Evolutionary Agent-Based Simulation of the Introduction of New Technologies in Air Traffic Management

Logan Yliniemi
Oregon State University
Corvallis, OR, USA
logan.yliniemi@engr.orst.edu

Adrian Agogino
UCSC at NASA Ames
Moffett Field, CA, USA
adrian.k.agogino@nasa.gov

Kagan Tumer
Oregon State University
Corvallis, OR, USA
kagan.tumer@oregonstate.edu

ABSTRACT

Accurate simulation of the effects of integrating new technologies into a complex system is critical to the modernization of our antiquated air traffic system, where there exist many layers of interacting procedures, controls, and automation all designed to cooperate with human operators. Additions of even simple new technologies may result in unexpected emergent behavior due to complex human/machine interactions. One approach is to create high-fidelity human models coming from the field of human factors that can simulate a rich set of behaviors. However, such models are difficult to produce, especially to show unexpected emergent behavior coming from many human operators interacting simultaneously within a complex system. Instead of engineering complex human models, we directly model the emergent behavior by evolving goal directed agents, representing human users. Using evolution we can predict how the agent representing the human user reacts given his/her goals. In this paradigm, each autonomous agent in a system pursues individual goals, and the behavior of the system emerges from the interactions, foreseen or unforeseen, between the agents/actors. We show that this method reflects the integration of new technologies in a historical case, and apply the same methodology for a possible future technology.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Performance, Experimentation

Keywords

Emergent Behavior; Multiagent Learning

1. INTRODUCTION

Air traffic management is an incredibly complex problem, with many layers of interacting procedures, controls, and automations, all interacting in real-time with human operators. Developing a new piece of technology to fit into this framework is a challenging process in which many unforeseen consequences. Thus, the ability to simulate new technologies within an air traffic management framework is of the utmost importance.

One approach is to create a highly realistic simulation for the entire system [13, 14], but in these cases, every technology in the system must be modeled, and their interactions must be understood beforehand. If this understanding is lacking, it may be difficult to integrate them into such a framework [16]. With a sufficiently complex system, missed interactions become a near-certainty.

Another approach is to use an evolutionary approach in which a series of simple agents evolves a model the complex behavior of the system through their interactions. Such agents have been used to gain insight into such systems as macroeconomics [3, 4, 8] and supply chain management [2]. However, the problem of defining the policies that the agents should follow is non-trivial, especially in the case where the agents may act in a heterogeneous manner: development of such agents can quickly become cyclical, with changes to one agent necessitating changes to others. These approaches are also prone to over-engineering and can quickly become too tailored to the specific problem to offer any generality or reusability.

Instead, in this work we use an evolutionary algorithm on agent policies to allow the agents to discover policies that meet certain designer-specified goals. This then trades off designer workload and required expertise for computing time. Such agents can develop both simple policies that directly meet system goals as well as more complex, emergent behaviors. We demonstrate this as a lens for examining new potential technologies by examining the historical case of the Norden Bombsight, and another situation which represents a possible future technology.

The major contributions of this work are to:

- Show that even simple behavioral models can be evolved that have predictive value in assessing technology introduction by exploring predicted outcomes of technology infusion
- Show that even with such simple evolved behavioral models, complex "emergent" behavior (not predicted or expected) may arise

The remainder of this work is organized as follows: In Section 2 we present background on evolutionary policy search and goal-based agents, as well as introducing the historical case that we use as a framework for our simulations. In Section 3 we present the domains used in this paper. In Section 4 we describe the algorithms used. In Section 5 we include the experimental parameters used for our experiments. In Section 6 we include our experimental results and Finally, in Section 7, we draw the conclusion of this work.

GECCO'14, July 12-16, 2014, Vancouver, BC, Canada.

2. BACKGROUND

This work employs an evolutionary algorithm to search for agent policies. In this section we describe these concepts and introduce the historical case of the Norden Bombsight.

2.1 Learning Agents / Multiagent System

Learning agents are autonomous actors in a system which make observations about their environment, reason based on these observations, perform actions that affect their environment, and use feedback from the environment to update their policies [15]. In a learning multiagent system, multiple agents are simultaneously performing this process. These agents exist in other agents' observations, and all actions taken affect the common environment [17].

2.2 Evolutionary Algorithms

We use an evolutionary algorithm in this work, in which agent policies are represented by neural networks. We evaluate their fitness with a calculation specific to each case that we present (Section 4). Once completed, the policies with the lowest fitness are removed from the system and replaced with slightly-modified versions of more successful policies. Over time, the population tends toward more desirable policies [1].

2.3 Agent Simulation / Goal-Based Evaluation

Romano et al. state that "Intelligent agents can act as basic building blocks of an artificial society and be used to study the emergency of the social behaviour. Agents can be autonomous, flexible, adaptable and goal-directed" [12]. This lets "Behaviours 'emerge' from the actions of these diverse units in constant motion" [12].

We take a similar approach in this work, where we merely set forth goals for the agents to achieve, and allow their collective search for good policies to lead to behaviors that emerge. In many cases these behaviors align with designer predictions, but unexpected behaviors can also emerge. In this way, the system designer need only describe the desired outcome of the system behavior, and not how the agents should go about reaching those goals.

Agent simulation has been used for modeling complex systems like macro-economics [3, 4, 8]. In these situations, simple agents interacting with the market mirror real-life observed phenomena, like the creation of "bubbles" in the market [8]. Other agent-based simulations have been used to investigate such systems as supply chains [2], network security [9], maritime piracy [7], and evacuation of dangerous environments [5, 11].

Authors investigating systems with agent-based simulation have argued for the need of simplicity in the agent representation as well [8, 9]. As LeBaron states, learning agents are put to use "that are simple enough for easier analysis and interpretation, yet rich enough to pursue many of the experiments in evolution and heterogeneity present in older, more complex frameworks" [8]. It is not the complexity of an individual agent, but rather the interaction between these agents in their shared environment, that creates the complex emergent behaviors that can be observed.

