Distributed Agent-Based Air Traffic Flow Management

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

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

ABSTRACT

Air traffic flow management is one of the fundamental challenges facing the Federal Aviation Administration (FAA) today. The FAA estimates that in 2005 alone, there were over 322,000 hours of delays at a cost to the industry in excess of three billion dollars. Finding reliable and adaptive solutions to the flow management problem is of paramount importance if the Next Generation Air Transportation Systems are to achieve the stated goal of accommodating three times the current traffic volume. This problem is particularly complex as it requires the integration and/or coordination of many factors including: new data (e.g., changing weather info), potentially conflicting priorities (e.g., different airlines), limited resources (e.g., air traffic controllers) and very heavy traffic volume (e.g., over 40,000 flights over the US airspace).

In this paper we use FACET – an air traffic flow simulator developed at NASA and used extensively by the FAA and industry – to test a multi-agent algorithm for traffic flow management. An agent is associated with a fix (a specific location in 2D space) and its action consists of setting the separation required among the airplanes going though that fix. Agents use reinforcement learning to set this separation and their actions speed up or slow down traffic to manage congestion. Our FACET based results show that agents receiving personalized rewards reduce congestion by up to 45% over agents receiving a global reward and by up to 67% over a current industry approach (Monte Carlo estimation).

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Multiagent systems*

General Terms

Application, Algorithms, Performance

AAMAS'07 May 14–18 2006, Honolulu, Hawaii, USA.

Keywords

Air Traffic Control, Multiagent Systems, Reinforcement Learning, Optimization

1. INTRODUCTION

The efficient, safe and reliable management of our ever increasing air traffic is one of the fundamental challenges facing the aerospace industry today. On a typical day, more than 40,000 commercial flights operate within the US airspace [14]. In order to efficiently and safely route this air traffic, current traffic flow control relies on a centralized, hierarchical routing strategy that performs flow projections ranging from one to six hours. As a consequence, the system is slow to respond to developing weather or airport conditions leading potentially minor local delays to cascade into large regional congestions. In 2005, weather, routing decisions and airport conditions caused 437,667 delays, accounting for 322,272 hours of delays. The total cost of these delays was estimated to exceed three billion dollars by industry [7].

Furthermore, as the traffic flow increases, the current procedures increase the load on the system, the airports, and the air traffic controllers (more aircraft per region) without providing any of them with means to shape the traffic patterns beyond minor reroutes. The Next Generation Air Transportation Systems (NGATS) initiative aims to address this issues and, not only account for a threefold increase in traffic, but also for the increasing heterogeneity of aircraft and decreasing restrictions on flight paths. Unlike many other flow problems where the increasing traffic is to some extent absorbed by improved hardware (e.g., more servers with larger memories and faster CPUs for internet routing) the air traffic domain needs to find mainly algorithmic solutions, as the infrastructure (e.g., number of the airports) will not change significantly to impact the flow problem. There is therefore a strong need to explore new, distributed and adaptive solutions to the air flow control problem.

An adaptive, multi-agent approach is an ideal fit to this naturally distributed problem where the complex interaction among the aircraft, airports and traffic controllers renders a pre-determined centralized solution severely suboptimal at the first deviation from the expected plan. Though a truly distributed and adaptive solution (e.g., free flight where aircraft can choose almost any path) offers the most potential in terms of optimizing flow, it also provides the most radical departure from the current system. As a consequence, a shift to such a system presents tremendous difficulties both in terms of implementation (e.g., scheduling and airport capacity) and political fallout (e.g., impact on air traffic con-

trollers). In this paper, we focus on agent based system that can be implemented readily. In this approach, we assign an agent to a "fix," a specific location in 2D. Because aircraft flight plans consist of a sequence of fixes, this representation allows localized fixes (or agents) to have direct impact on the flow of air traffic[1]. In this approach, the agents' actions are to set the separation 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 impact on the overall air traffic flow. Agents learn the most appropriate separation for their location using a reinforcement learning (RL) algorithm [15].

In a reinforcement learning approach, the selection of the agent reward has a large impact on the performance of the system. In this work, we explore four different agent reward functions, and compare them to simulating various changes to the system and selecting the best solution (e.g, equivalent to a Monte-Carlo search). The first explored reward consisted of the system reward. The second reward was a personalized agent reward based on collectives [3, 17, 18]. The last two rewards were personalized rewards based on estimations to lower the computational burden of the reward computation. All three personalized rewards aim to align agent rewards with the system reward and ensure that the rewards remain sensitive to the agents' actions.

