Announced strategy types in multiagent RL for conflict-avoidance in the national airspace

Carrie Rebhuhn
Oregon State University
rebhuhnc@engr.orst.edu

Matthew Knudson
NASA Ames Research Center
matthew.knudson@nasa.gov

Kagan Tumer
Oregon State University
kagan.tumer@oregonstate.edu

ABSTRACT

Automated conflict-avoidance for unmanned aerial systems is quickly becoming feasible due to near-future advances in communication guarantees. The NextGen Implementation Plan introduced by the FAA lays out requirements for air traffic in the US airspace to be implemented by 2020. This includes mandates for the introduction of the automatic dependent surveillance broadcast (ADS-B) on each system in the airspace, which is a plane-installed system that broadcasts information about a plane in the airspace. Optionally, a pilot can choose to install the traffic information service broadcast (TIS-B) system, which allows a plane to observe ADS-B information for a 15-mile radius.

This enhanced communication lends itself well to the formation of a multiagent system to model conflict-resolution in the air. But with a variety of different potential reactions by pilots in a conflict situation, a robust response structure must be developed. In this work, we introduce agents to make distributed decisions regarding rerouting for conflict-avoidance, and explore the possible benefit of *stereotyping* in this domain. We demonstrate also that the availability of avoidance type information can improve performance by reducing agent noise in a simple system reward formulation.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Multiagent reinforcement learning, unmanned autonomous systems, national airspace, conflict-avoidance

1. INTRODUCTION

The air traffic density in the national airspace (NAS) is increasing beyond the current capabilities of air traffic controllers. Centralized human control at each sector in the airspace works well for low plane-to-controller ratios, but with the growth commercial air traffic and the future introduction of autonomous systems in the airspace, it is clear that control algorithms must begin to handle some of the safety in the airspace. The FAA NextGen Implementation Plan promises regulation mandating that automatic dependent surveillance broadcast (ADS-B) systems be installed on all aircraft, providing a method by which air traffic control systems can easily gather information about sector congestion. More promising from a multiagent standpoint is the introduction of traffic information service broadcast (TIS-B), which provides distributed automatic ADS-B information to planes within a 15-mile radius of other planes. Using this locally-available information, we can construct conflict-avoidance in the national airspace as a distributed multiagent problem.

The problem of conflict-avoidance has been approached in several different ways. Geometric conflict-avoidance techniques plan out paths that avoid conflict situations while maintaining optimal trajectories [26]. These give guarantees about optimality, but they do not address the potential for changes in information about the predicted path. Game theory has also been attempted for conflict-avoidance, but this requires accurate models of opponent behavior in order to develop payoff matrices, and due to the potentially large size of the state space in the air traffic domain the problem does not scale well.

Policies developed by more robust techniques such as learning offer a promising alternative to deal with the potential stochasticity of a real-world implementation of the conflict-avoidance domain. Stochasticity in this domain could arise from pilot noncooperation, miscommunication, or inexperience, and so it is necessary for developed algorithms operating within a human-populated system to be robust to this type of failure. To model this type of behavior, we include agents with simple hard-coded policies within our domain to model predictable but non-optimal behaviors within the system. Using this set of policy types introduces an enforced hetereogeneity of policies within the system.

Multiagent systems have always had difficulty handling hetereogeneity between agents, and the NAS has a high potential for heterogeneity. Modeling each agent is also not useful because each plane may be seen only a couple times during a simulation. Different *types* of planes may be seen more frequently. This does not explicitly give responses, but implicitly defines a behavior type by the system. By dividing into types we can leverage this information without explicitly modeling responses. The use of typecasting (also referred to as 'stereotyping' or 'generalized agent mod-

Appears in: *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), Lomuscio, Scerri, Bazzan, Huhns (eds.), May, 5–9, 2014, Paris, France.*

eling') has been shown to perform well in domains where agents do not encounter the same other agents frequently. In the NAS agents may have conflicts with other agents that behave similarly to agents encountered earlier, but are not identical.

The contributions of this work are:

- We use multiagent learning to develop general conflict-avoidance policies in a modeled random airspace.
- We demonstrate the performance benefits of the inclusion of strategy type identification. This motivates future work which would model the agent classification more thoroughly.

The rest of the paper is organized as follows. Section 2 covers background information on the conflict-avoidance domain, previous approaches to stereotype development, and an overview of difference rewards, a reward shaping technique that we will use as a comparison in this paper. Section 3 covers implementation of reinforcement learning in the conflict-avoidance domain, as well as the development of the rewards and simulator structure. Section 4 covers the definition of the strategy types which exist among the agents in the mlutiagent system. Section 5 covers the results from three different scales, indicating the scalability of this learning approach. Finally, Section 6 overviews the findings from the experimentation and Section 7 gives a brief outline of future directions for this research.

