Evolving Distributed Agents for Managing Air Traffic

Adrian Agogino
UCSC, NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035, USA
adrian@email.arc.nasa.gov

Kagan Tumer
Oregon State University
204 Rogers Hall
Corvallis, OR 97331, USA
kagan.tumer@oregonstate.edu

ABSTRACT

Air traffic management offers an intriguing real world challenge to designing large scale distributed systems using evolutionary computation. The ability to evolve effective air traffic flow strategies depends not only on evolving good local strategies, but also on ensuring that those local strategies result in good global solutions. While traditional, direct evolutionary strategies can be highly effective in certain combinatorial domains, they are not well-suited to complex air traffic flow problems because of the large interdependencies among the local subsystems. In this paper, we propose an evolutionary agent-based solution to the air traffic flow problem. In this approach, we evolve agents both to learn the right local flow strategies to alleviate congestion in their immediate surroundings, and to prevent the creation of congestion "downstream" from their local areas. The agent-based approach leads to better and more fault-tolerant solutions. To validate this approach, we use FACET, an air traffic simulator developed at NASA and used extensively by the FAA and industry. On a scenario composed of three hundred aircraft and two points of congestion, our results show that an agent based evolutionary computation method, where each agent uses the system evaluation function, achieves 40% improvement over a direct evolutionary algorithm. In addition by creating agent-specific "difference evaluation functions" we achieve an additional 30% improvement over agents using the system evaluation.

Categories and Subject Descriptors

J.7 [**Computing Applications**]: Physical Sciences and Engineering—*Aerospace*

General Terms

Application, Algorithms, Performance

Keywords

Air Traffic Control, Multiagent Systems, Evolution

GECCO'07, July7–11, 2007, London, England, United Kingdom.

1. INTRODUCTION

The ability to effectively control air traffic with traditional control/optimization algorithms is decreasing rapidly as the air traffic levels and aircraft heterogeneity increases and restrictions on flight plans decrease. New strategies are needed to cope with this added complexity and provide robust safety levels while ensuring that air traffic delays do not reach unacceptable levels. Indeed, even at current air traffic levels, in 2005 alone there was an estimated 322,272 hours of delays within the United States airspace with a total cost estimated to exceed three billion dollars by industry [6]. With an expected increase in air traffic, unless the air traffic management processes are overhauled, these delays are expected to become significantly worse. The Next Generation Air Transportation Systems (NGATS) initiative is designed to address future issues in air traffic management without requiring major infrastructure changes (e.g., airports, runways, and towers) or adding large numbers of additional air traffic controllers. To accomplish this, new robust algorithms are needed that can safely manage complex air traffic flows while making optimal use of current infrastructure.

Evolutionary computation has been shown to be a powerful tool in solving complex problems such as pole balancing, rocket control, assembly line balancing and robotics [2, 7, 12, 8, 16]. Due to its success in complex domains, the use of evolutionary algorithms is a promising approach in the air traffic flow problem. However, using a single, centralized evolutionary algorithm to create a control policy for the entire airspace is problematic for three reasons:

1. A centralized algorithm provides a single point of failure and is not robust against communication failures that are inevitable in a system spread over large geographic areas;

2. A single evolutionary controller will struggle to discover effective solutions in a state space large enough to encompass the US national air space; and

3. A single evolutionary controller is both slow to optimize and slow to modify to adapt to changing conditions.

Instead of using a centralized evolutionary algorithm, we propose an adaptive approach based on local agents that is a better fit to this naturally distributed problem. In this architecture (shown in Figure 1), each agent is responsible for a local area and maintains its own population of partial solutions. At the beginning of each trial (consisting of mul-

tiple time steps), each agent selects a member from its population as its "representative". All representatives are then put together to form a full solution that is used to manage traffic during the trial. At the end of the trial, the full solution is evaluated and each agent receives an evaluation that rates the performance of the selected representative. The agents then update their populations (e.g., by either retaining that representative or not). To apply this multi-agent approach to the air traffic flow problem, the airspace is broken down into agents by assigning each agent to a "fix," a specific location in 2D. Because aircraft flight plans consist of a sequence of fixes, this representation allows localized fixes (represented by agents) to have direct impact on the flow of air traffic[1]. In this approach, the agents' actions are to set the separation distances that approaching aircraft are required to keep. This simple agent-action pair allows the agents to slow down or speed up local traffic and allows agents to a have significant influence on system performance. Agents maintain a population of separation distances and use an evolutionary algorithm to evolve the population to determine the most appropriate separation action for their location.

