predicts adequate vertical separation until 29 s when it issues a climb advisory. However, the climb advisory is too late and the encounter results in a vertical separation of 84 ft. ACAS X issues a DND at 16 s when the own aircraft's vertical rate is -1500 ft/min and resolves the encounter with 585 ft of vertical separation.

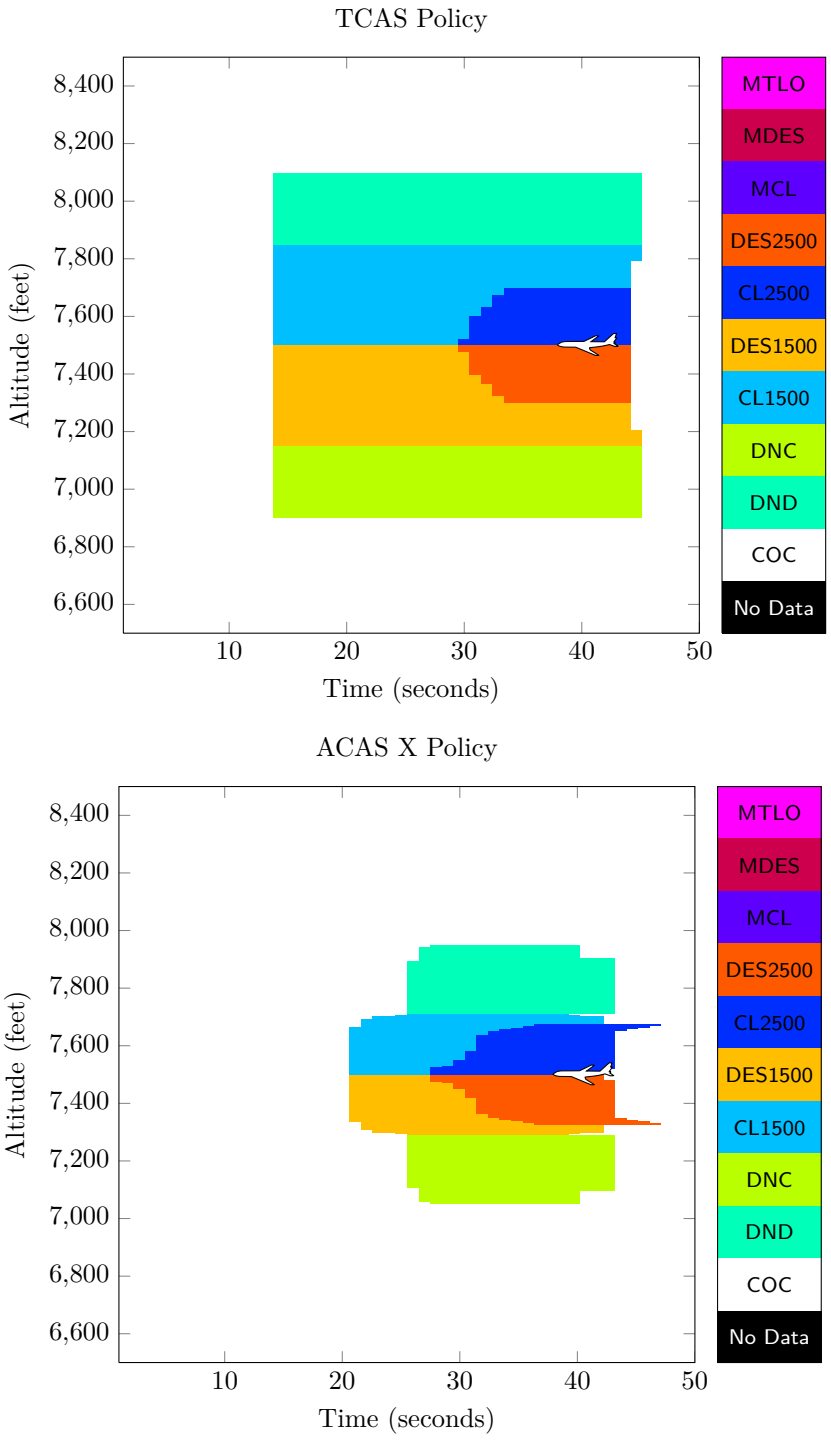
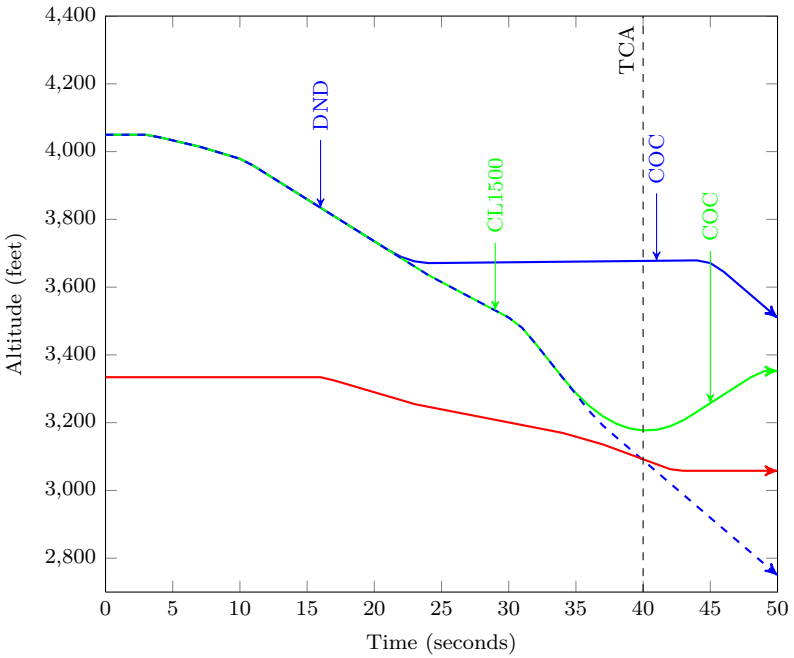
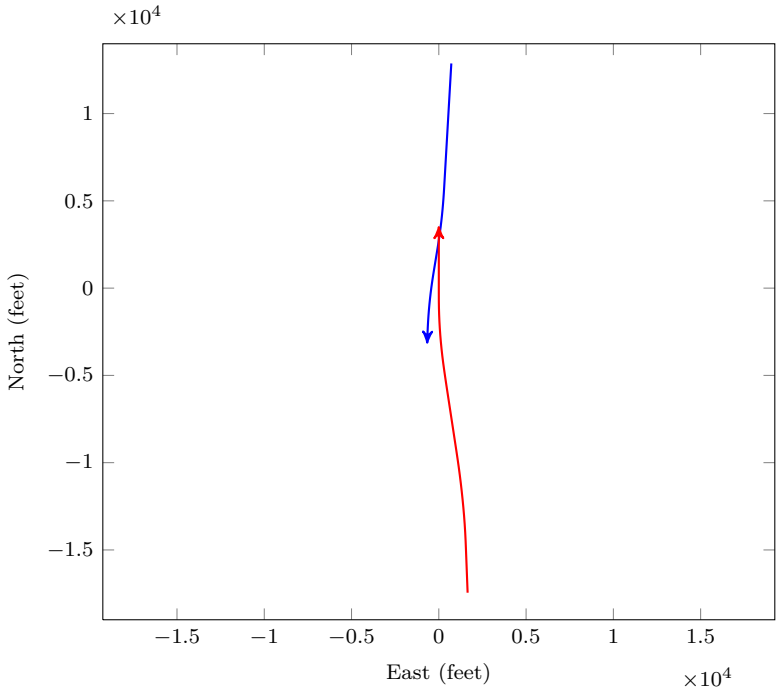


Figure 3-2: Policy plot of ACAS X and TCAS against an unequipped intruder. The intruder starts at 7500 ft and maintains level flight. The own aircraft is also initially level. TCA occurs at 40 s.



(a) Vertical profile for both ACAS and TCAS.



(b) Horizontal profile.

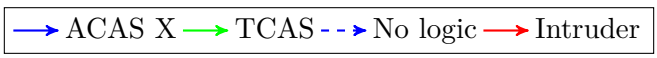


Figure 3-3: Example of a two aircraft encounter where ACAS X resolves the encounter and TCAS does not. The intruder is a Mode S unequipped aircraft.

Chapter 4

Coordination

Much of the development of ACAS X has focused primarily on encounters with a single unequipped intruder. If the intruder is equipped with a collision avoidance system, then safety can be significantly improved. However, the maneuvers recommended by the systems must be coordinated. If both the own aircraft and intruder issue the same advisory, then the likelihood of an NMAC increases. Coordination of advisories requires communication between aircraft or implicit coordination of the collision avoidance systems.

When discussing advisories, the terms compatible and coordinated are used interchangeably. This section also assumes that the own aircraft is equipped with a CAS and there is only one intruder that is also equipped with a CAS. Chapter 5 discusses situations with multiple intruders.

ACAS X must use a communication architecture compatible with TCAS. Therefore, limited information is shared between collision avoidance systems on different aircraft. The only shared information is a Vertical Resolution Advisory Complement (VRC), Cancel Vertical Resolution Advisory Complement (CVC), and Vertical Sense Bits (VSB). A VRC is transmitted to an intruder when the own aircraft has a resolution advisory. The VRC can only be a DO NOT CLIMB or a DO NOT DESCEND message and corresponds to the opposite sense of the advisory issued. For example, if TCAS selects a climb advisory against an intruder, the TCAS system sends a message to the intruder containing a DO NOT CLIMB VRC. A CVC is sent to cancel the

previous VRC message. The VSB is used to check consistency with the VRC and CVC fields before the coordination information is used. Full state information is not shared and is obtained from noisy sensors on the own aircraft and quantized altitude measurements from the intruder's transponder.

The sense of an advisory is determined by the direction of the commanded vertical rate. Every advisory has a sense of UP, DOWN, or NONE. If an advisory has a commanded vertical rate that is not bounded in the positive direction only, then its sense is UP. If the vertical rate is not bounded in the negative direction only, then its sense is DOWN. All other advisories have a sense of NONE. For example, a DND would have an UP sense while a COC would have a sense of NONE. Two advisories are compatible if the senses are different.

With no sharing of state information and a decentralized selection of actions, Dec-MDPs and Dec-POMDPs provide an appropriate way to model the problem. However, as discussed in Section 2.3.2, optimal solution methods and even approximate solution methods are not computationally tractable. Other methods include modeling the problem as a MMDP and a forced cooperation scheme similar to what TCAS uses. Both of these approaches have been previously applied to ACAS X. Kochenderfer and Chryssanthacopoulos found that the benefit of the added complexity of the MMDP was small and was outperformed by a simpler forced cooperation scheme [42].

Forced cooperation is discussed in greater detail in Section 4.1. An implicit coordination scheme is explored in Section 4.2 where advisories between two ACAS X equipped aircraft are guaranteed to issue compatible advisories under perfect state information. Section 4.3 presents an iterative policy approach that factors in maneuvering intruder dynamics. The remaining sections investigate the interoperability and robustness of different schemes and the chapter concludes with a summary and discussion.

4.1 Forced Cooperation

TCAS uses a forced cooperation scheme, which involves restricting the choice of advisories to those compatible with the advisory issued by the other aircraft. The method used on TCAS is quite robust, and no coordination failures are known to have occurred in TCAS operation (i.e., there has not been a situation where two coordinated aircraft have issued incompatible advisories). To determine which aircraft is forced to cooperate, a unique address is used to determine the master aircraft and slave aircraft. A slave aircraft is forced to issue only compatible advisories with a master aircraft. The only exception for TCAS is when the slave issues an advisory first. When a slave aircraft issues an advisory before a master, TCAS limits the master's first advisory to only compatible advisories with the slave.

The roles of the aircraft are determined based on the 24 bit ICAO Mode S address of the aircraft where the aircraft with the lower Mode S address is designated the master and the other aircraft is the slave. The aircraft share the sense of their advisories through the VRC. A forced cooperation scheme can be implemented on ACAS X by modifying the offline table and updating different modes online, or entirely online by raising the cost of incompatible advisories.

4.1.1 Offline Approach

Forced cooperation offline requires the addition of a state variable that represents the advisory sense of the other aircraft. The new state variable is the mode. Since a VRC is received when an intruder issues an advisory, the sense of the intruder's advisory is known, and thus the mode is known. If a VRC was not received, a belief over the mode could be maintained based on the maneuvers of the intruder aircraft.

For a basic forced cooperation scheme, only three states for the mode are needed:

- mode 0: intruder does not have an advisory or has an advisory with no sense,
- mode 1: intruder has an advisory with an up sense, and
- mode 2: intruder has an advisory with a down sense.

When the mode is one, all up sense advisories are penalized and when the mode is two, all down sense advisories are penalized. To simplify the integration of the mode state variable into the existing ACAS X framework, the dynamics used in the offline optimization assume that aircraft do not transition between modes. This implementation does not affect the unequipped performance of the logic.

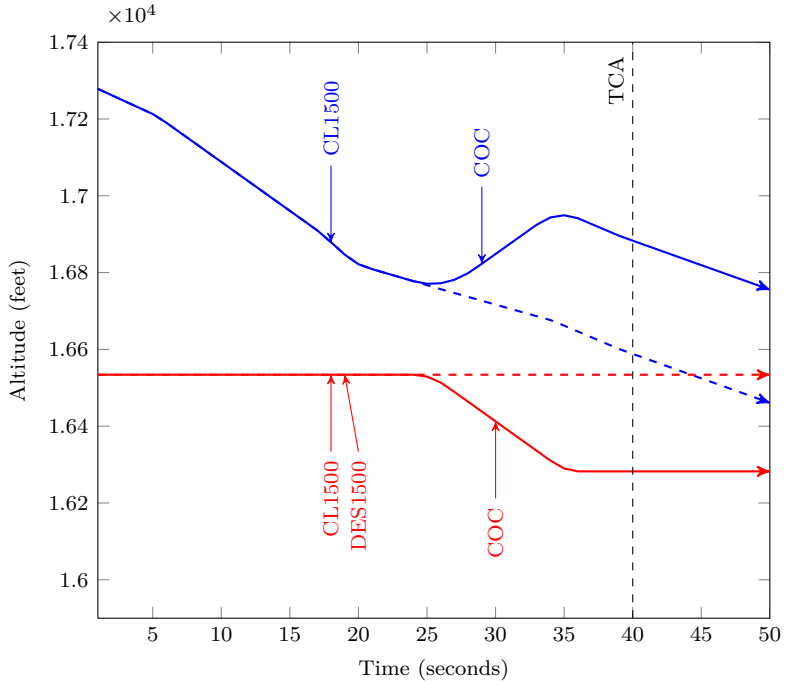
For basic forced cooperation, the master always remains in mode 0 while the slave aircraft changes mode based on received VRCs. For unequipped intruders, the own aircraft would always be in mode zero.

