Discovering Novel Fighter Combat Maneuvers in Simulation: Simulating Test Pilot Creativity

R. E. Smith
The Intelligent Computing Systems Centre
The University of The West of England
Bristol, UK
email: robert.smith@uwe.ac.uk

B. A. Dike
The Boeing Company
St. Louis, Missouri
email: bruce.a.dike@boeing.com

B. Ravichandran
A. El-Fallah
R. K. Mehra
Scientific Systems
Woburn, MA
Email: ravi, adel, rkm@ssci.com

Abstract

This paper reports the authors' ongoing experience with a system for discovering novel fighter combat maneuvers, using a genetics-based machine learning process, and combat simulation. In effect, the genetic learning system in this application is taking the place of a test pilot, in discovering complex maneuvers from experience. The goal of this work is distinct from that of many other studies, *in that innovation, and discovery of novelty, is, in itself valuable.* This makes the details of aims and techniques somewhat distinct from other genetics-based machine learning research.

The paper discusses the details of the application, and the motivations and details of the techniques employed. Results are presented for systems where one player adapts to a fixed strategy opponent, and where two players co-adapt. General implications of this work for other adaptive behavior applications are discussed.

1 INTRODUCTION

New technologies for fighter aircraft are being developed continuously. Often, aircraft engineers can know a great deal about the aerodynamic performance of new fighter aircraft that exploit new technologies, even before a physical prototype is constructed or flown. Such aerodynamic knowledge is available from design principles, from computer simulation, and from wind tunnel experiments.

However, knowing the aerodynamic impact of a new technology is distinct from knowing how the plane will perform in combat. The mapping from aerodynamics to effective maneuvers in a new fighter aircraft is exceedingly complex. Discovering this mapping involves learning, adaptation, and the transference of knowledge from the maneuvers of other fighter aircraft. It is a creative task, usually accomplished in the cockpit, through the well-trained mind of an experienced pilot.

Evaluating the impact of new technologies on actual combat can provide vital feedback to designers, to customers, and to future pilots of the aircraft in question. However, due to the complex mapping discussed above, this feedback typically comes at a high price. While designers can use fundamental design principles (i.e., tight turning capacity is good) to shape their designs, often times good maneuvers lie in odd parts of the aircraft performance space, and in the creativity and innovation of the pilot.

Therefore, the typical process would be to develop a new aircraft, construct a one-off prototype, and allow test pilots to experiment with the prototype, developing maneuvers in simulated combat. Clearly, the expense of such a prototype is substantial. Moreover, simulated combat with highly trained test pilots has a substantial price tag. Therefore, it would be desirable to discover the maneuver utility of new technologies, without a physical prototype.

There are several ways to approach this goal. One is to use man-in-the-loop combat simulation, where a real pilot flies a simulation of the new plane. This avoids the need for a physical prototype, but still includes the cost of trained fighter pilots, serving their role in simulated cockpits. Moreover, there is no guarantee that pilots will exhibit their instinctive, adaptive abilities on the ground, even in the most convincing, domed combat simulators. Certainly, the pilots would feel more at home with there senses embedded in the sights, sounds, and feelings of a real flight.

Another approach would be to analytically consider the mapping from aerodynamics to complex maneuvers. One could do this through combinations of differential game theory and optimal control. In general, such approaches involve modeling the combination of combat rules and aerodynamic performance in a mathematically tractable form. Of course, this is very difficult. Moreover, the process of making the combat situation tractable is likely to introduce structure into the problem, which will bias analysis towards rather obvious maneuvers as solutions. If the model is reconstructed to allow for new maneuver possibilities, it is likely that analytical techniques will find good maneuvers for these possibilities. However, the tractable modeling process is always likely to bias towards maneuvers that are actually *implicit* in the model. Such processes are unlikely to yield truly novel combat maneuvers.

A third approach is that pursued in the authors' past work, and in new work presented here. In this approach, an adaptive, machine learning system takes the place of the test pilot in simulated combat. This approach has several advantages.

- As in the analytical approach, this approach requires a model. However, in this case the model need only be accurate for purposes of combat simulation. It need not present a mathematically tractable form.
- Moreover, the approach is similar to that of man-in-the-loop simulation, except in this case the machine learning "pilot" has no bias dictated by past experiences with real aircraft, no prejudices against simulated combat, and no tendency to tire of the simulated combat process after hundreds or thousands of engagements.
- Moreover, one overcomes the constraints of real-time simulation.

This paper considers ongoing work in this area in terms of its unique character as a machine learning and adaptive systems problem. To recognize the difference between this problem and more typical machine learning problems, one must consider its ultimate goal. This work is directed at filling the role of test pilot in the generation of innovative, novel maneuvers. The work is not directed at online control. That is to say, the machine learning system is not intended to generate part of an algorithm for controlling a real fighter aircraft. Like the test pilot in simulated combat, the machine learning system can periodically fail, without worry that the associated combat failure will result in loss of hardware and personnel. In many ways, the machine learning system is even less constrained than the test pilot, in that it can experiment with maneuvers that would be dangerous in simulated fighter combat with real aircraft.

Moreover, the goal of this work is the process of innovation and novelty, rather than discovering optimality. It is difficult to define what is even meant by "the optimal combat maneuver". The system can have performance losses and gains as it learns, and still be useful, if it discovers "interesting" maneuvers along the way. Although performance feedback is necessary to drive the adaptive process in this system, neither optimizing this feedback, nor monotonic increase in its value, are necessarily required for the system to have real-world utility.

