A Cooperative Multi-Agent Approach to Free Flight

Jared C. Hill
jayrodhill@gmail.com

F. Ryan Johnson
scovich@gmail.com

James K. Archibald
jka@ee.byu.edu

Richard L. Frost
rickf@ee.byu.edu

Wynn C. Stirling
wynn@ee.byu.edu

Electrical and Computer Engineering
Brigham Young University
Provo, UT 84602 USA

ABSTRACT

The next generation of air traffic control will require automated decision support systems in order to meet safety, reliability, flexibility, and robustness demands in an environment of steadily increasing air traffic density. Automation is most readily implemented in *free flight*, the segment of flight between airports. In this environment, centralized control is impractical, and on-board distributed decision making is required. To be effective, such decision making must be cooperative. Satisficing game theory provides a theoretical framework in which autonomous decision makers may coordinate their decisions. The key feature of the theory is that, unlike conventional game theory which is purely egotistic in its structure, it provides a natural mechanism for decision makers to form their preferences by taking into consideration the preferences of others. In this way, a controlled form of *conditional altruism* is possible, such that agents are able to compromise so that every decision maker receives due consideration in a group environment. Simulations demonstrate that reliable performance can be achieved with densities on the order of 50 planes per ten thousand square miles.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Control theory*; J.2 [**Computer Applications**]: Physical Sciences and Engineering—*Engineering*

General Terms

Algorithms, Design, Experimentation, Performance, Theory

Keywords

Air traffic control, free flight, satisficing, distributed control

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

1. INTRODUCTION

Recent advancements in navigation aids, communication technologies, and computing power are making possible a new concept of air traffic control (ATC), namely *free flight*, an operating environment where pilots are allowed to select their flight path in real time [1]. Ideally, free flight will result in an increase in efficiency without reducing the safety of air travel. The annual cost to the airline industry due to ATC-caused delays is estimated to be $5.5 billion [2]. As air traffic density increases, so too does the loss resulting from inefficiencies and the workload of air traffic controllers. If appropriately coupled with anticipated improvements in automation, free flight can reduce airline operating costs and reduce controller workload.

The most important issue in ATC is conflict detection and resolution. Much of the current research for automating ATC focuses on this problem, particularly in the area between airports. This environment is an appropriate candidate for automation because plane density is low compared with areas near airports and because rigid scheduling is unnecessary. Automation would give pilots, dispatchers, and airline managers more responsibility in air traffic control. Airline managers could plan their routes to minimize distances traveled. Airline dispatchers, who routinely observe weather patterns, would have the flexibility to suggest course changes that avoid dangerous wind and storms. Pilots could define their own cruising altitudes and speeds for greater fuel efficiency. The workload of air traffic controllers would be reduced considerably, as individual planes would be responsible for immediate threat avoidance. Air traffic controllers would continue to monitor aircraft separation, control the air space surrounding airports, and schedule runways.

Much of the recent research in ATC has focused on solutions for conflict avoidance that are based on fixed rule sets that dictate actions based on situational geometry. This approach can achieve arbitrarily good performance in a fixed scenario, such as two intersecting flows of aircraft (see Section 6.2), but acceptable performance in arbitrary situations cannot be guaranteed. Both centralized and distributed algorithms have been proposed to aid in conflict resolution, but previous schemes involve little if any cooperation. As we will show in this paper, a cooperative multi-agent algorithm can effectively and efficiently resolve most conflicts, even with high traffic densities.

A centralized ATC approach would have to deal with potentially thousands of aircraft spread over a vast geographic area to ensure resolution of all possible conflicts. Furthermore, as the number of planes increases, the complexity of conflict resolution grows and quickly becomes computationally intractable. Similarly, the possibility that a single failure could disable a significant portion of the centralized system creates a highly undesirable risk [3].

In contrast, a distributed system, where agents act cooperatively to maintain spatial separation, could be more robust in the face of ATC node failures. Each agent would need to focus only on a relatively small subset of the global problem. By taking into consideration only those aircraft that pose potential threats, the need for a fully global solution is eliminated. At a minimum, a suitable distributed solution should have several characteristics: (1) the aircraft must coordinate their decisions in avoiding collisions; (2) the avoidance maneuvers must be generally applicable and not limited to specific geometric situations; (3) the approach must be realistic and not oversimplify the problem; (4) the solution must scale to high traffic densities.

In order to implement a cooperative system, aircraft must be able to communicate. Fortunately, recent advances in communication systems make this possible. For example, the Mode S transponder is a selective broadcast messaging system that allows ground control to identify each plane using a unique 24-bit id. Originally developed in 1975, it achieved widespread use in the 1980's as a way of broadcasting a plane's altitude to ground control. In the 1990's, changes were made that allowed the plane to broadcast its GPS information as well. Expanding this capability to include heading, velocity, and destination in the broadcast is reasonable given current technology.

