Safety Validation of Sense and Avoid Algorithms using Simulation and Evolutionary Search

Xueyi Zou, Rob Alexander, John McDermid

Department of Computer Science, University of York, UK
{xz972,rob.alexander,john.mcdermid}@york.ac.uk

Abstract. We present a safety validation approach for Sense and Avoid (SAA) algorithms aboard Unmanned Aerial Vehicles (UAVs). We build multi-agent simulations to provide a test arena for UAVs with various SAA algorithms, in order to explore potential conflict situations. The simulation is configured by a series of parameters, which define a huge input space. Evolutionary search is used to explore the input space and to guide the simulation towards challenging situations, thus accelerating the process of finding dangerous faults of SAA algorithms and supporting the safety validation process. We applied our approach to the recently published Selective Velocity Obstacles (SVO) algorithm. In our first experiment, we used both random and evolutionary search to find mid-air collisions where UAVs have perfect sensing ability. We found evolutionary search can find some faults (here, interesting problems with SVO) that random search takes a long time to find. Our second experiment added sensor noise to the model. Random search found similar problems as it did in experiment one, but the evolutionary search found some interesting new problems. The two experiments show that the proposed approach has potential for safety validation of SAA algorithms.

Keywords: Sense and Avoid, Unmanned Aerial Vehicles, safety validation, multi-agent simulation, evolutionary search, Genetic Algorithm

1 Introduction

Amazon, the world's largest online retailer, announced its “Prime Air” plan in 2013, where Unmanned Aerial Vehicles (UAVs) will be used to deliver goods to customers. However, this is only a fiction now: UAVs are not currently permitted to access civilian airspace in most countries due to safety considerations. One of the safety concerns is the UAV's inability to avoid mid-air collision with other aircraft. To alleviate this concern, UAVs must provide what is referred to as a Sense and Avoid (SAA) capability. In [1], SAA is defined as “*the capability of a UAV to remain well clear from and avoid collisions with other airborne traffic. Sense and Avoid provides the functions of self-separation and collision avoidance to establish an analogous to “see and avoid” required by manned aircraft*”.

As for collision avoidance, a wide variety of approaches have been proposed in the general field of robotics [2-6] that have the potential to be adapted for UAVs. The safety of these approaches, however, is by no means well understood. Considering the strict safety requirements in the aviation sector, a collision avoidance algorithm cannot be accepted and deployed without rigorous safety validation.

Validation is the process of determining whether a product (e.g. a piece of implemented software or a system) has the desired properties; desired, that is, in hindsight, rather than with reference to a pre-defined specification. Validation is different from verification, which is often conducted at the end of each development stage to determine whether a product of that stage (e.g. specification, computer model, design, and implementation etc.) is consistent with an explicit specification (or reference model). It is entirely possible that a product passes verification but fails validation, for example when the specification has not captured what the user actually wants or needs.

By *safety* validation, we mean the process of determining whether a product will behave safely during operation in terms of protecting itself, the environment it inhabits, and humans. Safety validation is difficult, firstly because users (regulators, operators, bystanders) often cannot provide precise safety requirements for complex, novel systems. Secondly, the operational environment of the product may be too complex to predict the range of possible operational scenarios in advance, which makes it hard to fill in the missing requirements.

For SAA algorithms, the conventional approaches for safety validation are simulation and flight test. Due to the high cost of flight test, it can only be conducted for a very limited time and thus gives limited assurance, although it does have the great advantage of testing real aircraft behaviours. Simulation is more cost-effective and thus can cover a far larger part of the possible operational situations, albeit subject to limitations in the fidelity of the simulation.

In this paper, we present a safety validation approach for SAA algorithms based on multi-agent simulation and evolutionary search. Arnold and Alexander previously presented an approach to testing autonomous robot control software [7], which they claimed to also have potential for validation. They randomly created a diverse range of situations and executed them in the Player/Stage robot simulator to test whether the robot behaved safely. The work presented in this paper is an advancement of [7], specifically in that we use evolutionary search to guide the simulation towards challenging situations. We believe that this has greater potential to reveal safety issues than randomized simulations. Moreover, we test our approach using a promising new collision avoidance algorithm (Selective Velocity Obstacles), rather than the quite simple algorithm (Smoothed Nearness Diagrams) tested in [7].

The paper is organized as follows: in section 2 we identify the challenges for such a safety validation approach, in section 3 we explain our proposed approach, and in section 4 we describe experimental use of our approach to validate the safety of the Selective Velocity Obstacles approach. Section 5 summaries the paper and outlines our future plans.

2 Challenges for an Automated Safety Validation Approach

Ideally, a safety validation approach would reveal all safety issues (dangerous faults) of the validated system (if there are any). It would do so efficiently, and would give confidence regarding the extent to which all the credible faults have been revealed. In this paper we attack only a small piece of the puzzle – given a space of situations and a system under validation, how we can efficiently home in on hazardous situations that we haven't seen before?

As stated in [7], to reveal as many faults as possible, a wide range of diverse test situations should be generated. It is important to favour situations that have a high likelihood of causing dangerous behaviours of the validated system; otherwise, it would be easy for an approach to spend most of its time in generating safe situations, which is computationally inefficient. In this paper, we use evolutionary search to generate test situations with a high collision risk.