The following sections introduce the details of this system, and discuss how its differences in goals affect the techniques employed. Final sections discuss the implications of this work, and suggest how machine innovation could serve as a new, interesting area for further research.

2 FIGHTER AIRCRAFT MANEUVERING

To understand motivations for the work presented here, and the results presented, one must have access to a few concepts of fighter aircraft maneuvering. This section provides an overview. For more comprehensive information, see [2].

One critical tactical measure in air combat is target *aspect angle*, which is the angle between a target aircraft's line-of-sight, and the line-of-sight to the attacking aircraft. If the target aspect angle is 0, then the target aircraft is pointed directly at the attacker. If the target aspect angle is 180, then the attacker is on the target's tail. Tactical advantage in close-in air combat can be measured as the difference between the target aspect angle and ownship aspect angle, since the most desirable condition for the attacker is to point directly at the target while the target is pointed directly away from the attacker.

An important, new, technological aspect of many modern fighter aircraft is the use of *post-stall technology* (PST). PST refers to systems such as thrust vectoring and advanced flight controls, which enable the pilot to fly at extremely high *angles-of-attack*, well beyond the normal stall limits of conventional aircraft. Angle-of-attack refers to the angle between the aircraft's velocity vector and the aircraft's nose, as measured in the geometric plane containing the aircraft's nose and vertical tail. Using PST flight modes, pilots have developed an entirely new class of combat maneuvers.

PST maneuvers include:

- **The Herbst Maneuver,** which is the most well-known of PST maneuvers. In a Herbst-maneuver, the aircraft quickly reverses direction through a combination of high angle-of-attack and rolling. The Herbst maneuver is named after Wolfgang Herbst, one of the original developers of PST.

- **The Helicopter Gun Attack,** where the PST aircraft points its nose at a circling adversary while staying in the center of the adversary's turning circle.

- **The Cobra Maneuver,** which was first demonstrated by the Russian pilot Pougachev. In the Cobra, the aircraft makes a very quick pitch-up from horizontal to 30 degrees past vertical. The airspeed of the aircraft slows dramatically as the plane continues its horizontal travel. The pilot then uses thrust vectoring to help pitch the aircraft's nose down and resume normal flight angles. This allows the aircraft to rapidly strip airspeed, causing a pursuing fighter to overshoot.

- **The PST Hammerhead Turn,** where the aircraft reverses direction by performing a maneuver resembling a backflip.

One typical attribute of PST tactics is *out-of-plane maneuvering*, where the attacking aircraft flies in a different maneuvering plane than the target. The *maneuvering plane* is determined by an aircraft's velocity vector and aerodynamic lift vector. A frequent characteristic of PST maneuvers is a continually changing maneuver plane.

PST maneuvers are primarily useful for air-to-air gun attacks, because the gun is fixed in alignment with the aircraft's nose, requiring the aircraft to be pointed directly at the target. Air-to-air missile attacks do not require a similar degree of nose pointing because missiles turn after launch in the direction of the target. Missiles are the preferred mode of attack for air combat. However, air-to-air gun attacks may be required in tactical situations if the target is inside the minimum missile range, or if the target employs countermeasures which defeat missiles.

Due to their complex dynamics, PST maneuvers are difficult to characterize without the use of 3-D visualization tools. The primary visualization tool used in this study was Agile-Vu, a 3-D graphics animation package developed by the U.S. Navy for analyzing aircraft flight trajectories. Agile-Vu provides the user with the ability to examine maneuvers from a variety of viewpoints and determine the degree of out-of-plane maneuvering.

In the experiments presented here, two aircraft are employed. One is the X-31 experimental fighter plane, which has significant PST characteristics. The other is an F-18, which has more "standard" fighter characteristics.

3 GENETICS-BASED MACHINE LEARNING

Although *genetic algorithms* GAs [2,4] are most often used in optimization, a growing area of GA application is in machine learning. Optimization can be distinguished from learning in several important respects. In optimization, a system like the GA is given a fixed set of parameters to tune, and its only feedback is the utility of various parameter settings. In a machine learning problem, a system must interact with an environment that provides utility-related feedback, and feedback that indicates some time-varying state of the environment. In general, the system must find several high-utility sets of parameters that correspond to various state-feedback signals. One approach to machine learning problems is to convert them to optimization problems by assigning a full set of parameters for every possible environmental state, and simply optimizing. In general this approach is impractical. Instead, one would like for the learning system to generalize with sets of parameters that perform well across a range of environmental states. Because the appropriate degree of generalization is itself an unknown, the effective number of parameters needed is unknown as well. Moreover, it might be necessary to obtain these parameters gradually, in "chunks" that build towards a final, generalized solution. Thus, the problem is difficult to cast as one of typical optimization.

This work considers GAs in a machine learning context. A thorough review of genetics-based machine learning cannot be included here, for the sake of brevity. However, a concise review of *learning classifier systems* (LCSs), the primary genetics-based machine learning technique, is provided in the following section.

3.1 LEARNING CLASSIFIER SYSTEMS

Consider the following method for representing a state/action pair in a reinforcement learning problem [3]: encode an environmental state in binary, and couple it to an action that can be taken in the environment, which is also encoded in binary. In other words, the string

0 1 1 0 / 0 1 0

represents one of 16 states and one of eight actions. This string can also be seen as a rule that says "IF in state 0 1 1 0, THEN take action 0 1 0". In a LCS, such a rule is called a classifier. One can easily associate a cost value, or other performance measures, with any given classifier.