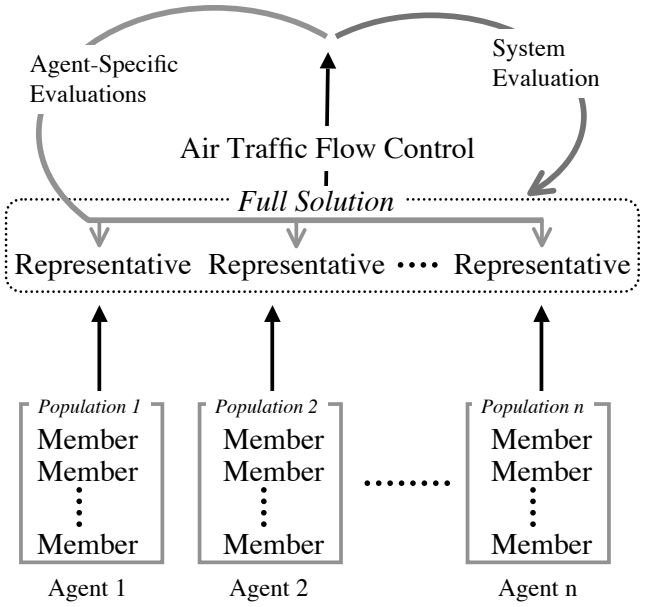

Figure 1: Schematic of agent-specific evaluations. Each agent has its own population of partial solutions. For each trials those partial solutions are put together to form a full solution to the air traffic problem. The solution is then used for that trial. The full system performance is evaluated and agents then receive an agent-specific evaluation which evaluates the partial solutions chosen by each agent.

The advantage of deploying a system where each agent evolves its own population is in having an extra degree of freedom: we can now have a separate evaluation function for each agent. Of course, this extra freedom comes at a potential cost. If the agent evaluation functions are not selected carefully, there is no reason to assume that each agent optimizing its own evaluation function will also optimize the system evaluation function. One of the most natural evaluation functions for an agent to use is the system evaluation, where each agent uses the identical evaluation function used to rate the performance of the entire system. Though this approach provides acceptable solutions, it is slow to converge can scale poorly [18]. We show how improved performance can be obtained when each agent uses its own agent-specific evaluation function based on collectives [20]. In this paper we focus on how an agent-specific evaluation dubbed the "difference evaluation" can be used in this domain [1].

Previous work in this domain fell into one of two distinct categories: (i) The first principles based modeling approaches used by domain experts [4, 9, 11, 15] ; and (ii) algorithmic approaches explored by the learning, evolution and/or agents community [19, 5, 10, 14]. In [4] "geometric optimization" was proposed, where the geometry of a particular air space pattern was utilized to create policies that reduced conflict. In [9] dynamic programming was used over a model of air traffic routes. In [10] a multi-agent approach to the "free flight" problem was proposed, where agents used utilities that balanced their local needs with the system goals. Closely related to the domain of this paper, in [19] a reinforcement learning approach was introduced to relieve air traffic congestion in a distributed way. The work presented in this paper fits in the second category, but in order to show the real world applicability of our approach, we use FACET, a simulator introduced and widely used (i.e., over 40 organizations and 5000 users) by work in the first category [3, 13].

In this paper we present a new way to evolve a distributed air traffic flow management algorithm that can be readily implemented and test that algorithm using the FACET air traffic simulator. In Section 2, we describe the air traffic flow problem and the simulation tool, FACET. In Section 4, we present the evaluation function characteristic and the agent-based approach, focusing on both the selection of the agents and the selection of their evaluation functions. In Section 5 we present results from a domain with three hundred aircraft and two points of congestion, explore different tradeoffs of the system objective function, and discuss the scaling properties of the different agent evaluations. Finally, in Section 6, we discuss the implications of these results and how the described work can enable the FAA to reach its stated goal of increasing the traffic volume by threefold without significantly modifying the existing infrastructure.

2. AIR TRAFFIC FLOW MANAGEMENT

With over 40,000 flights operating within the United States airspace on an average day, the management of traffic flow is a complex and demanding problem. Not only are there concerns for the efficiency of the system, but also for fairness (e.g., different airlines), adaptability (e.g., developing weather patterns), reliability and safety (e.g., airport management). In order to address such issues, the management of this traffic flow occurs over four hierarchical levels: 1) Separation assurance (2-30 minute decisions); 2) Regional flow (20 minutes to 2 hours); 3) National flow (1-8 hours); and 4) Dynamic airspace configuration (6 hours to 1 year). Because of the strict guidelines and safety concerns surrounding aircraft separation, we will not address that control level in this paper. Similarly, because of the business and political impact of dynamic airspace configuration, we will not ad-

[1]We discuss how flight plans with few fixes can be handled in more detail in Section 2.

dress the outermost flow control level either. Instead, we will focus on the regional and national flow management problems, restricting our impact to decisions with time horizons between twenty minutes and eight hours. The proposed algorithm will fit between long term planning by the FAA and the very short term decisions by air traffic controllers.

The continental US airspace consists of 20 regional centers (handling 200-300 flights on a given day) and 830 sectors (handling 10-40 flights). The flow control problem has to address the integration of policies across these sectors and centers, account for the complexity of the system (e.g., over 5200 public use airports and 16,000 air traffic controllers) and handle changes to the policies caused by weather patterns. Two of the fundamental problems in addressing the flow problem are: (i) modeling and simulating such a large complex system as the fidelity required to provide reliable results is difficult to achieve; and (ii) establishing the method by which the flow management is evaluated, as directly minimizing the total delay may lead to inequities towards particular regions or commercial entities. Below, we discuss how we addressed both issues, namely, we describe FACET, a widely used simulation tool, and discuss our system evaluation function.