2.4 The Norden Bombsight

To assist in examining the potential effects of a new technology on a modern complex system, it is useful to consider the effects that new technologies have had on complex systems in the past. In this work we consider the well-documented historical case of the Norden Bombsight.

In World War II, the Norden Bombsight was used by the United States Army Air Force (USAAF) for their precision daylight raids over Europe. While it had a number of drawbacks (such as requiring visual contact with the desired target), it was widely adopted and used in the European theatre, with the USAAF ordering 90,000 units, and training 50,000 bombardiers to use them at a development cost of 1.5 billion USD [6].

The basic functionality of the Norden Bombsight was an analog computer that was capable of calculating the location which a bomb should be dropped to land on the sighted target, given inputs of airspeed, altitude, and crosswind velocity [6]. This bombsight offered an advantage over earlier tactics, which consisted of a visual release method without the benefit of the analytical power of the Norden. Even so, a potential problem arises: If a group of bombers in formation are attempting to release their payloads to all contact the same area, as they approach the release point, the separation of the planes will reduce to a potentially dangerous level.

In the case of the physical world, new techniques are not adopted without fully considering potential safety risks. This, combined with the fact that the 50,000 trained bombardiers had varying skill levels at the analog programming required by the Norden, resulted in the decision by 1944 that the lead bomber should be the only one using the Norden. The other bombers merely held formation with respect to the lead, and released their payload when they saw the lead bomber do so. This reduced the accuracy of the formation as a function of the size of the formation and the ability of the pilots to hold their position with respect to the bomber ahead of them, but removed the problems of separation assurance.

Though the Norden Bombsight was not as revolutionary as it was originally touted to be, nor was its use as successful as originally hoped, it still represented a new technology in an already-existing, complex system, which changed the behaviors of the actors in the system. This is our basis for including this historical example, which forms the framework for our Norden Bombsight Domain (Section 3.1) in this work.

3. DOMAINS OF STUDY

In this work we use two domains of study to show how agents can be evolved to discover emergent behavior.. The first is the Norden Bombsight Domain (Section 3.1), which models the historical example of the Norden Bombsight described in Section 2.4. The second is the "Civilian 'Cloud' Domain", a related domain which represents a possible future case of technology integration.

3.1 Norden Bombsight Domain (NBD)

In the Norden Bombsight Domain, a series of planes must cooperatively traverse a corridor from south to north, and pass as closely as possible over a single point (target location) at the far end of the corridor. The planes all travel at a constant speed. Each plane agent senses local information using the following state variables:

1. dx: X-distance from target location
2. dy: Y-distance from target location
3. V_x: X-velocity
4. V_y: Y-velocity
5. p_x: X-distance to nearest other plane
6. p_y: Y-distance to nearest other plane
7. c_x: X-distance to centroid of k nearest cloud centers
8. c_y: Y-distance to centroid of k nearest cloud centers

and each plane agent takes action by selecting the bearing that they wish to proceed toward. In the Norden Bombsight Domain, the cloud states (7 and 8) have no bearing on the goals of the planes, but they are included in the state to keep the simulations presented in this work homogeneous.

After each agent chooses an action, system dynamics described in Section 4 are then calculated, and the plane agents move to their location for the next timestep. This process is repeated until all planes exit one of the edges of the corridor or a set number of timesteps has passed. At the end of each episode, the agents in use are scored based on their performance measured on the goals set forth for them. We use the exact same agent structure in 3 separate cases, while changing the goals slightly in each case. The cases for the Norden Bombsight Domain mirror the historical case of the Norden Bombsight.

3.1.1 Case 1: Pre-Norden Bombsight

Before the integration of the Norden Bombsight, the USAAF used pilot and bombardier visual cues to identify the target, but had no precise way of calculating the ideal launch point.

We model this by making the agents' goals to (i) pass through the north end of the corridor, with no emphasis on passing over the precise target point, and (ii) maintain separation assurance with the other aircraft in flight.

3.1.2 Case 2: Full Norden Integration

With the integration of the Norden Bombsight, the USAAF had the ability to precisely identify the target location, and the Norden Bombsight itself was used as an autopilot for the plane on the final approach, where it considered only its relative position to the precise target location when giving control commands to the bomber, without considering the other planes in the formation in any way.

We model this by making each agent's goal in this case to (i) pass over the target point as precisely as possible, while ignoring any contribution from separation assurance.

3.1.3 Case 3: Norden + Separation

The USAAF eventually settled on using the Norden Bombsight on the lead bomber in the flight (with inactive backup devices on other members of the formation), while all other bombers were to maintain formation with respect to the lead plane. This had two benefits: it reduced the system to an effectively centralized controller, and removed the separation assurance problems that would have been introduced by full integration (Case 2).

We model this as a multiagent system wherein the agents still have the ability to override the autopilot for the sake of separation assurance. The agents' goals in this case are to (i) pass over the target point as precisely as possible while (ii) maintaining separation assurance with the other aircraft in flight.

3.2 Civilian "Cloud" Domain (CCD)

While the Norden Bombsight Domain allows ready comparison to historical reality, we include a second, related domain to show the general applicability of the methods used, and to remove any possible bias due to the knowledge of historical outcomes that might be present in the Norden Bombsight Domain.

In the Civilian Cloud Domain, the setup is much the same: a set of planes must traverse a corridor from south to north using the same state information as allowed in the Norden Bombsight Domain, and would ideally like to pass over a waypoint at the end of the corridor. The complication arises by the introduction of areas of the corridor that are preferably avoided. These areas may represent turbulence, storms, other incidental planes in holding patterns, or areas of poor visibility. For the ease of the reader, we collectively term all of these areas to avoid "clouds", and represent them as a single goal in the simulation.