4.1.2 Online Approach

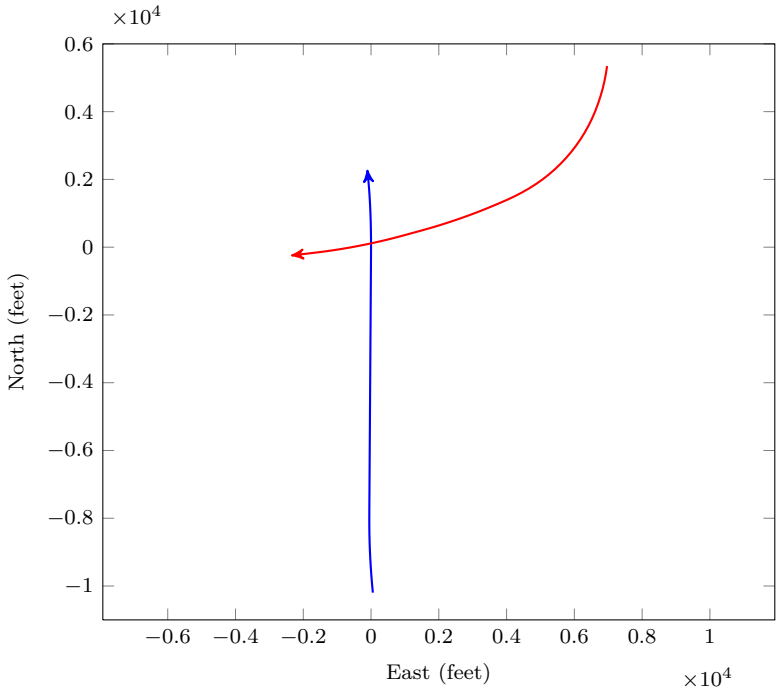
Forced cooperation online does not require any change to the offline table. All penalties are applied online based on received VRCs. For the basic scheme, the master aircraft never receives an online cost while the slave aircraft receives a cost of infinity for incompatible advisories. A negative effect of implementing forced cooperation online is since the offline optimization remains unchanged, the expected costs are computed assuming that future actions are unimpeded by cooperation restrictions dictated by another aircraft.

4.1.3 Example Encounter

Figure 4-1 is an example encounter with forced cooperation. Both aircraft are equipped with ACAS X using a basic forced cooperation scheme. The blue aircraft has the lower Mode S address and is the master. At 18 s both aircraft have CL1500 as the lowest cost action. At the next cycle, 19 s, the slave receives a DO NOT CLIMB VRC from the master. For the online approach, the slave performs the state-action value lookup as normal and then assigns all up sense advisories a cost of infinity. For the offline approach, the received VRC means the slave is now in mode one. The slave would then perform the state-action value lookup with a mode of one. In both cases, the lowest cost action for the slave at 19 s is DES1500. In this basic implementation, the master is not affected by received VRCs from the slave.



(a) Vertical profile.



(b) Horizontal profile.

Figure 4-1: Example of a two aircraft encounter where both aircraft are equipped with ACAS X using a forced cooperation scheme.

4.1.4 Forced Cooperation Improvements

Improvements can be made either offline or online to the basic forced cooperation scheme. This section describes four ways to consider the collision avoidance problem and improve upon basic forced cooperation.

Cooperative First Advisory. If the slave issues an advisory before the master, then it is likely that the slave is maneuvering to comply with its advisory. Reversing the advisory would lose any benefit from the maneuvering. Performance can be improved if the master is forced to be compatible with the slave when the slave issues an advisory first. This improvement can be implemented online by modifying the original online cost. The same three-mode scheme for the offline optimization could also be used, but the master aircraft would enter non-zero modes when the slave issues an advisory first.

Penalty for Master Aircraft. One way to factor in the dynamics of an equipped intruder is to penalize non-compatible advisories for the master aircraft. Similar to the cooperative first advisory improvement, when a VRC is received from an intruder, we can assume that the intruder is maneuvering due to its advisory. This improvement would result in fewer reversals when the intruder has a resolution advisory, but will lower the robustness to non-compliant intruders. The penalty for master aircraft can occur offline with two additional mode states or online with little change to the original online cost.

Forced Reversals. When two aircraft issue simultaneous advisories, the slave is forced to change its advisory. The next best action might be COC. An advisory immediately terminating and then reissuing later with the reversed sense would be confusing to a pilot. To avoid this scenario a large cost for COC can be added online when a reversal is forced due to cooperation with a master. This improvement could be implemented offline as well with no increase in state space size.

Additional Cooperation Costs. The offline dynamics assume an unequipped intruder. Not considering the dynamics of an equipped intruder can affect encounters where the relative altitude between aircraft is small. When coordinated encounters

occur where the intruder receives an advisory and the relative altitude is small, the logic can switch advisories before allowing the intruder aircraft to adequately respond.

Penalizing switching from an advisory after a received VRC for a period of time acts as a delay to allow the intruder to comply with its advisory. After a certain number of cycles, the own aircraft would have a better estimate of the state of the intruder. To maintain robustness to non-compliance, reversals to maintain advisories are penalized along with an exception to reversing due to maneuvering intruders.

Implementing the additional cooperation costs offline would require a very large increase in the size of the state space. This improvement should be implemented online. The number of cycles that the same VRC is received along with the intruder's vertical rate upon receiving a differing VRC needs to be tracked. The stored vertical rate is used to aid in detecting significant maneuvers of the intruder in the same sense as the advisory issued by the own aircraft.

4.1.5 Results

The variations of forced cooperation were simulated on 1.3×10^6 equipped vs. equipped encounters generated from the high fidelity encounter model discussed in Section 1.1.1. The variations implemented were:

- *No Strat:* no coordination strategy,
- *Basic FC:* the basic forced cooperation scheme with no improvements implemented online (same scheme as previously implemented for ACAS X [42]),
- *Basic FC Offline:* the basic forced cooperation scheme with no improvements implemented offline,
- *FC2:* forced cooperation online with the "cooperative first advisory," "penalty for master aircraft," and "forced reversals" improvements implemented online,
- *FC2 Offline:* forced cooperation offline with the "cooperative first advisory" and the "penalty for master aircraft" improvements implemented offline and the "forced reversals" improvement implemented online,

Table 4.1: Safety evaluation of forced cooperation coordination schemes

Scheme	Risk Ratio	Induced RR	State Space Size
No Strat	5.33×10^{-2}	3.13×10^{-2}	1.51×10^{7}
Basic FC	6.58×10^{-3}	4.56×10^{-3}	1.51×10^{7}
Basic FC Offline	6.69×10^{-3}	4.50×10^{-3}	4.53×10^{7}
FC2	2.92×10^{-3}	1.96×10^{-3}	1.51×10^{7}
FC2 Offline	2.28×10^{-3}	1.35×10^{-3}	7.55×10^{7}
Full FC	1.85×10^{-3}	1.01×10^{-3}	1.51×10^{7}
Full FC Offline	1.83×10^{-3}	9.95×10^{-4}	7.55×10^{7}

- *Full FC:* forced cooperation online with all improvements implemented online, and
- *Full FC Offline:* FC2 Offline with the "forced reversals" and "additional cooperation costs" improvements implemented online.

Table 4.1 presents the risk ratio, induced risk ratio, and state space size of the coordination schemes implemented. The additions to the basic forced cooperation scheme provide a large improvement in safety. Also, the offline implementation outperforms the online implementation as expected, except for the basic implementation which still has improved induced risk ratio performance. The improvement for the full offline implementation is small and comes with a significant increase in the size of the state space.

4.2 Implicit Coordination

If collision avoidance systems generate compatible advisories without any form of communication of action selection, then they are implicitly coordinated. ACAS X, like TCAS, is not implicitly coordinated with itself. Since altitude rates of the two aircraft are used instead of the relative altitude rate, the difference in the modeled dynamics of the aircraft result in a loss of symmetry in the offline table. If the offline table was symmetric across the encounter, then two ACAS X aircraft would be implicitly coordinated.

Implicit coordination of the first advisory can be tested for two ACAS X systems by assuming perfect state information and looking up various state-action values for

states from opposing perspectives. For example, assume there are two aircraft, A and B. Aircraft A has an altitude of h_{A} and an altitude rate of $\dot{h}_{\mathrm{A}}$. Aircraft B has an altitude of h_{B} and an altitude rate of $\dot{h}_{\mathrm{B}}$. Both aircraft would have the same τ and for determining the first advisory, both aircraft would be in an advisory state of COC, s^1_{RA}. Aircraft A would see the state of the system as $(h_{\mathrm{A}} - h_{\mathrm{B}}, \dot{h}_{\mathrm{A}}, \dot{h}_{\mathrm{B}}, \tau, \mathrm{s}^1_{\mathrm{RA}})$ while aircraft B would see the state as $(h_{\mathrm{B}} - h_{\mathrm{A}}, \dot{h}_{\mathrm{B}}, \dot{h}_{\mathrm{A}}, \tau, \mathrm{s}^1_{\mathrm{RA}})$. Performing lookups across the state space from opposing perspectives finds that ACAS X is not implicitly coordinated with itself.

Implicit coordination under perfect state information can be guaranteed by taking advantage of the symmetry of the action and state spaces of ACAS X and manipulating the state-action value lookups. To simplify the manipulation, the maintain advisory is removed from the action set. The remaining advisories have a distinct opposite and as a result every advisory state has a corresponding opposite as well. The opposing actions and advisory states are summarized in Table 4.2.

To guarantee implicit coordination, the lookup of the state-action values for the master is performed as normal. For the slave, the lookup of the state-action values is conducted from the opposite perspective and then the values are swapped for the opposite actions. Therefore, the best action for the master is guaranteed to be coordinated with the best action for the slave.

This implicit coordination scheme was simulated on 1.3×10^6 equipped vs. equipped encounters generated from the high fidelity encounter model discussed in Section 1.1.1. The simulations were conducted with noisy TCAS sensors and with no sensor noise and perfect vertical state information. All simulations were performed with the reduced action set. Any simulation with "FC" in the name was simulated with communication of the VRC and all other variations were simulated assuming no communication except for the Mode S address of the aircraft. The schemes simulated were:

- No Strat NN: no coordination strategy with no sensor noise and perfect vertical state information,

Table 4.2: Opposing actions and advisory states

Action / Advisory State	Corresponding Opposite
Actions	
COC	COC
DNC	DND
DES1500	CL1500
SDES1500	SCL1500
SDES2500	SCL2500
Advisory States	
COC	COC
DNC not responding	DND not responding
DNC responding	DND responding
DES1500 not responding	CL1500 not responding
DES1500 responding	CL1500 responding
SDES1500 not responding	SCL1500 not responding
SDES1500 responding	SCL1500 responding
SDES2500 not responding	SCL2500 not responding
SDES2500 responding	sCL2500 responding

- No Strat: no coordination strategy with noisy TCAS sensors,
- Impl NN: implicit coordination scheme with no sensor noise and perfect vertical state information,
- Impl: implicit coordination scheme noisy TCAS sensors,
- Impl NN with FC: Impl NN simulation will the full online forced cooperation scheme discussed in Section 4.1.5,
- Impl with FC: Impl simulation with the full online forced cooperation scheme discussed in Section 4.1.5,
- FC NN: same as Full FC from Section 4.1.5 with the reduced action set, no sensor noise, and perfect vertical state information, and
- FC: same as Full FC from Section 4.1.5 with the reduced action set.

The results for the simulations summarized in Table 4.3. The implicit coordination scheme outperforms no strategy by a factor of 6.5 with no noise; however the performance of the implicit coordination scheme degrades when sensor noise is introduced. Since the aircraft have different views of the world, implicit coordination can

no longer be guaranteed. The slave aircraft still determines the state-action values from the master's perspective, but must rely on its own state estimation to do so. The original table lookup with the full forced cooperation scheme outperforms the implicit coordination method.

The implicit coordination method performs table lookups at false operating points for the slave. If the table was symmetric, then the FC and Impl with FC simulations would be very similar. The lookup from the intruder's perspective, which is not the appropriate operating point from the offline optimization, is likely the reason there is a decrease in performance from the FC simulation to the Impl with FC.

Table 4.3: Safety evaluation of a simple implicit coordination scheme

Scheme	Risk Ratio	Induced RR
No Strat NN	5.63×10^{-2}	2.57×10^{-2}
No Strat	5.48×10^{-2}	3.32×10^{-2}
Impl NN	8.63×10^{-3}	4.24×10^{-3}
Impl	3.48×10^{-2}	1.91×10^{-2}
Impl NN with FC	4.07×10^{-3}	2.69×10^{-3}
Impl with FC	6.46×10^{-3}	4.56×10^{-3}
FC NN	1.95×10^{-3}	1.10×10^{-3}
FC	2.52×10^{-3}	1.64×10^{-3}

4.3 Iterative Policy Approach

The methods described thus far for coordination on ACAS X have not modeled the dynamics of an equipped intruder. Assuming an intruder is equipped with a CAS and the pilot complies with the suggested advisories would reduce robustness to non-compliant intruders and decrease safety for unequipped intruders. However, modeling a compliant, equipped intruder has potential to improve performance under normal, equipped-equipped scenarios. This section presents one way to develop a state-action value table by modeling an equipped intruder without taking an MMDP or Dec-POMDP approach. The new table can then be used in conjunction with a forced cooperation scheme to ensure compatible advisories.