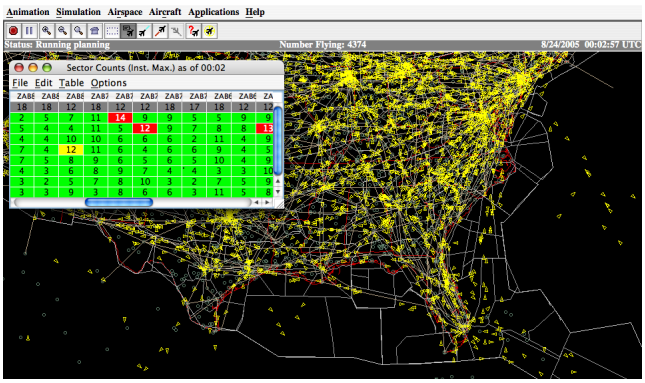

Figure 2: FACET screen-shot displaying traffic routes and air flow statistics.

2.1 FACET

FACET (Future ATM Concepts Evaluation Tool), a physics based model of the US airspace was developed to accurately model the complex air traffic flow problem [3]. It is based on propagating the trajectories of proposed flights forward in time. FACET can be used to either simulate and display air traffic (a 24 hour slice with 60,000 flights takes 15 minutes to simulate on a 3 GHz, 1 GB RAM computer) or provide rapid statistics on recorded data (4D trajectories for 10,000 flights including sectors, airports, and fix statistics in 10 seconds on the same computer) [13]. FACET is extensively used by the FAA, NASA and industry (over 40 organizations and 5000 users) [13].

FACET simulates air traffic based on flight plans and through a graphical user interface allows the user to analyze congestion patterns of different sectors and centers (Figure 2). FACET also allows the user to change the flow patterns of the aircraft through a number of mechanisms, including metering aircraft through fixes. The user can then observe the effects of these changes to congestion. In this paper, agents use FACET directly through "batch mode", where agents send scripts to FACET asking it to simulate air traffic based on metering orders imposed by the agents. The agents then produce their evaluations based on feedback received from FACET about the impact of these meterings.

2.2 System Evaluation

The system performance evaluation function we use in this paper focuses on delay and congestion but does not account for fairness impact on different commercial entities. Instead it focuses on the amount of congestion in a particular sector and on the amount of measured air traffic delay. The linear combination of these two terms gives the full system evaluation function, $G(z)$ as a function of the full system state z. More precisely, we have:

$$G(z) = -((1-\alpha)B(z) + \alpha C(z)) \ , \tag{1}$$

where $B(z)$ is the total delay penalty for all aircraft in the system, and $C(z)$ is the total congestion penalty. The relative importance of these two penalties is determined by the value of α, and we explore various trade-offs based on α in Section 5.

The total delay, B, is a sum of delays over a set of sectors S and is given by:

$$B(z) = \sum_{s \in S} B_s(z) \tag{2}$$

where

$$B_s(z) = \sum_t \Theta(t - \tau_s) k_{t,s}(t - \tau_s) \ , \tag{3}$$

where $k_{s,t}$ is the number of aircraft in sector s at time t, τ_s is a predetermined time, and $\Theta(\cdot)$ is the step function that equals 1 when its argument is greater or equal to zero, and has a value of zero otherwise. Intuitively, $B_s(z)$ provides the total number of aircraft that remain in a sector s past a predetermined time τ_s, and scales their contribution to count by the amount by which they are late. In this manner $B_s(z)$ provides a delay factor that not only accounts for all aircraft that are late, but also provides a scale to measure their "lateness". This definition is based on the assumption that most aircraft should have reached the sector by time τ_s and that aircraft arriving after this time are late. In this paper the value of τ_s is determined by assessing aircraft counts in the sector in the absence of any intervention or any deviation from predicted paths.

Similarly, the total congestion penalty is a sum over the congestion penalties over the sectors of observation, S:

$$C(z) = \sum_{s \in S} C_s(z) \tag{4}$$

where

$$C_s(z) = a \sum_t \Theta(k_{s,t} - c_s) e^{b(k_{s,t} - c_s)} \ , \tag{5}$$

where a and b are normalizing constants, and c_s is the capacity of sector s as defined by the FAA. Intuitively, $C_s(z)$ penalizes a system state where the number of aircraft in a sector exceeds the FAA's official sector capacity. Each sector capacity is computed using various metrics which include the number of air traffic controllers available. The exponential penalty is intended to provide strong feedback to return the number of aircraft in a sector to below the FAA mandated capacities.

3. EVALUATION FUNCTION PROPERTIES

While the goal of the system is to maximize the system evaluation function, the individual agents are designed to maximize their own agent-specific evaluation functions. Our goal is to create agent-specific evaluation functions so that agents that evolve to discover solutions that are deemed "good" by their own evaluation functions, also provide solutions deemed "good" by the system evaluation function. To help with this task we first illustrate some important properties of agent-specific evaluation functions based on work described in [20] and in the context of previous multi-agent control work described in [1] and [18].