We present three different cases in the Civilian Cloud Domain to exhibit the types of behaviors that learning agents can develop.

3.2.1 Case 4: Explicit Cloud Avoidance, No Spacing

In this case, we model a system with each plane equipped with a cloud-avoiding autopilot as it navigates through the corridor. Similar to Case 2, we define that these autopilots do not coordinate with each other in any way.

We model this by making their goals (i) to make it through the corridor to the target point successfully, while (ii) avoiding clouds. Avoiding other aircraft is not involved in their utility function.

3.2.2 Case 5: Explicit Cloud Avoidance + Spacing

In a second iteration of the Civilian Cloud Domain, we model a system in which each plane is equipped with a cloud-avoiding autopilot which *does* have the explicit goal of avoiding separation violations with other planes.

Their goals are to (i) make it through the corridor successfully, while (ii) maintaining separation assurance from other planes, and (iii) avoiding clouds.

3.2.3 Case 6: Implicit Cloud Avoidance

In our final case of the Civilian Cloud Domain, we model a system in which the cloudy areas of the airspace are able to be sensed, but avoiding clouds is not an explicit goal of the agents. However, *in this case only*, we remove their ability to sense other planes while in cloudy regions of the airspace (within 50 units of a cloud center), creating potentially dangerous conditions that are not explicitly within the utility function of the agents.

In this case, the agents' goals are to (i) make it through the corridor successfully, while (ii) maintaining separation assurance from other planes. Though the cloudy areas of the space aren't directly within the utility function, planes within the cloudy regions may experience un-sensed separation violations that occur, making the cloudy regions potentially dangerous.

4. SIMULATION DETAILS

In both simulation domains, we use the same evolutionary algorithm to execute a simulation. Certain steps do change depending on the case. The overall simulation algorithm is presented as Algorithm 1. The calculation of the utility based on the goals of the agent for each case is shown in Algorithms 2 and 3.

4.1 Simulation Algorithms

In Algorithm 1, a population of neural networks is formed, and a subset is selected to perform the first simulation. In this simulation, at each timestep each agent senses information about its surroundings into the state vector S_a, and evaluates its neural network on the state vector $(\mathcal{NN}(S_a))$, before scaling its output to be in the range [0:2π]. This value represents the desired heading $\hat{\theta}_a$. The actual heading for this timestep θ_a is then calculated as a weighted average between θ_a for the previous timestep and $\hat{\theta}_a$ for the current timestep, with a weighting factor ρ. In the event that θ_a and $\hat{\theta}_a$ lie on opposite sides of the [0:2π] discontinuity, the smallest angle between the two is measured and used for this calculation. The plane's x- and y-location are then updated. At this point, timestep-level utilities are calculated (per $\bigstar_1$ steps in Algorithm 2 or 3).

After the final timestep, the episode-end utilities are calculated ($\bigstar_2$), and the final utility for each agent used during that simulation is calculated ($\bigstar_3$). After all agents have been used in one of the simulations, the m lowest-performing networks are removed, and m new copies of surviving networks are produced. These new networks have their weights modified with a 20% chance per link, by adding a random number drawn from a normal distribution with zero mean and standard deviation 0.2.

This process is repeated for each generation. This process is uniform across all 6 cases presented here. The only changes to the process are the steps marked $\bigstar_{\{1,2,3\}}$ and the sensing step in Case 6, in which agents cannot detect planes when they are within 50 units of a cloud center.

4.2 Goal Evaluation

The primary difference between the cases is the goals that the agents are trying to meet, and how these affect their fitness calculation. In each case, the primary goal is to traverse the corridor from south to north. To emphasize this over all other considerations, the reward for successfully exiting the corridor to the north ($u_{success}$) is multiplied by a very large number (R), such that all solutions that successfully complete this challenge will have a higher fitness than those that do not. This is a form of lexicographic ordering [10], with completion prioritized higher than all other goals, which are weighted against each other. For ease, before applying weights to the goals, they are normalized in the range [0:1], with the maximum attainable goal taking on a value of 1.

4.2.1 Norden Bombsight Domain Goals

Algorithm 2 lays out the goals for the first 3 cases, which are executed in the Norden Bombsight Domain. In the first case, at each timestep the separation from the nearest plane is calculated, and the separation assurance utility u_{sep} is updated. At the end of the run, the binary success measure $u_{success}$ is calculated, which is 1 if the plane successfully passes through the north end of the corridor, and 0 if the plane either does not exit the corridor within the allotted time, or exits out of one of the other sides. Finally, these two rewards are combined to form the agent's overall utility U, which is used to determine which agents will be eliminated.

In the second case, the separation calculation is replaced by a calculation for the closest distance the agent passed over the target location, leading to u_{target}. The function $dist(target)$ returns the distance to the target, scaled between 0 and 1, with 1 being the maximum possible distance to the target. This value and $u_{success}$ are the only contributions to the overall utility U in Case 2.

In the third case, both u_{sep} and u_{target} are calculated, and they are combined in the final U calculation with weights W_{sep} and W_{target}, respectively, which allows the system designer to specify the perceived tradeoff between passing directly over the target and maintaining separation assurance throughout the flight.