Our approach to this distributed cooperative problem is based on satisficing game theory [4], which provides the basis for effective algorithms for multi-agent decision making. Satisficing game theory differs from conventional game theory in that it allows individuals or groups to condition their own preferences on the preferences of others. This feature allows for an inherently cooperative approach to conflict resolution. By exchanging the information and goals of each agent, cooperation between agents can be realized even in a completely distributed system. This paper briefly introduces satisficing theory, recounts previous work in automating collision avoidance, presents our new satisficing solution, and reports the results of simulation involving a variety of scenarios.

2. SATISFICING

Agents based on satisficing theory have two roles, or *personas*. One persona focuses on achieving the fundamental goal of the decision problem, regardless of cost, while the other persona focuses on conserving resources and reducing costs without worrying about achieving the goal. Together, these two conflicting personas represent a decision maker, who must balance the desire to achieve a goal with the cost of doing so. We require two utilities to account for the preferences of these two personas. One utility characterizes the *selectability* of the options available to the decision maker; that is, the degree of effectiveness of the options with respect to achieving the goal (e.g., staying on a direct path to the destination) without worrying about the cost of doing so. The other utility characterizes the *rejectability* of the options; that is, the degree to which resources are consumed (e.g., time delays, exposure to hazards). These two utilities are normalized to be mass functions. In the multi-agent case, they are multivariate mass functions that permit the simultaneous characterization of a multi-agent decision system. Because the utilities have the same mathematical structure as probability mass functions, we may characterize relationships such as independence and conditioning that are analogous to the probabilistic notions. That is, they posses the same syntax as probabilities, but with different semantics.

Let p_S and p_R denote selectability and rejectability mass functions, respectively, and let p_{SR} denote the joint mass function when simultaneously taking into consideration both selectable and rejectable attributes. For an n agent system, the joint selectability/rejectability is a mass function with $2n$ variables of the form

$$p_{S_1 S_2 \cdots S_n R_1 R_2 \cdots R_n}(u_1, u_2, \ldots, u_n; v_1, v_2, \ldots, v_n),$$

where $S_1 S_2 \cdots S_n$ corresponds to the collection of selectability personas and $R_1 R_2 \cdots R_n$ corresponds to the collection of rejectability personas. The variables u_i, $i = 1, \ldots, n$ correspond to the options available to the ith agent *as viewed from the perspective of its selectability* and the variables v_i, $i = 1, \ldots, n$ correspond to the options available to the ith agent *as viewed from the perspective of conserving resources.* This *interdependence function* captures all of the decision-making considerations that may affect a multi-agent system, and its construction is an essential component of satisficing game theory. Its construction can often be guided by appealing to the influences that exist between agent personas.

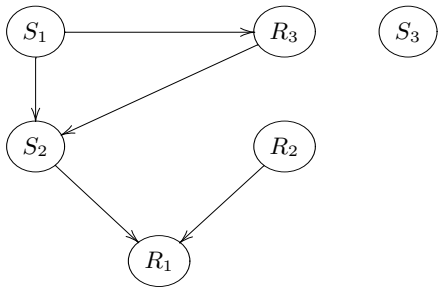

Figure 1: Praxeic network for three agent system

To illustrate, consider the directed acyclic graph displayed in Figure 1, which corresponds to a three-agent system (and hence with three selectability and three rejectability personas). In this system, the selectability of S_1 influences S_2 and R_3. Furthermore, R_3 influences S_2, and both S_2 and R_2 influence R_1. Finally, S_3 neither influences nor is influenced by any other persona. The interdependence function of this influence structure may be expressed as

$$p_{S_1 S_2 S_3 R_1 R_2 R_3} = p_{R_1|S_2 R_2} \cdot p_{S_2|S_1 R_3} \cdot p_{R_3|S_1} \cdot p_{S_1} \cdot p_{R_2} \cdot p_{S_3}$$

where arguments have been suppressed in the interest of brevity. The conditional mass functions represent the influence flows between nodes of the graph. For example, $p_{R_1|S_2 R_2}(v_1|u_2; v_2)$ expresses the amount of rejectability that Agent 1 should ascribe to option v_1, given that Agent 2 were to select option u_2 in interest of achieving its own goal and reject option v_2 on the basis of conserving resources. The

network presented in Figure 1 and its corresponding interdependence function is equivalent to a Bayesian belief network. These influence networks are called *praxeic* networks to emphasize that the linkages deal with the costs and benefits of taking action, rather than dealing with beliefs.

2.1 Multi-Agent Decision Making

For a single agent system, the notion of maximizing expected utility appears incontrovertible. When extending this concept to the multi-agent case, a standard approach is for all decision makers to maximize expected utility and to assume that all others will do likewise; the result is the Nash equilibrium. Such solutions, however, can be overly pessimistic. (In the well-known Prisoner's Dilemma game, the Nash solution yields poor results for both players). While utility maximization may be appropriate for adversarial situations, it does not naturally fit situations where the decision makers are disposed to cooperate, even though they may not be conflict-free. In such situations, decision makers must be able to weigh the tradeoffs between their own selfish interests and the overall well-being of the group. Satisficing game theory provides a mathematically rigorous way to make such compromises in a controlled way.