Now consider generalizing over actions by introducing a "don't care" character (#) into the state portion of a classifier. In other words, the string

1 1 # / 0 1 0

is a rule that says “IF in state 0 1 1 0 OR state 0 1 1 1 OR state 1 1 1 0 OR state 1 1 1 1, THEN take action 0 1 0”. The introduction of this generality allows a system to represent clusters of states and associated actions. However, one can keep only a limited list of these "cluster templates" (which are also called "rules", or classifiers). The search problem involved is deciding which rules to keep, and thus design an effective, rule-dictated maneuver strategy. This is a search problem that naturally lends itself to the GA.

The GA is used in a LCS to locate a good set of generalizations (cluster templates, or generalized rules). Note that this problem requires a diverse set of cooperative classifiers to dictate a strategy. Thus, the LCS applied to this problem must *co-evolve* an appropriate set of classifiers [4]. By using the GA to search for such strings, one can search for ways of clustering states together such that they can be assigned joint performance statistics [4][5].

Figure 1: Structure of a Learning Classifier System

Figure 1 shows a diagram of the structure of an LCS. In a stimulus-response LCS (like the system used in this paper), at each time step a message is posted by the system's detectors, representing the state of the environment. Classifiers are matched against this external message. From amongst the matching classifiers, some conflict resolution (CR) method is used to select a single classifier to act. That classifier’s action is submitted to the environment, through the system’s effectors. If reward (performance feedback) is received from the environment, it is distributed by a credit allocation (CA) system. Some LCSs also allow for internal messages that represent communication between classifiers on successive time steps. However, this possibility is not considered here.

The GA is applied periodically after a specified number of operational cycles. Typically, a high-fitness portion of the classifier list serves as the GA population. Newly produced classifiers replace a low-fitness portion of the list.
Given the outline of the basic learning system structure, one must consider why an LCS is appropriate for discovering rules that dictate combat maneuver logic. The major motivation for this approach is the extensive theoretical and empirical evidence that GAs have robust adaptive ability in ill-conditioned search spaces [6][7]. Moreover, such systems allow for discovery of co-adapted rules that dictate complex maneuvers. Finally, all these abilities only depend on direct, performance only feedback, making the system ideal for learning from experience in simulation.

4 THE LCS USED HERE

The authors have extensive, ongoing experience with using LCSs for acquiring novel fighter aircraft maneuvers. This is a successful, ongoing project, satisfying real-world goals for industry, NASA, and The United States Air Force [8,10].

By way of introduction to the system used here, consider the basic problem of one-versus-one fighter aircraft combat. Two aircraft start their engagement at some initial configuration and velocity in space. In our simulations, an engagement lasts for a pre-specified amount of time (typically 30 seconds). An engagement is divided into discrete time instants (in our simulation, 1/10th second instants). At each instant, each "aircraft" must observe the state of the environment, and make a decision as its own action for that instant. A score for a given aircraft can be calculated at the end of an engagement by comparing that aircraft's probability of damaging its opponent to its own probability of being damaged.

Given this basic outline of the problem at hand, we will introduce details of the fighter aircraft LCS.

1.1. The Combat Simulation

The LCS interacts in simulated, 1-versus-1 combat, through AASPEM, the Air-to-Air System Performance Evaluation Model. AASPEM is a U.S. Government computer simulation of air-to-air combat, and is one of the standard models for this topic.

1.2. Detectors and Classifier Conditions

The conditions of the classifier are from the traditional {1, 0, #} alphabet, defined over the encoding of the environmental state shown in Table 1.

Table 1 Encoding of Environmental State in the Fighter Aircraft LCS.

	Bins (represented in binary, plus the # character)							
3 bits:	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Own Aspect Angle (Degrees)	< 45	45 to 90	90 to .35	135 to 180	180 to 215	215 to 270	270 to 315	315 to 360
Opponent Aspect Angle (Degrees)	< 45	45 to 90	90 to 135	135 to 180	180 to 215	215 to 270	270 to 315	315 to 360
2 bits:	**00**	**01**	**10**	**11**				
Range (1000 feet)	< 1	1 to 4.5	4.5 to 7.5	> 7.5				
Speed (100 knots)	< 2	2 to 3.5	3.5 to 4.8	> 4.8				
Delta Speed (100 knots)	< -.5	-.5 to .5	.5 to 1	> 1				
Altitude (1000 feet)	< 10	10 to 20	20 to 30	> 30				
Delta Altitude (1000 feet)	< -2	-2 to 2	2 to 4	> 4				
Climb Angle	< -30	-30 to 30	30 to 60	> 60				
Opponent Climb Angle	< -30	-30 to 30	30 to 60	> 60				
Total: 20 bits								

Note that although this encoding is *crisp*, its coarseness, and the open-ended bins sometimes used in the encoding, have a linguistic character that is similar to that of a *fuzzy* encoding. That is, concepts like "high, low, and medium" are embodied in the encoding.

1.3. Effectors and Classifier Actions

The classifier actions directly fire effectors (there are no internal messages). Actions are from the traditional {1,0,#} syntax, and correspond to the coarse encoding shown in Table 2.

Table 2: **Action encoding in the Fighter aircraft LCS.**

	Action Values (suggested to AASPEM)							
3 bits:	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Relative Bank Angle (Degrees)	0	30	45	90	180	-30	-45	-90
Angle of Attack (Degrees)	0	10	20	30	40	50	60	70
2 bits:	**00**	**01**	**10**	**11**				
Speed (100 Knots)	1	2	3.5	4.8				
Total: 8 bits								

Note that these action values are only *suggested* to the system that simulates the fighter combat. In many situations, it may be impossible for the aircraft to obtain the exact value suggested by the active classifier. Fortunately, the nature of the underlying fighter combat simulation returns an aerodynamically feasible setting of these action parameters, regardless of conditions. This feature of the system is discussed in more detail later.