Previous work in this domain fell into one of two distinct categories: The first principles based modeling approaches used by domain experts [5, 8, 10, 13] and the algorithmic approaches explored by the learning and/or agents community [6, 9, 12]. Though our approach comes from the second category, we aim to bridge the gap by using FACET to test our algorithms, a simulator introduced and widely used (i.e., over 40 organizations and 5000 users) by work in the first category [4, 11].

The main contribution of this paper is to present a distributed adaptive air traffic flow management algorithm that can be readily implemented and test that algorithm using FACET. In Section 2, we describe the air traffic flow problem and the simulation tool, FACET. In Section 3, we present the agent-based approach, focusing on the selection of the agents and their action space along with the agents' learning algorithms and reward structures. In Section 4 we present results in domains with one and two congestions, explore different trade-offs of the system objective function, discuss the scaling properties of the different agent rewards and discuss the computational cost of achieving certain levels of performance. Finally, in Section 5, we discuss the implications of these results and provide and map the required work to enable the FAA to reach its stated goal of increasing the traffic volume by threefold.

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 address 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 present FACET a widely used simulation tool and discuss our system evaluation function.

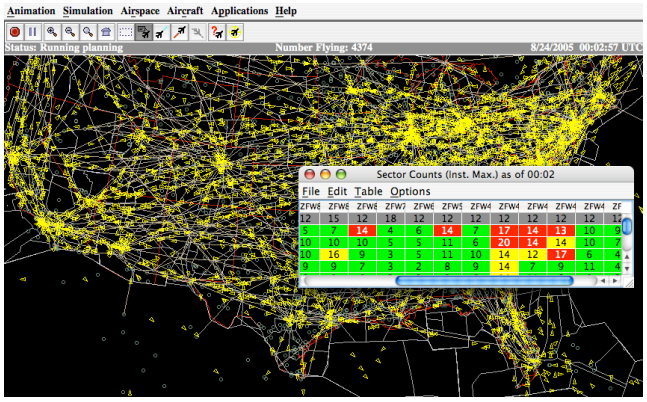

Figure 1: FACET screenshot displaying traffic routes and air flow statistics.

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 [4]. 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

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

on the same computer) [11]. FACET is extensively used by the FAA, NASA and industry (over 40 organizations and 5000 users) [11].

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 1). 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 rewards based on receive feedback from FACET about the impact of these meterings.

2.2 System Evaluation

The system performance evaluation function we selected 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 4.

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 FAAs 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. AGENT BASED AIR TRAFFIC FLOW

The multi agent approach to air traffic flow management we present is predicated on adaptive agents taking independent actions that maximize the system evaluation function discussed above. To that end, there are four critical decisions that need to be made: agent selection, agent action set selection, agent learning algorithm selection and agent reward structure selection.

3.1 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 the agents would not be able to sample their state space sufficiently, learning would be prohibitively slow. 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 reward is easier.
3. Because Aircraft flight plans consist of fixes, agent will have the ability to affect traffic flow patterns.
4. They can be deployed within the current air traffic routing procedures, and can be used as tools to help air traffic controllers rather than compete with or replace them.

Figure 2 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.

3.2 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, 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 determining the separation (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.

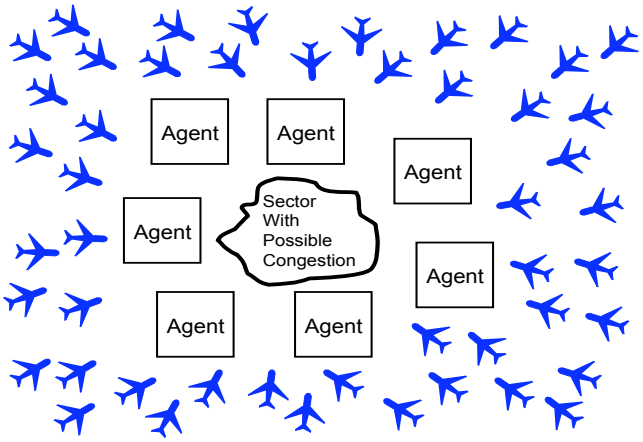

Figure 2: Schematic of agent architecture. The agents corresponding to fixes surrounding a possible congestion become "live" and start setting new separation times.

