A survey of multi-agent systems used for air traffic control

Sütő Evelyne

December 22, 2019

Contents

1 Introduction 1
1.1 About agent oriented systems 1
1.2 Why use agent oriented systems for air traffic control 2

2 MAS applications for ATC 3
2.1 Air traffic flow management 3
2.1.1 Agents 4
2.1.2 Agent actions 4
2.1.3 Agent learning 5
2.1.4 Agent specific rewards 5
2.1.5 About the experiments 5
2.2 Conflict resolution 5
2.2.1 Distributed problem solvers for ATC 5
2.2.2 Autonomous Agents for Air-Traffic Deconfliction 7

3 Conclusion 8

Abstract

Intelligent air traffic flow management is one of the fundamental challenges facing the Federal Aviation Administration (FAA) today. FAA estimates put weather, routing decisions and airport condition induced delays at 1,682,700 h in 2007,resulting in a staggering economic loss of over $41 billion, and in 740 million gallons of fuel being wasted.

In order to solve this problem NASA proposed the Free Flight system, which would allow pilots to choose their own routes, altitude and speed and essentially gives each aircraft the freedom to self-optimise. The Free Flight system's complex architecture makes it ideal to use multi-agent systems in their implementation. The feature of high autonomy and flexibility of agents makes them suitable for air traffic control systems. Free Flight systems depend on having two main components to solve traffic control and scheduling, Air traffic flow management and Conflict resolution.

We will present different agent oriented implementations for air traffic flow management and conflict resolution. We will also explore some of their experiments and its successes for the tasks at hand.

1 Introduction

1.1 About agent oriented systems

Agent-oriented techniques (AOT) offer an approach aimed at supporting the whole software development process. In particular, they enable analysis, design, and implementation of large and complex systems by providing abstraction levels that make it simpler and more natural to deal with the scale and complexity of problems in these systems. The goal of AOT is to handle all phases with a single, uniform concept, namely that of agents. Systems modelled by a collection of agents are called multi-agent systems (MAS). [Burmeister et al., 1997]

AOT can use techniques provided by distributed systems (like sharing resources and synchronization), but they complement these techniques by making subsystems more autonomous and enabling them to actively coordinate their activities instead of being coordinated by design. AOT extend object-oriented techniques in that the analysis, design and realization of complex systems are performed on a higher level of abstraction, and agents are active, concurrent objects embedded in a society. Agents are characterized by four important elements: internal states, message types in communication, protocols for coordination, and roles within the society that they take part in. [Burmeister et al., 1997]

When implementing an agent we have many architectures available to choose from which will in the end define these four elements as well.

Logic based agents

In a logic based architecture we try to model the environment with the help of logical formulas and their actions will be equivalent with logical deduction or theorem proving when interacting with this environment. These agents can be considered deliberate agents. The internal state is assumed to be a database of formulae of classical first -order predicate logic. These formulas can be thought of as the belief system of the agent about its environment. While the idea of logic based agents sounds good the calculational complexity that is needed to implement such theorem provers makes them very challenging to use in real-time environments. [Weiss, 1999]

Reactive agents

Reactive agents have been introduced as a solution for the calculational complexity challenges introduced with the approach of logic based agents. Agent' s decision-making is realised through a set of task accomplishing behaviours; each behaviour may be thought of as an individual action. Each of these behaviour modules is intended to achieve some particular task. Many behaviours can "fire" simultaneously. We need a mechanism to choose between the different actions selected by these multiple actions. Since these agents react to the changes of the environment we need to have a precise implementation of the environment which is generally hard to achieve. This reactive behaviour makes it hard to include the learning into their algorithms. [Weiss, 1999]

Belief-desire-intention agents

BDI architectures are practical reasoning architectures. The basic components of a BDI architecture are data structures representing the beliefs, desires, and intentions of the agent, and functions that represent its deliberation (deciding what intentions to have e.g. deciding what to do) and means-ends reasoning (deciding how to do it). Intentions play a central role in the BDI model: they provide stability for decision making, and act to focus the agent s practical reasoning. A major issue in BDI architectures is the problem of striking a balance between being committed to and over-committed to an intentions: the deliberation process must be finely tuned to its environment, ensuring that in more dynamic, highly unpredictable domains, it reconsiders its intentions relatively frequently- in more static environments, less frequent reconsideration is necessary. [Weiss, 1999]