Let us consider a set of n decision makers, and let U_i denote the set of options available to agent i, $i = 1, \ldots, n$. A *satisficing game* is the triple $(n, U_1 \times \cdots \times U_n, p_{S_1 \cdots S_n R_1 \cdots R_n})$. To solve this game, we must compute the joint selectability and rejectability marginals as, respectively

$$p_{S_1 \cdots S_n}(u_i, \ldots, u_n) = \sum_{v_1 \in U_1} \cdots \sum_{v_n \in U_n} p_{S_1 \cdots S_n R_1 \cdots R_n}(u_i, \ldots, u_n, v_i, \ldots, v_n) \quad (1)$$

and

$$p_{R_1 \cdots R_n}(v_i, \ldots, v_n) = \sum_{u_1 \in U_1} \cdots \sum_{u_n \in U_n} p_{S_1 \cdots S_n R_1 \cdots R_n}(u_i, \ldots, u_n, v_i, \ldots, v_n). \quad (2)$$

Similarly, we then compute the individual selectability and rejectability marginals as

$$p_{S_i}(u_i) = \sum_{u_1 \in U_1} \cdots \sum_{u_{i-1} \in U_{i-1}} \sum_{u_{i+1} \in U_{i+1}} \cdots \sum_{u_n \in U_n} p_{S_1 \cdots S_n}(u_1, \ldots, u_{i-1}, u_i, u_{i+1}, \ldots, u_n). \quad (3)$$

and

$$p_{R_i}(u_i) = \sum_{u_1 \in U_1} \cdots \sum_{u_{i-1} \in U_{i-1}} \sum_{u_{i+1} \in U_{i+1}} \cdots \sum_{u_n \in U_n} p_{R_1 \cdots R_n}(u_1, \ldots, u_{i-1}, u_i, u_{i+1}, \ldots, u_n). \quad (4)$$

These unconditioned utilities can be computed from the interdependence function using methods applied to Bayesian networks, including Pearl's Belief Propagation Algorithm [5] and the Sum-Product rule for factor graphs [6].

The *jointly satisficing solution at caution level q* is the subset of all option vectors such that the joint selectability is at least as great as the caution level multiplied by the joint rejectability, that is

$$\Sigma_q = \{u \in U : p_S(u) \geq q \cdot p_R(u)\}.$$

The individually satisficing solutions for each agent are obtained from the marginal selectability and rejectability functions, yielding the *individually satisficing solutions*:

$$\Sigma_q^i = \{u_i \in U_i : p_{S_i}(u_i) \geq q \cdot p_{R_i}(u_i)\}.$$

The jointly satisficing set comprises all joint options that provide a group benefit that exceeds (as modulated by q) the group cost, and the individually satisficing sets comprise all individual options that provide an individual benefit that exceeds (as modulated by q) the cost to the individual. If the intersection of the two sets is empty, q can be gradually reduced to increase the number of options available in the individual satisficing sets. This allows the individuals to sacrifice self-interest for the benefit of the entire group.

Satisficing theory thus provides a natural mechanism for an agent to take into consideration what other agents *want* as well as what they can *do* when forming its own preferences. It therefore becomes possible to evaluate "what-if" scenarios involving multiple agents. The agent can detect potentially cooperative situations and respond accordingly, enabling it to consider approaches to problems that are simply not available with optimization-based decision-making. As a result, satisficing agents, which make no claim to optimality, can sometimes outperform "optimal" solutions to cooperative multi-agent problems.

For example, agents representing two aircraft on a collision course could infer that neither wants the collision to occur. Only one plane needs to make a heading change to avoid a collision, but if the planes do not cooperate, both may turn. Each satisficing agent determines its influence flow dynamically using some ranking metric, such as accumulated delay. Each agent computes the selectability and rejectability of its options using Equations 3 and 4 and makes its decision. This will result in only the lower ranking agent making a heading change, because the higher ranking agent's selectability is independent of the lower ranking agent. However, if the higher ranking agent were already planning to change heading (to get back on course, say), the other agent would not make a heading change. This is an example of multi-lateral decision-making improving the performance of the system.

2.2 Altruism

Cooperative behavior almost always requires some amount of altruism from the agents involved. This is difficult to model using traditional game theoretic approaches, which are based on exclusive self-interest. In the previous example, exclusive self-interest would dictate that each agent let the other take evasion action in order to avoid the cost of changing course, perhaps leading to both agents taking evasive action later to avoid the cost of a collision.