2. RELATED WORK

The work presented in this paper has roots in many fields, including air traffic conflict-avoidance, agent modeling, and reward shaping. We briefly cover key topics that are relevant to the work in this paper.

2.1 Conflict-Avoidance

There are two main approaches to developing algorithms to ensure safety in the airspace; the first is to control the *density* of aircraft which are passing through different sectors in order to balance the load on regional air traffic controllers, and the second is to develop optimal *rerouting* procedures in order to avoid a predicted separation losses between aircraft. These problems are coupled, in that as the density of the aircraft moving through sector decreases the probability of conflicts occurring (and therefore the difficulty of solving this problem) also decreases [3]. These approaches are called air traffic flow management (ATFM) and conflict-avoidance respectively. In this work we focus on the problem of conflict-avoidance.

Conflict-avoidance is currently performed at ground stations during plane flight. While many approaches to ATFM focus primarily on the airspace as an abstract scheduling problem, the problem of conflict-avoidance focuses instead on preventing temporal and spatial conflicts from occurring within an airspace by re-planning conflicted routes, not necessarily restricting this re-planning to passing through fixes. This approach focuses on safety at the route-planning level rather than the scheduling level, and agents have many more options for maneuvers to avoid conflicts. Optimality plays a major role as well; aircraft must also consider the cost of maneuvers taken to ensure the safety of the aircraft [2].

One approach is to treat this as a pairwise problem, where one aircraft will conflict with another in a given time horizon and a reroute or velocity change must occur in order to avoid this conflict. [21]. This unfortunately does not explicitly tackle the problem of 'cascading' conflicts, which are conflicts generated by rerouting between two planes. Approaches which address this include plotting courses for avoidance maneuvers using mixed integer programming [11, 8], distributed search strategies [22, 24], optimal control [26], game theory [18], and negotiation [23].

These approaches work well when the conflict between aircraft is well-defined and has no uncertainty. However in real flight there can be differences between the announced flight plans and the actual trajectory taken. The FAA separation restrictions accommodate this uncertainty in flight, however as the airspace becomes more crowded, it will become impossible to attain the desired throughput while maintaining the current safety restrictions. In order to maintain safety in the airspace while allowing a higher throughput, it is necessary to automatically identify which types of planes need higher safety allowances, and to predict ways these planes may deviate from their courses that may not be communicated to controllers in a timely manner. The information on air traffic necessary to make these more sophisticated decisions should be available to each plane through the (mandatory by 2020) ADS-B implementation, and the (optional) TIS-B subscriptions [15].

2.2 Reputation, Modeling, and Typecasting

Agent modeling has roots in game theory with the concept of *reputation* [17], which is used for inferring the strategy type of another agent in iterated play. Strategies of competing agents are typically hidden, so an agent that can accurately identify (and subsequently exploit) the strategy of its opponent has a clear advantage. By observing several of the actions of the opponent agents, an agent using *reputation* can develop a belief distributions over strategies known to it. Fudenberg and Levine [16] prove that if an agent's playing type is known, the best performance it can hope to achieve is its Stackelberg payoff, but if this type is not known it can potentially do better.

While reputation typically focuses on fitting other agents to known probabilistic models of play, opponent modeling is a two-step process which focuses more on the building of these models and then using subsequent observations to fit to the observed models. Opponent modeling has been the focus of recent interest in the application of Poker [5, 4, 25, 27, 20]. Poker is a domain in which the *style* of play of other players greatly impacts the potential reward. This makes obtaining some information on the player types crucial to performing well in this domain. Teofilo et al. [27] showed that a Q-learning agent whose state incorporated predefined player types, which were calculated based on the frequency of past plays, outperformed basic/intermediate playing strategies. Lazaric et al. [20] point out that if the model of behavior is sufficiently inaccurate, the use of the defined model may inhibit learning in Q-learning agents.

Barrett et al. developed a generalized teammate modeling method, which was shown to promote collaboration on a diverse set of hand-created agents. Using the estimated teammate models they then used a planning algorithm to take actions [7]. Barret et al. also a study on the performance of several ad hoc teamwork between several agents which had varying levels of awareness about other teammates of several types. They showed that even if models are incorrect or incomplete, they can still be used to provide

good performance [6].