Layered agents

Given the requirement that an agent be capable of reactive and pro - active behaviour, an obvious decomposition involves creating separate subsystems to deal with these different types of behaviours. Layering represents a natural decomposition of functionality, it is easy to see how reactive, pro-active, social behaviour can be generated by the reactive, pro-active, and social layers in an architecture. We may choose from Vertical and Horizontal layering as well. When choosing horizontal architecture we can easily model many behaviours with different layers, but since these layers are active concurrently we need to implement a synchronisation of these layers in order to achieve coherent output. Vertical architectures are well suited for problems that need to flow through multiple steps, where each layer can model one of the needed steps that are built on each others success, because of this the architecture is very intolerant to failures, because each layer depends on the correctness of its ancestor. [Weiss, 1999]

1.2 Why use agent oriented systems for air traffic control

As we mentioned in 1.1 one of the properties of agents is its high autonomy. This autonomy makes them well suited to use in complex systems as individual components. It is also easier to use them in distributed systems because of their autonomy and their ability to flexibly interact with other systems and the environment.

The domain of traffic control has many properties which makes it a good candidate for multi agent systems such as: geographically and functionally distributed elements, subsystems which have a high degree of autonomy and highly dynamic environment and elements. [Burmeister et al., 1997]

The growing air traffic control has been a problem since the 1990s. The air traffic management systems were mostly relying on human resources using little automation. One of the first improvements were the instrumental flight rules, which helped the air control being more effective, but still was relying on human resources. With the appearance of GPS and several types of satellite navigation systems such as ADS-B (Automatic dependent surveillance broadcast) today the automation of air traffic control was beginning to be possible to automatise the air traffic management. These technologies also make possible an innovative concept for air traffic management called *Free flight*. *Free Flight* allows pilots to choose their own routes, altitude and speed and essentially gives each aircraft the freedom to self-optimise. Aircraft exibility will be restricted only in congested airspace, or to prevent unauthorised entry of special use airspace (such as military airspace). The goal is to create a decentralised system by using on-board equipment which allows aircrafts to share some of the workload, such as navigation, weather prediction and aircraft separation, with ground controllers. In *Free Flight* airspace, each aircraft is surrounded by two virtual zone, the protected zone and the alert zone. A conflict or loss of separation between two aircraft occurs whenever the protected zones of the aircraft overlap. In *Free flight* each aircraft communicates state and intent data to each other using an ADS-B datalink for conflict prediction, and then coordinate to resolve potential conflicts. State and intent data could be uncertain. Coordination among the aircraft is in the form of manoeuvres which are finite sequences of flight modes like heading, altitude or speed changes for each aircraft. The main thrust of our conflict resolution algorithms is to verify that a manoeuvre successfully resolves the conflict by computing the set of initial conditions for which the manoeuvre is safe, meaning that the protected zones of the conflicting aircraft do not overlap. [Tomlin et al., 1998]

As we can see from the description of the *Free flight system* air traffic control management is a well suited domain for multi agent systems, where each aircraft can act as an agent, each has their internal state and goal, they communicate through different types of messages, each needs to follow the same protocol and rules and they need to cooperate to resolve confligts that might arise.

2 MAS applications for ATC

2.1 Air traffic flow management

Current air traffic management relies on a centralized, hierarchical routing strategy that performs flow projections ranging from 1 to 6 hour. Therefore, the system is not only slow to respond to changes, but is also at the limit of its capacity. It is difficult to see how the current systems and algorithms can accommodate the expected increase in air traffic. There is therefore a strong need to explore new, distributed and adaptive solutions to the air flow control problem. A multiagent approach is an ideal fit to this distributed problem where the interaction among the aircraft, airports and traffic controllers renders a predetermined centralized solution. However, since our current implementations use centralized archotectures implementing a multiagent system would mean a big effort. [Agogino and Tumer, 2012]

Air traffic flow management has to have an efficient system which takes into consideration fairness (for different airlines), reliability, safety and adaptability. Having such a complex system becomes very challenging to test. The researchers use a simulation tool called FACET (Future ATM Concepts Evaluation Tool), a physics based model of the US airspace that was developed to accurately model the complex air traffic flow problem. It is based on propagating the trajectories of proposed flights forward in time. FACET can be used to either simulate and display air traffic or provide rapid statistics on recorded data. The effectiveness of the system is then calculated based on the linear combination of the amount of congestion and air traffic delay.

$$G(z) = (B(z) + \alpha C(z))$$

where B(z) is the total delay penalty for all aircraft in the system, and C(z) is the total congestion penalty. The multi agent system developed in [Agogino and Tumer, 2012] uses adaptive agents taking independent actions that maximize the system evaluation function. In this model, each agent first chooses an action. The results of all the agents actions are then simulated in FACET. From the simulation all congestion and lateness values are observed, a system reward is computed and agents compute their appropriate rewards. These rewards are then used to modify the agents control policies, which are then used to choose the next action.