A second challenge is "situation coverage" – testing the maximum proportion of potentially dangerous situations that the system could ever encounter [7]. Here, we partially address this using an encounter model of possible situation types to generate a broad distribution of specific situations.

A third challenge is simulation fidelity, in particular whether there are faults in the system that the simulation cannot reveal because they depend on details that are not modelled. In this paper, we explore this issue in a simple way – we run simulations with both infallible sensors and sensors subject to random noise, and look at how the results of the latter simulations are richer (in terms of the range of hazardous situations found).

In section 5, we discuss briefly how we will further address these issues in our future work.

3 Proposed Method

The proposed method is the integration of multi-agent simulation and evolutionary search. We build multi-agent simulations to provide a test arena for UAVs with various SAA algorithms, in order to explore potential conflict situations. The simulation is configured by a series of parameters, which define a huge input space. Evolutionary search is used to explore the input space and to guide the simulation towards challenging situations, thus accelerating the process of finding faults and supporting the safety validation process.

Multi-agent Simulation

We use MASON (See *http://cs.gmu.edu/~eclab/projects/mason/*) as our multi-agent simulation framework. In a typical multi-agent simulation, there are three basic elements: agents, environment, and their interactions. The agents in our simulation are UAVs with various kinds of SAA algorithms. They have attributes, such as maximum and minimum speed, maximum turning rate, etc., and they also have behaviours, such as sensing other UAVs and avoiding them. The environment in our simulation is simplified as a 2-D rectangular horizontal flight area with length and width set according to the range of the "Traffic Advisory (TA)" and "Resolution Advisory (RA)" regions of the TCAS (traffic alert and collision avoidance system). Apart from UAVs, some other entities in the environment are waypoints for navigation, and the start point and destination of each UAV. The interactions between the UAVs are only via the sense and avoid algorithms. We have not modelled any explicit communication between UAVs. The interactions between UAVs and the environment include UAVs following waypoints and generating new waypoints for collision avoidance.

To validate the safety of SAA algorithms, the simulation should simulate different encounters for the SAA algorithm to handle. We developed "encounter generators" that can generate three kinds of encounters, each involving two UAVs: (1) head on encounters, (2) crossing encounters, and (3) tail approach encounters. We refer to one of the UAVs as the "subject" UAV and the other as an intruder. Using any one of the encounter generators, the intruder's start point, velocity vector and destination can be decided on the premise that the subject UAV's start point, velocity vector and destination have been fixed. The three encounters are explained as follows:

1. The head on encounter is where the subject UAV and the intruder approach each other in opposite directions, as illustrated in Fig. 1 (a). The intruder can approach the subject UAV from either the left side or the right side with a certain offset.
2. The crossing encounter is where the subject UAV and the intruder approach each other at an encounter angle ranging from 0° (exclusive) to 180^0 (exclusive) from either the left or the right side, as illustrated in Fig. 1 (b). If the encounter angle equals 180^0, it is a head on encounter without offset. If the encounter angle equals 0°, it is a tail approach encounter without offset, which will be discussed next.
3. The tail approach encounter is where the intruder overtakes or is overtaken by the subject UAV flying on parallel tracks, as illustrated in Fig. 1 (c). The intruder can overtake or be overtaken by the subject UAV from the left side or the right side with a certain offset.

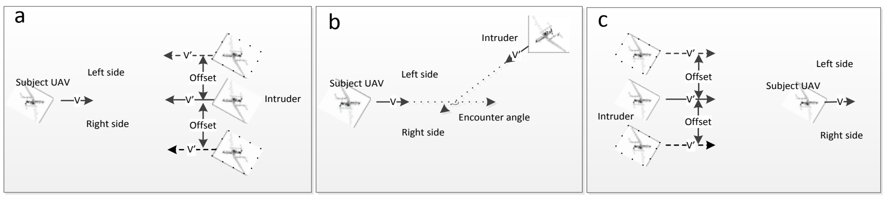

Fig. 1. (a) Head on; (b) Crossing; (c) Tail approach.

Some global agents are utilized to monitor the simulation: a "proximity measurer" measures the nearest distance of each UAV to other UAVs in every simulation step and the most dangerous proximity of each UAV to others in a simulation run; an "accident detector" monitors the simulation and logs accidents and removes UAVs disabled by a collision. These global monitoring agents play an important role in guiding the search (which will be described later) towards challenging situations.

As stated above, the simulation is configured by a series of parameters, which can be divided into three categories:

1. parameters for one or more encounters, e.g. the parameter to decide which encounter should be simulated in a simulation run, and the parameters used to generate that encounter;
2. parameters for the subject UAV, e.g. the subject UAV's destination, its maximum and minimum speed, maximum acceleration and turning rate;
3. parameters for the intruders, e.g. the intruder's (or intruders') maximum and minimum speed, maximum acceleration and turning rate.

Evolutionary Search

The evolutionary search part of the approach is implemented by using ECJ (See *http://cs.gmu.edu/~eclab/projects/ecj/*), which is a Java-based evolutionary computation research system. We have experimented with Genetic Algorithms (GA).