The use of typecasting is essential in cases where agents are encountered briefly within the system. Typecasting has been employed with the concept of 'trust', which is expressed as a confidence that the other agent will follow through with an agreed action. Burnett et al. uses a tree model to represent a stereotype, and these stereotypes are shared and updated by the agent community [9]. This is further developed by their concept of *stereotypical reputation*, which gives a mechanism by which agents which have no set opinion can use the opinions developed by others in the system about *stereotypes* of other agents in order to develop their own trust evaluations [10]. Denzinger et al. [14] use stereotypes with a periodic reevaluation of the chosen stereotype, and may switch between different stereotypes.

Typecasting has also proved useful in learning robust policies in poker. In [5], Bard et al. present a method of learning robust responses to several player types learned offline, which provides implicit modeling rather than relying on an explicit model of an agent's actions. Similarly, in this work we learn *responses* to defined types through inclusion of type information into the state space. This prevents the need for a detailed or explicit model of the other agents.

2.3 Difference Rewards

Reinforcement learning in a cooperative multiagent system focuses on finding a joint action that most benefits the collective system. Learning agents adapt their strategies through repeatedly taking actions and getting a reward for these actions. For policies developed in multiagent learning systems, a key element to promoting coordination is the selection of the reward functions.

Difference rewards have been successful largely due to the fact that they directly address two key problems in reward shaping; factoredness and learnability. Factoredness is the degree to which an agent is rewarded for helping the system as a whole, and learnability is the degree to which the reward given to the agent actually addresses the effects of that agent's action. Difference rewards incentivize cooperation with the system goal, as well as ensuring that the reward delivered is an information-rich reflection of that individual agent's performance.

The difference reward's formulation is given by [1]:

$$D_j(\vec{s},\vec{a}) = G(\vec{s},\vec{a}) - G(\vec{s}_{-j} + c_j, \vec{a}_{-}j) \quad (1)$$

where $G(\vec{s},\vec{a})$ is the global evaluation function and $G(\vec{s}_{-j} + c_j, \vec{a}_{-}j)$ is the global evaluation function without the effects of agent j. Intuitively, the difference evaluation function gives the impact of agent j on the global evaluation function, as the second term removes the portions of the $G(\vec{s},\vec{a})$ not dependent on agent j. The difference objective provides effective agent-specific feedback for learning agents within a multiagent system. The difference objective has provided excellent results in many domains, including distributed sensor network control [13], rover control [12], and air-traffic control [1]. It is important to note that though the difference reward has been applied extensively to the problem of *air-traffic control*, it has only recently been applied to the problem of *conflict-avoidance*.

3. PROBLEM DEFINITION

Conflict-avoidance in the UAS domain becomes a complex problem when there is a high density of planes in the system. Planes must select a deviation from their path in order to avoid conflict with another plane, but must also reach a destination within a reasonable amount of time. In the context of modeling the UAS as a multiagent system, an agent must select the parameters of a *diversion waypoint* in a way that avoids conflict without taking an unnecessary amount of extra distance or causing additional conflicts as a result of this evasive action.

3.1 Reinforcement Learning in the NAS

In this work we use reinforcement learning agents to map plane states (relative to the plane's nearest neighbor) to conflict-avoidance actions (waypoints) through Q-learning. The purpose of agents in our conflict-avoidance domain is to select waypoints in a way that avoids conflict propagation without drastically sacrificing path optimality. There are a set of agents $\mathcal{A}$, each of which make conflict resolution decisions for a single vehicle in the system. We call the joint state of all of these agents $\vec{s}$.

Each agent i has a state s_i, which is described in relation to its nearest neighbor, such that $s_i = \{\delta_{n(i)}, \Theta_{n(i)}, h_{n(i)}, \mathcal{T}_{n(i)}, p_i, g_i\}$, where $\delta_{n(i)}$ is the xy-planar distance to the nearest neighbor of i, $\Theta_{n(i)}$ is the relative heading of the nearest neighbor of i, $h_{n(i)}$ is the relative height (z-position) of the nearest neighbor of i, $\mathcal{T}_{n(i)}$ is the type of the nearest neighbor, p_i is the agent's position, and g_i is the goal position of the agent. The position p_i and goal position g_i of the agent is not useful for the task of conflict-avoidance, so for the purpose of Q-learning we do not distinguish between states with different p_i or g_i values. The type information $\mathcal{T}_{n(i)}$ is used to distinguish different states in Q-learning in our *type-partitioned* experiments, but we compare to when this is not used to distinguish types in our *type-free* experiments.