3.1 Factoredness and Agent Sensitivity

Let the **system evaluation function** be given by $G(z)$, where z is the state of the full system (e.g., the actions of the agents along with the environmental state). Let the **agent evaluation function** for agent i be given by $g_i(z)$. First we want the agent-specific evaluation functions of each agent to have high *factoredness* with respect to G, intuitively meaning that an action taken by an agent that improves its agent-specific evaluation function also improves the system evaluation function (i.e. G and g_i are aligned). Formally, the degree of factoredness between G and g_i is given by:

$$\mathcal{F}_{g_i} = \frac{\int_z \int_{z'} u[(g_i(z) - g_i(z'))\ (G(z) - G(z'))]dz'dz}{\int_z \int_{z'} dz'dz} \qquad (6)$$

where z' is a state which only differs from z in the state of agent i, and $u[x]$ is the unit step function, equal to 1 when $x > 0$. Intuitively, a high degree of factoredness between g_i and G means that an agent evolved to maximize g_i will also maximize G.

Second, the agent evaluation function must be more sensitive to changes in that agent's fitness than to changes in the fitness of all the other agents. Formally we quantify the *agent-sensitivity* of evaluation function g_i, at z as:

$$\lambda_{i,g_i}(z) = E_{z'}\left[\frac{\|g_i(z) - g_i(z - z_i + z'_i)\|}{\|g_i(z) - g_i(z' - z'_i + z_i)\|}\right], \qquad (7)$$

where $E_{z'}[\cdot]$ provides the expected value over possible values of z', and $(z - z_i + z'_i)$ notation specifies the state vector where the components of agent i have been removed from state z and replaced by the components of agent i from state z'. So at a given state z, the higher the agent-sensitivity, the more $g_i(z)$ depends on changes to the state of agent i, i.e., the better the associated signal-to-noise ratio for i. Intuitively then, higher agent-sensitivity means there is "cleaner" (e.g., less noisy) selective pressure on agent i. Ideally we want evaluation functions that are both factored and highly agent-sensitive (Figure 3).

As an example, consider the case where the agent evaluation function of each agent is set to the system evaluation function, meaning that each agent is evaluated based on the fitness of the full system. Such a system will be factored by the definition of Equation 6. However, the agent fitness functions will have low agent-sensitivity, because the fitness of each agent will depend on the fitness of all other agents, leading to a small numerator and large denominator in Equation 7.

3.2 Difference Evaluations

Instead of using the system evaluation for agent evolution, we can use agent-specific evaluation functions that are more

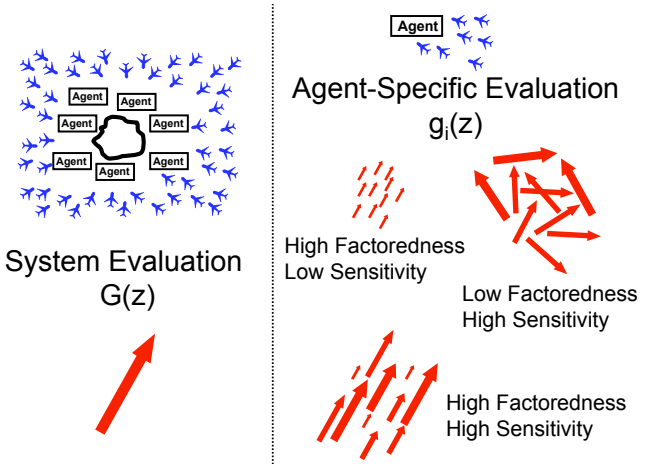

Figure 3: Properties of Agent-Specific Evaluations. Direction of an arrow represents the goal of evaluation function. Size of an arrow represents sensitivity of the evaluation to agent's action. As a system designer we are concerned with maximizing the system evaluation function (left). For agents to be able to effectively maximize system evaluation, their evaluations should be aligned with the system evaluation (high factoredness) and supply a strong signal (high sensitivity).

sensitive to the actions of the agent. Consider **difference** evaluation functions, which are of the form [18, 20]:

$$D_i \equiv G(z) - G(z - z_i + c_i)\ , \qquad (8)$$

where z_i is the action of agent i. All the components of z that are affected by agent i are replaced with the fixed constant c_i [2].

In many situations it is possible to use a c_i that is equivalent to taking agent i out of the system. Intuitively this causes the second term of the difference evaluation to evaluate the performance of the system without i and therefore D evaluates the agent's contribution to the system performance. There are two advantages to using D: First, because the second term removes a significant portion of the impact of other agents in the system, it provides an agent with a "cleaner" signal than G. This makes the evaluation function more agent-sensitive. Second, because the second term does not depend on the actions of agent i, any action by agent i that improves D, also improves G. The difference evaluation is therefore fully factored.

4. AGENT BASED AIR TRAFFIC FLOW

The multi-agent approach to air traffic flow management we present is predicated on agents evolving independent solutions that maximize the system evaluation function discussed above. To that end, there are three critical decisions that need to be made:

- Agent selection;

[2]This notation uses zero padding and vector addition rather than concatenation to form full state vectors from partial state vectors. The vector "z_i" in our notation would be $z_i e_i$ in standard vector notation, where e_i is a vector with a value of 1 in the ith component and is zero everywhere else.

- Agent action set selection; and
- Agent evolution algorithm selection.

In this section, we provide a detailed description for each of these decisions.