To use GA, first, the initial population is set up with *n* individuals, with each individual's genome representing the settings of the simulation parameters identified above. Then each individual of the population is evaluated by a simulation run and the fitness of that individual can be calculated. According to the fitness, the selection process will (re)sample *n* individuals from the population, and the selected individuals' genome will be "crossed-over" and mutated. After these genetic operations, the individuals will be used to form the next generation of the population, which will replace the old population. This process goes on until it runs out of time or the ideal individual(s) has been found.

The fitness of an individual is calculated by applying a "fitness function" to it. Defining a good fitness function is a crucial task in GA work, as it will ultimately determine where the search moves towards. In our case, a good fitness function should favour those individuals that embody hazardous situations, while avoiding premature convergence (i.e. avoiding the population becoming very homogenous). Since the main concern of SAA is mid-air collision, we define a fitness function based on the nearest distance between pair of UAVs during each simulation run observed by our "proximity measurer".

Similarities to Existing Approaches

It is noted that our approach shares commonalities with search-based software testing where meta-heuristic search techniques are employed for automatic generation of test data [8]. Whilst none of the results from that work are directly comparable to what we have presented here, we have adapted some of the ideas for our work. In search-based software testing, test data are generated as the input of a piece of code (or software) while in this paper, we use evolutionary search to generate input data (e.g. configuration parameters) for multi-agent simulations. This is because, for SAA algorithms, safety cannot be analysed without consideration of other UAVs and the environment.

Similar approaches have also been used in the ASHiCS (Automating the Search for Hazards in Complex Systems) project [9] and by Alam et al. in [10, 11] to conduct safety analysis of ATM (Air Traffic Management) systems. However, their main concern is to identify the combination of airspace configurations and Air Traffic Controller's actions that can result in a high collision risk. This is different from our work that is to identify safety issues of SAA algorithms.

4 Experiments and Findings

4.1 Case Study Introduction

We confine the experiments to two UAV encounters, where the two UAVs run the same SAA algorithm, though it is possible for our approach to handle multiple UAVs with heterogeneous SAA algorithms. We have tested our proposed approach on the recently published Selective Velocity Obstacles (SVO) [4] algorithm for collision avoidance. We selected SVO because it improves the widely studied Velocity Obstacles [3] approach to accommodate the common right-of-way rules of the air while providing collision avoidance capability. Full details of SVO are provided in [4]; below, we provide a brief summary.

A velocity obstacle is the set of all velocity vectors of an agent (UAV) that will result in a collision with other agents (or obstacles) at some moment in time, assuming that the other agents maintain their current velocity vectors [3]. It follows that if the agent chooses a velocity vector outside its velocity obstacle, then a collision will not occur in a certain time horizon.

SVO is designed for cooperative collision avoidance, where each UAV in an encounter cooperatively avoids each other while obeying the right-of-way rules. The rules are as follows [4, 12]:

1. On a converging encounter, the one on the right has the right-of-way;
2. On a head-on encounter, both aircraft should move to the right side;
3. The one that is about to be overtaken has the right-of-way;
4. Avoidance manoeuvres should not go over, under, or in front of other aircraft that have the right-of-way, except when it is clear.

Here three types of encounters are defined: Converging, Head-on, and Overtaking as illustrated in Fig. 2. Note they are different from the encounters defined by our simulation as illustrated in Fig. 1, which we think will help in revealing faults. SVO defines a way to selectively avoid the other UAV(s) by defining three manoeuvre modes [4], which are

1. **Avoid**, where the host UAV takes an manoeuvre to avoid collision with others;
2. **Maintain**, where the host UAV keeps its current velocity vector;
3. **Restore**, where the collision avoidance system gives back the control to the original controller/pilot.

It is noted that, for a UAV to use the SVO approach, the only information it needs about the others is their current positions, velocity vectors and shapes. It is assumed that each UAV has perfect sensing ability when fitted with ADS-B[1] to enable the cooperative collision avoidance (comments on the capabilities of ADS-B are outside the scope of this paper). In the experiments, we added some dynamic constraints on the UAVs, which were converted from the performance data of Global Hawk given in [6], as shown in Table 1. For SVO, when in the "Avoid" mode, it is desirable for the host UAV to select a new velocity vector outside its velocity obstacle induced by others but still obey the right-of-way rules. However, considering the dynamic constraints, we assume that each UAV can only avoid others by turning right 2.5deg/s. This means that during a simulation run, the magnitude of each UAV's velocity vector keeps constant and only the direction of the velocity vector will change. Another consideration for this is that we follow the policy given in [4] and set the other manoeuvres, such as the "climb and descend" for non-cooperative situations and speed and direction change for conflict resolution[2]. Note that, between simulation runs, the velocity magnitude also varies.