Agents select an action $a_i = \{\tau, m, t\}$, where τ is the action type (heading change, or altitude change), m is the magnitude of this change, and t is the duration of the redirection. Agents use $\epsilon-$greedy action selection to choose an action a_i based on their a state s_i, and then receive a reward $R(\vec{s},\vec{a})$ based on the system state $\vec{s}$ and the action taken by the agent. This reward is used to update each agent's Q-table using the update:

$$Q_i^{t+1}(s_i^t, a_i^t) = Q_i^t(s_i^t, a_i^t) + \alpha(R(\vec{s}^{t+1}, \vec{a}^t) + \gamma \max_a Q_t(s_i^{t+1}, a) - Q_t(s_i^t, a_i^t)) \quad (2)$$

where $Q_t(s_i^t, a_i^t)$ is Q-value for the state and action taken by an agent at time t, α is the learning rate (between 0 and 1), $R(\vec{s}^{t+1}, \vec{a}^t$ is the reward for the joint state of the system at time $t+1$ and on the set of actions taken by all agents $\vec{a}^t$ at time t, and γ is the learning discount factor.

Actions that agents take in this domain result in the creation of a *diversion waypoint*, which is a temporary redirection of the vehicle from its target. When a diversion waypoint is created, the agent's plane continues toward this waypoint until either another conflict is detected and a different waypoint is selected, or the duration of the waypoint change has ended. If a new conflict is detected, this waypoint may be replaced by a different choice. If the duration ends, the plane will refix on and continue toward its original goal. Goal acquisition results in an instantaneous redirection to another target so that an extended time horizon can

be explored. This procedure of redirecting after moving to the waypoint separates the task of conflict-avoidance from the task of optimal navigation, although the reward captures some of this extra cost. We use the reward signal to shape the behavior toward optimal navigation as well as conflict avoidance, as we explain in Section 3.2.

3.2 Reward Design

The objective of learning within the conflict-avoidance domain is to minimize costs of flight path alteration while keeping at a safe distance from other planes. This can be framed as a cost minimization problem, where the agents (UAS) in the system are working together to minimize the system utility. We formalize this system utility in this domain as:

$$G(\vec{s},\vec{a}) = \sum_{i\in\mathcal{A}} d_{extra}(a_i) + \frac{|\mathcal{A}|L}{\sum_{i\in\mathcal{A}} d_{neighbor}(s_i)} \tag{3}$$

where $G(\vec{s},\vec{a})$ is the global reward dependent on the states $\vec{s}$ and actions $\vec{a}$, $d_{extra}(a_i)$ is the distance cost incurred by the action of agent i, $a_i \in \vec{a}$, $d_{neighbor}(s_i)$ is the euclidean distance of the nearest neighbor of agent i, $|\mathcal{A}|$ is the number of agents and L is the edge length of the testing map. The first term incentivizes minimization of detour cost in the system, while the second term incentivizes maximization of the average distance between planes across the system, weighted by the edge length of the world. Multiplication by the edge length effectively scales the second term to the first term.

Global rewards are useful for promoting coordination, however they do not address noise introduced by other agents. Difference rewards have had success in the past in reducing the agent noise in the system. We formulate a difference reward based on the global reward that we developed in Equation 3.2 and the difference reward given by Equation 1. We use a zero-padded counterfactual. Conflicts within this system have others which are affected, so removing these from the system modifies the reward in a subset of agents which have a conflict with the agent (C_j, unique from lowercase c_j which is the counterfactual term).

$$D_j(\vec{s},\vec{a}) = \left(\sum_{i\in\mathcal{A}} d_{extra}(a_i) + \frac{|\mathcal{A}|L}{\sum_{i\in\mathcal{A}} d_{neighbor}(s_i)}\right) - \left(\sum_{i\in\mathcal{A}, i\notin C_j} d_{extra}(a_i) + \frac{|\mathcal{A}\notin C_j|L}{\sum_{i\in\mathcal{A}, i\notin C_j} d_{neighbor}(s_i)}\right) \tag{4}$$

$$D_j(\vec{s},\vec{a}) = \sum_{i\in C_j} d_{extra}(a_i) + \frac{|\mathcal{A}\in C_j|L}{\sum_{i\in C_j} d_{neighbor}(s_i)} \tag{5}$$

where $D_j(\vec{s},\vec{a})$ is the difference reward for agent j. This aids in coordination by identifying specifically the agents with which agent i is in conflict, and penalizing it under the assumption that if it were not there the conflict would not exist. This is a strong assumption, because another agent in the system besides i may also be in conflict with one of the agents in C_j, but in the interest of computational speed this is taken as a good approximation.