The iterative policy approach (IPA) is similar to Joint Equilibrium-based Search for Policies (JESP) [73] and was inspired by level-k thinking [74]. With JESP, each

agent starts with a policy. All but one policy is held fixed while the remaining agent calculates the best policy in response to the others. Level-k thinking is a model of strategic behavior where decisions are based on beliefs about what other players will do.

During the offline optimization, instead of modeling the intruder as unequipped, the intruder can be modeled as if it was equipped with a variation of ACAS X. To perform this approach exactly, the state space would have to be expanded to track the intruder's s_{RA}. However, an approximate approach can be made by assuming the s_{RA} of the intruder is always COC and that the intruder immediately responds to an advisory. To combat the wrongly modeled pilot delay, the acceleration response to the advisories is reduced by a factor of eight. So, instead of responding with $1/4\,\text{g}$ acceleration after 5 s and $1/3\,\text{g}$ acceleration after 3 s, the pilot response is modeled as an immediate $1/32\,\text{g}$ or $1/24\,\text{g}$ acceleration.

To determine the best action for the intruder, the state-action values of the intruder are found by referencing a previously generated table. The intruder's view of the world is determined by transposing the own aircraft's state to the intruder's perspective.

Once a new table is generated, then that table can be used to generate another, improved table until a local optima is found. This approach was implemented starting with the original ACAS X table and a total of 7 iterations were performed. Since the intruder is modeled with equipped dynamics, the cost of an NMAC was increased to combat a conservative policy. Algorithm 3 is an outline for this approach. The algorithm is similar to Algorithm 1 in that a new, improved policy is generated each iteration. When calculating the new policy, any policy generation method such as value iteration or policy iteration can be used.

Figure 4-2 compares policies of the original ACAS X with IPA ACAS X. The policies were generated with no noise and perfect vertical rate estimation. The intruder starts at 7500 ft and maintains level flight. The own aircraft also maintains level flight. The IPA policy results in a smaller alerting region and a smaller region

Algorithm 3 IPA Policy Iteration

```
function IPAPolicyIteration(π_0)
    n ← 0
    N ← maximum number of iterations
    repeat
        Assume intruder has policy π_n with s_RA = COC
        Calculate π_{n+1}
        n ← n + 1
    until π_n = π_{n+1} or n ≥ N
    return π_{n+1}
```

of stronger advisories (i.e. CL2500). The IPA policy being more conservative than the original ACAS X logic is expected due to assuming an equipped intruder.

This IPA table was simulated on 1.3×10^6 equipped vs. equipped encounters and 1.5×10^6 equipped vs. unequipped encounters generated from the high fidelity encounter model discussed in Section 1.1.1. The IPA table was used in conjunction with the online full forced cooperation scheme. The results of the IPA are presented in Table 4.4. The results are compared to the online full forced cooperation scheme with the original ACAS X table. Since the IPA table is different from the original ACAS X table, the results against Mode S and Mode C intruders are also shown.

Table 4.4: Performance evaluation of the IPA table

	Equipped vs. Equipped		Equipped vs. Mode S		Equipped vs. Mode C	
Metrics	Original	IPA	Original	IPA	Original	IPA
RR	1.85×10^{-3}	1.73×10^{-3}	1.50×10^{-2}	1.62×10^{-2}	2.08×10^{-2}	2.60×10^{-2}
Induced RR	1.01×10^{-3}	8.84×10^{-4}	7.65×10^{-3}	9.01×10^{-3}	1.24×10^{-2}	1.80×10^{-2}
Pr(Alert)	2.21×10^{-1}	2.17×10^{-1}	2.09×10^{-1}	2.00×10^{-1}	2.19×10^{-1}	2.15×10^{-1}
Pr(Reversal)	1.10×10^{-4}	1.80×10^{-4}	4.52×10^{-4}	8.87×10^{-4}	5.03×10^{-4}	9.09×10^{-4}

The IPA table results a 6.5 % improvement in risk ratio with equipped intruders, but worse performance against unequipped intruders (8 % and 25 % for Mode S and Mode C intruders, respectively). The improvements gained for the equipped-equipped encounters is small. Also, since the offline dynamics assume a cooperative equipped intruder, we expect the robustness to non-compliant intruders to decrease. The robustness is discussed in Section 4.4.

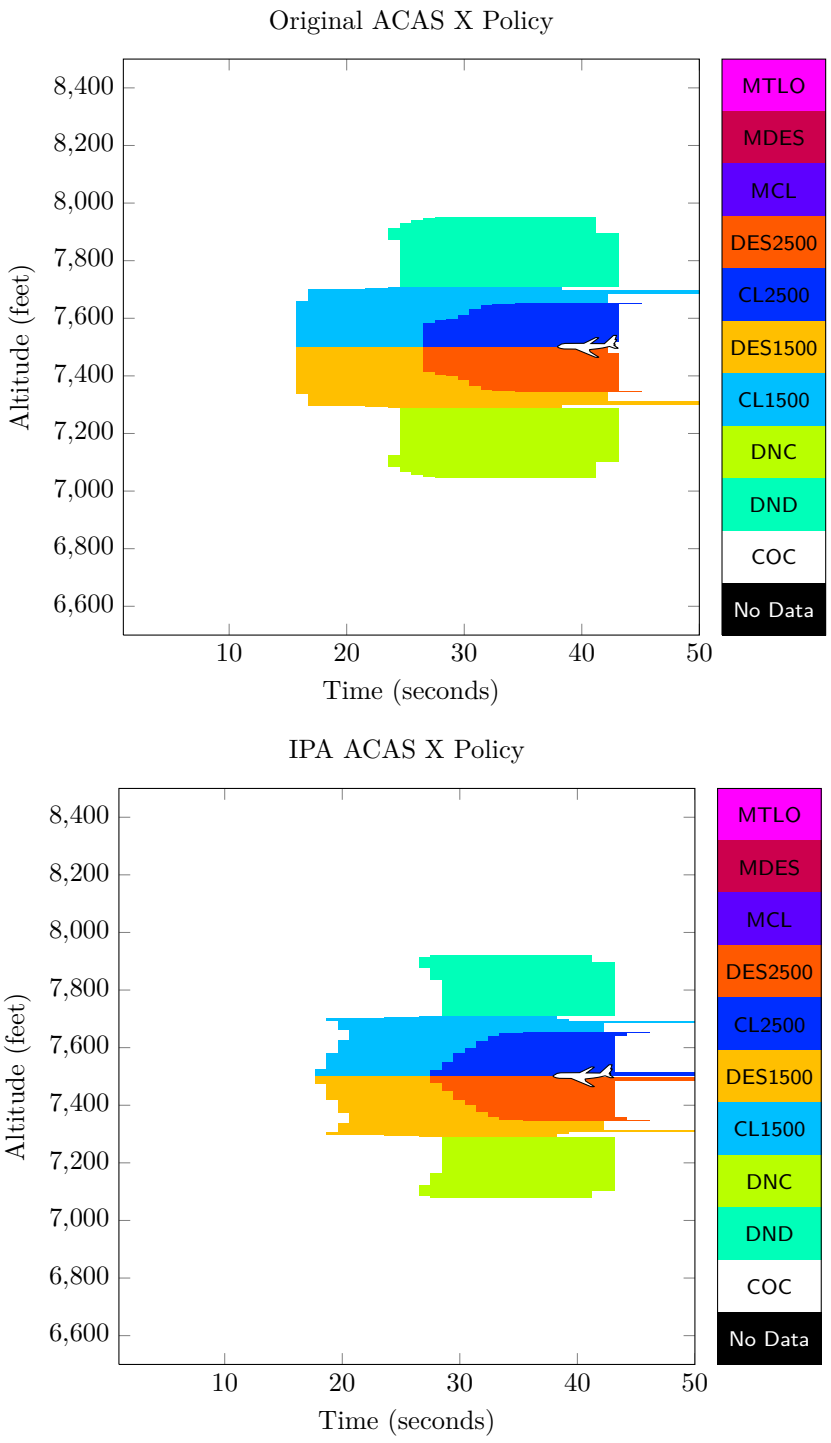


Figure 4-2: IPA and normal ACAS X policies for a master aircraft in an equipped-equipped encounter. The intruder starts at 7500 ft and maintains level flight. The own aircraft is also initially level. TCA occurs at 40 s.

4.4 Robustness Analysis

Coordinated encounters where an intruder does not respond to an advisory is an area of concern, especially after the Überlingen mid-air collision in 2002. Of the coordination strategies discussed, only the iterative approach assumes dynamics of an equipped aircraft. Therefore, robustness to non-compliant intruders is expected to be high.

If the own aircraft is the master, then forced cooperation should not be affected by a non-compliant intruder. However, the "additional cooperation costs" improvement discussed in Section 4.1.4 does consider the received VRC message. If the intruder is the master, forced cooperation limits when the own aircraft is allowed to reverse. The robustness of the full forced cooperation scheme is analyzed in this section and compared to TCAS.

The simulation setup is similar to the previous simulations except the pilot response time was varied. The pilot response was varied for either the master or the slave, but not at the same time. The normal pilot response of an initial delay of 5 s and all subsequent delays of 3 s was used unless specified. In Table 4.5 the pilot response column describes the pilot response varied for that simulation. "Normal" refers to a standard simulation. "None" means the pilot response was turned off for that particular aircraft and "$7-5$" means the initial delay was set to 7 s and the subsequent delay was set to 5 s for either the master or slave. The simulations were conducted using TCAS, and ACAS X. The results are summarized in Table 4.5.

In every scenario, ACAS X with full forced cooperation scheme significantly outperformed TCAS. When the master aircraft or the slave aircraft still responded to its advisories but with a delayed response, ACAS X saw a larger percent increase in risk ratio than TCAS. Yet, ACAS X was safer than TCAS in the same scenarios. As expected, IPA ACAS X degraded in performance and was worse than the original ACAS X when the intruder had a delayed response or was non-compliant. This degradation was expected due to assuming an equipped intruder that always responded to a fixed policy.

Table 4.5: Safety evaluation of coordination robustness

	TCAS		IPA ACAS X		ACAS X	
Pilot Response	Risk Ratio	Induced RR	Risk Ratio	Induced RR	Risk Ratio	Induced RR
Normal	2.54×10^{-3}	5.40×10^{-4}	1.73×10^{-3}	8.84×10^{-4}	1.85×10^{-3}	1.01×10^{-3}
7–5 Master	5.72×10^{-3}	1.75×10^{-3}	5.15×10^{-3}	4.04×10^{-3}	4.88×10^{-3}	3.69×10^{-3}
None Master	5.76×10^{-2}	2.62×10^{-2}	2.96×10^{-2}	1.97×10^{-2}	2.67×10^{-2}	1.50×10^{-2}
7–5 Slave	5.30×10^{-3}	1.94×10^{-3}	6.04×10^{-3}	4.88×10^{-3}	4.72×10^{-3}	3.55×10^{-3}
None Slave	3.98×10^{-2}	2.34×10^{-2}	2.42×10^{-2}	1.53×10^{-2}	2.19×10^{-2}	1.13×10^{-2}

4.5 Interoperability

To test the interoperability of ACAS X with TCAS, two different interoperability tests were conducted. Both simulations involved one ACAS X equipped aircraft that used the full forced cooperation scheme implemented online and one TCAS equipped aircraft. The Mode S addresses were varied to make the ACAS X equipped aircraft the master and vice versa. The results are summarized in Table 4.6. The master aircraft is the first aircraft and is referred to as AC 1 in the metrics.

Table 4.6: Performance evaluation of interoperability with standard TCAS sensor noise

Metrics	ACAS X Master	TCAS Master	Both ACAS	Both TCAS
RR	2.59×10^{-3}	2.09×10^{-3}	1.85×10^{-3}	2.54×10^{-3}
Induced RR	1.52×10^{-3}	1.05×10^{-3}	1.01×10^{-3}	5.40×10^{-4}
Pr(Alert)	4.89×10^{-1}	4.87×10^{-1}	2.21×10^{-1}	4.99×10^{-1}
Pr(Alert AC 1)	1.70×10^{-1}	4.71×10^{-1}	1.94×10^{-1}	4.72×10^{-1}
Pr(Alert AC 2)	4.73×10^{-1}	1.72×10^{-1}	1.97×10^{-1}	4.73×10^{-1}
Pr(Reversal)	5.99×10^{-4}	2.03×10^{-3}	2.08×10^{-3}	6.22×10^{-3}
Pr(Reversal AC 1)	2.30×10^{-5}	1.11×10^{-3}	1.10×10^{-4}	6.92×10^{-4}
Pr(Reversal AC 2)	5.94×10^{-4}	1.21×10^{-3}	1.99×10^{-3}	6.19×10^{-3}
Pr(Strengthening)	6.78×10^{-3}	6.24×10^{-3}	3.87×10^{-3}	1.08×10^{-2}
Pr(Strengthening AC 1)	1.67×10^{-3}	4.87×10^{-3}	2.08×10^{-3}	5.84×10^{-3}
Pr(Strengthening AC 2)	5.72×10^{-3}	1.75×10^{-3}	2.19×10^{-3}	6.30×10^{-3}

Since the full forced cooperation scheme implemented in ACAS X is very similar to that of TCAS, the interoperability of the two collision avoidance systems was not a concern. The simulation results support that initial hypothesis. The addition of ACAS X to the encounter either maintains or improves safety. The benefit of an ACAS X aircraft can be seen from the operational metrics. The probability of alert of the ACAS X aircraft in each scenario is much lower than the TCAS equipped aircraft. In addition, the probability of reversing and strengthening also decrease with the addition of an ACAS X aircraft.

TCAS does well at not inducing NMACs, but fails to resolve many more encounters than ACAS X. As a result, scenarios that involve ACAS X have a large decrease in unresolved NMACs and have more induced NMACs. The increase in induced NMACs is a problem that might warrant further investigation despite the improvement in overall safety.