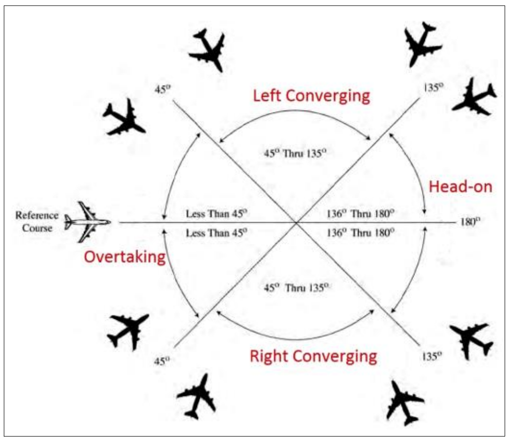

Fig. 2. UAV encounter types, adapted from manned air traffic [13]

Max speed	92.6m/s	Min speed	51.4m/s
Normal speed	77.2m/s	Max turning rate	2.5deg/s

Table 1. UAV performance limits

Collision avoidance manoeuvres in some typical encounters are shown in Fig. 3. In the figures in this paper, the subject UAV always starts from the middle of the left side. The points in the diagram were generated by the SVO algorithm to denote the waypoints the host UAV should navigate by – the bigger red points generated from

[1] Abbreviation for Automatic Dependent Surveillance-Broadcast, a cooperative surveillance technology with which a UAV will send its real time information, such as position and velocity, to its peers via a radio frequency.

[2] Conflict resolution resolves situations where the distance between two UAVs becomes or is forecasted to become less than the minimum desired separation distance. It happens before collision avoidance.

Avoid” modes, the smaller orange points from “Maintain” modes, and the black hollow points from “Restore” modes.

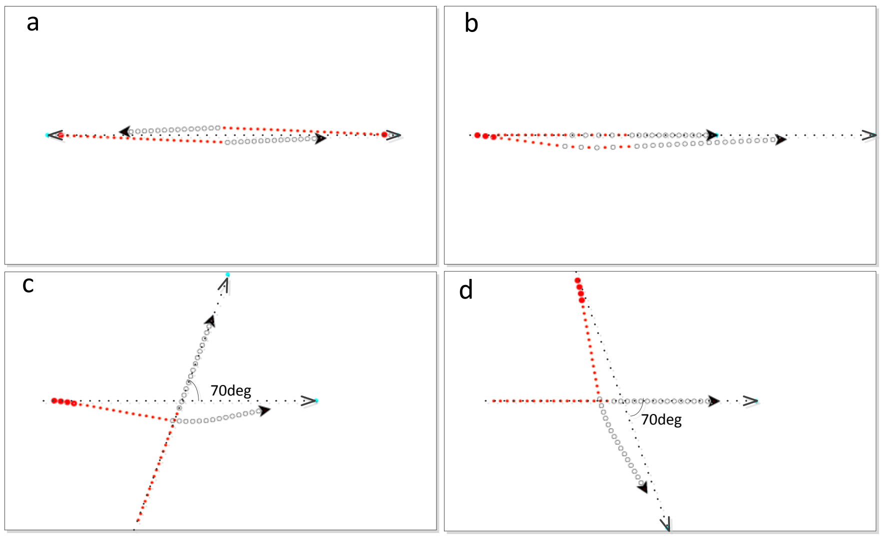

Fig. 3. (a) In a Head-on encounter, each UAV avoids the other; (b) In an Overtaking encounter, the front one has the right-of-way; (c) In a right Converging encounter, the intruder has the right-of-way; (d) In a left Converging encounter, the subject UAV has the right-of-way

4.2 Experiment 1: perfect sensing ability

Experiment 1 was conducted under the assumption that each UAV has perfect sensing ability – they know both their own and the other UAV’s real time position and velocity vector.

Experiment 1.1

We first used random search as pre-treatment to find some “obvious” mid-air collisions. We conducted random search 3 times, with 250,000 uniformly distributed sample points (simulation runs) each time. Overall there were 9 mid-air collisions, all of which happened in crossing encounters. Examples and their parameter settings are shown in Table 2.

From Table 2 one pattern can be found – the encounters are all left side crossing (according to Fig. 1) with encounter angles around 46^o; and the subject UAV’s speed is very high (92.6 is the maximum speed for this kind of UAV) while the intruder’s speed is very low (51.4 is the minimum speed for this kind of UAV).

When we scrutinize all these encounters, a typical situation is shown in Fig. 4. The situation is “Left Converging” according to Fig. 2, where the subject UAV has the right-of-way. The intruder made a right turn manoeuvre. But since the subject UAV was at high speed and the intruder was at low speed, the manoeuvre was not enough to avoid a collision.

	Subject UAV speed	*Is right side*	*Encounter angle*	*Intruder speed*
Trial 1	92.00	NO	46.15	54.34
Trial 2	90.70	NO	45.18	54.30
	....	....	....	....
Trial 3	89.86	NO	45.27	52.70
	....	....	....	....
	92.60	NO	46.75	55.50
Average	90.98		46.01	54.33

Table 2. Mid-air collisions and their parameter settings revealed in experiment 1.1

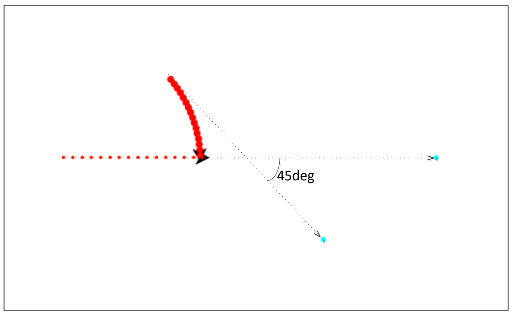

Fig. 4. A typical encounter found in experiment 1.1

It is noted that of all the 3*250,000=750,000 random searched points, random search found only 9 "obvious" mid-air collisions. Either there are few obvious collision situations, or random search has difficulty finding more challenging situations.