One common solution is to modify the agent's utilities so that it perceives altruistic acts as self-serving. This leads to one-sided *categorical altruism*. The altruistic agent will *always* behave altruistically, even when the agents it intends to help do not take advantage of the sacrifice.

In contrast, a satisficing agent is fully aware of its own interests, but also considers the preferences of the agents it interacts with. It displays *conditional altruism* by considering various factors before making sacrifices, including: (a) the level of trust in (or willingness to cooperate with) particular agents, (b) the likelihood that other agents will actually be benefited by a sacrifice, (c) the amount of flexibility afforded by an agent's circumstances at the time, and (d) whether or not the agent cares about the decision the

other agent makes.

Conditional altruism allows an agent to balance individual and group interests in well-defined ways. It also allows an agent to behave cooperatively while still protecting itself from exploitation — if it does not believe that cooperation is possible or likely, it will act in its self-interest.

3. PREVIOUS WORK

A variety of simulation environments and algorithms have been created by other researchers to address collision detection and resolution. Krozel *et al.* describe three algorithms, beginning with a centralized controller [7]. The system performs a look ahead, a prediction of what conflicts will occur in the next eight minutes if all planes were to continue on their current heading, and then groups the planes according to these interactions. Permutation sequences are established for all aircraft in a cluster. These are used to rank the planes, each plane needing only to avoid planes with a higher ranking. If no conflict free paths for all planes can be found, that permutation is considered to have failed and the next is processed. To limit the computation, a maximum of 100 permutations are processed if the cluster size is greater than four. If no sequence provides a conflict free path, the permutation with the least conflicts is selected.

The authors also present two decentralized strategies. Both algorithms use a "frontside/backside" approach in which a plane may go in front or behind a plane to avoid conflict. The "myopic" algorithm finds the closest conflict and chooses the side that requires the smaller heading change. If any new conflicts are created, they will be resolved during the next round. The "look-ahead" strategy checks to see if the more efficient maneuver will cause a conflict later on. If a conflict is detected, the algorithm checks the opposite side. If that also leads to a conflict, the algorithm starts from the original solution and increments 2 degrees until a conflict free path is found. Evaluated in a simulation of open air space with randomly generated planes, the algorithms are acceptable with moderate traffic densities, but no level of performance can be guaranteed with high densities because only a subset of the solution set is searched.

Dugail, Mao, and Feron introduce a geometric algorithm and prove that all conflicts can be resolved with bounded maneuvers [8]. The example scenario describes two flows of air traffic intersecting at right angles in the middle of the simulated air space. The initial separation in flows are uniformly distributed over a 5 to 10 mile distance. A 100 mile radius circle is drawn around the point of intersection. As each plane enters the circle, it is required to make one movement that ensures that it will not conflict with planes already in the circle. Once the initial maneuver is completed, the plane must continue to fly on a straight path until exiting the circle, ensuring it will create no new conflicts. This algorithm is specialized to this scenario, and it is uncertain how it would behave under other circumstances.

Resmerita, Heymann, and Meyer present a resource allocation approach to collision detection and resolution called DOR [9, 10]. It partitions the airspace into distinct "cells" that may only be occupied by one aircraft at a time, ensuring separation. These cells become the vertices of an undirected graph whose edges are paths between cells. Agent trajectories are directed, timed graphs that overlay the airspace graph. Before an aircraft enters the system, it registers itself with a central controller that maintains a list of all aircraft and their trajectories. The controller then distributes resources (timed access to cells) as planes request them, eliminating any need for agents to communicate with each other.

Conflict resolution becomes necessary when an agent desires a resource that has already been allocated. First, the agent will attempt alternate paths. If no acceptable path can be constructed, the controller will request that agents holding the disputed resources attempt to free them by choosing alternate paths of their own. The resulting cascade of resource shuffling has two possible outcomes: either the resources for an acceptable path can be freed, in which case the agent acquires them and proceeds, or it fails and the agent is not allowed to enter the system. This algorithm is optimal with respect to agent path quality (as determined by each agent). Each agent in the system follows one of its optimal paths; if an optimal path cannot be found, it does not enter. The system that results is in Nash equilibrium — no agent can unilaterally improve its path without causing a conflict. Because the algorithm is computationally intense, depends on a centralized controller, and requires every agent to have full knowledge of the system, it is not well suited for a distributed system.

Pallottino, Feron, and Bicchi describe a geometric approach to collision detection and resolution [11, 12]. Path planning is modeled as a set of linear constraints on either velocity or heading changes to be optimized with respect to total flight time and course deviations, respectively. The complexity of their algorithm is $O(n^2)$; it can be solved in several seconds with up to 15 agents involved in simultaneous conflict threats. They also prove that a decentralized adaptation of the algorithm is possible given a proper look-ahead distance. In this case the constraints imposed on an aircraft depend on its state with respect to other nearby craft. The authors consider the three-agent case, which requires eight different constraint formulations. The safety of the decentralized scheme is proven by considering worst-case maneuvering requirements during state transitions as other aircraft become visible. The algorithm can quickly become unwieldy as the number of constraint formulations that must be considered grows as $O(2^n)$ where n is the number of visible aircraft.