4.1 Agent Selection

Selecting the aircraft as agents is perhaps the most obvious choice for defining an agent. That selection has the advantage that agent actions can be intuitive (e.g., change of flight plan, increase or decrease speed and altitude) and offer a high level of granularity, in that each agent can have its own policy. However, there are several problems with that approach. First, there are in excess of 40,000 aircraft in a given day, leading to a massively large multi-agent system. Second, as each agent would take relatively few actions, it would not be able to sample its environment sufficiently, resulting in a prohibitively slow evolutionary process. As an alternative, we assign agents to individual ground locations throughout the airspace called "fixes." Each agent is then responsible for any aircraft going through its fix. Fixes offer many advantages as agents:

1. Their number can vary depending on need. The system can have as many agents as required for a given situation(e.g., agents coming "live" around an area with developing weather conditions).
2. Because fixes are stationary, collecting data and matching behavior to evaluation is easier.
3. Because aircraft flight plans consist of fixes, agents will have the ability to affect traffic flow patterns.
4. Because fixes are part of the current flight management procedures, algorithms that modify them directly can be readily deployed (e.g., as tools to help air traffic controllers rather than compete with or replace them).

Figure 4 shows a schematic of this agent based system. Agents surrounding a congestion or weather condition affect the flow of traffic to reduce the burden on particular regions.

4.2 Agent Actions

The second issue that needs to be addressed, is determining the action set of the agents. Again, an obvious choice may be for fixes to "bid" on aircraft approaching their location, affecting their flight plans. Though appealing from a free flight perspective, that approach makes the flight plans too unreliable and significantly complicates the scheduling problem (e.g., arrival at airports and the subsequent gate assignment process).

Instead, we set the actions of an agent to determine the separation distance (distance between aircraft) that aircraft have to maintain, when going through the agent's fix. This is known as setting the "Miles in Trail" or MIT. When an agent sets the MIT value to d, aircraft going towards its fix are instructed to line up and keep d miles of separation (though aircraft will always keep a safe distance from each other regardless of the value of d). When there are many aircraft going through a fix, the effect of issuing higher MIT values is to slow down the rate of aircraft that go through the fix. By increasing the value of d, an agent can limit the amount of air traffic downstream of its fix, reducing congestion at the expense of increasing the delays upstream.

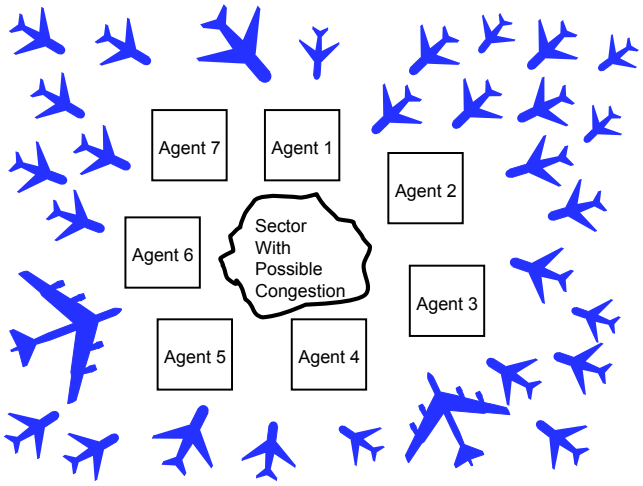

Figure 4: Schematic of agent architecture. The agents corresponding to fixes surrounding a possible congestion become "live" and start setting new separation times.

4.3 Agent Evolution

The objective of each agent is to evolve the best values of the miles in trail distance d that will lead to the best system evaluation, G. In this paper we assume that each agent evolves using its own evolutionary algorithm with its own population of values of d. Since each member of the population is comprised of a single number, a sophisticated genetic algorithm is not needed. Instead each agent uses a simple evolutionary algorithm to evolve its population using its own fitness evaluation function. The evolutionary algorithm is based on a population of fixed sized. At the beginning of each trial the agent chooses the best member of the population with probability $(1 - \epsilon)$ and a random member of the population with probability ϵ. At the end of the trial it replaces the worst member of the population and a copy of the best member and mutates the the copy with probability ϵ.

5. EXPERIMENTS

In this paper we test the performance of our agent based air traffic optimization method on a series of simulations using the FACET air traffic simulator. In all experiments we test the performance of three different evolutionary methods. The first method is an evolutionary algorithm with one agent. With this method there is one population for the entire system, and each member of the population contains a full solution, specifying the mile in trail distances for all the fixes. The other two methods are agent based methods where the agents are maximizing one of the following evaluations:

1. The system evaluation, $G(z)$, as define in Equation 1.
2. The difference evaluation $D_i(z)$, assuming that agents can calculate counterfactuals.

In the agent based methods, each agent has its own population of miles in trail values and an agent's action is to choose one of these values at the beginning of a trial. The miles in

trail values for all the agents constitute a policy for the air space.

These methods are tested on an air traffic domain with 300 aircraft. The aircraft are going through two points of congestion over a four hour simulation, with 200 going over one point of congestion and 100 going through the other point of congestion. The second congestion is less severe than the first one, so agents have to form different policies depending which point of congestion they are influencing. The points of congestion are created by setting up a series of flight plans that cause the number of aircraft in the sectors of interest to be significantly more than the number allowed by the FAA. Agents are responsible for reducing congestion while trying to minimize delay.