It was not clear whether or not the "obvious" situations found so far constitute all the possible situations that will result in a mid-air collision for the SVO algorithm. We explored this in experiments 1.2 and 1.3.

Experiment 1.2

Experiment 1.2 was intended to find new, subtler, situations that will result in mid-air collisions other than those found in experiment 1.1 using random search. To this end, if we sampled a point that corresponded to the class of collision situations found in 1.1, it was discarded without ever being simulated. These points were identified based on them satisfying all of the following conditions:

1. It is a left side crossing encounter;
2. The subject UAV's speed minus the intruder's speed is more than 18m/s;
3. The encounter angle is greater than 45^o, but it is less than 51.2^o.

The numbers above were estimated from the numbers in the "Average" row of Table 2 with some extra margin. Thus we excluded the "obvious" dangerous encounters already identified, ensuring that the search were only looking for "new" problems.

We conducted random search 3 times, with 250,000 sample points each time. Of all the sampled points, we found *no* mid-air collision and thus nothing interesting.

Experiment 1.3

The purpose of experiment 1.3 was the same as experiment 1.2, but evolutionary search was used instead. The point discarding conditions were the same as those in experiment 1.2. Whenever a new individual was created that matched all of the conditions, it was immediately awarded the worst possible fitness value (i.e. 0) without ever being simulated.

In the experiment, since we only considered two UAV encounters, the objective was thus to minimize the average of the minimum distances[3] of each UAV to the other (the minimum distances were actually equal). Formally, it was defined as:

$$minimize Z = \frac{min_{i\in[0,s]}\{X_{jk_i}\}+min_{i\in[0,s]}\{X_{kj_i}\}}{2}$$

where s is the number of simulation steps; X_{jk_i} is the distance to collision between the subject UAV and the intruder in the i^{th} simulation step; and X_{kj_i} is the reverse.

In this experiment, the fitness function was defined as:

$$f(ind) = \frac{1}{1+Z}$$

where the value of $f(ind)$ is the fitness of each individual, and when Z equals 0 this fitness function reaches its maximum, 1, meaning there is a mid-air collision.

In this experiment, we set the number of generations to be 500, each generation with 500 individuals. So the number of total sample points is the same as before. We made 3 trials. Each trial took less than 3 minutes to compute using an ordinary desktop, slightly longer than the previous random searches, which took about 2.5 minutes.

From the log we can see a fast increase in average fitness in the beginning generations as illustrated by the blue curves in Fig. 5. It means over these generations, the average minimum distance between the two UAVs decreased quickly and the evolutionary search was guiding the simulation towards more challenging situations. The figure also shows that the average fitness curves all reached a near-plateau before 100 generations, but their values varied. The third trial got the greatest value.

As shown by the orange curves in Fig. 5. (a) and (b), the first two trials did not find any mid-air collision, but the third trial found many (Fig. 5. (c)). Note that in Fig. 5. (c), the average fitness near-plateaus before the average number of accidents. This is because one accident only happens and is counted when the fitness of that individual is exactly 1 (i.e. the distance between the two UAVs is exactly 0).

When we checked these mid-air collisions found in trial 3, we found that the genomes (parameter settings) were almost the same – the genomes that code for accident scenarios were almost clones of each other. This is because when the GA finds a good individual, it will have a high probability to focus on the individual and make some minute modifications to it. So GA has a strong tendency to converge. But if the initial genomes are not very good, the minute modifications are not enough to find some better individuals and an evolutionary search may fail to find the best individuals in a finite number of generations ("premature convergence" in GA terms), which

[3] Here, the distance was scaled for simulation visualization purpose. Whenever a collision happened the distance was set to 0.

was what happened in the first two trials. We tried to overcome this by using a much bigger initial population in experiment 2.3, see below.