3.3 Agent Learning

The objective of each agent is to learn the best values of d that will lead to the best system performance, G. In this paper we assume that each agent will have a reward function and will aim to maximize its reward using its own reinforcement learner [15] (though alternatives such as evolving neuro-controllers are also effective [1]). For complex delayed-reward problems, relatively sophisticated reinforcement learning systems such as temporal difference may have to be used. However, due to our agent selection and agent action set, the air traffic congestion domain modeled in this paper only needs to utilize immediate rewards. As a consequence, a simple table-based immediate reward reinforcement learning is used. Our reinforcement learner is equivalent to an ϵ-greedy Q-learner with a discount rate of 0 [15]. At every episode an agent takes an action and then receives a reward evaluating that action. After taking action a and receiving reward R an agent updates its Q table (which contains its estimate of the value for taking that action [15]) as follows:

$$Q'(a) = (1 - l)Q(a) + l(R), \tag{6}$$

where l is the learning rate. At every time step the agent chooses the action with the highest table value with probability $1 - \epsilon$ and chooses a random action with probability ϵ. In the experiments described in this paper, α is equal to 0.5 and ϵ is equal to 0.25. The parameters were chosen experimentally, though system performance was not overly sensitive to these parameters.

3.4 Agent Reward Structure

The final issue that needs to be addressed is selecting the reward structure for the learning agents. The first and most direct approach is to let each agent receive the system performance as its reward. However, in many domains such a reward structure leads to slow learning. We will therefore also set up a second set of reward structures based on agent-specific rewards. Given that agents aim to maximize their own rewards, a critical task is to create "good" agent rewards, or rewards that when pursued by the agents lead to good overall system performance. In this work we focus on difference rewards which aim to provide a reward that is both sensitive to that agent's actions and aligned with the overall system reward [2, 17, 18].

3.4.1 *Difference Rewards*

Consider **difference** rewards of the form [2, 17, 18]:

$$D_i \equiv G(z) - G(z - z_i + c_i) \ , \tag{7}$$

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 reward 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 benefit has been dubbed "learnability" (agents have an easier time learning) in previous work [2, 17]. 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. This term which measures the amount of alignment between two rewards has been dubbed "factoredness" in previous work [2, 17].

3.4.2 *Estimates of Difference Rewards*

Though providing a good compromise between aiming for system performance and removing the impact of other agents from an agent's reward, one issue that may plague D is computational cost. Because it relies on the computation of the counterfactual term $G(z - z_i + c_i)$ (i.e., the system performance without agent i) it may be difficult or impossible to compute, particularly when the exact mathematical form of G is not known. Let us focus on G functions in the following form:

$$G(z) = G_f(f(z)), \tag{8}$$

where $G_f()$ is non-linear with a known functional form and,

$$f(z) = \sum_i f_i(z_i) \ , \tag{9}$$

where each f_i is an unknown non-linear function. We assume that we can sample values from $f(z)$, enabling us to compute G, but that we cannot sample from each $f_i(z_i)$.

[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.

In addition, we assume that G_f is much easier to compute than $f(z)$, or that we may not be able to even compute $f(z)$ directly and must sample it from a "black box" computation. This form of G matches our system evaluation in the air traffic domain. When we arrange agents so that each aircraft is typically only affected by a single agent, each agent's impact of the counts of the number of aircraft in a sector, $k_{t,s}$, will be mostly independent of the other agents. These values of $k_{t,s}$ are the "$f(z)$s" in our formulation and the penalty functions form "G_f." Note that given aircraft counts, the penalty functions (G_f) can be easily computed in microseconds, while aircraft counts (f) can only be computed by running FACET taking on the order of seconds.

To compute our counterfactual $G(z - z_i + c_i)$ we need to compute:

$$G_f(f(z - z_i + c_i)) = G_f\left(\sum_{j \neq i} f_j(z_j) + f_i(c_i)\right) \quad (10)$$

$$= G_f\left(f(z) - f_i(z_i) + f_i(c_i)\right) \quad (11)$$

Unfortunately, we cannot compute this directly as the values of $f_i(z_i)$ are unknown. However, if agents take actions independently (it does not observe how other agents act before taking its own action) we can take advantage of the linear form of $f(z)$ in the f_is with the following equality:

$$E(f_{-i}(z_{-i})|z_i) = E(f_{-i}(z_{-i})|c_i) \quad (12)$$

where $E(f_{-i}(z_{-i})|z_i)$ is the expected value of all of the fs other than f_i given the value of z_i and $E(f_{-i}(z_{-i})|c_i)$ is the expected value of all of the fs other than f_i given the value of z_i is changed to c_i. We can then estimate $f(z - z_i + c_i)$:

$$\begin{aligned} f(z) - f_i(z_i) + f_i(c_i) &= f(z) - f_i(z_i) + f_i(c_i) \\ &+ E(f_{-i}(z_{-i})|c_i) - E(f_{-i}(z_{-i})|z_i) \\ &= f(z) - E(f_i(z_i)|z_i) + E(f_i(c_i)|c_i) \\ &+ E(f_{-i}(z_{-i})|c_i) - E(f_{-i}(z_{-i})|z_i) \\ &= f(z) - E(f(z)|z_i) + E(f(z)|c_i) \ . \end{aligned}$$

Therefore we can evaluate $D_i = G(z) - G(z - z_i + c_i)$ as:

$$D_i^{est1} = G_f(f(z)) - G_f(f(z) - E(f(z)|z_i) + E(f(z)|c_i)) \ , \quad (13)$$

leaving us with the task of estimating the values of $E(f(z)|z_i)$ and $E(f(z)|c_i))$. These estimates can be computed by keeping a table of averages where we average the values of the observed $f(z)$ for each value of z_i that we have seen. This estimate should improve as the number of samples increases. To improve our estimates, we can set $c_i = E(z)$ and if we make the mean squared approximation of $f(E(z)) \approx E(f(z))$ then we can estimate $G(z) - G(z - z_i + c_i)$ as:

$$D_i^{est2} = G_f(f(z)) - G_f(f(z) - E(f(z)|z_i) + E(f(z))) \ . \quad (14)$$

This formulation has the advantage in that we have more samples at our disposal to estimate $E(f(z))$ than we do to estimate $E(f(z)|c_i))$.

4. SIMULATION RESULTS

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 five different methods. The first method is Monte Carlo estimation, where random policies are created, with the best policy being chosen. The other four methods are agent based methods where the agents are maximizing one of the following rewards:

1. The system reward, $G(z)$, as define in Equation 1.
2. The difference reward $D_i(z)$, assuming that agents can calculate counterfactuals.
3. Estimation to the difference reward $D_i^{est1}(z)$, where agents estimate the counterfactual using $E(f(z)|z_i)$ and $E(f(z)|c_i)$.
4. Estimation to the difference reward $D_i^{est1}(z)$, where agents estimate the counterfactual using $E(f(z)|z_i)$ and $E(f(z))$.

These methods are first tested on an air traffic domain with 300 aircraft, where 200 of the aircraft are going through a single point of congestion over a four hour simulation. Agents are responsible for reducing congestion at this single point, while trying to minimize delay. The methods are then tested on a more difficult problem, where a second point of congestion is added with the 100 remaining aircraft going through this second point of congestion.

In all experiments the goal of the system is to maximize the system performance given by $G(z)$ with the parameters, $a = 50$, $b = 0.3$, 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. In all experiments to make the agent results comparable to the Monte Carlo estimation, the best policies chosen by the agents are used in the results. All results are an average of thirty independent trials 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.

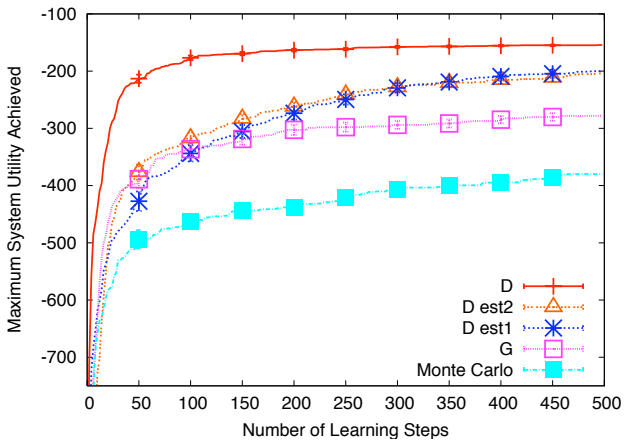

Figure 3: Performance on single congestion problem, with 300 Aircraft , 20 Agents and $\alpha = .5$.