In all experiments the goal of the system is to maximize the system performance given by $G(z)$ with the parameters, $a = 50$, $b = 0.3$, $c_1 = 18$, $c_2 = 15$, τ_{s_1} equal to 200 minutes and τ_{s_1} equal to 175 minutes. These values of τ are obtained by examining the time at which most of the aircraft leave the sectors, when no congestion control is being performed. Except where noted, the trade-off between congestion and lateness, α is set to 0.5. Expanding equation 1 and plugging in the constants we get the following equation for the system evaluation used in the experiments:

$$\begin{aligned} G(z) = & - \frac{1}{2}\sum_t \Theta(t-200)k_{t,1}(t-200) \\ & - \frac{1}{2}\sum_t \Theta(t-175)k_{t,2}(t-175) \\ & - 25\sum_t \Theta(k_{t,1}-18)e^{0.3(k_{t,1}-18)} \\ & - 25\sum_t \Theta(k_{t,2}-15)e^{0.3(k_{t,2}-15)} . \end{aligned} \quad (9)$$

In the experiments the evolution parameter ϵ is set to 0.25. In all experiments the best policies chosen by the agents are used in the results. All results are an average of fifty independent experiments with the differences in the mean $(\sigma/\sqrt{n})$ shown as error bars, though in most cases the error bars are too small to see. All conclusions are statistically significant with $p < 0.01$.

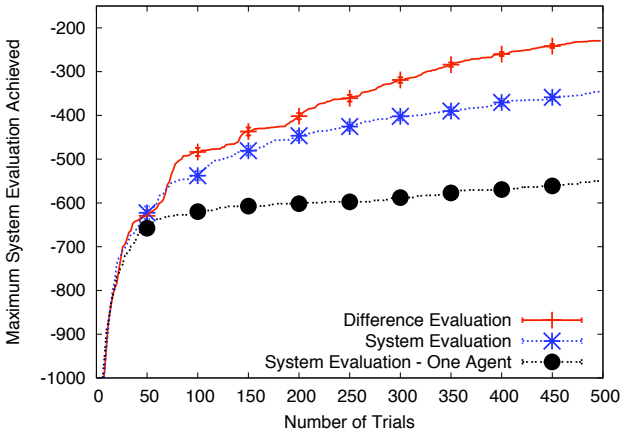

Figure 5: Performance on two congestion problem, with 300 Aircraft, 20 Fixes and $\alpha = .5$.

5.1 Results

The results displayed in Figures 5 show the performance of the four algorithms when there are twenty fixes (and therefore 20 agents, except for the one agent solution). Clearly the centralized solution using a single agent with a single evolutionary algorithm performs the worst. This is not surprising since it is difficult for the single evolutionary algorithm to evolve a solution with twenty coupled variables. In contrast both the agent based methods performed significantly better than the straight (single agent) evolutionary algorithm.

Among the agent based methods, agents using difference evaluations perform better than agents using the system evaluation. Again this is not surprising, since with twenty agents, an agent directly trying to maximize the system evaluation has difficulty determining the effect of its actions on its own evaluation. Even if an agent takes an action that reduces congestion and lateness, other agents at the same time may take actions that increase congestion and lateness, causing the agent to wrongly believe that its action was poor. In contrast agents using the difference evaluation have more influence over the value of their own evaluation, therefore when an agent takes a good action, the value of this action is more likely to be reflected in its evaluation.

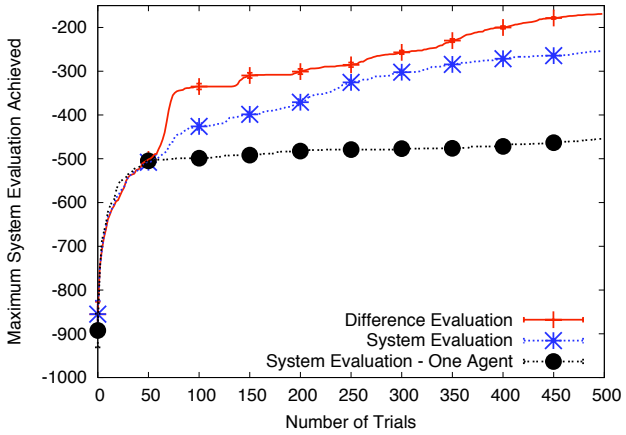

Figure 6: Performance on two congestion problem, with 300 Aircraft, 50 Fixes and $\alpha = .5$.

To verify that the performance improvement of our methods is maintained when there is a different number of agents, we perform additional experiments where we increased the number of agents used by the agent based solutions. We did this by creating a domain with 50 fixes instead of 20, thereby increasing the number of agents to 50 in the agent based solutions. The results displayed in Figure 6 show that indeed the relative performances of the methods are comparable when the number of agents is increased to 50. Figure 7 shows scaling results and demonstrates that the conclusions hold over a wide range of number of agents.