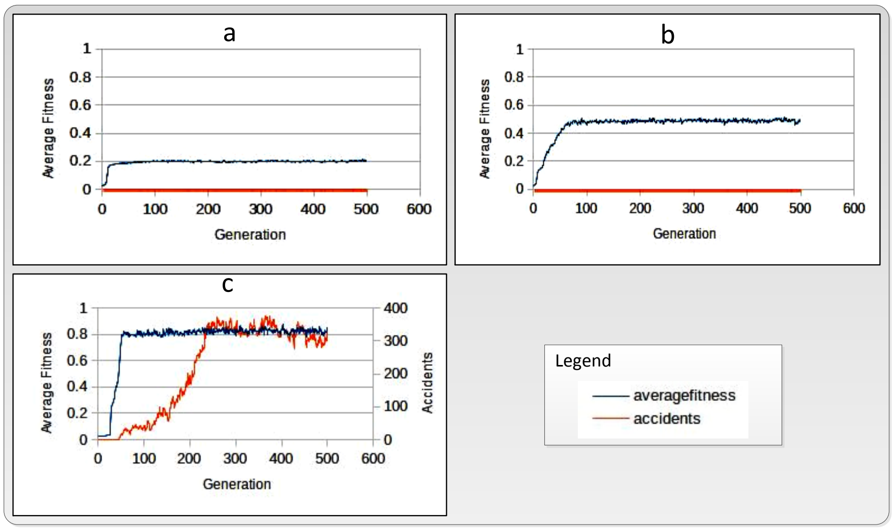

Fig. 5. Average fitness and accidents of each generation in experiment 1.3

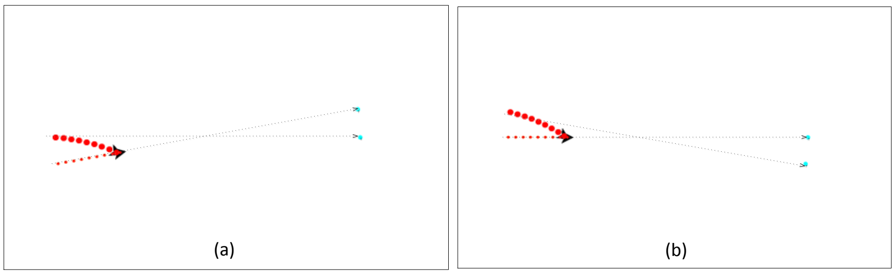

Fig. 6. Typical encounters found in experiment 1.3

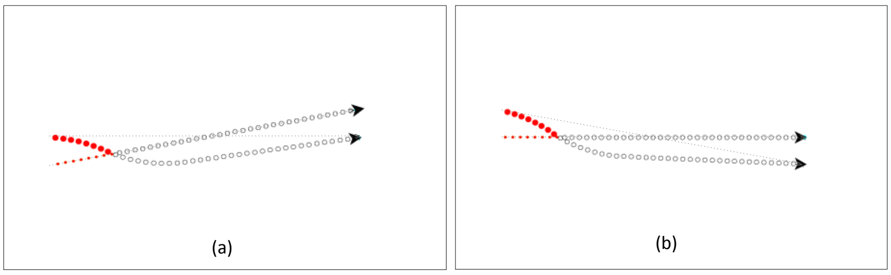

Fig. 7. Collisions shown in Fig. 6 can be avoided with a slightly larger turning rate

Two typical encounters that resulted in mid-air collisions are shown in Fig. 6. These encounters are not so interesting, as the initial positions of the two UAVs are too close. But even in such close initial positions conditions, if the UAV's maximum turning rate is a bit greater than 2.5deg/s, say 3deg/s, all the collisions can be avoided, as shown in Fig. 7.

Fig. 8 shows the average fitness of each generation using GA (data from trial 1) and random search (data from trial 1, 2, 3 of experiment 1.2). Since the fitness represents the nearest proximity between the two UAVs during a simulation run, we can conclude that the evolutionary search is more efficient in finding subtler challenging situations than random search.

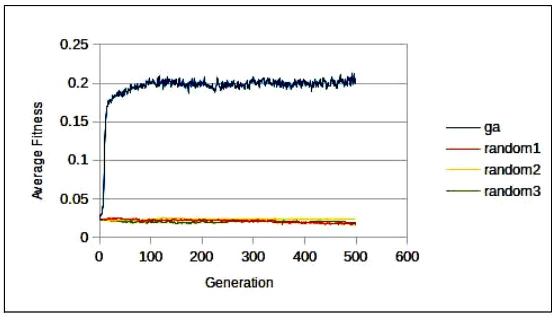

Fig. 8. Average fitness of GA and random searches

So far, two rough patterns of encounters have been revealed that are likely to result in mid-air collisions. The two patterns are summarized as follows:

1. Pattern 1 is left 45^o crossing encounters, where the intersection of the following conditions is true:
   - It is a left side crossing encounter;
   - The subject UAV's speed minus the intruder's speed is more than 18m/s;
   - The encounter angle is greater than 45^o, but it is less than 51.2^o.

2. Pattern 2 is close initial positions encounters, where the intersection of the following conditions is true:
   - The encounter angle is less than 20^o;
   - The subject UAV's speed minus the intruder's speed is less than 5m/s.

4.3 Experiment 2: sensor value uncertainty

Experiment 2 was conducted without making the perfect sensing ability assumption. Here we simply add Gaussian noise to the sensing result of the other UAV's position and velocity vector. The mean (μ) of the Gaussian noise is 0, and the standard deviation (σ) is 0.05*{*real value*}. The sensing rate is as TCAS, which is 1Hz.

Experiment 2.1

Again, we first used random search to find "obvious" mid-air collisions. We conducted random search 5 times, with 250,000 sample points each time.