Algorithm 1 Simulation Process for NBD and CCD

Initialize population of n neural networks.
for $generation = 1 \rightarrow max\_generations$ **do**
  Select P neural networks not used yet this generation
  Assign one of the P neural networks to each plane
  for $simulation = 1 \rightarrow sims\_per\_gen$ **do**
    for $timestep = 1 \rightarrow max\_timesteps$ **do**
      for $a = 1 \rightarrow P$ **do**
        Sense: $S_a \leftarrow \{d_x, d_y, V_x, V_y, p_x, p_y, c_x, c_y\}$
        Choose action: $\hat{\theta}_a \leftarrow \mathcal{NN}(S_a) \in \{0{,}2\pi\}$
        Adjust plane heading: $\theta_a \leftarrow \theta_a + \rho * (\theta_a - \hat{\theta}_a)$
        Update x-location: $x_a \leftarrow x_a + |V|\cos(\theta_a)$
        Update y-location: $y_a \leftarrow y_a + |V|\sin(\theta_a)$
        Calculate timestep-level goal values $\bigstar_1$
      end for
    end for
    for $a = 1 \rightarrow P$ **do**
      Calculate end-of-simulation goal values $\bigstar_2$
      Calculate agent utility U $\bigstar_3$
    end for
  end for
  Remove m neural networks with lowest utility U
  Create m neural networks, identical to surviving networks
  Mutate the newly created networks
end for

4.2.2 Civilian Cloud Domain Goals

Algorithm 3 lays out the goals for the final 3 cases, which are executed in the Civilian Cloud Domain.

In Case 4, the goals are to proceed through the space without coming too near the cloud regions, and pass over a target. No consideration is given to the separation between airplanes.

In Case 5, we give the agents explicit utility for staying out of the cloudy regions of the airspace, with u_{cloud}, which is again aggregated into U with weight W_{cloud}.

In Case 6, we do not explicitly give utility for avoiding cloudy regions, but the agents benefit from avoiding these regions because their sensors for the nearest plane (which does form part of the utility calculation) are only effective outside of these regions: Staying outside of cloudy regions becomes an implicitly coupled goal.

Algorithm 2 Goals for NBD Cases

Case 1: Pre-Norden Bombsight
$\bigstar_1$: if($|p_x, p_y| < \beta$), $u_{sep} \leftarrow u_{sep} - (|p_x, p_y| - \beta)^2$
$\bigstar_2$: if($run\_success$), $u_{success} = 1$
$\bigstar_3$: $U = u_{success} * R + u_{sep}$
Case 2: Full Adoption of Norden Bombsight
$\bigstar_1$: $u_{target} = max(u_{target}, 1 - dist(target))$
$\bigstar_2$: if($run\_success$), $u_{success} = 1$
$\bigstar_3$: $U = u_{success} * R + u_{target}$
Case 3: Norden Bombsight + Separation
$\bigstar_1$: if($|p_x, p_y| < \beta$), $u_{sep} \leftarrow u_{sep} - (|p_x, p_y| - \beta)^2$
$\bigstar_1$: $u_{target} = max(u_{target}, 1 - dist(target))$
$\bigstar_2$: if($run\_success$), $u_{success} = 1$
$\bigstar_3$: $U = u_{success} * R + W_{sep} * u_{sep} + W_{target} * u_{target}$

Algorithm 3 Goals for CCD Cases

Case 4: Cloud Avoidance, No Separation Assurance
$\bigstar_1$: $u_{target} = max(u_{target}, 1 - dist(target))$
$\bigstar_1$: if($|p_x, p_y| < \beta$), $u_{sep} \leftarrow u_{sep} - (|p_x, p_y| - \beta)^2$
$\bigstar_2$: if($run\_success$), $u_{success} = 1$
$\bigstar_3$: $U = u_{success} * R + W_{cloud} * u_{cloud} + W_{target} * u_{target}$
Case 5: Cloud Avoidance with Separation Assurance
$\bigstar_1$: $u_{target} = max(u_{target}, 1 - dist(target))$
$\bigstar_1$: if($|p_x, p_y| < \beta$), $u_{sep} \leftarrow u_{sep} - (|p_x, p_y| - \beta)^2$
$\bigstar_1$: $\forall_i$ if($dist(C_i) < \beta_{cloud}$), $u_{cloud} \leftarrow u_{cloud} - dist(C_i)^{-2}$
$\bigstar_2$: if($run\_success$), $u_{success} = 1$
$\bigstar_3$: $U = u_{success} * R + W_{target} * u_{target} + W_{sep} * u_{sep} + W_{cloud} * u_{cloud}$
Case 6: Implicit Cloud Avoidance
$\bigstar_1$: if($|p_x, p_y| < \beta$), $u_{sep} \leftarrow u_{sep} - (|p_x, p_y| - \beta)^2$
$\bigstar_1$: $u_{target} = max(u_{target}, 1 - dist(target))$
$\bigstar_2$: if($run\_success$), $u_{success} = 1$
$\bigstar_3$: $U = u_{success} * R + W_{sep} * u_{sep} + W_{target} * u_{target}$

5. SIMULATION SETUP

In all experiments, we use a population of $n = 110$ agents of 8-input, 6-hidden unit, 1-output neural networks that are assigned

to $P = 10$ planes over $sims\_per\_gen = 11$ simulations each generation, for $max\_generations = 500$ generations. In each simulation, the policies used are randomly selected from those that have not been used in that generation.

We instruct the planes to maintain $\beta_{plane} = 15$ units of separation from each other at all times, and the planes move at a rate of 10 units per timestep. We use an effective cloud radius of $\beta_{cloud} = 50$ units. After each generation, $m = 10$ agents are removed from the population and replaced with mutated copies of more-fit agents.

The agents start at the south end of the corridor, and are directed toward a target in the center of the north end of the corridor, 1000 units away. All agents start with heading $\theta_a = \pi/2$ (North). The system dynamics are governed by the planes travelling at $|V| = 10$ units per timestep, and a heading adjustment coefficient of $\rho = 0.5$. Cases 1 and 2 involve no weights. In Cases 3-6, all weights are set to a value of 1.0 unless otherwise noted in the results.

6. RESULTS

We present the outcome of the simulation for each case presented in Section 3. The first three cases mimic the historical Norden Bombsight, while the last three cases serve as a test case, without the designer-side bias introduced by historical knowledge.

In every case we perform 30 statistical runs, and then present the simulation output for the statistical run which attained the sum of agent performance closest to the mean value across all 30 runs. We offer these plots for each of the 6 cases representing the different goal combinations that the agents were tested with (Figures 1–9).