3.3 Simulator Dynamics

We use a point-mass simulator to generate conflict scenarios within the domain. The objective is to prevent loss

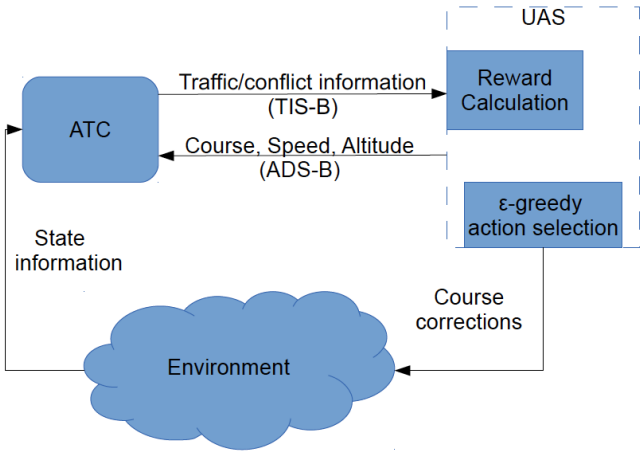

Figure 1: A high-level overview simulation of the UAS conflict-avoidance domain. The ATC represents a centralized communication structure, while decisions on course corrections for conflict-avoidance are made by the UAS, and therefore are decentralized.

of separation in the airspace by calling on agents to reroute when a potential future conflict is announced by an air traffic controller. UAS within the simulator start with random targets and random starting locations within an area of airspace, and the simulator detects the conflicts between the different UAS. When the conflicts in the system are detected, the simulator, which represents an air traffic controller (ATC) in the national airspace, calls *all* agents to make an action choice. All agents, rather than just the conflicted ones, are called to make decisions to address the problem of conflict-propagation in the system. The air traffic controller provides a central information source, but the decisions about how to modify the course are made in a decentralized manner. The simulation process is shown at a high level in Figure 1.

Our simulation preserves the core system dynamics, but simplifies the physics and geometry of the domain for the purposes of algorithm testing. Targets for the planes are created randomly within a square area, with a zero altitude. For purposes of keeping congestion constant as the problem scales to a larger airspace with more agents, the edge length of the square area is given by Equation 6:

$$L = \frac{|\mathcal{A}| dxy_{conflict}}{2} \tag{6}$$

where $dxy_{conflict}$ is the minimum allowable horizontal distance between planes. In this simulation $dxy_{conflict} = 100m$.

Planes are also created randomly with zero altitude. Planes may ascend to a height of 1,000m, and descend as low as 0m within the simulation space. This simulates the airspace restrictions imposed currently on planes, either by regulation or by physical capabilities of the plane. No restrictions are placed on the lateral movement of the plane, as this would unnecessarily hinder the movement of planes created at the edges of the target creation area. Takeoff and landing procedures are also not modeled, as they typically have a prescribed methodology at the airport.

In our formulation of this problem we ignore many variables present in true flight simulation, such as aerodynamics, turning radii, and airspeed limitations. However we maintain the core information necessary to for realistic decision-making in this domain. Planes with ADS-B devices can communicate their course, speed, and altitude to a ground controller. Planes equipped with TIS-B also have access to this information within a 15mi radius and +- 3500 ft altitude. By 2020, ADS-B capabilities will be required to fly in the NAS [15], and we assume a learning agent would have access to TIS-B services as well. The broadcasting of types is also reasonable: the identifier information transmitted by the ADS-B system may be used in conjunction with a simple database of planes and their associated type (i.e. small aircraft, jet liner, etc.).

4. IMPLEMENTATION OF AGENT TYPES

Our objective in this work is not only to use reinforcement learning in the UAS domain, but to also explore the inclusion of abstract strategy information in the state space. In this section we define several heuristic strategy types.

An agent's type relates to its behavior in the system; in an abstract way an agent announces its general strategy in the system. Knowing this strategy reduces some of the uncertainty in the system, and the nearest-neighbor agents become more than just background noise. Because our approach to conflict-avoidance generically observes types of planes, rather than estimating their next action, deviations from the policy can be accommodated by the reinforcement learners. This gives information about agent strategies without having an explicit and rigorous model of the other planes.

Table 1 outlines the three different heuristic types we use in our implementation. These *heuristic* types represent fixed strategies of UAS for conflict-avoidance, and they do not learn to change their action. The first type is a counterclockwise-avoiding agent, which will always select to adjust its heading counterclockwise (by an amount between 0 and $+90^\circ$) in order to avoid another agent. The second type is a clockwise-avoiding agent, which will always select a clockwise (by an amount between 0 and $-90(^\circ)$) action to avoid a conflict. The third agent type will always change its altitude in order to avoid a conflict.