As an example of the encoding, consider the classifier shown below:

```
01010010110111010010 / 11010010
```

This classifer decodes to:

```
IF
         (90 > OwnAspectAngle > 35) AND
         (215 > OpponentAspectAngle > 180) AND
         (7.5 > Range > 4.5) AND
         (Speed > 4.8) AND
         (0.5 > DeltaSpeed -0.5) AND
         (Altitude > 30) AND
         (2 > DeltaAltitude > -2) AND
         (ClimbAngle < -30) AND
         (60 > OpponentClimbAngle> 30)
THEN ATTEMPT TO OBTAIN
         (RelativeBankAngle = -45) AND
         (AngleOfAttack = 40) AND
         (Speed = 3.5)
```

1.4. Match-and-Act

In our system if no classifiers are matched by the current message, a default action for straight, level flight is used. There is no "cover" operator [13].

1.5. Credit Allocation

- At the end of an engagement, the "measure of effectiveness" score for the complete engagement is calculated (see below).
- This score is *assigned* as the fitness for *every* classifier that acted during the engagement (and to any duplicates of these classifiers).
- Note that this score *replaces* the score given by averaging the parent scores when the GA generated the rule. Thus, rules that do not fire simply "inherit" the averaged fitness of their GA parents [9].

1.6. Measures of Effectiveness

Our efforts have included an evaluation of different measures of effectiveness within the genetics-based machine learning system, to determine the relative sensitivity of the process. Initial candidate measures included exchange ratio, time on advantage, time to first kill, and other relevant variables.

The measure of effectiveness ultimately selected to feedback into the GA fitness function was based on the following steps. The base score was based on a linear function of average angular advantage (opponent target aspect angle minus ownship target aspect angle) over the engagement, as shown in Figure 2.

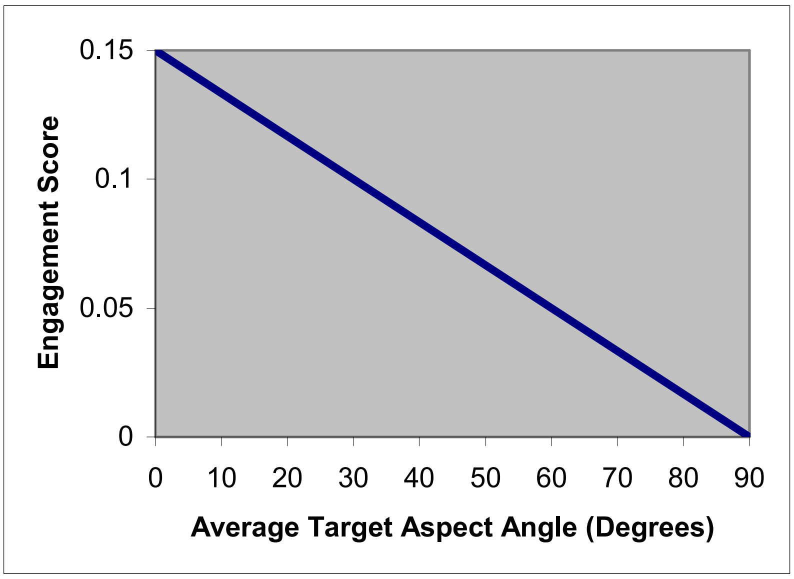

Figure 2: **Measure of effectiveness, which is used as an input to the GA fitness function.**

To encourage maneuvers that might enable gun firing opportunities, an additional score was added when the target was within 5 degrees of the aircraft's nose. A tax was applied to non-firing classifiers to discourage the proliferation of parasite classifiers that contain elements of high-performance classifiers but have insufficient material for activation. All non-firing classifiers that were identical to a firing classifier were reassigned the firing classifier's fitness.

1.7. GA Activation

- The GA acts at the end of each 30-second engagement.
- The GA is panmictic (it acts over the entire population).
- In some of our experiments, the *entire* classifier list is replaced each time the GA is applied. This has been surprisingly successful, despite the expected disruption of the classifier list. In recent experiments, we have used a generation gap of 0.5 (replacing half of the classifier population with the GA). This is still a substantially larger portion than are replaced in many LCSs.
- A new, GA-created classifier is assigned a fitness that is the average of the fitness values of its "parent" classifiers.
- The GA used employed tournament selection, with a tournament size ranging from 2 to 8.
- Typical GA parameters are a crossover probability of 0.95, and a mutation rate of 0.02 per bit position. When a condition bit is selected for mutation, it is set to one of the three possible character values (1, 0, or #), with equal probability. Note that this actually yields an effective mutation probability of (0.02)(2/3)=0.0133. Children rules replaced randomly selected rules in the population.

1.8. Conflict Resolution

The matching rule with the highest fitness/strength is selected to act *deterministically*.

1.9. Combat Simulation Starting Conditions

A two-tier approach was employed for defining run conditions and learning system parameters. First, a baseline matrix of initial positions, relative geometries, and energy states was identified in conjunction with NASA requirements. The primary source document for this step was the X-31 Project Pinball II Tactical Utility Summary, which contained results from manned simulation engagements conducted in 1993 at Ottobrunn, Germany [14]. Initial findings from the X-31 Tactical Utility Flight Test conducted at Dryden Flight Research Center were also used to compare with results from this project. The baseline, starting condition, test matrix, as shown in Figure 3, was based on X-31 manned simulation and flight test conditions, and was tailored to the X-31 performance envelope, flight test range constraints, and other testing considerations.