In the first 4 trials, all the collision situations found can either be categorized as pattern 1 or pattern 2, except one. No collision was found in trial 5. The one exception is a left side crossing according to Fig. 1, where even though the subject UAV and the intruder's speeds are very close (i.e. 85.84m/s and 83.04m/s), their encounter angle is larger (28.56^o) than that in pattern 2. This exceptional encounter recurred as shown in Fig. 9 (a).

According to SVO, this is an Overtaking encounter, where the speeds of the UAVs are very close. Due to the sensor noise, the intruder sometimes decided its speed was greater than the subject UAV's and took avoidance manoeuvres while in fact it shouldn't have. The result is that the intruder's right turn avoidance manoeuvres cancelled out some of the effect of the subject UAV's and they collide sometime in the future. But if there were no sensor noise, the collision would not have happened as shown in Fig. 9 (b).

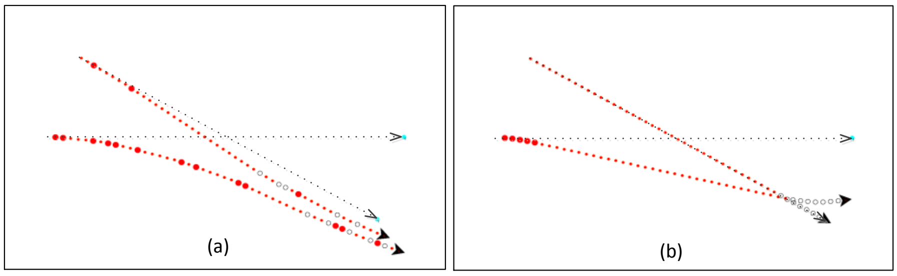

Fig. 9. (a) Trajectory with sensor noise; (b) Trajectory without sensor noise

Again we need to ask whether or not the situations found so far constitute all the possible situations that will result in a mid-air collision under sensor noise. We explored this in experiments 2.2 and 2.3.

Experiment 2.2

Experiment 2.2 tried to find subtler situations that will result in mid-air collisions other than those found in experiment 2.1 using random search. We conducted random search 5 times, with 250,000 sample points each time. Of all the sampled points, we found *no* mid-air collision.

We then checked some of the nearest mid-air approaches and found another situation that may lead to mid-air collision – the intruder approaches the subject UAV from the *right* side with an encounter angle a little greater than 45^o; and the intruder has a high speed while the subject UAV has a low speed. This is actually the same as those identified in pattern 1 except the intruder approaches from the right side. It follows that the random search should have found some collisions in this situation as it did in experiment 1.1 considering that we have searched such a huge number of sample points. One explanation for this could be that with the Gaussian noise added, more uncertainty was added and the set of possible paths through the simulation became far larger than before.

Experiment 2.3

Experiment 2.3 tried to find even subtler situations that will result in mid-air collisions other than those found in experiment 2.1 and 2.2 using evolutionary search. To achieve this, we noted that GA has a strong tendency to converge and the existence of some good initial genomes determines whether it can find the "best" individuals in a finite number of generations. (See our earlier comment on this in experiment 1.3). We set the search to run for 50 generations (ten times fewer than before), each generation with 5000 individuals (ten times more than before). The number of total sample points is also the same as experiment 2.1 and 2.2. The fitness function is the same as that of experiment 1.3.

We made 5 trials, of which all but the third trial found mid-air collisions. A typical collision is shown in Fig. 10 (a). This is a little like those identified in pattern 1, except that the encounter angle is a little greater (51.7^o for this typical encounter). Due to sensor noise, sometimes the intruder decided to "Maintain" its velocity while in fact it should have made an "Avoid" manoeuvre.

When we observed this encounter without sensor noise, we found the trajectory as shown in Fig. 10 (b). The intruder did avoid the subject UAV, but it could not get to its target due to the maximum turning rate constraint. So it kept circling around the target. This is undesirable and also forms a hazard, because it may cause the UAV to run out of fuel and finally crash. As can be seen from the figure, this happened in the "Restore" stage and it is actually not the responsibility of the collision avoidance system but the autopilot (or other controllers). This problem can be solved by letting the UAV take a Dubins Curve [14] to its target.

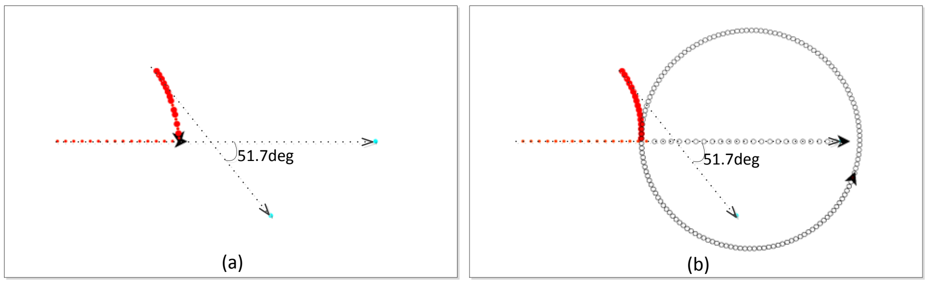

Fig. 10. A typical encounter in experiment 2.3, (a) with sensor noise; (b) without sensor noise

4.4 Findings

Through the experiments, we found the following:

1. Whether with random search or evolutionary search, our multi-agent simulations have the ability to reveal safety issues of a SAA algorithm (SVO). Using the encounters generated by our "encounter generators", SAA algorithms can be tested in different situations;
2. Even though random search can reveal some relatively obvious safety issues, evolutionary search has the ability to guide the simulations towards much subtler challenging situations for SVO to handle. With the combination of the two, the safety validation process has the potential to be accelerated;
3. Some plausible safety issues of SVO have been revealed by our approach – it is dangerous to let low speed UAV avoid high speed UAV in some situations; the 45^o encounter angle for crossing is a dangerous boundary value for SVO; the SVO algorithm is sensitive to sensor noise on velocity.