4.6 Discussion

This chapter presented various approaches to extend ACAS X to handle encounters with a single intruder equipped with a collision avoidance system. Overall, the IPA performed best with an equipped intruder, but degraded performance for unequipped and non-compliant intruders. The full forced cooperation was a rather simple approach that performed extremely well in safety, operational metrics, and robustness. The offline method of the full forced cooperation did provide an improvement over the online method, but required the state space to be expanded by a factor of five.

Due to the simple nature of the online full forced cooperation scheme and the good performance, it is the best fit for ACAS X. The scheme is suboptimal in general, but is suitable for ACAS X for several reasons. A variation of this scheme has been used by TCAS for many years, provides robustness against non-compliant intruders, and allows for the offline development to be focused on encounters with unequipped intruders.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Multithreat

Multithreat encounters are rare, but they will increase in frequency as the airspace becomes more dense. During the development of ACAS X, there has been some research into unequipped multithreat scenarios [42], [75]. For multithreat scenarios involving equipped intruders, the coordination of advisories is important.

Previous data mining has documented 3803 multithreat encounters over a nine month period, of which 95 % were three aircraft encounters [4]. Most multithreat situations start as two aircraft encounters and evolve into a three or more aircraft encounter. Since a majority of actual multithreat encounters involve three aircraft, the multithreat logic for ACAS X must perform well with two intruder scenarios. However, all the approaches for three aircraft encounters should extend easily to an arbitrary number of intruders.

This chapter discusses the challenges with multiple unequipped and equipped intruders. First, we will discuss the problem in general and review potential solution methods. Then we will provide more detail on cost fusion techniques and apply this approach to ACAS X. The addition of an action specifically for multithreat encounters will be discussed. Finally, the section will end with results from simulations of ACAS X on realistic encounters from a high fidelity encounter model along with some stress testing results.

5.1 Potential Solution Methods

As discussed in Chapter 4, an appropriate way to model the problem would be as a Dec-MDP or a Dec-POMDP. However, the solution methods are computationally intractable. Also, as the number of aircraft increases, the computation required to generate a policy would also increase. Therefore, we need to consider methods that scale well to an arbitrary number of intruders.

TCAS treats each threat individually, with the same threat detection, initial sense selection, and initial strength selection logic that would be used with a single intruder. The multithreat portion of the logic attempts to reconcile the senses and strengths associated with each intruder before displaying a composite advisory to the pilots. When all threats have the same sense, the logic simply uses the individual advisory with the greatest strength. When the senses of the individual advisories differ, TCAS uses a set of rules to either (1) identify a single sense for all threats or (2) issue a "dual-negative advisory" that places vertical rate limits in both directions.

A similar approach could be used for ACAS X. However a large benefit of ACAS X over TCAS is the ability to accommodate the anticipated evolution of the airspace and re-optimize the logic as necessary with little development effort. Due to the complexity of the pseudocode with TCAS, it is difficult to modify the logic. If a command arbitration approach was used for ACAS X to handle multithreat scenarios, the ease of interchanging new logic tables might be lost. A specific example would be if a different action set is used. A command arbitration approach would likely result in advisory specific rules. Therefore, changing an advisory set will result in an overhaul of the multithreat logic.

The next section explains an alternative method where costs associated with individual intruders are fused.

5.2 Cost Fusion

An n-aircraft multithreat scenario can be separated into n single threat encounters. The single threat logic of ACAS X can be executed as a sub-agent for each individual threat. The outputs from the n single threat encounters could either be actions or state-action values. Research has shown that utilizing state-action values from sub-agents of a complex system can result in better performance than trying to fuse or arbitrate over suggested actions from the sub-agents [76], [77]. The process of using state-action values from sub-agents solving a portion of the problem is called cost fusion.

Cost fusion computes the state-action costs $C^*(s, a)$ for all actions a by using the state-action costs for intruder i, $C^*(s_i, a)$, assuming intruder i is the only threat. The state-action costs from multiple intruders are fused to arrive at the global state-action cost function $C^*(s, a)$. Fusing the costs requires defining a function f that combines costs associated with multiple intruders. That is,

$$C^*(s, a) = f(C^*(s_1, a), \ldots, C^*(s_N, a)), \tag{5.1}$$

where N is the number of intruders. After fusing the utilities, the optimal action is computed using

$$\pi^*(s) = \arg\min_a C^*(s, a). \tag{5.2}$$

The fusion of the costs occurs after the online and offline costs are combined. Therefore, $C^*(s_i, a)$ represents the sum of the offline and online costs of taking action a for intruder i alone. Previous work on ACAS X assumed that $C^*(s_i, a)$ represented just the offline cost of taking action a for intruder i alone.

Two utility fusion methods were studied for the multithreat logic. The first method, the min-sum strategy, defines f to be a summation:

$$C^*(s, a) = \sum_i C^*(s_i, a). \tag{5.3}$$

Defining f in this way leads to counting penalties such as alert costs and reversal costs

Table 5.1: Costs for a two-intruder example

Intruder	No alert	Climb	Descend
1	13	11	7
2	15	0	8
sum	28	**11**	15
max	15	11	**8**

multiple times. The cost of alerting, for example, would be reflected in the state-action costs for each intruder. Adding these utilities together amounts to incurring the alert cost multiple times, though in reality the collision avoidance system can only alert once at any given time. This may cause the system to delay issuing the alert. Delaying an alert can be undesirable because there may be fewer available options to resolve the conflict further into the encounter. When more intruders are present, the importance of alerting earlier is magnified.

The second method, the min-max strategy, avoids accumulating penalties for each intruder by defining f as follows:

$$C^*(s, a) = \max_i C^*(s_i, a). \tag{5.4}$$

Table 5.1 is an example contrived to illustrate the difference between the two methods. There are two intruders and three actions (no alert, climb, and descend) from which to select at the current time. The table shows the cost for each intruder and for each action. The min-sum method issues the climb advisory because it is effective in preventing conflict with the second intruder (hence the low cost), even though following the climb may lead to conflict with the first intruder. The min-max method selects the descend action because the higher cost for executing the descend is 8 while the highest cost for executing the climb is 11.

One property of the cost fusion methods is that they do not alert any earlier than the single-threat policy on which they are built. This may be undesirable because it may be necessary to alert a little earlier to pass above or below all intruders.

Theorem 1. *If the optimal action for each intruder $\pi^*(s_1), \ldots, \pi^*(s_N)$ is COC, then the min-sum and min-max decomposition methods will also have an optimal action of COC.*

Proof. Since $\pi^*(s_i) = \text{COC}$, then for all intruders i,

$$C^*(s_i, \text{COC}) \leq C^*(s_i, a), \forall a. \tag{5.5}$$

- *Min-Sum*

  Equation (5.5) implies

  $$C^*(s, \text{COC}) = \sum_i C^*(s_i, \text{COC}) \leq \sum_i C^*(s_i, a) = C^*(s, a), \forall a. \tag{5.6}$$

  Therefore, $\pi^*(s) = \arg\min_a C^*(s, a) = \text{COC}$.

- *Min-Max*

  The proof will be by contradiction. From the theorem statement, Eq. (5.5) still holds. Now, we will assume, with out loss of generality, that

  $$\pi^*(s) = \tilde{a} \neq \text{COC}. \tag{5.7}$$

  Therefore, $C^*(s, \tilde{a}) < \arg\max_i C^*(s_i, \text{COC})$. Based on the definition of f for the min-max method, we know that there exists an intruder k such that $C^*(s, \tilde{a}) = C^*(s_k, \tilde{a})$. From Eq. (5.5) we know

  $$C^*(s_k, \text{COC}) < C^*(s_k, \tilde{a}). \tag{5.8}$$

  Thus, $C^*(s, \tilde{a}) < \arg\max_i C^*(s_i, \text{COC})$ for all intruders such that $i \neq k$. With out lost of generality, assume that $C^*(s_j, \text{COC}) = \arg\max_i C^*(s_i, \text{COC})$. From Eq. (5.8), we know that $j \neq k$.

By Eq. (5.7), the definition of f, and Eq. (5.5),

$$C^*(s_k, \tilde{a}) > C^*(s_j, \tilde{a}) > C^*(s_j, \text{COC}). \tag{5.9}$$

However, we said that

$$C^*(s_k, \tilde{a}) = C^*(s, \tilde{a}) < \arg\max_i C^*(s_i, \text{COC}) = C^*(s_j, \text{COC}), \tag{5.10}$$

which gives us a contradiction.

□

Figure 5-1 shows the policies for the min-max and min-sum fusion methods. The intruders are at 7300 ft and 7700 ft and fly level. The horizontal geometry is head-on and TCA for both intruders is at 40 s. The own aircraft is initially level. As expected, double-counting costs makes the alerting region for the min-sum method smaller. The alerting region for the min-max method is similar to the individual policies, but still varies. The min-max method delays alerting a little longer when compared to the individual policies, especially when the own aircraft is between the intruders.

One problem with cost fusion techniques is the influence of non-critical agents on decisions. For example, suppose a second intruder enters an encounter, but is far away and does not pose a threat. The third aircraft is not a concern; however, fusing the costs of that intruder, despite it not being a factor, affects the action selection process. To mitigate the problem of non-threatening aircraft impacting the alert behavior, a form of arbitration is incorporated in which the costs of an intruder are only considered if the own aircraft would alert against that intruder in isolation. This process allows the cost fusion to alert at the same time as it would in an individual encounter. This arbitration is different than what was previously proposed for ACAS X [42].

Figure 5-2 shows an example of the two fusion strategies with this type of arbitration. Once arbitration is introduced, the policies between the two fusion schemes become very similar. The geometry of the encounter is the same as presented in Figure 5-1. Because the fusion scheme is only used when both aircraft are threats, there

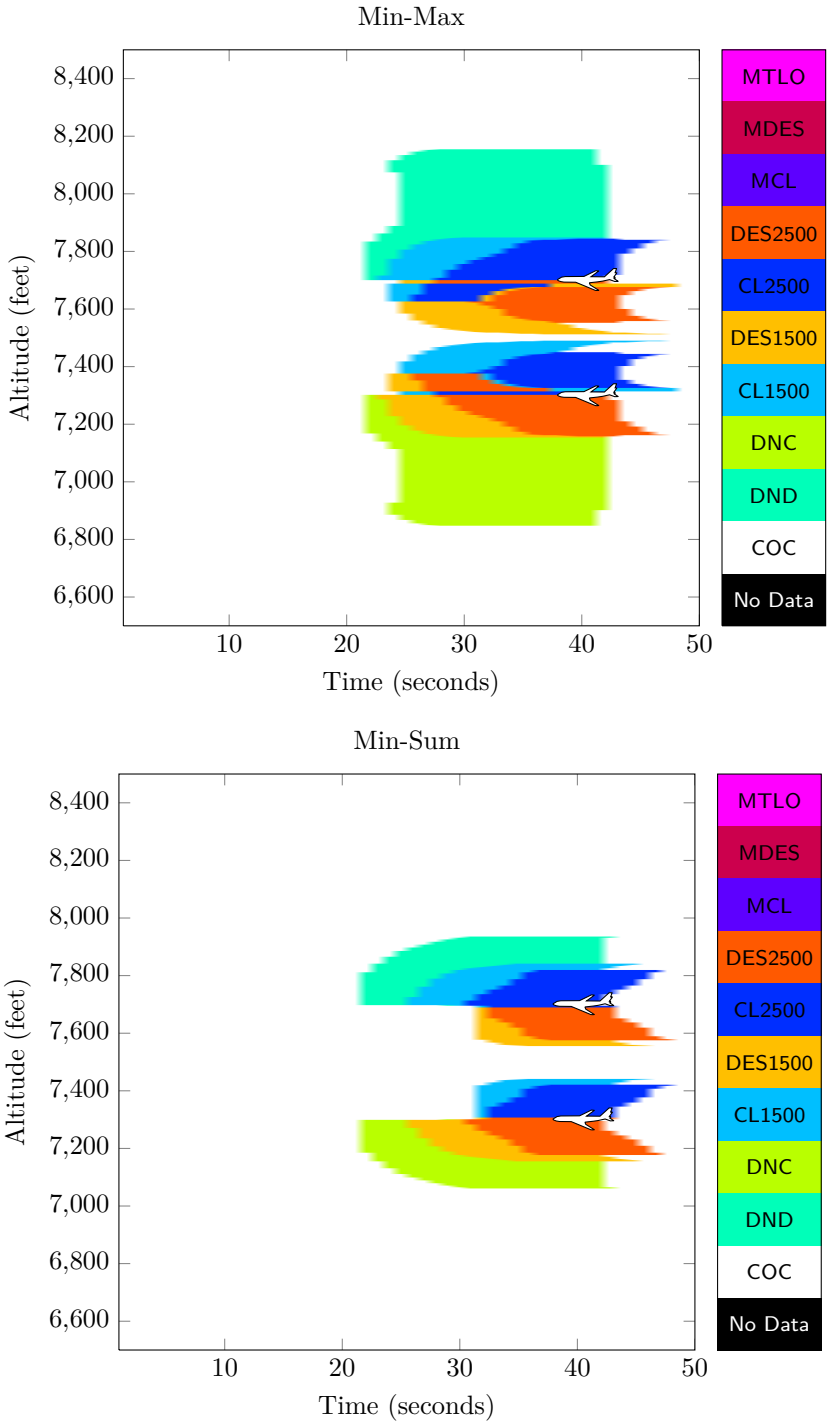


Figure 5-1: Policy plot using the min-sum and min-max cost fusion schemes. The intruders start at 7300 ft and 7700 ft and fly level. The horizontal geometry is head-on and both intruders' horizontal TCA to the own aircraft occur at 40 s. The own aircraft is initially in level flight.

is a delay when the own aircraft is in the middle. This affect can be easily identified when the intruders are separated by a greater altitude.

5.3 Multithreat Level-Offs

The single threat advisory set does not contain any dual-negative advisories that place speed limits in both senses. TCAS uses a variety of these advisories to resolve multithreat encounters. An example of a dual-negative advisory is a multithreat level-off (MTLO). An MTLO advisory limits the up sense to a maximum vertical rate of 250 ft/min and limits the down sense to a minimum vertical rate of −250 ft/min. An MTLO is often appropriate in sandwich encounters, where there are threats both above and below. A sandwich encounter often results in an up sense being issued against one intruder and a down sense against another. Often a single sense advisory is inadequate, and so an MTLO is issued, allowing the aircraft to pass between the two intruders.