4. SYSTEM EVALUATION TECHNIQUES

Before presenting our algorithm for conflict resolution, we discuss evaluation metrics used to evaluate other schemes. Given the importance of safety in managing air traffic, the most important characteristic of conflict resolution schemes is clearly their effectiveness in avoiding safety incidents. This can be established empirically through simulation, or, ideally, proven using some analysis technique. Unfortunately, proofs are often possible only after making unrealistic assumptions that make the problem tractable, such as instantaneous lateral translation [13]. Some published algorithms are proven correct in a specific scenario, such as the conflicts occurring between two fixed flows of aircraft at a specific point [8], or the conflicts that can occur between a small number of agents (typically 2 or 3) [12]. Given these limitations, it appears that previously proposed algorithms are impractical for realistic ATC applications despite proofs of their correctness. The complexity inherent in automating ATC suggests that an optimal solution is intractable and unrealistic.

System efficiency is a useful and intuitive metric of solu-

tion quality [7]. It can be thought of as the degree to which an aircraft can fly the straight line path between successive waypoints. Deviation from the straight line path results in a time delay, as do changes in velocity. We define system efficiency as

$$SE = \frac{1}{N} \cdot \sum_{i=1}^{N} \left(\frac{t_i}{t_{d_i} + t_i} \right)$$

where t_i is the ideal flight time for the straight line path and t_d is the difference between actual and ideal flight time for each of the N aircraft. In general, as more planes are added to a system implementing free flight, the number of conflict avoidance maneuvers will increase and the system efficiency will be reduced.

Another metric of interest, although seldom addressed directly in evaluations of other proposed schemes, is *calculation time* [11]. The usefulness of an algorithm is limited by its ability to run in real time, even if it produces extremely desirable results. Clearly air traffic management is an application with hard real-time constraints.

Some researchers also define a metric of *stability*, a measure of the extent to which solutions to present conflicts lead to new conflicts in the future [7]. System stability is measured as the number of conflict alerts triggered when planes fly their nominal (straight-line) trajectories divided by the number of conflict alerts triggered when the planes are implementing the conflict resolution scheme. A conflict alert occurs when a plane detects a new conflict with the projected path of another plane in its vicinity. The metric gives insight into the likelihood of a conflict resolution algorithm overloading the system. A full comparative analysis of the stability of our satisficing algorithm is ongoing and will be described elsewhere.

5. THE ALGORITHM

The algorithm presented here is by no means the only solution or even the best solution that could be devised using a satisficing approach, but our studies show that it does give good results.

Each second (or *round*), every agent is required to choose one of five available options. The option set consists of: a 5 degree turn to the left, a 2.5 degree turn to the left, no turn, a 2.5 degree turn to the right, and a 5 degree turn to the right. These turning constraints limit the aircraft to a minimum of 72 seconds per 360 degree turn. At the beginning of every decision round, each plane is assumed to have a current list of information from all other planes within fifty miles. Conflict detection and cooperative avoidance utilizes this broadcast information, which includes the current heading, current velocity, current altitude, next waypoint (desired direction), and current delay time.

As mentioned in Section 2, an important part of satisficing is defining influence flows that adequately and succinctly describe the system. Traditional Bayesian network tools are useful for static influence flows; because the set of neighboring planes changes frequently, ATC applications require dynamic influence flows. Using simple criteria such as distance to destination, delays already accumulated, and time in the air, aircraft rank each other so that the dynamic influence flows are acyclic. After the planes are ranked, the list is pruned to decrease the total calculations required. For best performance, it is important to consider only those agents that can influence the current agent. First, all planes that have a lower rank than the current plane are removed. Next, all planes with which the currently considered plane has no projected conflicts are removed from the list.

The selectability of an option is defined as the extent to which that option allows the agent to achieve a goal. In our algorithm, selectability is based on heading and the selectabilities of higher ranking agents. An option that allows an agent to follow its nominal route more closely has a higher selectability. Options that take the agent further from its destination naturally have lower selectabilities.

The formulation of selectability allows for cooperation. Each agent examines the selectability of higher ranking agents. If one of them strongly prefers a direction that will lead to a separation violation, the current agent lowers its selectability for those headings that would result in a conflict. This allows for the conditional altruism previously discussed. Note that the lower ranking agent does not have to sacrifice and take a less selectable heading just because a higher ranking plane *might* choose a conflicting option. The preference of the higher ranking plane is also considered.

The rejectability of an option is defined as the cost of choosing that option. In our ATC scenario, rejectability is determined by possible conflicts. If an option will lead to a possible conflict with another plane, that option is assigned a higher rejectability. Rejectability values are scaled based on the severity of the incident (near miss or collision) and also by the projected time to incident.