Interpretation of Figures.

Each of the figures represents each aircraft by a distinct color-symbol combination. The lines plot out each timestep, with the symbols appearing every 10 timesteps. The planes proceed from the bottom of the figure through the top. It is important to note that if a plane travels in the x-direction, even if it appears to intersect the path created by another plane, in many cases it is in the same location many timesteps after the first plane was there.

Clouds are represented by grey circles, which represent the maximum radius at which the cloud will have effect on an agent's reward. Being closer to a cloud center is worse than being near the edge, and passing over multiple clouds simultaneously has an additional negative effect on the agent's reward.

6.1 Case 1 Results

In Case 1, the agents seek to (i) pass through the north end of the corridor, with no emphasis on passing over the target point, and (ii) maintain separation assurance with the other aircraft in flight. This represents the case in which the USAAF did not have an accurate method for determining the ideal release point for their payloads.

The agents do not show a preference to where they exit the corridor, and are only concerned with staying spaced out with respect to the other aircraft. The agents quickly converge on a solid policy: move straight forward to the end of the corridor, and perform slight corrections when too close to another plane. Other, more complex methods may exist that could be discovered by a thorough consideration of the problem by a human expert, but the simplest policies are both easier to arrive at, and tend to be the most consistent performers, resulting in this straightforward solution.

6.2 Case 2 Results

Within Case 2, the agents operate with the sole goal: (i) pass over the target point as precisely as possible, while ignoring any contribution from separation assurance. This represents a potential outcome if the USAAF were to adopt a full-scale integration of the Norden Bombsight

This leads to agent policies where the agents learn to direct themselves directly over the target location with no heed for other concerns. This means that the planes would be travelling over the target at very nearly the same time, making a very unsafe situation for all involved. This result agrees with the USAAF determination not to use the Norden Bombsight aboard all of the aircraft within the formation, and is another case where a very simple policy leads to high individual utility.

6.3 Case 3 Results

In Case 3, the agents have the two explicit goals of (i) to pass over the target point as precisely as possible while (ii) maintain-

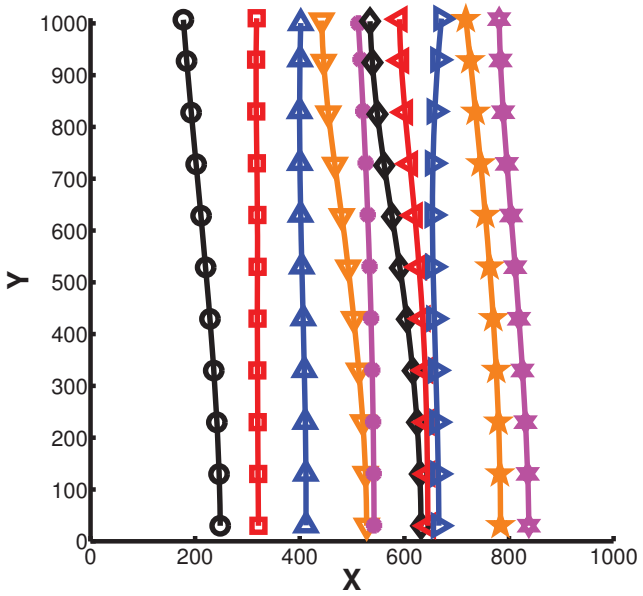


Figure 1: Case 1 — Pre-Norden Bombsight.

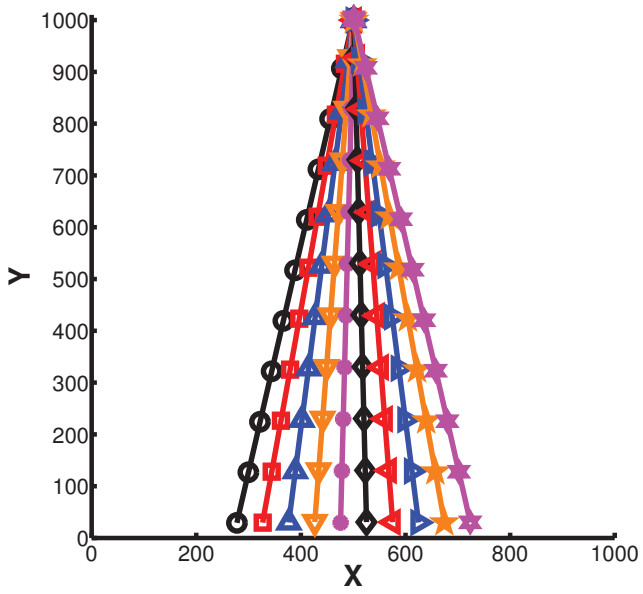


Figure 2: Case 2 — Full Norden Integration.

ing separation assurance with the other aircraft in flight. These two goals are conflicting, however, so we examine the effect of different weights on the paths chosen by the simulated aircraft. This represents a partial integration of the Norden Bombsight.

A number of interesting phenomena arise. When spacing is given a near-zero weight $W_{sep} = 0.1$ (Figure 3), the simulation reveals that the aircraft will converge to the center target point at the top of the corridor, as in Case 2. Some signs of separation concerns do appear, however, as Figure 3 shows one of the aircraft peeling off from the rest, in favor of maximizing its spacing utility over the target utility. The agents also do not converge to as tight of a group at the target point as in Case 2.

When we move the weight to consider separation $W_{sep} = 0.5$, we find that there is a tradeoff that the agents begin to make, shown by the simulation in Figure 4: there is less convergence toward the target point, and some of the agents choose to stay away from the congested center area in favor of maintaining separation assurance.

As we increase the weight for separation equal to that of the target weight $W_{sep} = 1.0$, the simulation shows that the planes favor maintaining their spacing more than passing over the center point, and spread out further. However, Figure 5 shows that the agents begin to develop a secondary strategy: they take a longer path toward the target, so that they arrive in the area at a later point in time. Notice particularly in Figure 5 the three aircraft starting at the right cross all the way behind the other planes to end up to the left of their target. This longer path takes them longer to traverse than the planes taking a more direct path, so they avoid separation violations in this way.