Note that each of these represents a separate initial condition for a fighter combat simulation. Each learning run consists of repeated engagements, all starting from the same one of these conditions.

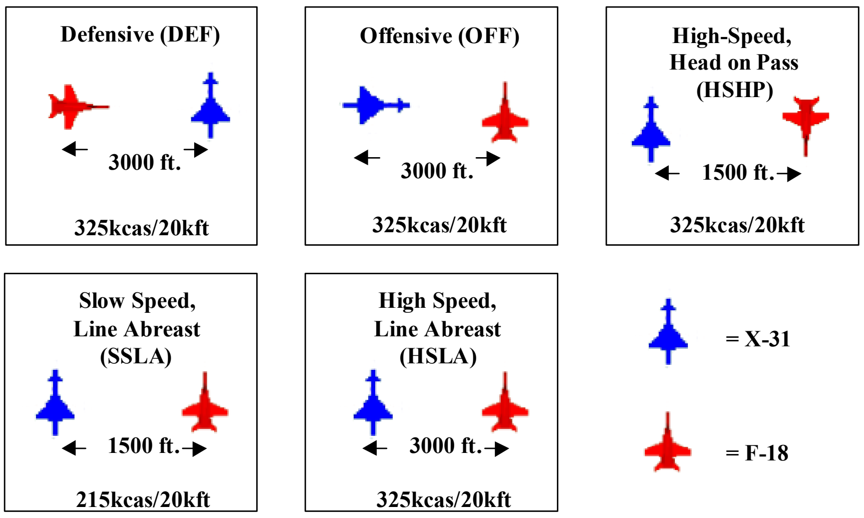

Figure 3: Baseline test matrix of initial conditions for the combat simulations.

The first four start conditions (Defensive (DEF), Offensive (OFF), Slow-Speed Line Abreast (SSLA), and High-Speed Line Abreast (HSLA)) were derived directly from the Pinball II project. The fifth start condition, High Speed Head-On Pass (HSHP), was added to the matrix to provide an additional geometry which would not exclusively result in a close turning fight. The opponent aircraft was an F/A-18. The baseline matrix formed a set of core conditions to generate X-31 tactics results for a balanced cross-section of tactically relevant conditions. The test conditions specified the initial geometries, X-31 and opponent speeds, altitudes and ranges.

2. "One-Sided Learning" Results

Results from the system outlined in Section 1 are extensively documented elsewhere [8,9]. However, results are reprinted here for clarification.

In our early efforts [8], only one of the fighter aircraft employs the genetic learning system, while the other employs fixed, but reactive, standard combat maneuvers that are embedded in AASPEM. Simply stated, these fixed maneuvers instruct the opponent to execute the fastest possible turn to point it's own nose at the LCS-controlled aircraft, while attempting to match it's altitude.

In this "one-sided learning" configuration, the system has found a variety of novel fighter aircraft maneuvers that were positively evaluated by actual fighter test pilots. One maneuever discovered by the LCS is shown in Figure 4. Moreover, the system discovers maneuvers from a variety of well-recognized fighter aircraft strategies [8]. For instance, the result shown in Figure 4 is a variant of the well-documented "Herbst maneuver".

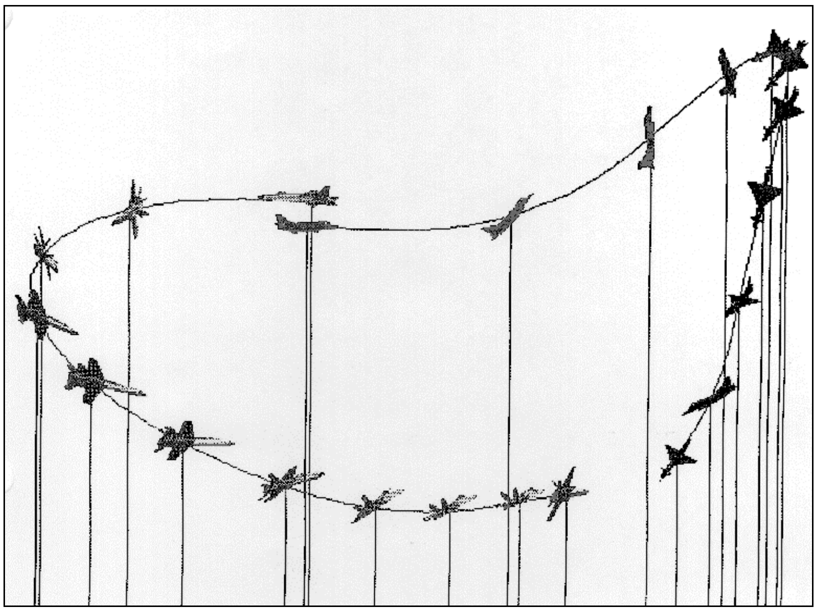

Figure 4: A maneuver evolved by the LCS, starting from the HSHP starting position (see Figure 3). The aircraft on the left is following a fixed, but reactive strategy, while the aircraft on the right is following a strategy evolved by the LCS.

3. "Two-Sided Learning" Results

In more recent work with the fighter combat LCS, we have allowed both opponents to adapt under the action of the GA [10]. This ongoing effort complicates the fighter combat problem, and the interpretation of simulation results. These complexities are best seen from the perspective of extant literature on two player games.
In many ways, the one-versus-one fighter combat scenario is like a continuous version of the *iterated prisoners' dilemma* (IPD) [17,18]. In the IPD, the game takes place in distinct rounds, in which each player can take one of two actions: cooperate or defect. If both players cooperate, both players get a moderate reward. If one player defects while the other cooperates, the former gets a large reward, and the latter gets a large punishment (negative reward). If both defect, both get an even larger punishment.