ATC must be viewed as a risk-averse environment. While any separation failure is unacceptable, planes must ultimately arrive at their destinations, so successful algorithms must balance safety and efficiency. In our algorithm, each agent chooses the option that maximizes the difference between its selectability and rejectability. Thus, agents prefer options that are low risk; options leading to possible conflicts will receive low selectability and high rejectability, thus making them highly unattractive.

Two separate algorithms with these same influence flows were created. The "full" model uses all the information available to create influence networks for each visible plane. Although necessarily incomplete (since it cannot see all influences on other planes), the model is able to make reasonable approximations of the preferences of higher-ranking planes. However, this approach can become computationally intractable with high traffic densities. A "simplified" model was developed with reduced computational overhead. In this model, higher ranking planes are grouped together based on their desired heading change (their most selectable option), and then each group is treated as a single agent. Although the efficiency and safety margins are slightly lower than those of the full model, the simplified model is better suited for very dense traffic.

By design, both algorithms are independent of the geometry or spatial relationships of any fixed scenario. They have been simulated using a variety of scenarios as reported below in Section 6.

5.1 A Simple Example

Consider the scenario in Figure 2 with two agents on a collision course. Both agents are currently on a heading that will take them directly to their destinations. Agent one, denoted A_1, is 5 minutes behind schedule and headed to point Q. Agent two, A_2, is on schedule and headed to

point P. This means that A_1 outranks A_2, so when A_2 calculates its selectability, it will take A_1's selectability into account.

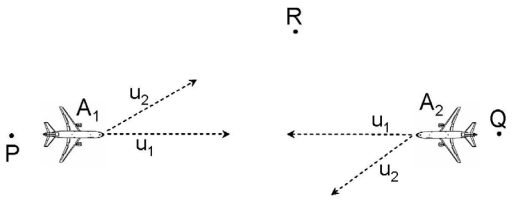

Figure 2: Two agents, each with two options

When the two planes see each other and conflict alerts are triggered, each will calculate the selectability and rejectability of its two options (limited in this example). A_1's selectability, p_{S_1}, is simple. Flying straight, u_1, will have the highest selectability since that will take it in the direction of its destination. A_1's rejectability, p_{R_1}, will be a uniform distribution because it outranks A_2. Therefore, A_1 will choose to go straight and rely on A_2 to resolve the incident.

A_2's decision requires the following to be computed for each of its available options:

$$p_{S_2}(u_i) = \sum_{u_n \in U} f_S(u_i, u_n) \cdot p_{S_1}(u_n)$$

$$p_{R_2}(u_i) = \sum_{u_n \in U} f_R(u_i, u_n)$$

where u_i is the choice A_2 is considering, U is A_1's option set, and $p_{S_1}(u_n)$ is A_1's selectability of the option u_n. The function $f_S(u_i, u_n)$ has value one if A_1 choosing u_n and A_2 choosing u_i will result in no incident and is zero otherwise. Similarly, the function $f_R(u_i, u_n)$ gives a value in the range of zero (no incident) to one (immediate incident) depending on the time until a projected incident assuming A_1 chooses u_n and A_2 chooses u_i. Of A_2's two options, u_2 will be assigned the largest selectability, and u_1 will be assigned the largest rejectability. As a result, A_2 will choose to turn left and the collision will be averted.

Consider the changes if A_1 has just completed a conflict avoidance maneuver and its destination is actually point R. In this case, its most selectable option will be u_2, and A_2 will no longer see its option u_1 as resulting in a conflict. Thus, A_1 will turn and A_2 will fly straight. A_2's altruism in deferring to A_1 is conditional on A_1's choosing an option that causes a conflict.

6. RESULTS

Our simulation environment is similar to many used by other researchers [7, 8, 13]. All aircraft are constrained to fly at the same altitude. Changes in altitude, although an efficient and effective conflict resolution maneuver in reality, are not allowed; this enriches the problem by requiring fewer aircraft to create a dense airspace. All aircraft fly at the same velocity, 500 miles per hour. Heading changes are assumed to take place instantaneously. Aircraft must maintain a minimum separation of 5 miles. If two planes come within five miles of each other, a near miss is recorded. If the separation distance is 300 feet or less, a collision is recorded.

The simulator is designed to test completely distributed systems in a variety of scenarios. A central thread is used to simulate the communication framework that would be distributed among the aircraft in a real system. The communication thread compiles and disseminates all information to the planes, each of which runs on its own thread. Each agent thread receives the information about its neighbors, then makes a decision based on that information. The plane then updates its heading and position and sends the updated information back to the communication thread to be distributed to the other planes.