5 Conclusions and Future Work

We have described a safety validation approach for SAA algorithms using multi-agent simulation and evolutionary search. Through experiments we have shown that our approach can reveal faults that random simulation takes a long time to find, and thus that our approach may accelerate the safety validation process. In the process, we found some safety issues with the SVO algorithm.

When building simulations, we treat SAA algorithms as black boxes. The information on positions, velocities and shapes of UAVs is provided as input to the algorithm and the next waypoint the host UAV should navigate to is returned as output. Therefore, this approach can be easily used to assess a variety of SAA algorithms as long as they follow that input and output protocol (or can be adapted to do so).

The collision avoidance algorithm analysed in this paper is relatively simple, and thus the fitness function used in this paper is straightforward – only the nearest proximity to the other UAV is considered. In the future, we will study more sophisticated algorithms (e.g. the ACAS X algorithm [5]) and devise risk measurements that accommodate factors beyond simple proximity. We will then base our fitness function on these risk measurements to lead the simulation towards high risk situations.

In the experiments, the GA was not well-tuned and sometimes it would lead to premature convergence to local maxima (or minima). We will explore ways to overcome this in the future by adaptively controlling the crossover and mutation probabilities (e.g. as discussed in [15]).

According to section 2, this work partially addresses the challenge of efficiency and touches on challenges of fidelity and coverage. Our future work will tackle these further, the latter two in particular by creating more complex encounter generators that produce richer situations (including equipment failure and other degraded modes). Also, we will consider multi-body encounter problems and the use of 3D simulations. In this way we hope to contribute to the development of effective SAA algorithms, and to provide a cost-effective approach for validation of this important class of algorithm.

Acknowledgements The first author would like to thank the China Scholarship Council (CSC) for its partial financial support for his PhD study.

References

1. Federal Aviation Administration, U.S. Department of Transportaion: Integration of Civil Unmanned Aircraft Systems (UAS) in the National Airspace System (NAS) Roadmap, First Edition. (2013)
2. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine, 4, 23-33 (1997)
3. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. The International Journal of Robotics Research 17, 760-772 (1998)
4. Jenie, Y.I., Van Kampen, E.-J., de Visser, C.C., Chu, Q.P.: Selective Velocity Obstacle Method for Cooperative Autonomous Collision Avoidance System

for Unmanned Aerial Vehicles. AIAA Guidance, Navigation, and Control (GNC) Conference. American Institute of Aeronautics and Astronautics (2013)

5. Kochenderfer, M.J., Chryssanthacopoulos, J.: Robust airborne collision avoidance through dynamic programming. Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-371 (2011)

6. Temizer, S., Kochenderfer, M.J., Kaelbling, L.P., Lozano-Pérez, T., Kuchar, J.K.: Collision avoidance for unmanned aircraft using Markov decision processes. AIAA Guidance, Navigation, and Control Conference. American Institute of Aeronautics and Astronautics (2010)

7. Arnold, J., Alexander, R.: Testing Autonomous Robot Control Software Using Procedural Content Generation. Computer Safety, Reliability, and Security, pp. 33-44. Springer (2013)

8. McMinn, P.: Search-based software test data generation: a survey. Software testing, Verification and reliability 14, 105-156 (2004)

9. Clegg, K., Alexander, R.: The discovery and quantification of risk in high dimensional search spaces. Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion, pp. 175-176. ACM (2013)

10. Alam, S., Lokan, C., Abbass, H.: What can make an airspace unsafe? characterizing collision risk using multi-objective optimization. IEEE Congress on Evolutionary Computation (CEC), pp. 1-8 (2012)

11. Alam, S., Lokan, C., Aldis, G., Barry, S., Butcher, R., Abbass, H.: Systemic identification of airspace collision risk tipping points using an evolutionary multi-objective scenario-based methodology. Transportation Research Part C: Emerging Technologies 35, 57-84 (2013)

12. Federal Aviation Administration: Federal Aviation Regulations (FAR) Chapter I, subchapter F Air Traffic and General Operating Rules, 91.113 Right-of-way rules: Except water operations. (1989)

13. Federal Aviation Administration: JO 7110.65U, Air Traffic Control, Chapter 1: General. In: U.S. Department of Transportation (ed.), (2012)

14. Dubins, L.E.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of mathematics, 497-516 (1957)

15. Srinivas, M., Patnaik, L.M.: Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics, 24, 656-667 (1994)