An MTLO could be added to the advisory set, but then the logic could issue an MTLO during single intruder encounters, which is undesirable. We can determine when an MTLO should be issued based on a series of general, simple checks. When discussing the checks, the best action for each intruder when the intruder is considered in isolation is used and are refered to as individual actions. A total of three checks is used to determine when an MTLO should be issued for ACAS X.

Conflicting Actions. The first and possibly most obvious check, is that there has to be a minimum of two individual actions of opposite senses. For example, a CL1500 against intruder one and a DES1500 against intruder 2.

Conflicting VRCs. If an aircraft is receiving two or more conflicting VRCs and is forced to coordinate with the other aircraft, then an MTLO should be issued. For example, consider a three aircraft encounter where aircraft A has the lowest Mode S address, aircraft B has the second lowest Mode S address, and aircraft C has the highest Mode S address. If aircraft A has an up sense advisory against aircraft C, and aircraft B has a down sense advisory against aircraft C, then aircraft C is receiving DO

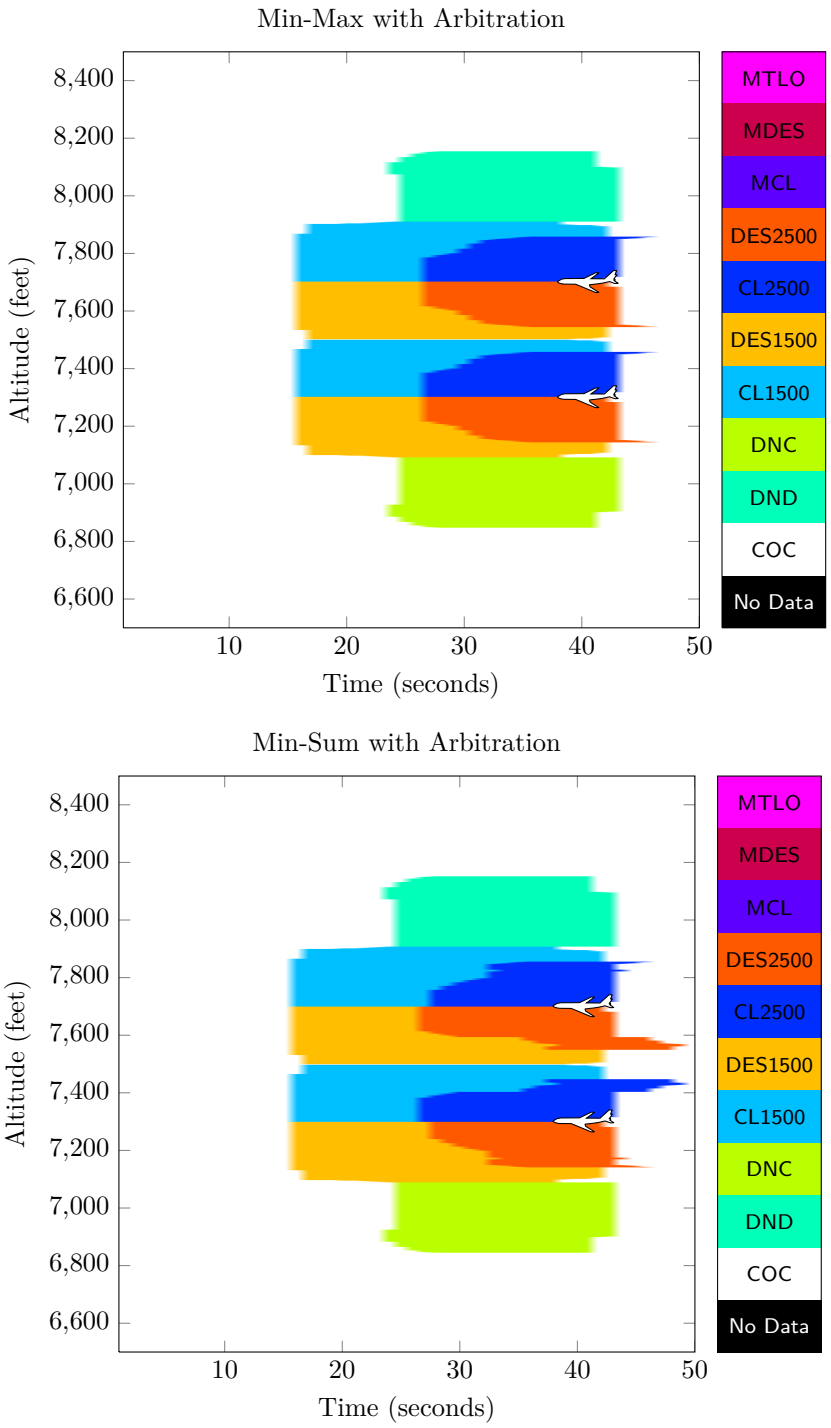


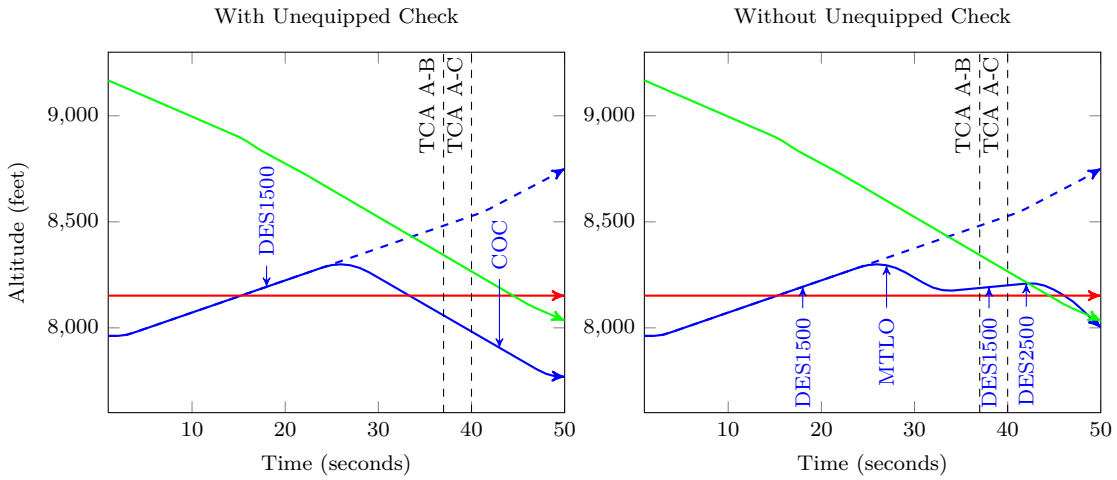
Figure 5-2: Policy plot using the min-sum and min-max cost fusion schemes with arbitration. The intruders start at 7300 ft and 7700 ft and fly level. The horizontal geometry is head-on and both intruders' horizontal TCA to the own aircraft occur at 40 s. The own aircraft is initially in level flight.

NOT CLIMB and DO NOT DESCEND VRCs from two master aircraft. Since pairwise coordination is required (using forced cooperation), the only logical choice is to issue an MTLO.

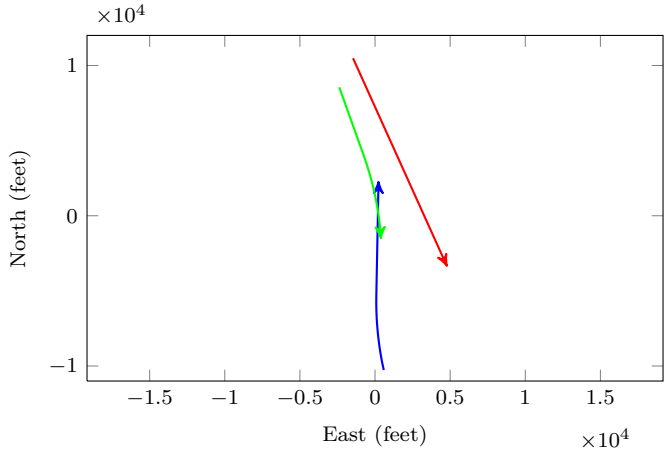
MTLOs with Unequipped Intruders. A situation where an MTLO is undesirable involves unequipped intruders. If an MTLO is issued when an unequipped intruder is within the vertical separation for an NMAC, then the own aircraft will fly level into a conflict. Prohibiting an MTLO within a vertical threshold of an unequipped intruder when the aircraft are not diverging at an acceptable rate ensures that separation will increase before an MTLO is issued.

Figure 5-3 shows an encounter where the prevention of an MTLO due to an unequipped intruder is helpful. In this example, an MTLO is prohibited if the aircraft is within 150 ft vertically of an unequipped intruder and the aircraft are not diverging by more than 250 ft/min. The intruders do not have enough separation at TCA to allow the own aircraft to split the two intruders. If the intruders were equipped with a CAS, then an MTLO could be issued because aircraft B would be expected to climb while aircraft C would be expected to descend. Since neither aircraft is equipped, then an MTLO should not be allowed. Not allowing the MTLO and keeping the descend advisory due to the unequipped check results in adequate separation and resolves the encounter.

Figure 5-4 shows an example policy of the complete ACAS X multithreat logic with the min-max strategy compared to a TCAS policy. Both of the policies were computed with no noise. The encounter geometries are the same as in Figure 5-1 and Figure 5-2. There are distinct differences in the ACAS X and TCAS policies. The most noticeable is the appearance that TCAS basically treats the two intruders as a single aircraft with a larger protection zone. The avoidance of sandwich encounters is very common with the TCAS multithreat logic. This tendency works well with two intruders, but tends to degrade performance when more intruders are introduced.



(a) Vertical profile.



(b) Horizontal profile.

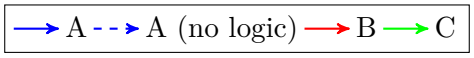


Figure 5-3: Example multithreat encounter where the "unequipped intruders" check plays a critical role.

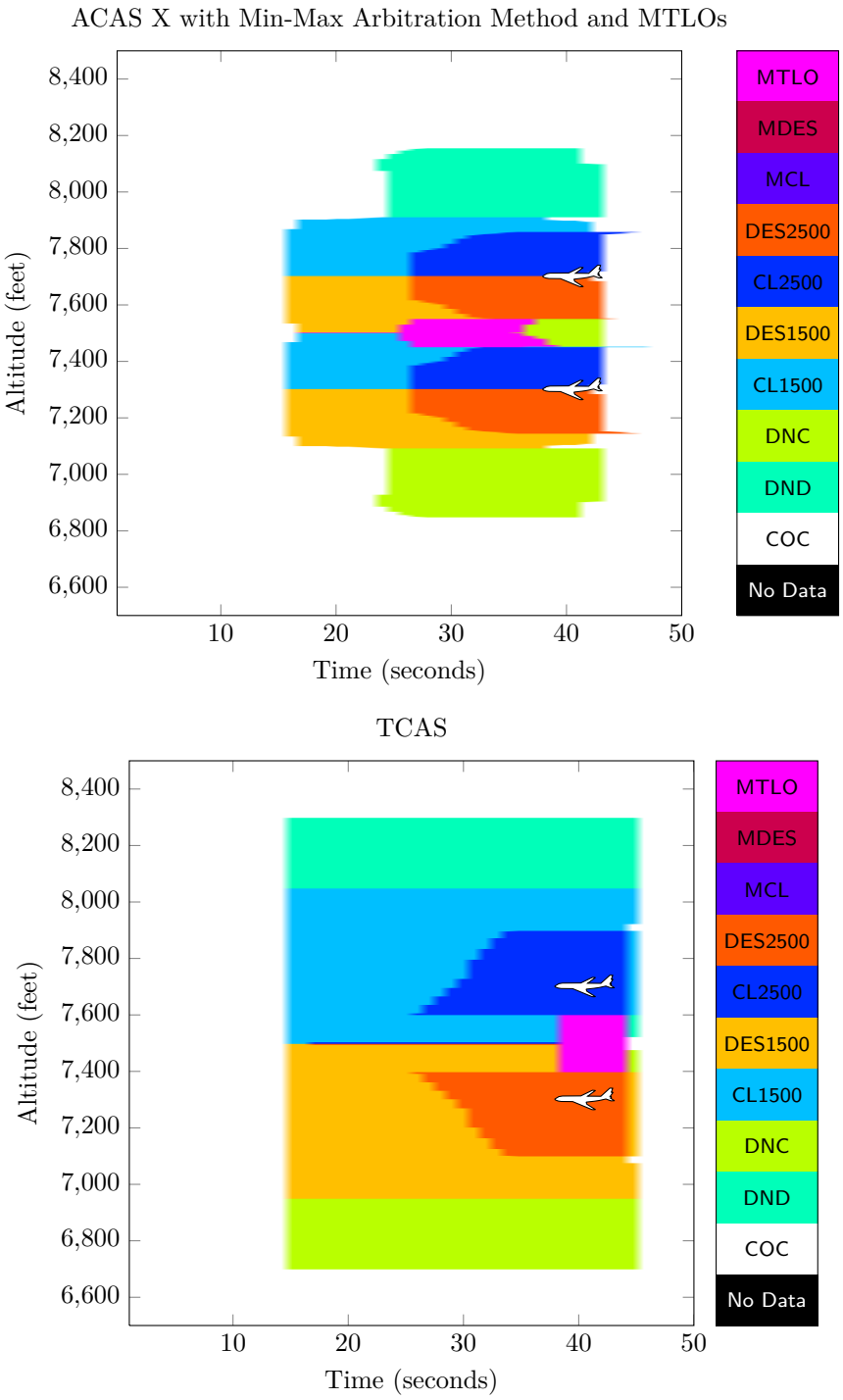


Figure 5-4: Policy plot of ACAS X using the min-max with arbitration and MTLOs method compared to TCAS. The intruders start at 7300 ft and 7700 ft and fly level. The horizontal geometry is head-on and both intruders' horizontal TCA to the own aircraft occur at 40 s. The own aircraft is initially in level flight.