This is taken to the extreme when we then reduce the target weight to $W_{target} = 0.01$ and the ideal separation to a larger value

Figure 3: Case 3 — Little weight on separation.

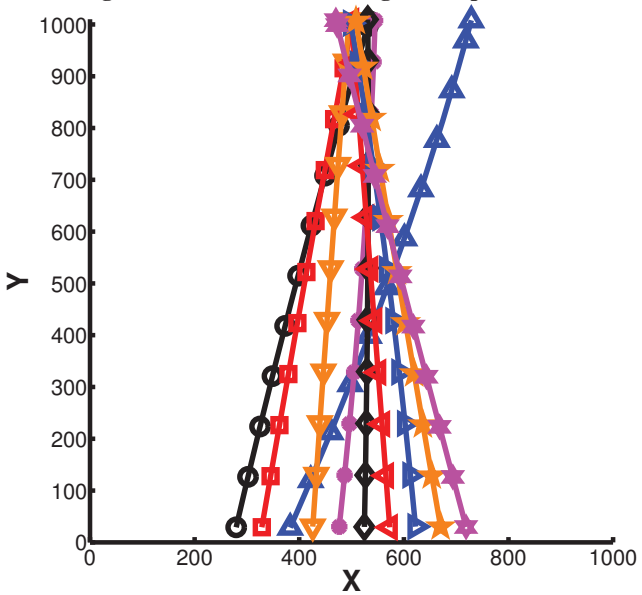


Figure 4: Case 3 — Moderate weight on separation.

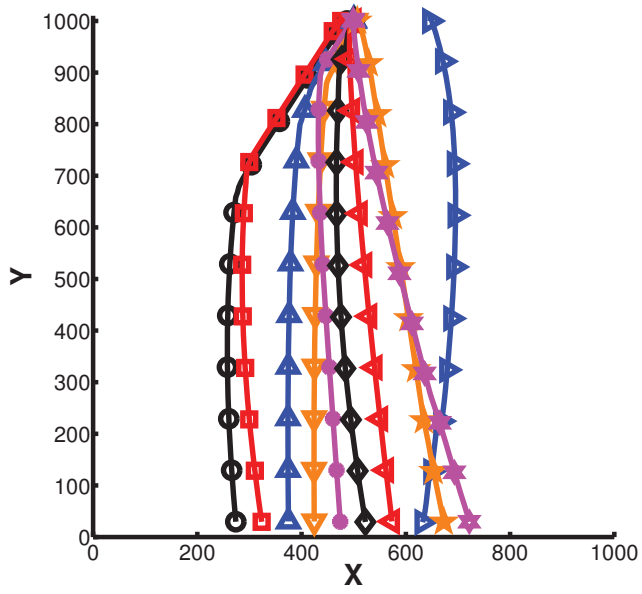


Figure 5: Case 3 — Large separation weight.

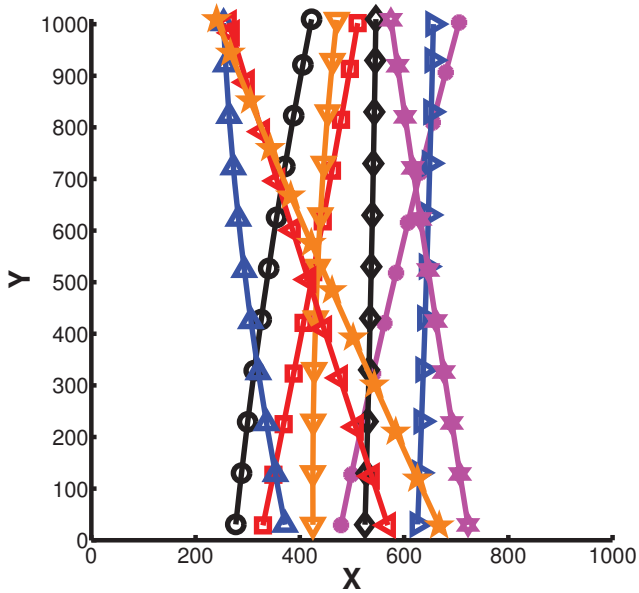
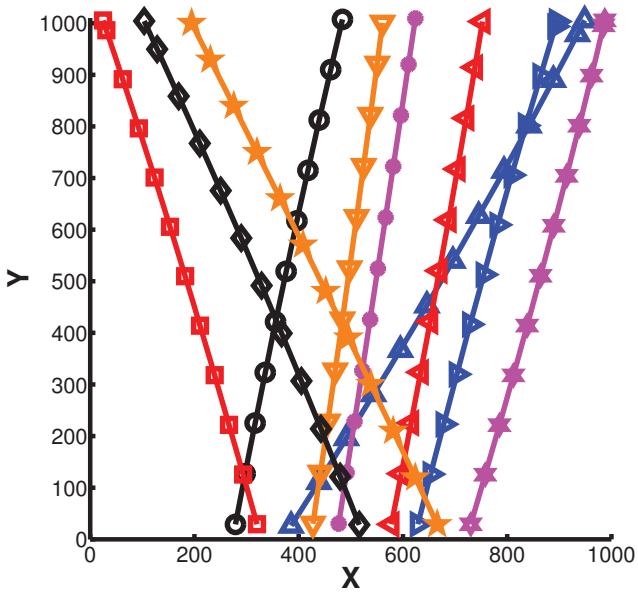


Figure 6: Case 3 — Large separation distance and weight.



$\beta_{plane} = 250$ units. With these settings, agents become solely concerned with staying in uncongested areas while traveling through the airspace. Figure 6 shows that a few of the simulated agents make large movements in the x-direction while traveling through the corridor, and the remaining agents spread out as they travel.