5.2 Varying Agents per Fix

In the previous experiments using the system utility, either one agent is assigned per fix or a single agent is assigned to all of the fixes. To test the performance of the multi-agent system between these extremes, we conduct an experiment where the number fixes assigned to a particular agent varies.

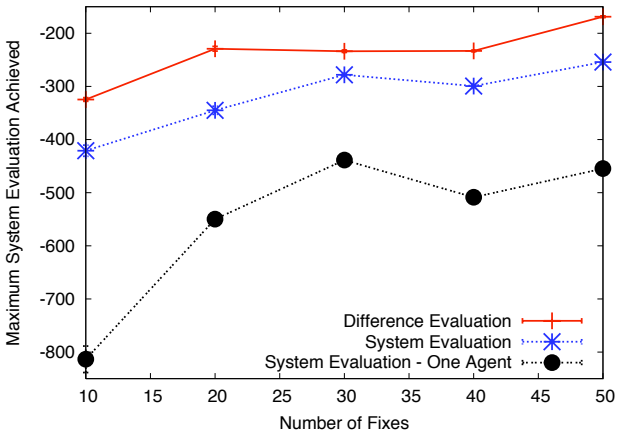

Figure 7: Impact of number of agents on system performance. Two congestion problem, with 300 Aircraft and $\alpha = .5$. Except for "One Agent" solution, there is one agent per fix.

The results for a twenty fix problem are shown in Figure 8. It is clear from the figure that there is nothing special about the two extremes. In this domain the results show that it is generally better to have more agents.

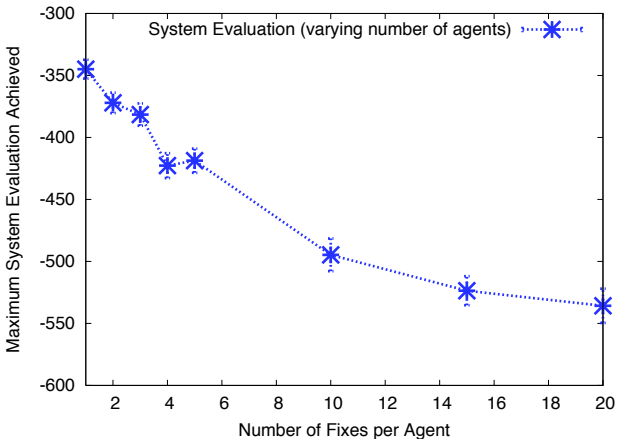

Figure 8: Impact of number of fixes assigned to each agent on system performance. Two congestion problem, with 300 Aircraft, $\alpha = .5$. and 20 fixes.

5.3 Penalty Tradeoffs

Recall that the system evaluation function used in the experiments is $G(z) = -((1-\alpha)D(z) + \alpha C(z))$, which comprises of penalties for both congestion and lateness. This evaluation function forces the agents to tradeoff these relative penalties depending on the value of α. With high α the optimization focuses on reducing congestion, while with low α the system focuses on reducing lateness.

Next, we perform a series of experiments where α ranges from 0.0 to 1.0 . Figure 9 shows the results which lead to three interesting observations:

- First, there is a zero congestion penalty solution. This solution has agents enforce large MIT values to block all air traffic, which appears viable when the system evaluation does not account for delays. All algorithms find this solution, though it is of little interest in practice due to the large delays it would cause.

- Second, if the two penalties were independent, an optimal solution would be a line from the two end points. Therefore, unless D is far from being optimal, the two penalties are not independent. Note that for $\alpha = 0.5$ the difference between D and this hypothetical line is as large as it is anywhere else (statistically equivalent to $\alpha = 0.75$), making $\alpha = 0.5$ a good choice for testing the algorithms in a difficult setting.

- Evolution using the system evaluation is particularly poor at handling multiple objectives. For both the single-agent and multi-agent solution, the performance degrades significantly for mid-ranges of α.

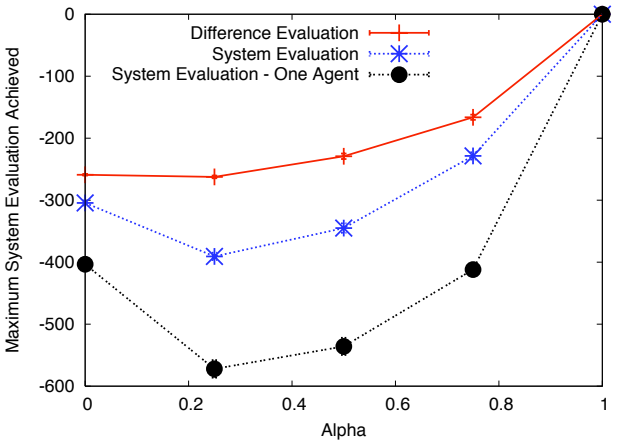

Figure 9: Tradeoff between objectives on two congestion problem, with 300 Aircraft and 20 Fixes. Note that evolution with the system evaluation is particularly bad at handling multiple objectives.