There is also a learning type, which does not have fixed actions, and instead learns to take optimal actions within the system. This type is also observable to other learning agents, and gives a much more vague indication of their nearest neighbor's next action in comparison to knowledge of the heuristic types. Scaling up of the problem is performed while keeping the ratio of agents consistent at 1:1:1:2, so there are 1.5 times as many heuristic types as learners.

5. RESULTS

In order to characterize learning in the UAS domain, we tested several different learning structures without the inclusion of type information in the state space. We then compared these to learning performance using a state space that included type information. Learning is performed over 5000 steps and 10 statistical runs. Here we present learning results using several problem scales, including 50, 100, and 200-agent systems.

To keep the learning scalability as the focus, we increased the world size proportionally to a square of the increase in agents. If the world size is kept constant, the congestion problem with one amount of agents might be trivial to solve, while the congestion problem with a slightly larger amount of agents would be impossible. We aimed to have a steady but manageable level of congestion throughout the system while we varied between a large number of agents.

Type	**Heading**	**Altitude**	**Maintain**
CCW	$+90^\circ$	0	false
CW	-90°	0	false
Up	0	+1000m	false
Type	ϵ	α	γ
Learner	0.1	0.1	0.9

Table 1: The three heuristic (non-learning) agent types are shown at the top, showing the actions taken by the agents at each conflict instance. There is also 'Learner' type agent within the system, which performed ϵ-greedy Q-learning with the given ϵ,α, and γ values. Different numbers of agents are tested keeping a constant constant proportion of CCW:CW:Up:Learner, 1:1:1:2 respectively.

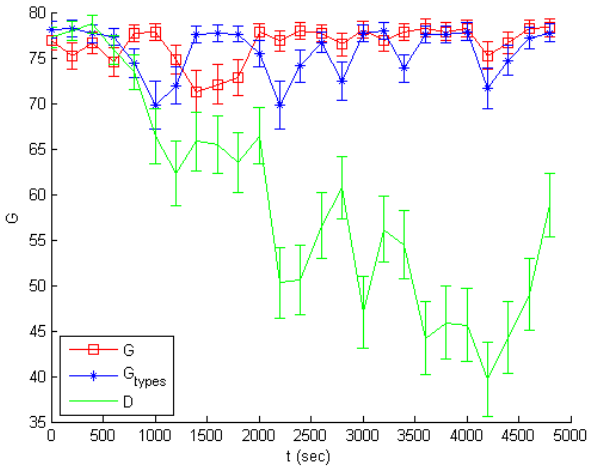

Figure 2: Learning conflict-avoidance with 50 agents over 5000 timesteps with 10 statistical runs. Error bars are given according to the mean-squared error.

Although we attempt to even out conflict frequencies through statistical runs, there is still a cyclic nature to conflict generation and resolution. The agents are only given feedback when there is a conflict in the system, and therefore this will in the global reward to be exceptional performance ($G = 0.0$) that will then encounter a conflict and appear to do worse.

Figure 2 shows the performance of 50 agents in our system. The difference reward significantly outperforms the global reward. The global reward that uses types performs somewhat better than the global reward without types, but does not do significantly better and ends up at similar values at the end of the run.

In Figure 3 we show performance of a system with 100 agents. Performance is similar for the first half of the run, but diverges around $t = 2500s$. Agents using the global re-

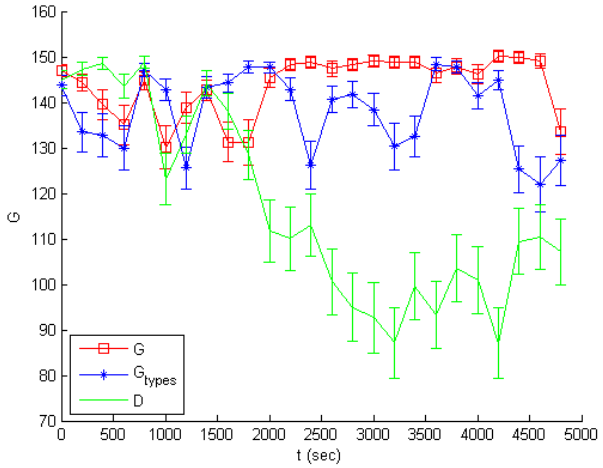

Figure 3: Learning conflict-avoidance with 100 agents over 5000 timesteps with 10 statistical runs. Error bars are given according to mean-squared error. Agents trained on the global reward without types perform poorly, and agents trained using the difference rewards without types perform well. Agents trained using the global reward with types are able to learn a somewhat better policy than global rewards without types, but do not attain the performance of difference rewards.