This strategy is a result that is not immediately apparent by inspection of the problem. Crossing behind other planes in flight is a somewhat unforeseen behavior that may not arise with hand-designed agents. The evolutionary algorithm, however, led to these policies through the agents simply seeking to maximize their utility because it was an easily accessible solution that offered improvement over more intuitive solutions.

6.4 Case 4 Results

In Case 4, we introduce the complication of clouds to avoid. This creates the agent goals for this case of (i) to make it through the corridor to the target point successfully, while (ii) avoiding clouds. Avoiding other aircraft is not involved in their utility function.

The simulation shows that the agents learning on these goals very successfully avoid the cloudy patch in the middle of the corridor, and pass near the center of the corridor end. Due to spacing being a non-concern for the agents in this case, they maintain very close distance to each other as they pass directly through the corridor.

6.5 Case 5 Results

Case 5 presents the agents with 3 goals to balance: to (i) make it through the corridor successfully, while (ii) maintaining separation assurance from other planes, and (iii) avoiding clouds.

This leads to an unexpected emergent behavior that may initially look quite disorganized, but has an underlying order. As the group of agents passes by the right side of the central cloud region, the agents peel off from the group one by one, making a complete turn before proceeding to the end of the corridor. As they turn at various points, they increase the spacing between themselves and the remainder of the formation, increasing their spacing utility, u_{sep}. In other simulations, we observed agent behaviors that were similar with a left-handed turn, as well as behaviors where the agents would split to two sub-formations performing this maneuver on either side of the cloud formation.

With even weight on all three goals, this unexpected policy of agents choosing to complete a full turn before proceeding to the end of the corridor creates a higher utility value U than a more conventional "laminar flow policy" like those seen in Figure 7. Executing the turns at different points in the corridor leads to an intermediate spreading of the planes, and also allow the agents to pass through the end of the corridor at different points in time, allowing more agents to pass closer to the center.

This type of emergent behavior manifested due to the evolutionary algorithm employed in this work, and outperformed our earlier attempts and solving the problem with rule-based agent policies. While we were subsequently able to develop hand-coded policies that were superior to the evolved policy, the concept for these policies was directly inspired by the results of the evolutionary algorithm. Furthermore, the evolutionary algorithm that developed these policies remains unchanged from the algorithm used in each of the other cases, while the superior hand-coded policies required both inspiration from the evolutionary algorithm and additional development time.

6.6 Case 6 Results

In Case 6, we remove the penalty of the cloudy regions, creating the goals of (i) make it through the corridor successfully, while (ii) maintaining separation assurance from other planes. We introduce a simulated complication due to the clouds: the agent's ability to sense other nearby planes is inhibited by the clouds, creating an implicit goal that they should avoid the cloudy areas in order to achieve their separation assurance goals. To make this case more challenging we change the allocation of clouds throughout the space so that the planes have a larger area to avoid.

The simulation reveals that the agents do learn to avoid the cloudy areas, though not quite as severely as when the goal was explicitly involved in their utility calculation. Passing through the cloudy area is not a bad decision in and of itself, but when multiple agents simultaneously choose to do so, they reduce each other's utility through the un-sensed separation violations.

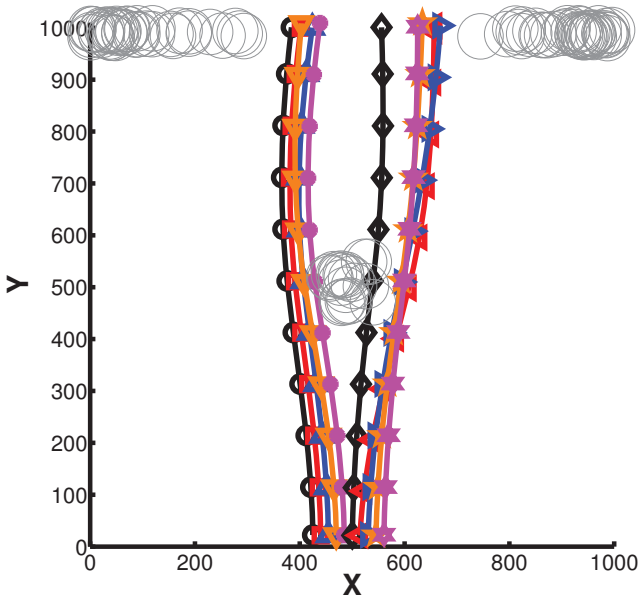


Figure 7: Case 4 — Explicit cloud avoidance.

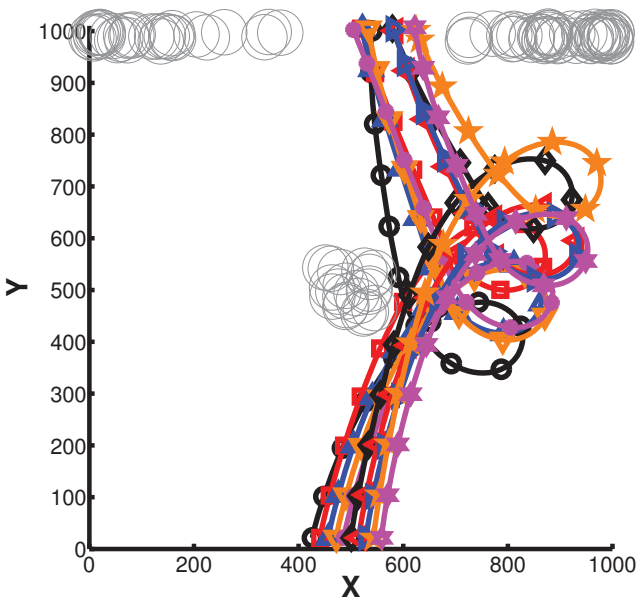


Figure 8: Case 5 — Explicit cloud avoidance and spacing.