6.1 Random Scenario

This scenario, based on a model in [7], reflects open airspace with no obstacles other than other aircraft. The radius of the world is 100 miles. Planes start at a position on a concentric circle with a 120 mile radius. Planes are removed when they arrive at their destination, a specific point on the inner circle. (The 20 mile buffer zone ensures that planes are not involved in a conflict immediately after generation.) The starting and destination points are chosen at random, thus creating an even distribution over the world. Simulated plane densities ranged from less than one plane per 10,000 square miles (5 planes total) to more than 50 planes per 10,000 square miles (80 planes total).

Table 1 shows the average results of running four simulations of the "full" model at each density, each lasting fifty minutes. At the maximum traffic density studied by Krozel *et al.* [7] of 25 aircraft per 10,000 square miles, the highest system efficiency reported (for their centralized approach) was approximately 97.2 percent. At that same traffic density, the best decentralized scheme had an efficiency of approximately 94.7 percent; no safety incidents were reported. The satisficing approach results in significantly better system efficiency as the traffic density increases.

Planes	Incidents	Efficiency (%)
80	.25/26	97.1
70	.25/20	97.7
60	.25/11	98.0
50	0/6	98.1
45	0/5	98.6
40	0/4	99.1
30	0/2	99.1
20	0/0	99.6

Table 1: Results for random flight scenario

6.2 Intersecting Flows

In this scenario, introduced in [8], two orthogonal traffic flows intersect at a point. The radius of the simulated world is 100 miles. One stream of aircraft starts at a fixed point on the southern end of the world, and the other stream starts at a fixed point on the western end of the world. Planes are generated slightly over 5 miles apart to allow them to maintain separation while maneuvering. The two flows intersect at a fixed point; all aircraft must cross the other traffic flow to reach their destinations. Something similar to this geometric arrangement is likely to arise in free flight as popular routes appear dynamically. The lead planes of each stream were generated in such a way that both would reach the point of intersection at the same time.

Table 2 shows average results of running five simulation runs for each agent over a five hour period. The constant flow scenario produces a new plane on each flow every forty seconds, corresponding to a distance gap of over five miles. In the remaining scenarios, occasional gaps were inserted randomly, creating streams of consecutive planes of length given by a gaussian random variable $\mathcal{N}(\mu, \infty)$. As the results show, the presence of even a few gaps in the otherwise continuous flow increases both safety and efficiency. The near miss radius in this simulation was set at five miles.

Scenario	Flights	Incidents	Efficiency (%)
Constant Flow	97	0/29	98.2
$\mu = 25$	97	0/26	98.8
$\mu = 20$	94	0/24	97.4
$\mu = 15$	93	0/6	99.8
$\mu = 10$	87	0/4	99.7

Table 2: Results for intersecting flights

Figure 3 is a screen shot of the simulation of the satisficing algorithm. Notice the "waves" that are formed as the planes maneuver to safely cross the intersection point. We note that the hard coded algorithm for this scenario as introduced in [8] produces a pattern very similar to that seen in the figure. Although it was not specifically tuned to produce these results in this situation, the satisficing algorithm exhibits very similar behavior to that of an algorithm hardwired for this intersecting scenario. This emergent behavior underscores the promise of satisficing as a solution to ATC management.

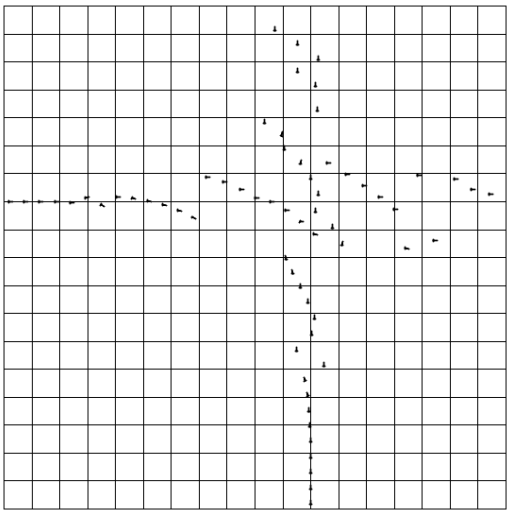

Figure 3: Intersecting traffic (10 mile grid)

6.3 Same Point - Same Time Scenario

This scenario tests robustness more than it models reality. Once again an open air space is created. A predetermined number of planes are evenly distributed around the outside of a circle with a radius of 25 miles. The destination for these planes is the exact opposite side of the circle from their originating position. This means all planes would prefer to travel through the center of the circle, and all are set to arrive there at the same time. Table 3 shows the results using the "simplified" algorithm, with the near miss radius set at five miles. The number of calculations required was too great to run the full model within reasonable time constraints.

Planes	Incidents	Efficiency (%)
12	0/0	97.9
14	0/1	97.3
16	0/1	95.5
18	0/3	94.4
20	0/7	93.6
22	0/8	91.4
24	0/8	86.9
26	0/14	85.6
28	0/14	84.5
30	0/18	84.4
32	0/19	85.7

Table 3: Results for the choke point scenario

Figure 4 shows a series of screen shots of the scenario with 32 planes attempting to navigate the same point. Similar to section 6.2, interesting wave-like patterns are created as the planes attempt to navigate the dense traffic without collisions.