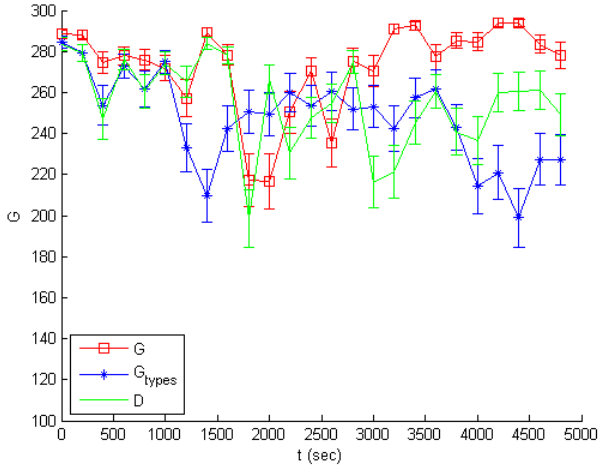

Figure 4: Learning conflict-avoidance with 200 agents over 5000 timesteps with 10 statistical runs. Error bars are given according to the mean-squared error.

ward are unable to learn with the amount of agent noise in the system. The use of types in the global reward appears to cut down some of the noise, however, by expanding the state space to include more local (nearest neighbor) information. The difference reward performs a similar function, cutting down on the noise of the agents through an additional computation.

In our largest system, Figure 4 shows the performance of 200 agents. This figure shows that the value of type identification greatly increases with the large increase in agents, and in contrast to the other experiments the difference reward does not outperform the global reward with types. This indicates that the qualities of type incorporation into the state space may increase in value with the problem size.

6. CONCLUSIONS

In this paper we use reinforcement learning in a complex UAS conflict-avoidance domain. This domain is characterized by a high level of stochasticity and unmodeled plane interactions, which are propagated abstractly to agents through the reward signal. We show the response of three different reward mechanisms to scaling.

Scaling up the number of agents in the system shows a shift between importance levels of different aspects of the system. For lower numbers of agents (50-100 agents), the importance of capturing the total effects of agents on the system was more helpful than considering their neighbor's strategy types. Conversely, at high numbers of agents (200 agents) the benefit obtained by the added strategy information was greater than the amount of benefit obtained from having a more agent-specific reward. This concept of a tradeoff, and what exactly defines these system qualities, will be explored more in future work.

This failure on the part of the difference reward to handle higher numbers of agents may also have had something to do in part with the accumulation of errors introduced by approximating an agent's absence in the system. This points out a difficulty in implementation–the difference reward must in many cases be somehow approximated rather than directly calculated. In contrast, the inclusion of types into the state space of the global reward is much easier given some communication of types, or a mechanism by which to obtain hidden types. It is much more difficult to calculate the state that would have existed if an agent had not been there, and application of the difference reward rarely examines downstream benefits. An approximation is sufficient in many cases, however, so the difference reward is feasible to implement on a real-world scenario. Comparing the global reward with types and the difference reward, one may be preferable depending on the specifics of the domain.

In this work we use the difference reward as a comparison metric, in order to ascertain the response of a well-formed reward both to the domain and to contrast the application of stereotyping. Though the formulation of the difference reward is simple, the counterfactual element $G(\vec{s}_{-j} + c_j, \vec{a}_{-}j)$ can be impossible to compute exactly in a real-world scenario. In contrast, our formulation of the system reward can be computed with sufficient communication, such as that offered by ADS-B and TIS-B. Encouragingly, our formulation of the system reward with strategy type inclusion shows some improvement over a basic system reward, and in some cases can even compete with the difference reward.

7. FUTURE WORK

This work provides a baseline for how agent typecasting can improve performance in complex reinforcement learning domains where types are accurately identified. This is representative of the UAS domain because of the technologies becoming available for communication between aircraft such as ADS-B and TIS-B. Other work which focuses on identification of types addresses the issue of automatically *developing* the strategy type based on observed information. By decoupling the learning performance from the identification of types, we are able to see the advantages of type identification without the unmeasured potential *mis*identification of typing methods.

Future work will extend our method to include automatic identification of types from observed behavior. This will allow us to take advantage of the typecasting process in domains in which strategies are either hidden or unknown to the neighboring agent. Also, in the future we hope to extend the use of types to more standard coordination problems such as the POI domain[19] to demonstrate its efficacy as domain-nonspecific technique.

8. REFERENCES

[1] A. Agogino and K. Tumer. Regulating air traffic flow with coupled agents. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Estoril,Portugal, May 2008.