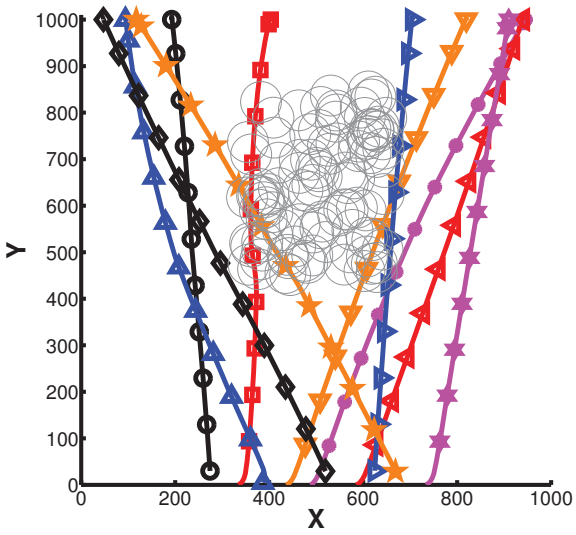


Figure 9: Case 6 — Implicit cloud avoidance.

7. CONCLUSION

In this work we have presented a method for the simulation of complex systems to predict the effects of new technology integration in air traffic management through the use of evolutionary algorithms. These evolutionary algorithms require the system designer only to specify a goal for the system, and the resultant behaviors achieve these goals in both expected and unforeseen manners.

We demonstrated the use of this technique in two domains, with the algorithm unchanged between the 6 cases tested. In the historical test cases, we arrive at the same conclusion that the historical system designers did: applying the Norden Bombsight to every bomber would be a poor idea, as many separation assurance problems would arise. Considering the separation between the planes as well as the precise location of the target is a better solution. This supports our claim that an agent-based evolutionary algorithm approach can assist in assessing the value of new technologies.

In the future technology case of the Civilian Cloud Domain, the simulation demonstrated that agents were able to balance multiple simultaneous goals for an expected outcome when we specified areas of the airspace to be avoided, in the form of clouds. When the agents also had to consider their separation from nearby planes, however, they developed a set of radical behaviors which gained higher utility than a more expected policy. This supports our claim that evolutionary algorithms may develop complex emergent behaviors, even with relatively simple agents.

Finally, the evolutionary algorithm successfully learned the implicit coupling between goals, even when one of the goals was not explicitly a part of the fitness function. In this way the agents learned to avoid the cloudy regions of the space because they negatively affected their sensing ability, even when we removed any direct consideration of the clouds from their utility function.

This type of approach — the use of agent-based evolutionary algorithms — offers the ability to analyze hypothetical cases in air traffic management before they are fully realized, and identify possibly dangerous interactions before risking human lives.

Acknowledgements.

This work was partially supported by the National Science Foundation under grant numbers CNS-0931591 and CNS-0930168.

8. REFERENCES

[1] C. A. C. Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, 191(11):1245–1287, 2002.

[2] O. Gaci and H. Mathieu. Simulation of a segregation strategy in a warehouse of dangerous goods by a multi- agent system. *IEEE Symposium on Computational Intelligence and Informatics*, 2011.

[3] M. Gallegati, G. Giulioni, and N. Kichiji. Complex dynamics and financial fragility in an agent based model. *Advances in Complex Systems*, 6(3):1–16, 2003.

[4] D. Delli Gatti, E. Gaffeo, and M. Gallegati. Complex agent-based macroeconomics: a research agenda for a new paradigm. *Journal of Economic Interaction and Coordination*, 5(2):111–135, 2010.

[5] E. Gelenbe and F-J. Wu. Large scale simulation for human evacuation and rescue. *Computers and Mathematics with Applications*, 64, 2012.

[6] M. Gladwell. The strange tale of the norden bombsight. TEDGlobal 2011, July 2011.

[7] M. Jakob, O. Vanek, O. Hrstka, and M. Pechoucek. Agents vs. pirates: Multi-agent simulation and optimization to fight maritime piracy. *AAMAS*, 2012.

[8] B. Lebaron. Heterogeneous gain learning and the dynamics of asset prices. *Journal of Economic Behavior and Organization*, 2012.

[9] J. Lin, J. Blythe, S. Clark, N. Davarpanah, N. Hughston, and M. Zyda. Unbelievable agents for large scale security simulation. *Association for the Advancement of Artificial Intelligence*, 2010.

[10] R.T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395, 2004.

[11] E. Mas, A. Suppasri, F. Imamura, and S. Koshimura. Agent-based simulation of the 2011 great east japan earthquake/tsunami evacuation: An integrated model of tsunami inundation and evacuation. *Journal of Natural Disaster Science*, 34(1), 2012.

[12] D. Romano, L. Lomax, and P. Richmond. NARCSim an agent-based illegal drug market simulation. *Games Innovations Conference*, pages 101–108, 2009.

[13] M. Sierhuis, W. J. Clancey, and R. J.J. van Hoof. Brahms: A multiagent modelling environment for simulating work processes and practices. *International Journal of Simulation and Process Modelling*, 3(3):134–152, 2007.

[14] M. Sierhuis, W. J. Clancey, R. J.J. van Hoof, C. H. Seah, M. S. Scott, R. A. Nado, S. F. Blumenberg, M. G. Shatto, B. L. Anderson, A. C. Bruins, C. B. Buckley, T. E. Diegelman, T. A. Hall, D. Hood, F. F. Reynolds, J. R. Toschlog, and T. Tucker. NASA's OCA mirroring system an application of multiagent systems in mission control. *AAMAS*, 2009.

[15] R. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[16] A. Walker and M. Wooldridge. Understanding the emergence of conventions in multiagent systems. *International Conference on Multiagent Systems*, 1995.

[17] M. Wooldridge. *An introduction to multiagent systems*. Wiley, 2002.