4.1 Single Congestion

In the first experiment we test the performance of the five methods when there is a single point of congestion, with twenty agents. This point of congestion is created by setting up a series of flight plans that cause the number of aircraft in

the sector of interest to be significantly more than the number allowed by the FAA. The results displayed in Figures 3 and 4 show the performance of all five algorithms on two different system evaluations. In both cases, the agent based methods significantly outperform the Monte Carlo method. This result is not surprising since the agent based methods intelligently explore their space, where as the Monte Carlo method explores the space randomly.

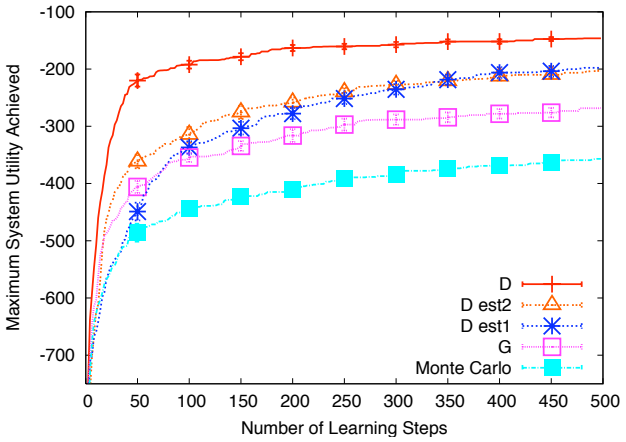

Figure 4: Performance on single congestion problem, with 300 Aircraft , 20 Agents and $\alpha = .75$.

Among the agent based methods, agents using difference rewards perform better than agents using the system reward. Again this is not surprising, since with twenty agents, an agent directly trying to maximize the system reward has difficulty determining the effect of its actions on its own reward. 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 reward have more influence over the value of their own reward, therefore when an agent takes a good action, the value of this action is more likely to be reflected in its reward.

This experiment also shows that estimating the difference reward is not only possible, but also quite effective, when the true value of the difference reward cannot be computed. While agents using the estimates do not achieve as high of results as agents using the true difference reward, they still perform significantly better than agents using the system reward. Note, however, that the benefit of the estimated difference rewards are only present later in learning. Earlier in learning, the estimates are poor, and agents using the estimated difference rewards perform no better then agents using the system reward.

4.2 Two Congestions

In the second experiment we test the performance of the five methods on a more difficult problem with two points of congestion. On this problem the first region of congestion is the same as in the previous problem, and the second region of congestion is added in a different part of the country. 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.

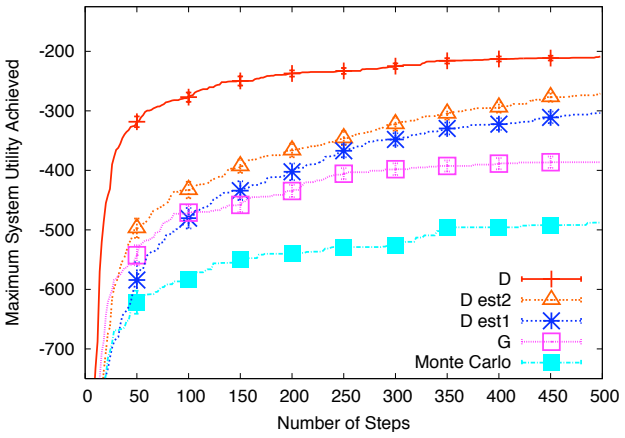

Figure 5: Performance on two congestion problem, with 300 Aircraft, 20 Agents and $\alpha = .5$.

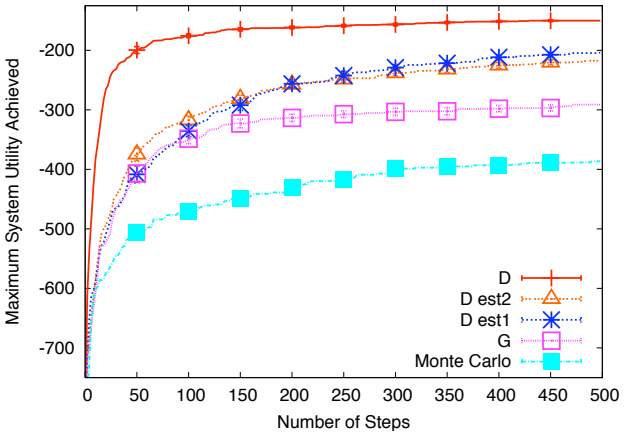

Figure 6: Performance on two congestion problem, with 300 Aircraft, 50 Agents and $\alpha = .5$.