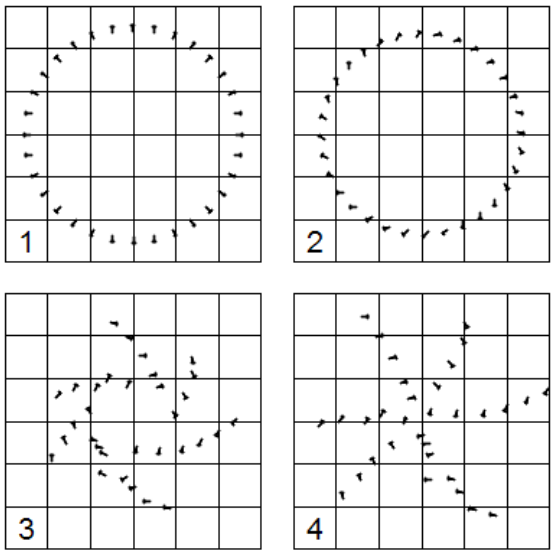

Figure 4: Choke point (10 mile grid)

6.4 Airport Scenario

This scenario is an attempt to model actual airspace more realistically. A predefined number of airports are generated at the outset. A circle with a radius of 40 miles is drawn around each airport. All airplanes coming or going from the airport will fly at an altitude of 5,000 feet. Airplanes that are merely passing over the airport fly at an altitude of 30,000 feet. This is done so that planes flying through the area will not interfere with planes during takeoff and landing. When a plane leaves the 40 mile safety radius after

takeoff, it is then assigned an elevation of 30,000 feet and must interact with the other aircraft at cruising altitude. As expected, planes that are generated at the same airport, and then have the same destination airport, tend to fly on the same path. This creates small "highways" in the air that act similarly to intersecting traffic flows. The following results are modeled with a one mile near miss metric.

Agent	Flights	Incidents	Delay (sec)
Full Model	1110	0/0	50
Simplified Model	1106	0/2.5	42

Table 4: Results for airport flight scenario

7. CONCLUSION

As air traffic densities continue to increase, decentralized decision-making algorithms will become more important. Such algorithms will give pilots more flexibility and responsibility. Due to the cooperative nature of air traffic management, an algorithm designed to work in free flight must not be purely egotistical. Satisficing theory offers an attractive framework within which multi-agent systems can be designed because it provides a natural mechanism for cooperation. Aircraft that implement a satisficing algorithm can consider the desires of others while making their own decisions, allowing for conditional altruism. Our simulation experiments included a variety of scenarios, and the results demonstrate the flexibility and performance offered by the satisficing algorithm, obtained without sacrificing safety.

8. REFERENCES

[1] National Research Council Panel on Human Factors in Air Traffic Control Automation, C. D. Wickens, A. S. Mavor, R. Parasuraman, and J. P. McGee, Eds., *The Future of Air Traffic Control: Human Factors and Automation*. National Academy Press, 1998.

[2] T. S. Perry, "In search of the future of air traffic control," *IEEE Spectrum*, vol. 34, no. 8, pp. 18–35, August 1997.

[3] L. Geppert, "Lost radio contact leaves pilots on their own," *IEEE Spectrum*, pp. 16–17, November 2004.

[4] W. C. Stirling, *Satisficing Games and Decision Making: With Applications to Engineering and Computer Science*. Cambridge University Press, 2003.

[5] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[6] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, 2001.

[7] J. Krozel, M. Peters, K. D. Bilimoria, C. Lee, and J. S. B. Mitchell, "System performance characteristics of centralized and decentralized air traffic separation strategies," *Fourth USA/Europe Air Traffic Management Research and Development Seminar*, 2001.

[8] D. Dugail, E. Feron, and K. Bilimoria, "Stability of intersecting aircraft flows using heading change maneuvers for conflict avoidance," *American Control Conference*, 2002.

[9] S. Resmerita, M. Heymann, and G. Meyer, "A framework for conflict resolution in air traffic management," in *IEEE Conf. on Decision and Control*, 2003, pp. 2035–40.

[10] S. Resmerita and M. Heymann, "Conflict resolution in multi-agent systems," in *IEEE Conference on Decision and Control*, 2003, pp. 2537–42.

[11] L. Pallottino, E. M. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 3–11, 2002.

[12] L. Pallottino, A. Bicchi, and S. Pancanti, "Safety of a decentralized scheme for free-flight ATMS using mixed integer linear programming," in *American Control Conference*, 2002.

[13] Z.-H. Mao, E. Feron, and K. Bilimoria, "Stability and performance of intersecting aircraft flows under decentralized conflict avoidance rules," *IEEE Transactions on Intelligent Transporation Systems*, vol. 2, pp. 101–109, 2001.