This is similar to the situation in the one-versus-one fighter combat scenario. Each aircraft has a choice of attacking the other (analogous to defecting), or evading the other (analogous to cooperating). This analogy is useful, since IPD has been studied a great deal, both with static and adaptive players. However, it is important to note that the current work may be one of the first explorations of a real-world, continuous time analog of IPD with two adaptive players.

The IPD literature for co-adaptive players shows us several interesting behaviors. Each of these is an artifact of the *red queen effect*, so-called, because the red queen in Alice in Wonderland states that in her world you must keep running just to stand still [19]. In an analogous way, the performance of each player in the two-sided learning problem is relative to that of its opponent. In other words, when one player adapts and the other uses a static strategy (as in our previous work), the performance of the adaptive player is absolute with respect to its opponent. However, when both players are adaptive, the performance ceases to have an absolute meaning. Instead, its meaning is only relative to the state of its current opponent. This is an important effect that must be considered in the interpretation of our current results.

Also because of the red queen effect, the dynamic system created by two players has several possible attractors. These include

- Fixed Points **Where each player adapts a static strategy, as is willing to cope with a (possibly inferior) steady state.**

- Periodic Behavior **Where each player cycles through a range of strategies, sometimes losing and sometimes winning.**

- Chaotic Behavior: **Where each player visits a random series of strategies, with no eventual escalation of tactics.**
- Arms Races **Where each player continuously ramps up the sophistication of its strategies.**

The latter is clearly the behavior we want our simulations to encourage. Our current results have (qualitatively) shown promise in this area (i.e., we have seen an escalation of strategies between the two aircraft).

A number of approaches to two-sided learning have been considered. In each approach, a "run" consists of 300 simulated combat engagements. Approaches employed include:

1. **Alternate freeze learning (ALT):** In each run one side is learning while the other is frozen (not altering its rule base). The frozen player uses the population (rules) obtained from the previous run, in which it was learning. The learning player starts each new run with a random set of rules. In other words, rules are learned “from scratch” against the rules learned in the last cycle by the other player.
2. **Alternate freeze learning with memory (MEM):** This learning scheme can be viewed as an extended version of the ALT learning. At the end of each run, the results of the 300 engagements are scanned to obtain the highest measure of effectiveness. The rules from the highest scoring engagement are used for the frozen strategy in the next run. Furthermore, these rules are memorized and are added to the population in the upcoming learning sequence runs. Thus, the system has memory of its previously learned behavior.
3. **Parallel learning (PAR):** Genetic learning goes on continuously for both players.

Each of the three two-sided algorithms were tested and evaluated on all five Pinball cases (see Figure 3). In all experiments a population of maneuvers was generated in an iterative process, beginning with an initial population of random classifiers. The maneuver population was successively improved using AASPEM air combat engagements to evaluate the maneuver.

A typical two-sided learning result (and one which indicates a learning "arms race") is shown in Figure 5 and Figure 6.

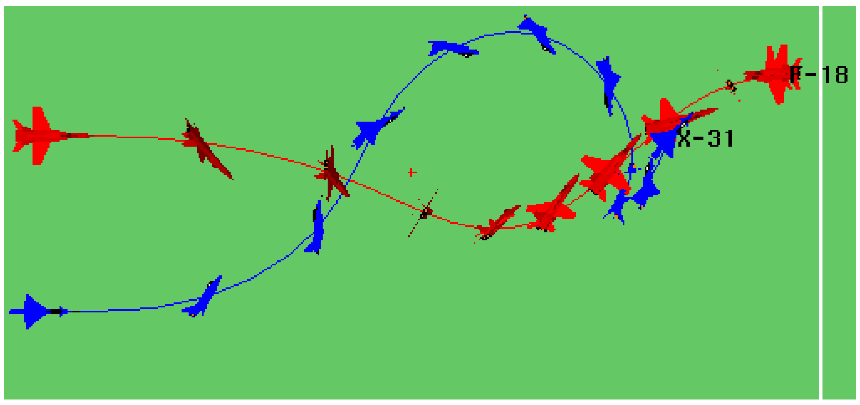

Figure 5: A coevolved result, where both aircraft are executing maneuvers that were discovered by the separate LCSs, starting from the SSLA initial condition (see Figure 3), and the ALT learning strategy. This is the "best" maneuver discovered by the Red Player, which is a reaction to the maneuver learned by the Blue Player.

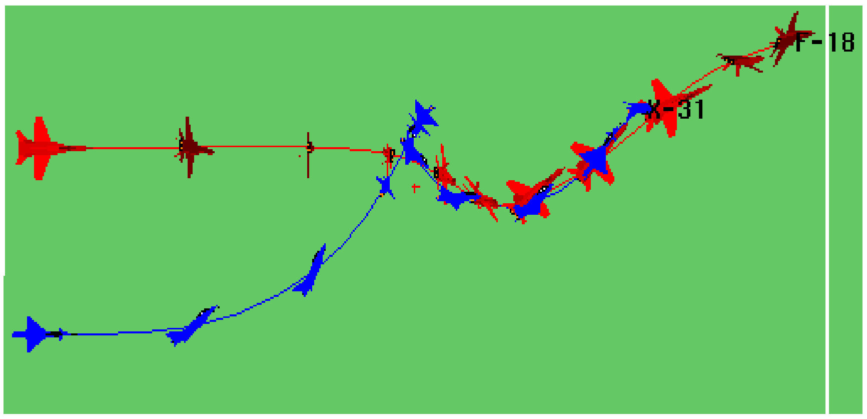

Figure 6: A coevolved result, where both aircraft are executing maneuvers that were discovered by the separate LCSs, starting from the SSLA initial condition (see Figure 3), and the ALT learning strategy. This is the "best" maneuver discovered by the Blue Player, which is a reaction to the "best" maneuver for the Red Player, which was discovered earlier by the Red Player's LCS (which is shown in Figure 5) .