5.4 Resolution Advisory State

The belief distribution over s_{RA} must always be updated based on the global action. Maintaining different beliefs over s_{RA} for each intruder could result in selecting inappropriate advisories. All s_{RA} updates occur with the global actions except when an MTLO is issued. If an MTLO is issued, the s_{RA} is updated with a DNC if the own vertical rate is greater than or equal to zero and DND if the own vertical rate is less than to zero.

Treating s_{RA} globally has some important consequences in multithreat situations. Suppose, for example, there are two intruders. The own aircraft alerts due to the first intruder. The second intruder is close, but is not considered a threat. Since s_{RA} is global, there is no longer a cost for alerting against the second intruder, which can result in premature alerting against the second intruder. However, this negative affect is relatively small.

5.5 Online Costs

The various online costs depend upon the previous advisory issued (see Section 3.6). In multithreat situations, each intruder is treated as if it were in isolation. If the intruder is not a threat, then the online costs are updated as if a clear of conflict was issued. If the best action globally is an MTLO and the intruder is a threat, then the online costs associated with that intruder are updated with either a DNC or DND, depending on the sense of the individual advisory. Otherwise, they are updated with the global action.

5.6 Multithreat Coordination

The communication between aircraft in a multithreat encounter is still limited to the VRC, VSB, CVC, and the Mode S address. Therefore, coordination must be done pairwise. A non-zero VRC will only be sent to an intruder if the individual advisory is an alert. For example, if the own aircraft issues a climb with respect to intruder A

and a COC with respect to B, then a do not climb VRC would be sent to A and no message would be sent to B.

If an MTLO is issued by the own aircraft, then different coordination messages are sent to the intruders. The coordination message sent to each intruder is based on the individual advisories toward those intruders considered in isolation. All intruders in which an up sense was desired would be sent a do not climb and all intruders in which a down sense was desired would be sent a do not descend VRC. If there was no advisory toward a particular intruder, then no message would be sent.

From a high level perspective, ACAS X does not differ greatly from TCAS in how it handles multithreat encounters. Both handle coordination in a pairwise manner and use single threat logic to resolve encounters individually, fusing those results to determine a global action. The fundamental differences are in the fusion process. However, those differences do not affect interoperability. Section 5.8.4 will discuss interoperability simulation results.

5.7 ACAS X Multithreat Execution

Figure 5-5 is a detailed block diagram on how ACAS X execution occurs with multiple intruders. A belief state and a set of online costs are kept separate for each intruder. The updates for the belief state and the online costs are determined from the outputs from the "Action Selection" component (i.e. individual actions and advisory).

The cost fusion and MTLO determination occurs in the "Action Selection" component. The summed action costs (online and offline) for each intruder are needed for this determination. From this diagram, we can see that the computational complexity for selecting an action at each cycle increases linearly with the number of intruders.

A plot of mean execution times to select an action is shown in Figure 5-6. The times were determined using the Min-Max with arbitration and MTLO fusion scheme. The timing included the "Individual Cost Estimation" and "Action Selection" components of Figure 5-5. Figure 5-6 validates that the complexity increases linearly with

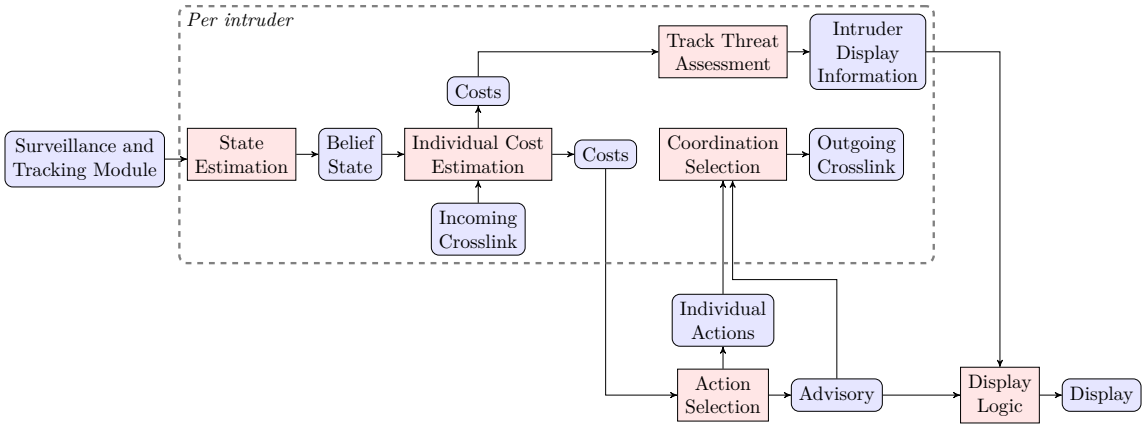


Figure 5-5: ACAS X multithreat execution.

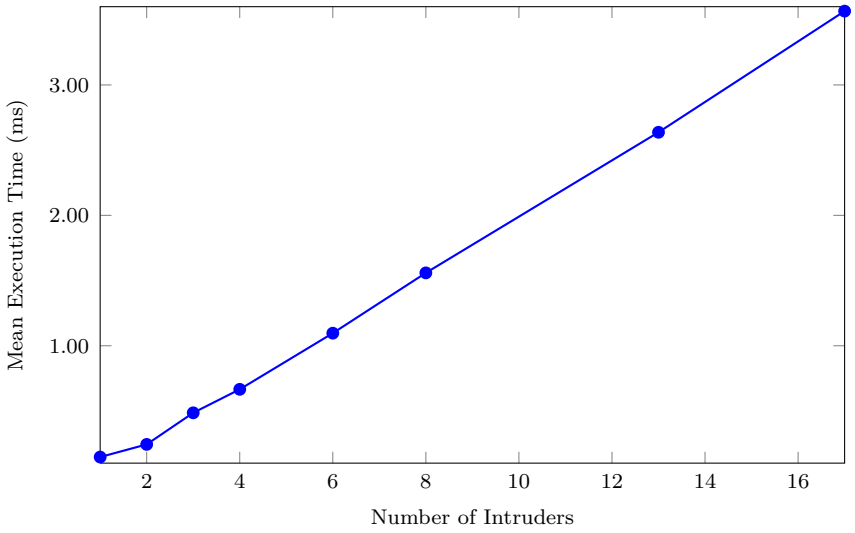


Figure 5-6: Mean execution time for ACAS X action selection for each cycle.

the number of intruders. The simulations were conducted on a single core of an Intel Xeon CPU operating at 3.33 GHz.

5.8 Simulation

Simulations were conducted using three-aircraft encounters generated from a high-fidelity model [4]. Two variations of stress test encounters were also used. In both stress test sets, the intruders are initially distributed (with some variance) around the first aircraft so that all aircraft will converge at the center in 40 s. For the first stress testing encounter set, there were no accelerations applied to the aircraft. For the

Table 5.2: Stress testing encounter set model parameters

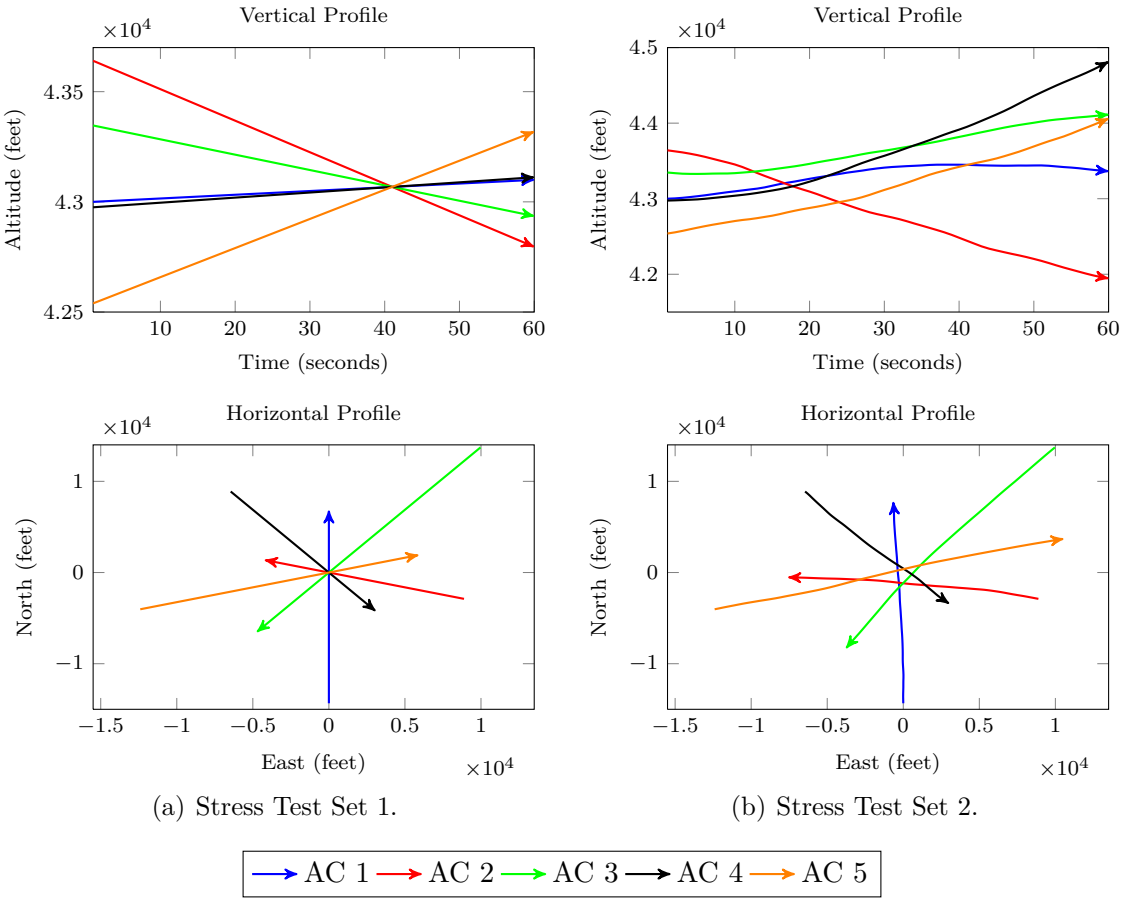
Parameters	Stress Test Set 1	Stress Test Set 2
Number of encounters	1×10^5	1×10^5
Initial vertical rate (ft/min)	Uniform $(-1000, 1000)$	Uniform $(-1000, 1000)$
Vertical acceleration noise (ft/s^2)	None	Normal $(0, 3)$
Initial speed (kt)	Uniform $(150, 450)$	Uniform $(150, 450)$
Horizontal acceleration noise (ft/s^2)	None	Normal $(0, 6)$

second set, the accelerations of the aircraft were white Gaussian noise sampled every second. The parameters of the two stress testing sets are summarized in Table 5.2. These models are not very realistic, but they provide a way to stress test the system to ensure proper behavior with an arbitrary number of intruders. An example of a stress testing encounter from both sets with 5 aircraft is shown in Figure 5-7.

If the encounter only involves one equipped aircraft, then the risk ratio is calculated with that aircraft. Since a CAS cannot affect the other two aircraft, only the equipped aircraft is considered when computing the NMAC rate. For ease of presenting the results, a three letter sequence will be used to signify the equipage of the aircraft involved. The aircraft listed first would have the lowest Mode S address and the aircraft listed last would have the highest Mode S address. For example, the sequence "XTS" signifies a three aircraft encounter of a master ACAS X aircraft, a slave TCAS aircraft and an unequipped aircraft with a Mode S transponder.

5.8.1 ACAS X vs. Two Unequipped Intruders

Table 5.3 summarizes the simulation results for scenarios where there were two unequipped intruders. "XSS Max" refers to the min-max fusion technique, while "XSS Sum" refers to the min-sum method. The min-max method outperforms the min-sum method for safety and alert rate and both methods outperform TCAS. The ACAS X min-max method reduces the risk ratio by 22 % while alerting 34 % less compared to TCAS.



(a) Stress Test Set 1.

(b) Stress Test Set 2.

Figure 5-7: Example of a five aircraft stress testing encounter. Left encounter is from stress test set 1 and right encounter is from stress test 2.

Table 5.3: Performance evaluation with two unequipped intruders

Metrics	XSS Max	XSS Sum	TSS
RR	9.34×10^{-2}	9.41×10^{-2}	1.20×10^{-1}
Induced RR	1.43×10^{-2}	1.45×10^{-2}	1.71×10^{-2}
Pr(Alert)	4.33×10^{-1}	4.33×10^{-1}	6.55×10^{-1}
Pr(Reversal)	3.03×10^{-3}	1.90×10^{-3}	1.24×10^{-2}
Pr(Strengthening)	2.97×10^{-2}	2.18×10^{-2}	6.20×10^{-2}

5.8.2 Two Equipped Aircraft and One Unequipped

Table 5.4 summarizes the simulation results for scenarios where two aircraft were equipped with ACAS X and there was a Mode S unequipped intruder. Both methods for ACAS X outperform TCAS in every category except for the Pr(Induced NMAC). A large portion of the induced NMACs occur when an advisory is issued due to the other equipped aircraft and causes a maneuver toward the unequipped aircraft. Often an MTLO is issued which would resolve the encounter if the aircraft was equipped, but because it is not, the aircraft continue on their paths and result in an NMAC.

The min-max method reduced the risk ratio by 11 %. The safety improvement of was achieved with a 27 % reduction in alerts, 82 % reduction in reversals, and a 48 % reduction in strengthenings compared to TCAS.

Table 5.4: Performance evaluation with two equipped aircraft and one unequipped intruder

Metrics	XXS Max	XXS Sum	TTS
RR	1.01×10^{-1}	1.01×10^{-1}	1.13×10^{-1}
Induced RR	1.50×10^{-2}	1.48×10^{-2}	1.26×10^{-2}
Pr(Alert)	5.88×10^{-1}	5.88×10^{-1}	8.03×10^{-1}
Pr(Reversal)	2.22×10^{-3}	2.13×10^{-3}	1.22×10^{-2}
Pr(Strengthening)	4.48×10^{-2}	3.38×10^{-2}	8.65×10^{-2}

5.8.3 Three Equipped Aircraft

Table 5.5 summarizes the simulation results for scenarios with three ACAS X aircraft. ACAS X outperforms TCAS in every metric and the min-max fusion method outperforms the min-sum method in the safety metrics. The min-max fusion method resulted in a reduction in Risk Ratio over TCAS by 8 % and a reduction in induced NMACs by 53 %. The improvements to the operational metrics by ACAS X was more

significant with a reduction in alerts by 26 %, a reduction in reversals by 66 %, and a reduction in strengthenings by 18 % compared to TCAS.