The results displayed in Figure 5 show that the relative performance of the five methods is similar to the single congestion case. Again agent based methods perform better than the Monte Carlo method and the agents using difference rewards perform better than agents using the system reward. To verify that the performance improvement of our methods is maintained when there are a different number of agents, we perform additional experiments with 50 agents. 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. Agents using D^{est2} perform slightly better than agents using D^{est1} in all cases but for 50 agents. This slight advantage stems from D^{est2} providing the agents with a cleaner signal, since its estimate uses more data points.

4.3 Penalty Tradeoffs

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

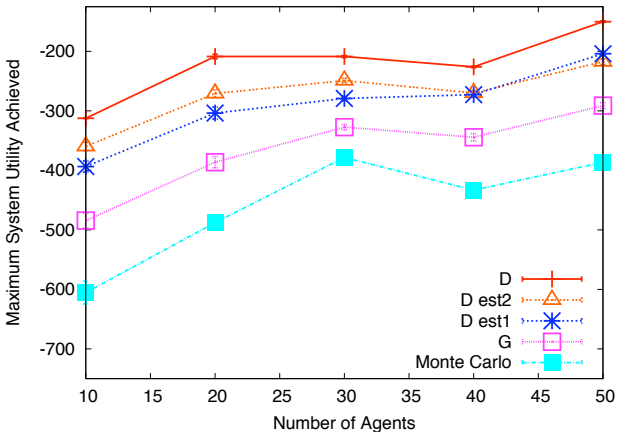

Figure 7: Impact of number of agents on system performance. Two congestion problem, with 300 Aircraft and $\alpha = .5$.

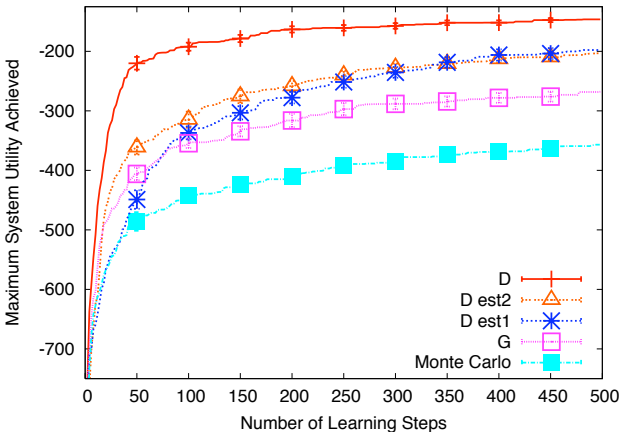

Figure 8: Performance on two congestion problem, with 300 Aircraft, 20 Agents and $\alpha = .75$.

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. To verify that the results obtained above are not specific to a particular value of α, we repeat the experiment with 20 agents for $\alpha = .75$. Figure 8 shows that qualitatively the relative performance of the algorithms remain the same.

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, making alpha=0.5 a reasonable choice for testing the algorithms in a difficult setting.
- Third, Monte Carlo and G are particularly poor at handling multiple objectives. For both algorithms, the performance degrades significantly for mid-ranges of α.

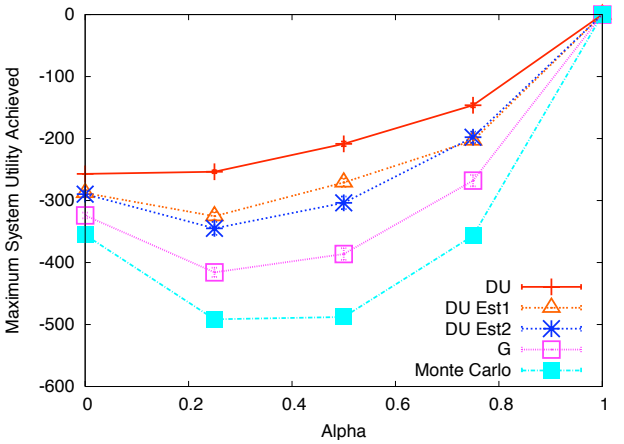

Figure 9: Tradeoff Between Objectives on two congestion problem, with 300 Aircraft and 20 Agents. Note that Monte Carlo and G are particularly bad at handling multiple objectives.

4.4 Computational Cost

The results in the previous section show the performance of the different algorithms after a specific number of episodes. Those results show that D is significantly superior to the other algorithms. One question that arises, though, is what computational overhead D puts on the system, and what results would be obtained if the additional computational expense of D is made available to the other algorithms.