Figure 5 shows the "best" maneuver (in terms of the measure of effectiveness discussed in Section 1.6) discovered for the Red Player. In this maneuver, the Red Player (which is an F-18) fails to have an advantage for more than the last half of the engagement due to agile maneuvers performed by the Blue Player (which is an X-31, a somewhat superior aircraft). The X-31 executes a nose dive maneuver followed by Herbst-type maneuver allowing it to gain an advantage. Although Red does not gain advantage, it is interesting to note its evasive behavior in reaction to blue’s maneuver.

Figure 6 is the Blue Player's "best" maneuver (once again, in terms of the measure of effectiveness), which is evolved after the result shown in Figure 5. It is likely that learning from the Red Player's evasion (Figure 5) resulted in the improved behavior seen in Figure 6. This illustrates the advantage of two-sided learning. However, note that the meaning of "best", in terms of a static measure of effectiveness, is not entirely clear in these simulations, due to the red queen effect. Therefore, further investigation of the two-sided learning system is needed. However, the two-sided learning system is already yielding valuable maneuver information, in relationship to the system's overall goals.

5 DIFFERENCES IN GOALS AND TECHNIQUES

We believe much of the success of the system presented above is due to the particular goal at which the system is directed. From the outset of the fighter aircraft LCS project, the goal has not been to directly *control* fighter aircraft. Instead, the focus has been on the discovery of novel maneuvers for their own sake. The utility of this goal is revealed when one considers the complexity of the aircraft combat task [7].

In a general sense, one can understand the aircraft's *aerodynamic* characteristics before an actual prototype is generated. However, given the complexity of the aircraft combat task, one cannot map this directly to maneuvers that are advantageous. Therefore, test pilots, and extensive combat trials (mock or real), are generally needed to discover the innovative maneuvers that will convert raw aerodynamics into combat success.

Therefore, the goal of or system simply the discovery of novel maneuvers, not optimality. There is, in general, no such thing as an optimal maneuver, and discovery of such a maneuver is not the system's goal. Discovering many *successful* combat maneuvers in this complex task has quantifiable advantages, including:

- Feedback to aircraft designers with regard to the combat advantages of various aerodynamic characteristics.
- Feedback to fighter pilots on the use of various aircraft characteristics.
- Feedback to customers on the advantages of a given aircraft's characteristics.

These advantages have real-world value. Moreover, they are advantages that could transfer to any number of other tasks where the discovery of novel strategies is difficult, and where new strategies have intrinsic values.

6 IMPLICATIONS OF THIS GOAL

In general, the goal of novelty discovery shifts the perception of what techniques are advantageous. For instance, the pursuit of *Q*-values that yield a Bellman optimal control strategy seems less critical in this setting. While it is true that one wants to pursue high utility maneuvers, it is not expected, that an "optimal" maneuver can be found. Moreover, enumerating all possible high-utility maneuvers is probably not possible. Finding classifier-based generalizations over the *entire* payoff landscape is probably not possible as well.

Most importantly, maintaining any *consistent* control strategy is not of great importance in this task. The frequent tendency of LCSs to have periodic failures is not a particular concern. Consider the typical, but previously unpublished result from the fighter aircraft system shown in Figure 7.

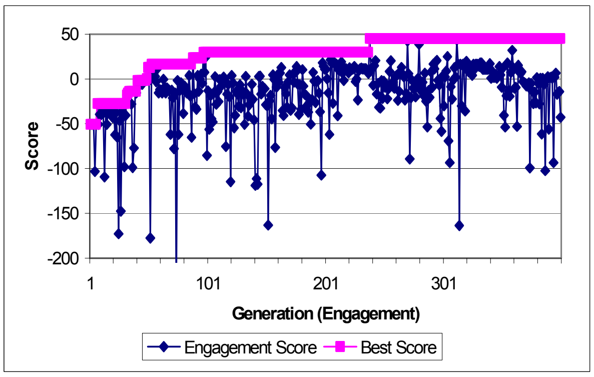

Figure 7 Engagement Score for the LCS aircraft versus Generation Number. In this typical run, only one aircraft is using the LCS, while the other uses standard combat maneuver logic, as in the run shown in Figure 4.

This figure shows that the fighter aircraft LCS frequently loses rules and suffers temporary losses of performance. However, the system moves to improved performance (and new maneuvers). Thus, it accomplishes the goal of *discovery,* but not necessarily the goal of control.

7 OBSERVATIONS AND SUGGESTIONS FOR FUTURE LCS EFFORTS

The work presented here is a part of an ongoing effort. The LCS presented is an artifact of several years of incremental design, and its performance is likely to benefit from more recent developments in LCS technology. However, the system continues to provide useful results in a complex, real-world domain.

The main key to this system's success is in the nature of the application itself. This task is one where discovering novelty has quantifiable value to designers, pilots, and customers (rather than at online control). Discovering similar application domains, where using genetics-based machine innovation has a real-world value, is an important area for future efforts.