6. DISCUSSION

In this paper we show how evolutionary computation can be effectively used to solve a difficult air traffic flow problem. The key to our success in this domain is in breaking up the full problem into local problems and addressing each local problem with an agent that evolves its own population using its own evaluation function. The distributed agent approach where each agent uses the system evaluation achieves a 40% increase in performance over a single evolutionary algorithm. Furthermore, we show that an additional 30% increase in performance is obtained by having the agents evolve using the "difference evaluation" function. Agents using the difference evaluation are more successful, because the value of their evaluation is more sensitive to their own actions. At the same time the difference evaluation is still fully "factored", so that while each agent evolves a solution that maximizes its own difference evaluation, as a whole the system is producing a solution that maximizes the system evaluation.

While the difference evaluation is highly useful, we have not directly addressed how to compute it, or the computation costs associated with computing it. In many cases it is computable using knowledge of the functional form of the system evaluation, without the need to recompute the entire system evaluation. For example, in the FACET air traffic simulator, there are mechanisms to update a change to a flight plan once a simulation has been run, that produce a result faster than running the entire simulation from scratch. In other cases a close approximation to the difference evaluation can be computed with very little computational cost over the initial evaluation of the system utility. In fact, it has previously been shown that after the system evaluation is computed, the difference evaluation can be quickly estimated using no additional runs of the FACET simulator, with only a small loss of system performance [19].

This paper has shown how a multi-agent system can evolve an effective solution to a specific air traffic flow configuration involving the metering of aircraft through fix locations. While this specific application can be used in many situations to relieve congestion, the multi-agent paradigm is highly flexible and can be used in numerous other air traffic flow configurations. For instance we are currently investigating how agents can evolve policies that regulate flow through sectors instead of fixes, which provides a more direct, one-to-one interface between sector controllers and agents. In addition agents can evolve in heterogeneous environments where different agents may have different controls. In such environments, air traffic operators could choose to "turn on" the agents they are the most comfortable with for local conditions within the airspace. The long term objective of this work is to both identify the best agent insertion points within the next generation air traffic systems and to use good evolutionary computation methods to provide solutions/recommendation to improve air traffic flow.

Acknowledgments: The authors thank Banavar Sridhar for his invaluable help in describing both current air traffic flow management and NGATS, and Shon Grabbe for his detailed tutorials on FACET.

7. REFERENCES

[1] A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In *The Genetic and Evolutionary Computation Conference*, pages 1–12, Seatle, WA, June 2004.

[2] E. J. Anderson and M. C. Ferris. Genetic algorithms for combinatorial optimization: The assembly line balancing problem. *OSRA Journal on Computing*, 6:161–173, 1994.

[3] K. D. Bilimoria, B. Sridhar, G. B. Chatterji, K. S. Shethand, and S. R. Grabbe. Facet: Future atm concepts evaluation tool. *Air Traffic Control Quarterly*, 9(1), 2001.

[4] Karl D. Bilimoria. A geometric optimization approach to aircraft conflict resolution. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Denver, CO, 2000.

[5] Martin S. Eby and Wallace E. Kelly III. Free flight separation assurance using distributed algorithms. In *Proceedings of Aerospace Conference, 1999*, Aspen, CO, 1999.

[6] FAA OPSNET data Jan-Dec 2005. US Department of Transportation website.

[7] D. Floreano and F. Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. In *Proc. of Conf. on Simulation of Adaptive Behavior*, 1994.

[8] F. Gomez and R. Miikkulainen. Active guidance for a finless rocket through neuroevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Chicago, Illinois, 2003.

[9] S. Grabbe and B. Sridhar. Central east pacific flight routing. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Keystone, CO, 2006.

[10] Jared C. Hill, F. Ryan Johnson, James K. Archibald, Richard L. Frost, and Wynn C. Stirling. A cooperative multi-agent approach to free flight. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1083–1090, New York, NY, USA, 2005. ACM Press.

[11] P. K. Menon, G. D. Sweriduk, and B. Sridhar. Optimal strategies for free flight air traffic conflict resolution. *Journal of Guidance, Control, and Dynamics*, 22(2):202–211, 1999.

[12] Daniel Merkle, Martin Middendorf, and Hartmut Schmeck. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 6(4):333–346, 2002.

[13] 2006 NASA Software of the Year Award Nomination. FACET: Future ATM concepts evaluation tool. Case no. ARC-14653-1, 2006.

[14] M. Pechoucek, D. Sislak, D. Pavlicek, and M. Uller. Autonomous agents for air-traffic deconfliction. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Hakodate, Japan, May 2006.

[15] B. Sridhar, G. Chatterji, and S. Grabbe. Benefits of direct-to in national airspace system. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Denver, CO, 2000.

[16] K. Stanley and R. Miikkulainen. Efficient reinforcement learning through evolving neural network topologies. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, San Francisco, CA, 2002.

[17] C. Tomlin, G. Pappas, and S. Sastry. Conflict resolution for air traffic management. *IEEE Transaction on Automatic Control*, 43(4):509–521, 1998.

[18] K. Tumer and A. Agogino. Coordinating multi-rover systems: Evaluation functions for dynamic and noisy environments. In *The Genetic and Evolutionary Computation Conference*, Washington, DC, June 2005.

[19] K. Tumer and A. Agogino. Distributed agent-based air traffic flow management. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Honolulu, Hawaii, May 2007. to appear.

[20] K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.