[2] J. K. Archibald, J. C. Hill, N. A. Jepsen, W. C. Stirling, and R. L. Frost. A satisficing approach to aircraft conflict resolution. *Trans. Sys. Man Cyber Part C*, 38(4):510–521, July 2008.

[3] M.A. Azzopardi and J.F. Whidborne. Computational air traffic management. In *Proceedings of the 30th AIAA/IEEE Digital Avionics Systems Conference (DASC2011), Best Paper Session*, pages 1.B.5–1, 2011.

[4] N. Bard and M. Bowling. Particle filtering for dynamic agent modelling in simplified poker. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI)*, pages 515–521, 2007.

[5] N. Bard, M. Johanson N. Burch, and M. Bowling. Online implicit agent modelling. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, AAMAS '13, pages 255–262, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[6] S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2011.

[7] S. Barrett, P. Stone, S. Kraus, and A. Rosenfeld. Teamwork with limited knowledge of teammates. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, July 2013.

[8] D. Bertsimas and S.S. Patterson. The air traffic flow management problem with enroute capacities. In *May-June 1998, pp. 406-422*, 1998.

[9] C. Burnett, T. J. Norman, and K. Sycara. Bootstrapping trust evaluations through stereotypes. In *In Proceedings. of 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 241–248, 2010.

[10] C. Burnett, T. J. Norman, and K. P. Sycara. Stereotypical trust and bias in dynamic multiagent systems. *ACM TIST*, 4(2):26, 2013.

[11] M. A. Christodoulou and S. G. Kodaxakis. Automatic commercial aircraft-collision avoidance in free flight: the three-dimensional problem. *IEEE Transactions on Intelligent Transportation Systems*, 7(2):242–249, 2006.

[12] M. Colby and K. Tumer. Shaping fitness functions for coevolving cooperative multiagent systems. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 425–432, 2012.

[13] M. Colby and K. Tumer. Multiagent reinforcement learning in a distributed sensor network with indirect feedback. In *Proceedings of the 12th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2013.

[14] J. Denzinger and J. Hamdan. Improving modeling of other agents using tentative stereotypes and compactification of observations. In *IAT*, pages 106–112. IEEE Computer Society, 2004.

[15] FAA. Nextgen implementation plan, 2012.

[16] D. Fudenberg and D. K.Levine. Maintaining a reputation when strategies are imperfectly observed. *Review of Economic Studies*, 59(3):561–79, July 1992.

[17] D. Fudenberg and D. M. Kreps. Reputation in the simultaneous play of multiple opponents. *Review of Economic Studies*, 54(4):541–68, 1987.

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

[19] M. Knudson and K. Tumer. Robot coordination with ad-hoc team formation (extended abstract). In *Proceedings of the Ninth International Joint Conference on Autonomous Agents and Multiagent Systems*, Toronto, Canada, May 2010.

[20] A. Lazaric, M. Quaresimale, and M. Restelli. On the usefulness of opponent modeling: the kuhn poker case study. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 3*, AAMAS '08, pages 1345–1348, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[21] R. A. Paielli. Automated generation of air traffic encounters for testing conflict resolution software. *AIAA Journal of Aerospace Information Systems*, 10(5), May 2013.

[22] J. Samek, D. Sislak, P. Volf, and Michal Pechoucek. Multi-party collision avoidance among unmanned aerial vehicles. In *Intelligent Agent Technology, 2007. IAT '07. IEEE/WIC/ACM International Conference on*, pages 403–406, 2007.

[23] D. Sislak, P. Volf, M. Pechoucek, and N. Suri. Automated conflict resolution utilizing probability collectives optimizer, May 2011.

[24] D. Sislak, P. Volf, and Michal Pechoucek. Agent-based cooperative decentralized airplane-collision avoidance. *Intelligent Transportation Systems, IEEE*

Transactions on, 12(1):36–46, 2011.

[25] F. Southey, M. Bowling, B. Larson, C. Piccione, N. Burch, D. Billings, and C. Rayner. BayesâĂŹ bluff: Opponent modelling in poker. In *In Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI*, pages 550–558, 2005.

[26] T. Tarnopolskaya and N. Fulton. Synthesis of optimal control for cooperative collision avoidance for aircraft (ships) with unequal turn capabilities, 2009.

[27] L. F. Teófilo, Nuno Passos, L. P. Reis, and H. L. Cardoso. Adapting strategies to opponent models in incomplete information games: a reinforcement learning approach for poker. In *Proceedings of the Third international conference on Autonomous and Intelligent Systems*, AIS'12, pages 220–227, Berlin, Heidelberg, 2012. Springer-Verlag.