On a more technical level, there are features of the current fighter aircraft LCS that deserve consideration in the design of other LCSs.

One should consider the "smoothed" interpretation of actions in the fighter combat LCS. The authors strongly suspect that this is a reason for the LCSs success on this task. Many of the action representations used in LCS research are "brittle". That is, rule deletion, and other forms of classifier set disruption, can easily cause less-than-graceful failure. In the fighter aircraft task, the actions of rules are implicitly interpreted in a linguistic, aerodynamically appropriate manner. This makes each classifier less sensitive changes in the composition of the classifier set, which are likely to occur, given the action of the GA.

One should also consider the fighter aircraft LCS credit allocation scheme. Given the efficacy and popularity of *Q*-learning and related algorithms (e.g., SARSA, which is essentially the implicit bucket brigade), it is easy to overlook that there are well-founded techniques that do not follow their form. In particular, the Monte-Carlo RL methods are just as firmly related to reinforcement learning theory. In fact, Sutton and Barto [11] indicate that such methods may be more appropriate than *Q*-learning and related methods, when one faces a non-Markovian task.

However, such methods are only appropriate for tasks with clearly defined episodes. The fighter aircraft task is clearly episodic, since engagements take place in a pre-defined time window. When one directs an LCS at an episodic task, epochal credit assignment schemes, like that used in the fighter aircraft LCS, may be of higher utility than methods based on *Q*-learning, SARSA, or the bucket brigade.

Like the *Q*-learning based methods used in LCSs, epochal schemes can only approximate the associated reinforcement learning techniques, since such techniques are typically based on table lookup. Given that the focus of the LCS is on finding generalized rules, associated reinforcement learning techniques, be they epochal or not, must be adapted to evolving, generalized rule sets. XCS [14] seems to have found an appropriate adaptation of *Q*-learning for an LCS. Finding an appropriate adaptation of well founded, epochal reinforcement learning schemes to LCS use is an area worthy of future research.

However, when considering the technical details of reinforcement learning control (that is, obtaining optimal strategies in reinforcement learning problems), one should not overlook the utility of the LCS approach for generating novel, innovated approaches to problems. Such goals may not fall within the rubric of strict optimality, but in many domains (like the fighter aircraft task), such open-ended discovery can have a real world, hard cash value. The applicability of the LCS approach to such creative tasks deserves further consideration.

ACKNOWLEDGMENTS

The authors gratefully acknowledge that this work is sponsored by The United States Air Force (Air Force F33657-97-C-2035 and Air Force F33657-98-C-2045). The authors also gratefully acknowledge the support provided by NASA for the early phases of this project, under grant NAS2-13994.

REFERENCES

1. Booker, L. B. (1992) Viewing Classifier Systems as an Integrated Architecture. Paper presented at The First International Conference on Learning Classifier Systems, Houston, Texas, October 1.
2. Goldberg, D. E. (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.
3. Grefenstette, J. J. (1988) Credit assignment in rule discovery systems based on genetic algorithms. Machine Learning 3. pp. 225-246.
4. Holland, J. H. (1992) Adaptation in Natural and Artificial Systems MIT Press.
5. Holland, J. H., Holyoak, K. J., Nisbett, R. E. and Thagard, P. R. (1986) Induction: Processes of inference, learning, and discovery. MIT Press, Cambridge, MA.
6. Holland, J. H. and Reitman, J. S. (1978) Cognitive systems based on adaptive algorithms. In Waterman, D. A. and Hayes-Roth, F., " Pattern directed inference systems". Academic Press, NY.
7. Shaw, R. L. (1998) Fighter Combat : Tactics and Maneuvering. United States Naval Institute Press.
8. Smith, R. E. and Dike B. A. (1995) Learning novel fighter combat maneuver rules via genetic algorithms. International Journal of Expert Systems, 8(3) (1995) 247-276.

9. Smith, R. E., Dike, B. A. and Stegmann, S. A. (1994) Inheritance in Genetic Algorithms, in: Proceedings of the ACM 1995 Symposium on Applied Computing. ACM Press. pp. 345-350.

10. Smith , R. E., Dike, B. A., Mehra, R. K., Ravichandran , B. and El-Fallah, A. (in press). Classifier Systems In Combat: Two-Sided Learning of Maneuvers For Advanced Fighter Aircraft. Computer Methods in Applied Mechanics and Engineering, Elsevier.

11. Sutton, R. S. and Barto, A. G. (1998) Reinforcement Learning: An Introduction. MIT Press.

12. Watkins, J. C. H. (1989). Learning with delayed rewards. Unpublished doctoral dissertation. King's College, London.

13. Wilson, S. W. (1994) ZCS: A zeroth-level classifier system, Evolutionary Computation 2(1). pp. 1-18.

14. Wilson, S. W. (1995). Classifier fitness based on accuracy. Evolutionary Computation, 3(1), 149-176.

15. Wilson, S. W. (1999) State of XCS Classifier System Research. Technical Report Number 99.1.1 (Unpublished), Prediction Dynamics, Concord, MA.

16. P. M. Doane, C. H. Gay and J. A. Fligg, Multi-system integrated control (MuSIC) program. final report. Technical report, Wright Laboratories, Wright-Patterson AFB, OH., 1989.

17. R. Axelrod, The Evolution of Cooperation. (Basic Books, New York, 1984)

18. R. D. Luce and H. Raiffa, Games and Decisions. (Dover Publications, 1989).

19. D. Floriano and S. Nolfi, S., God save the red queen!: Competition in co-evolutionary robotics, in: Proceedings of the Second International Conference on Genetic Programming, (1997) 398-406.