Table 5.5: Performance evaluation with three equipped aircraft

Metrics	XXX Max	XXX Sum	TTT
RR	8.22×10^{-2}	8.23×10^{-2}	8.92×10^{-2}
Induced RR	1.72×10^{-3}	1.82×10^{-3}	3.68×10^{-3}
Pr(Alert)	6.09×10^{-1}	6.09×10^{-1}	8.28×10^{-1}
Pr(Reversal)	2.99×10^{-3}	2.95×10^{-3}	8.86×10^{-3}
Pr(Strengthening)	5.46×10^{-2}	4.46×10^{-2}	6.68×10^{-2}

5.8.4 Interoperability

Interoperability was tested in both two and three aircraft scenarios by permuting the master-slave relationship among the aircraft. The results for the two equipped aircraft scenarios are summarized in Table 5.6, and the three equipped aircraft scenario results are presented in Table 5.7. All simulations used the min-max fusion method on ACAS X.

Similar to the results of Section 5.8.2, introducing an ACAS X equipped aircraft increases safety and outperforms TCAS in all operational metrics. However, it increases the Pr(Induced NMAC) slightly over the TTS scenario, but reduces it compared to the XXS scenario. Examining the Pr(NMAC) for each equipped aircraft in the individual scenarios shows that the ACAS X aircraft is always safer than a TCAS equipped aircraft while alerting much less. For the three equipped interoperability scenarios, the introduction of an ACAS X aircraft improves the overall safety. Safety is improved as more ACAS X aircraft are involved in the encounter.

Table 5.6: Performance evaluation of interoperability with two equipped aircraft and one unequipped intruder

Metrics	XTS	TXS	XXS	TTS
RR	1.03×10^{-1}	1.10×10^{-1}	1.01×10^{-1}	1.13×10^{-1}
Induced RR	1.22×10^{-2}	1.48×10^{-2}	1.50×10^{-2}	1.26×10^{-2}
Pr(NMAC AC 1)	9.87×10^{-4}	1.14×10^{-3}	9.96×10^{-4}	1.12×10^{-3}
Pr(NMAC AC 2)	1.04×10^{-3}	1.02×10^{-3}	9.82×10^{-4}	1.07×10^{-3}
Pr(Alert)	7.52×10^{-1}	7.54×10^{-1}	5.88×10^{-1}	8.03×10^{-1}
Pr(Alert AC 1)	4.18×10^{-1}	6.67×10^{-1}	4.41×10^{-1}	6.69×10^{-1}
Pr(Alert AC 2)	6.63×10^{-1}	4.15×10^{-1}	4.39×10^{-1}	6.66×10^{-1}
Pr(Strengthening)	6.68×10^{-2}	6.58×10^{-2}	4.48×10^{-2}	8.65×10^{-2}

Table 5.7: Performance evaluation of interoperability with three equipped aircraft

Metrics	XXT	XTX	TXX	XTT	TXT	TTX	XXX	TTT
RR	8.60×10^{-2}	8.74×10^{-2}	8.44×10^{-2}	8.73×10^{-2}	8.69×10^{-2}	8.84×10^{-2}	8.22×10^{-2}	8.92×10^{-2}
Induced RR	5.18×10^{-3}	6.50×10^{-3}	3.25×10^{-3}	4.95×10^{-3}	4.39×10^{-3}	5.96×10^{-3}	1.72×10^{-3}	3.68×10^{-3}
Pr(NMAC AC 1)	8.66×10^{-4}	8.64×10^{-4}	8.62×10^{-4}	8.65×10^{-4}	8.79×10^{-4}	8.73×10^{-4}	8.29×10^{-4}	8.91×10^{-4}
Pr(NMAC AC 2)	8.42×10^{-4}	9.01×10^{-4}	8.41×10^{-4}	8.87×10^{-4}	8.63×10^{-4}	9.02×10^{-4}	8.22×10^{-4}	8.96×10^{-4}
Pr(NMAC AC 3)	8.86×10^{-4}	8.70×10^{-4}	8.42×10^{-4}	8.79×10^{-4}	8.79×10^{-4}	8.91×10^{-4}	8.27×10^{-4}	9.03×10^{-4}
Pr(Alert)	7.75×10^{-1}	7.75×10^{-1}	7.75×10^{-1}	8.28×10^{-1}	8.29×10^{-1}	8.28×10^{-1}	6.09×10^{-1}	8.28×10^{-1}
Pr(Alert AC 1)	4.26×10^{-1}	4.25×10^{-1}	6.76×10^{-1}	4.03×10^{-1}	6.79×10^{-1}	6.78×10^{-1}	4.49×10^{-1}	6.83×10^{-1}
Pr(Alert AC 2)	4.26×10^{-1}	6.75×10^{-1}	4.25×10^{-1}	6.78×10^{-1}	4.04×10^{-1}	6.77×10^{-1}	4.48×10^{-1}	6.82×10^{-1}
Pr(Alert AC 3)	6.75×10^{-1}	4.25×10^{-1}	4.25×10^{-1}	6.79×10^{-1}	6.78×10^{-1}	4.03×10^{-1}	4.48×10^{-1}	6.83×10^{-1}
Pr(Strengthening)	6.19×10^{-2}	6.06×10^{-2}	5.94×10^{-2}	6.69×10^{-2}	6.54×10^{-2}	6.50×10^{-2}	5.46×10^{-2}	6.68×10^{-2}

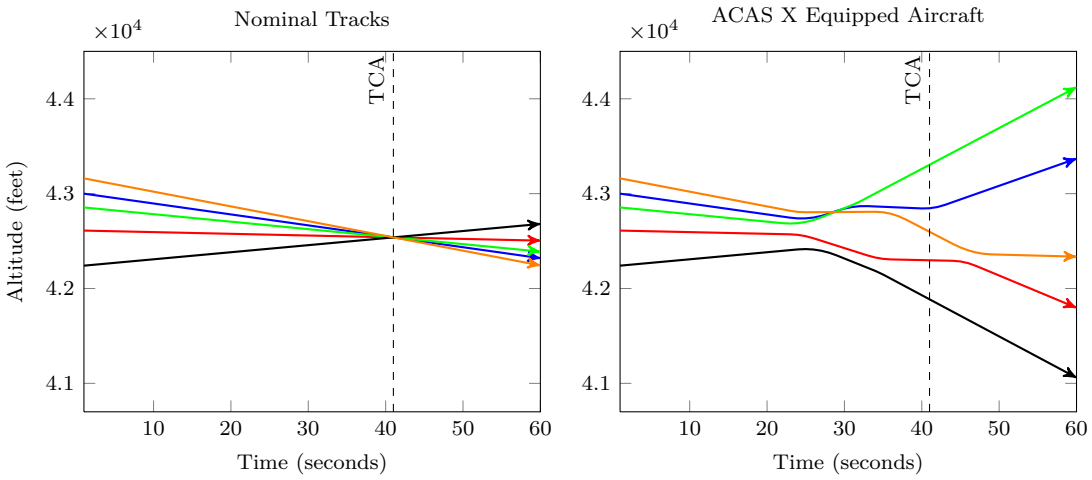
5.8.5 Stress Testing

Scenarios were generated that included 3 to 10 aircraft. As the number of aircraft increases, the computation scales linearly. For the stress testing, the aircraft were either all equipped with ACAS X or all equipped with TCAS. The ACAS X aircraft used the min-max fusion method. An example of a resolved encounter where all aircraft are equipped with ACAS X from stress test 1 is shown in Figure 5-8. The nominal vertical profile is shown on the left, while the vertical tracks with ACAS X equipped aircraft is on the right. For this encounter, ACAS X was able to successfully resolve the situation so no aircraft resulted in a NMAC. The minimum vertical distance between any aircraft at TCA is 248 ft. The advisories for each aircraft are omitted to avoid clutter.

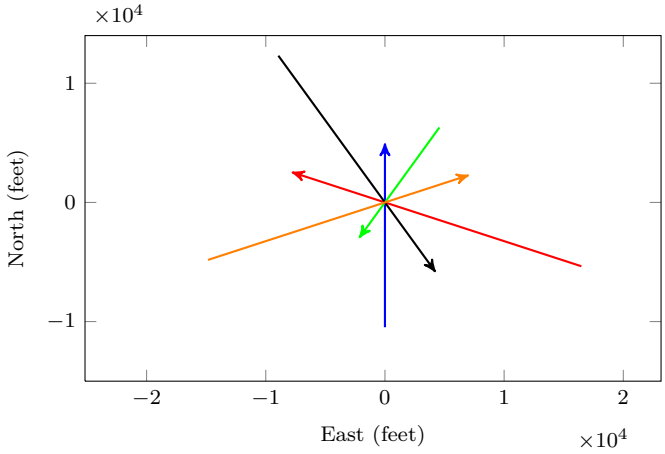
Figure 5-9 summarizes the results of stress testing ACAS X compared against TCAS. ACAS X significantly outperforms TCAS for encounters involving more than 3 aircraft. For stress test set 2, the encounters are not guaranteed to have an n-aircraft conflict at TCA due to the white noise accelerations. ACAS X is able to resolve these encounters considerably better than TCAS.

5.9 Discussion

This chapter explored options for expanding the ACAS X logic to multiple intruders. The result was improved performance over TCAS in almost all metrics evaluated. ACAS X outperformed TCAS in all metrics for every scenario.



(a) Vertical profile.



(b) Horizontal profile.

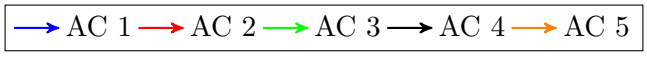


Figure 5-8: Example of a resolved five aircraft stress testing encounter by ACAS X.

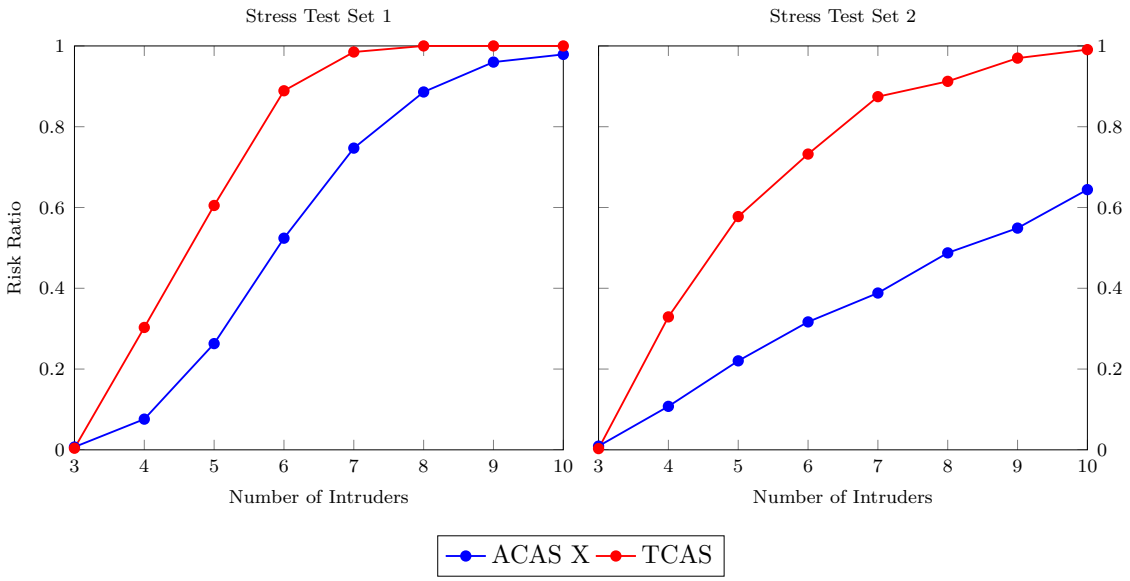


Figure 5-9: Risk ratio vs. number of intruders for both stress testing encounter sets.

Sandwich encounters are particularly challenging to resolve. The costs cannot be fused to gain an earlier alert in multithreat situations. A potential solution to the current cost fusion approach could be to reward alerting once multiple intruders reach a certain threshold. However, the current method still outperforms TCAS on realistic scenarios derived from an encounter model. In addition, stressing the logic showed a large increase in safety over TCAS and promising results for difficult encounters.

Chapter 6

Conclusion

6.1 Summary

The next generation air traffic collision avoidance system, ACAS X, was originally designed for single unequipped intruders. This thesis investigated different methods to extend ACAS X to coordinated encounters and multiple equipped intruders. The research was broken up into two separate parts—coordination and multithreat.

Various methods were explored for coordinated encounters involving two equipped aircraft. An iterative policy approach resulted in the best performance for equipped encounters, but degraded safety in unequipped scenarios. A forced cooperation scheme was implemented both offline and online. The offline approach outperformed the online approach, but required a large increase in state space size. Overall, the online forced cooperation scheme is a simple approach, performed better than TCAS, integrates well with TCAS aircraft, and provides robustness to non-compliant intruders.

To extend ACAS X to handle multiple intruders, cost fusion was explored with two different fusion methods. The cost fusion approach was modified to only fuse costs of intruders that were threats when considered in isolation. A multithreat specific action, an MTLO, was also added as an option for ACAS X. An MTLO is issued if a series of general, simple checks are passed. The Min-Max scheme with MTLOs performed well and outperformed TCAS.

The multithreat logic computation scales linearly with the number of intruders. This scalability enabled us to stress test the coordination and multithreat logic on scenarios involving up to ten intruders. The logic significantly outperformed TCAS with four or more intruders.