The computation cost of the system evaluation, G (Equation 1) is almost entirely dependent on the computation of the airplane counts for the sectors $k_{t,s}$, which need to be computed using FACET. Except when D is used, the values of k are computed once per episode. However, to compute the counterfactual term in D, if FACET is treated as a "black box", each agent would have to compute their own values of k for their counterfactual resulting in $n + 1$ computation of k per episode. While it may be possible to streamline the computation of D with some knowledge of the internals of FACET, given the complexity of the FACET simulation, it is not unreasonable in this case to treat it as a black box.

Table 1 shows the performance of the algorithms after 2100 G computations for each of the algorithms for the simulations presented in Figure 5 where there were 20 agents, 2 congestions and $\alpha = .5$. All the algorithms except the fully computed D reach 2100 k computations at time step 2100. D however computes k once for the system, and then once for each agent, leading to 21 computations per time step. It therefore reaches 2100 computations at time step 100. We also show the results of the full D computation at t=2100, which needs 44100 computations of k as D^{44K}.

Table 1: System Performance for 20 Agents, 2 congestions and $\alpha = .5$, after 2100 G evaluations (except for D^{44K} which has 44100 G evaluations at t=2100).

Reward	G	$\sigma/\sqrt{n}$	time
D^{est2}	-232.5	7.55	2100
D^{est1}	-234.4	6.83	2100
D	-277.0	7.8	100
D^{44K}	-219.9	4.48	2100
G	-412.6	13.6	2100
MC	-639.0	16.4	2100

Although D^{44K} provides the best result by a slight margin, it is achieved at a considerable computational cost. Indeed, the performance of the two D estimates is remarkable in this case as they were obtained with about twenty times fewer computations of k. Furthermore, the two D estimates, significantly outperform the full D computation for a given number of computations of k and validate the assumptions made in Section 3.4.2. This shows that for this domain, in practice it is more fruitful to perform more learning steps and approximate D, than few learning steps with full D computation when we treat FACET as a black box.

5. DISCUSSION

The efficient, safe and reliable management of air traffic flow is a complex problem, requiring solutions that integrate control policies with time horizons ranging from minutes up to a year. The main contribution of this paper is to present a distributed adaptive air traffic flow management algorithm that can be readily implemented and to test that algorithm using FACET, a simulation tool widely used by the FAA, NASA and the industry. Our method is based on agents representing fixes and having each agent determine the separation between aircraft approaching its fix. It offers the significant benefit of not requiring radical changes to the current air flow management structure and is therefore readily deployable. The agents use reinforcement learning to learn control policies and we explore different agent reward functions and different ways of estimating those functions.

We are currently extending this work in three directions. First, we are exploring new methods of estimating agent rewards, to further speed up the simulations. Second we are investigating deployment strategies and looking for modifications that would have larger impact. One such modification is to extend the definition of agents from fixes to sectors, giving agents more opportunity to control the traffic flow, and allow them to be more efficient in eliminating congestion. Finally, in cooperation with domain experts, we are investigating different system evaluation functions, above and beyond the delay and congestion dependent G presented in this paper.

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.

6. 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] A. Agogino and K. Tumer. Multi agent reward analysis for learning in noisy domains. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Utrecht, Netherlands, July 2005.

[3] A. K. Agogino and K. Tumer. Handling communiction restrictions and team formation in congestion games. *Journal of Autonous Agents and Multi Agent Systems*, 13(1):97–115, 2006.

[4] 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.

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

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

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

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

[9] 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.

[10] 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.

[11] 2006 NASA Software of the Year Award Nomination. FACET: Future ATM concepts evaluation tool. Case no. ARC-14653-1, 2006.

[12] M. Pechoucek, D. Sislak, D. Pavlicek, and M. Uller. Autonomous agents for air-traffic deconfliction. In *Proc of the Fifth Int Jt Conf on Autonomous Agents and Multi-Agent Systems*, Hakodate, Japan, May 2006.

[13] B. Sridhar and S. Grabbe. Benefits of direct-to in national airspace system. In *AIAA Guidance, Navigation, and Control Conf*, Denver, CO, 2000.

[14] B. Sridhar, T. Soni, K. Sheth, and G. B. Chatterji. Aggregate flow model for air-traffic management. *Journal of Guidance, Control, and Dynamics*, 29(4):992–997, 2006.

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

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

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

[18] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.