6.2 Further Work

The suggested online full forced cooperation scheme performs well, yet has a higher induced NMAC rate than TCAS. In addition, the induced NMAC rate for the two equipped multithreat scenarios was also higher than that of TCAS. Induced NMACs are viewed worse than unresolved NMACs by the community. Therefore, despite the improved overall safety, further studies into reducing the induced NMAC is needed.

The focus of this thesis was to develop safe and operationally acceptable coordination and multithreat strategies for ACAS X. During this process, a goal was to not increase the size of the state space. Further work exploring the benefits of expanding the state space should be conducted.

Expanding the state space to include multiple modes for equipped and unequipped intruders is a potential path. However, this approach would require the state space to at least double. In addition, expanding the state space to handle more intruders should be explored. As technology progresses, the storage capacity for the offline table will only increase, enabling a larger state space.

Another area to explore for coordination is generalizing the forced cooperation approach to maneuvers not in the vertical plane. Restricting the scope of actions to the vertical plane makes it easy to label cooperative and non-cooperative advisories. The forced cooperation scheme can work in more complex domains; however, cooperative and non-cooperative advisories would have to be defined.

The multithreat logic can be further explored in many ways. First, a more principled way of issuing an MTLO should be investigated. Simple, general checks provide a quick and easy way to implement multithreat specific advisories; however, other approaches that take advantage of the offline optimization more might perform better.

Another area to investigate with the multithreat logic is a possible real time solution method once encounters enter a multithreat situation. This would not require an addition to the state space, but would add more computation per action selection.

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] International Civil Aviation Organization, "Surveillance, radar and collision avoidance," in *International Standards and Recommended Practices*, 4th, vol. IV, annex 10, 2007.

[2] RTCA, *Minimum operational performance standards for Traffic Alert and Collision Avoidance System II (TCAS II)*, DO-185B, RTCA, Inc., Washington, D.C. 2008.

[3] 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.

[4] T. B. Billingsley, L. P. Espindle, and J. D. Griffith, "TCAS multiple threat encounter analysis," Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-359, 2009.

[5] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179–189, 2000. DOI: 10.1109/6979.898217.

[6] C. Scholten, "Elements for a new departure in air traffic control," PhD thesis, Technical University of Eindhoven, 1969.

[7] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986. DOI: 10.1177/027836498600500106.

[8] O. Khatib and L. Maitre, "Dynamic control of manipulators operating in a complex environment," in *Symposium on Theory and Practice of Robots and Manipulators*, 1980.

[9] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *IEEE International Conference on Robotics and Automation*, 1991.

[10] K. Zeghal, "A review of different approaches based on force fields for airborne conflict resolution," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 1998.

[11] E. Lalish and K. A. Morgansen, "Decentralized reactive collision avoidance for multivehicle systems," in *IEEE Decision and Control Conference*, 2008.

[12] ——, "Distributed reactive collision avoidance," *Autonomous Robots*, vol. 32, no. 3, pp. 207–226, 2012. DOI: 10.1007/s10514-011-9267-7.

[13] W. Kelly and M. Eby, "Advances in force field conflict resolution algorithms," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2000.

[14] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, "A probabilistic approach to aircraft conflict detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 199 –220, 2000. DOI: 10.1109/6979.898224.

[15] K. D. Bilimoria, "A geometric optimization approach to aircraft conflict resolution," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2000.

[16] G. Dowek, A. Geser, and C. Munoz, "Tactical conflict detection and resolution in a 3-D airspace," in *USA/Europe Air Traffic Management R & D Seminar*, 2001.

[17] G. Dowek, C. Munoz, and V. Carreno, "Provably safe coordinated strategy for distributed conflict resolution," in *AIAA Guidance Navigation, and Control Conference and Exhibit*, 2005.

[18] S. Luongo, F. Corraro, U. Ciniglio, V. Di Vito, and A. Moccia, "A novel 3D analytical algorithm for autonomous collision avoidance considering cylindrical safety bubble," in *IEEE Aerospace Conference*, 2010.

[19] R. Chamlou, "Future airborne collision avoidance — design principles, analysis plan and algorithm development," in *Digital Avionics Systems Conference*, 2009.

[20] E Frazzoli, Z.-H. Mao, J.-H. Oh, and E Feron, "Resolution of conflicts involving many aircraft via semidefinite programming," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 1, pp. 79–86, 2001. DOI: 10.2514/2.4678.

[21] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *European Control Conference*, 2001.

[22] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conference*, 2002.

[23] T. Schouwenaars, J. How, and E. Feron, "Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees," in *AIAA Guidance, Navigation and Control Conference*, 2004.

[24] A. Alonso-Ayuso, L. Escudero, and F. Martin-Campo, "Collision avoidance in air traffic management: a mixed-integer linear optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 47 –57, 2011. DOI: 10.1109/TITS.2010.2061971.

[25] W. Shi, J. Sun, and X. Song, "An improved model for air traffic flight conflict resolution," in *IEEE International Conference on Computer Science and Automation Engineering*, 2012.

[26] M. Oishi, C. Tomlin, V. Gopal, and D. Godbole, "Addressing multiobjective control: safety and performance through constrained optimization," in *Hybrid Systems: Computation and Control*, M. Benedetto and A. Sangiovanni-Vincentelli, Eds., vol. 2034, Springer Berlin Heidelberg, 2001.

[27] M. Christodoulou and C. Kontogeorgou, "A novel algorithm for collision avoidance in commercial aircraft using neural networks and non-linear programming," in *Mediterranean Conference on Control and Automation*, 2008.

[28] F. Borrelli, D. Subramanian, A. Raghunathan, and L. Biegler, "MILP and NLP techniques for centralized trajectory planning of multiple unmanned air vehicles," in *American Control Conference*, 2006.

[29] Y. Kuwata, T. Schouwenaars, A. Richards, and J. How, "Robust constrained receding horizon control for trajectory planning," in *AIAA Guidance, Navigation and Control Conference*, 2005.

[30] B. D. Luders, "Robust trajectory planning for unmanned aerial vehicles in uncertain environments," Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, 2008.

[31] Y. Kuwata, A. Richards, T. Schouwenaars, and J. How, "Distributed robust receding horizon control for multivehicle guidance," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 4, pp. 627 –641, 2007. DOI: 10.1109/TCST.2007.899152.

[32] J. P. Chryssanthacopoulos and M. J. Kochenderfer, "Analysis of open-loop and closed-loop planning for aircraft collision avoidance," in *IEEE International Conference on Intelligent Transportation Systems*, 2011.

[33] L. F. Winder, "Hazard avoidance alerting with Markov decision processes," PhD thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2004.

[34] L. Kaelbling and T. Lozano-Pérez, "Finding aircraft collision-avoidance strategies using policy search methods," MIT Computer Science and Artificial Intelligence Laboratory, Tech. Rep. 2009-043, 2009.

[35] T. B. Wolf and M. J. Kochenderfer, "Aircraft collision avoidance using monte carlo real-time belief space search," *Journal of Intelligent and Robotic Systems*, vol. 64, no. 2, pp. 277–298, 2011. DOI: 10.1007/s10846-010-9532-6.

[36] S. Temizer, M. J. Kochenderfer, L. P. Kaelbling, T. Lozano-Pérez, and J. K. Kuchar, "Collision avoidance for unmanned aircraft using Markov decision processes," in *AIAA Guidance, Navigation, and Control Conference*, 2010.

[37] H. Kurniawati, D. Hsu, and W. Lee, "SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Robotics: Science and Systems*, 2008.

[38] M. J. Kochenderfer, J. P. Chryssanthacopoulos, L. P. Kaelbling, and T. Lozano-Perez, "Model-based optimization of airborne collision avoidance logic," Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-360, 2010.

[39] M. J. Kochenderfer, J. P. Chryssanthacopoulos, and P. Radecki, "Robustness of optimized collision avoidance logic to modeling errors," in *IEEE/AIAA Digital Avionics Systems Conference*, 2010.

[40] 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*, 2010.

[41] ——, "Partially-controlled Markov decision processes for collision avoidance systems," in *International Conference on Agents and Artificial Intelligence*, 2011.

[42] ——, "Robust airborne collision avoidance through dynamic programming," Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-371, 2011.

[43] 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.

[44] M. Gariel, F. Kunzi, and R. Hansman, "An algorithm for conflict detection in dense traffic using ADS-B," in *IEEE/AIAA Digital Avionics Systems Conference*, 2011.

[45] P. Fiorini and Z. Shillert, "Motion planning in dynamic environments using velocity obstacles," *International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998. DOI: 10.1177/027836499801700706.

[46] Z. Shiller, F. Large, and S. Sekhavat, "Motion planning in dynamic environments: obstacles moving along arbitrary trajectories," in *IEEE International Conference on Robotics and Automation*, 2001.

[47] B. Kluge and E. Prassler, "Reflective navigation: individual behaviors and group behaviors," in *IEEE International Conference on Robotics and Automation*, 2004.

[48] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *IEEE International Conference on Robotics and Automation*, 2008.

[49] H. von Viebahn and J. Schiefele, "A method for detecting and avoiding flight hazards," in *SPIE Meeting on Enhanced and Synthetic Vision*, 1997.

[50] M. Prandini, J. Lygeros, A. Nilim, and S. Sastry, "A probabilistic framework for aircraft conflict detection," in *AIAA Guidance, Navigation, and Control Conference*, 1999.

[51] N. Kantas, A. Lecchini-Visintini, and J. Maciejowski, "Simulation-based Bayesian optimal design of aircraft trajectories for air traffic management," *International Journal of Adaptive Control and Signal Processing*, vol. 24, no. 10, pp. 882–899, 2010. DOI: 10.1002/acs.1204.

[52] G. Hoffmann and C. Tomlin, "Decentralized cooperative collision avoidance for acceleration constrained vehicles," in *IEEE Conference on Decision and Control*, 2008.

[53] D. Šišlák, J. Samek, and M. Pěchouček, "Decentralized algorithms for collision avoidance in airspace," in *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2008.

[54] D. Šišlák, P. Volf, and M. Pěchouček, "Agent-based cooperative decentralized airplane-collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 36–46, 2011. DOI: 10.1109/TITS.2010.2057246.

[55] B. Goode and M. Roan, "A differential game theoretic approach for two-agent collision avoidance with travel limitations," *Journal of Intelligent & Robotic Systems*, vol. 67, no. 3-4, pp. 201–218, 2012. DOI: 10.1007/s10846-012-9657-x.

[56] J. Archibald, J. Hill, N. Jepsen, W. Stirling, and R. Frost, "A satisficing approach to aircraft conflict resolution," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, no. 4, pp. 510–521, 2008. DOI: 10.1109/TSMCC.2008.919162.

[57] O. Sigaud and O. Buffet, Eds., *Markov Decision Processes in Artificial Intelligence*. Wiley, 2010.

[58] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[59] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1–2, pp. 99–134, 1998. DOI: 10.1016/S0004-3702(98)00023-X.

[60] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *International Conference on Machine Learning*, 1995.

[61] M. Hauskrecht, "Value-function approximations for partially observable Markov decision processes," *Journal of Artificial Intelligence Research*, vol. 13, pp. 33–94, 2000. DOI: 10.1613/jair.678.

[62] J. Pineau, G. Gordon, and S. Thrun, "Anytime point-based approximations for large POMDPs," *Journal of Artificial Intelligence Research*, vol. 27, no. 1, pp. 335–380, 2006. DOI: 10.1613/jair.2078.

[63] T. Smith and R. Simmons, "Heuristic search value iteration for POMDPs," in *Conference on Uncertainty in Artificial Intelligence*, 2004.

[64] M. T. Spaan and N. Vlassis, "Perseus: randomized point-based value iteration for POMDPs," *Journal of Artificial Intelligence Research*, vol. 24, no. 1, pp. 195–220, 2005. DOI: 10.1613/jair.1659.

[65] D. Silver and J. Veness, "Monte-carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems*, 2010.

[66] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, "Online planning algorithms for POMDPs," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008. DOI: 10.1613/jair.2567.

[67] D. S. Bernstein, S. Zilberstein, and N. Immerman, "The complexity of decentralized control of Markov decision processes," in *Conference on Uncertainty in Artificial Intelligence*, 2000.

[68] Z. Rabinovich, C. V. Goldman, and J. S. Rosenschein, "The complexity of multiagent systems: the price of silence," in *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003.

[69] F. A. Oliehoek, "Value-based planning for teams of agents in stochastic partially observable environments:" PhD thesis, University of Amsterdam, 2010.

[70] D. M. Asmar, M. J. Kochenderfer, and J. P. Chryssanthacopoulos, "Vertical state estimation for aircraft collision avoidance with quantized measurements," *AIAA Journal of Guidance, Control, and Dynamics*, in press 2013.

[71] M. J. Kochenderfer, *Algorithm description document for the airborne collision avoidance system X threat resolution module*, Federal Aviation Administration, TCAS Program Office, 2012.

[72] J. E. Holland, M. J. Kochenderfer, and W. A. Olson, "Optimizing the next generation collision avoidance system for safe, suitable, and acceptable operational performance," in *AIAA Guidance, Navigation, and Control Conference*, 2013.

[73] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella, "Taming decentralized POMDPs: towards efficient policy computation for multiagent settings," in *International Joint Conference on Artificial Intelligence*, 2003.

[74] D. O. Stahl and P. W. Wilson, "Experimental evidence on players' models of other players," *Journal of Economic Behavior & Organization*, vol. 25, no. 3, pp. 309–327, 1994. DOI: 10.1016/0167-2681(94)90103-1.

[75] J. P. Chryssanthacopoulos and M. J. Kochenderfer, "Decomposition methods for optimized collision avoidance with multiple threats," in *IEEE/AIAA Digital Avionics Systems Conference*, 2011.

[76] J. K. Rosenblatt, "Optimal selection of uncertain actions by maximizing expected utility," *Autonomous Robots*, vol. 9, no. 1, pp. 17–25, 2000. DOI: 10.1023/A:1008916000526.

[77] S. J. Russell and A. Zimdars, "Q-decomposition for reinforcement learning agents," in *International Conference on Machine Learning*, 2003.