Let's see the basic components of the system.

2.1.1 Agents

One of the most obvious choice for agents could be an aircraft itself since aircrafts have all the important information about the flight plan. However, this would create a massive multi agent system. For this reasons the researchers use an alternative option of assigning agent to individual ground locations throughout the airspace called 'fixes'. Each agent is then responsible for any aircraft going through its fix.

2.1.2 Agent actions

One obvious method would be for the fixes to bid on aircraft, affecting their flight plans. This simple action, however, would result in an unpredictable flight traffic changes and would be very challenging to define the bidding process in a way which could optimise the flights.

The researchers instead define three actions for the fixes.

- Miles in trail (MIT) Agents control the distance aircraft have to keep from each other wile approaching a fix.
- Ground delays An agent controls how long aircraft that will eventually go through a fix should wait on the ground. Imposing a ground delay will cause aircraft to arrive at a fix later. With this action, congestion can be reduced if some agents choose ground delays and others do not, as this will spread out the congestion.
- Rerouting An agent controls the routes of aircraft going through its fix, by diverting them to take other routes that will (in principle) avoid the congestion.

2.1.3 Agent learning

As in all reinforcement learning systems, the agents have as a collective goal to maximize the system performance. In order to achieve this each agent has their own reward function which they will maximize. The agents use the ϵ-greedy learning algorithm. After taking action a and receiving reward R an agent updates its value for action a, V(a).

$$V(a) = (1 - \lambda)V(a) + \lambda R$$

where λ is the learning rate. At every time step, the agent chooses the action with the highest table value with probability 1 - ϵ and chooses a random action with probability ϵ.

2.1.4 Agent specific rewards

The main difficulty is to choose a reward function for the agents which line up with the overall system reward but it's still sensitive to the agent's personal actions.

The chosen reward type was the difference reward.

$$D_i = G(z) - G(z - z_i + c_i)$$

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. 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 agents contribution to the system performance.

2.1.5 About the experiments

In their first experiment the researchers use the MIT value to control the separation between the aircrafts by choosing between 11 actions each representing different distances between 0 to 100 miles. In the experiments there are up to 20 agents used. They compare multiple reward function such as system reward, difference reward and difference reward estimates and come to the conclusion that the difference reward is the best reward function. The ground delay controlling method has a similar result to the MIT value change method. The same is true for flight rerouting methods as well.

As we can see from [Agogino and Tumer, 2012] multi agent systems can be incorporated to air traffic flow management sysems or they could replace the currently used centralized system.

2.2 Conflict resolution

2.2.1 Distributed problem solvers for ATC

As we already mentioned it can be quite natural to see an aircraft as an autonomous system which can represent an agent in a multi agent system.

Each aircraft has only a limited sensory input, so its knowledge of the world is never complete and it must continually gather information as it moves through an airspace. Information may be accumulated either by sensing or by communication. Agents can use messages to exchange information or instructions as well. Each agent has a specific goal in mind, move from point A to B to get to its destination. These goals can create shared conflicts when multiple aircraft get through the same airspace at the same time and they violate the minimum separation requierements between two aircrafts, so they enter each other's protected zone. These conflicts must be detected and resolved in a timely manner. The resolution of such conflicts needs negotiation and cooperation of the agents. Agents may gather information about a shared conflict, evaluate or interpret the information, develop a plan to avoid a projected conflict, or execute such a plan. Agents may be more or less appropriate for such roles, depending on their current processing load, state of knowledge, and their spatial constraints. [Cammarata et al., 1988]

In [Cammarata et al., 1988] the researchers use distributed problem solvers to achieve the conflict resolution task. The system contains fours subsystems in total. Each system implements the information distribution policies where the information should be sent to other aircraft selectively (no broadcasting), without waiting for a request, without expecting an acknowledgement, and without repeating the information a second time. This assumes that the communication on the system is error free. Each aircraft is allowed to send a maximum of 5 messages per 15 seconds of time.

The organizational policy is implemented by tree of the four systems which concludes the task centralization. The fourth system will adopt the task sharing policy. Under task centralization, the agents involved in any given conflict task will choose one of their number to play most of the roles. In particular, one agent will perform the evaluation role (do all the evaluation of the potential conflicts between aircraft), the plan-fixing role (attempt to devise a plan-fix to dissolve the entire conflict), and the actor role (act on the new plan). The selected agent is required to modify only his plan to resolve the conflict; thus the remaining agents perform no planning or actions. Instead, having agreed on the choice of a replanner, they adopt passive information-gathering roles, merely sending their intentions (plan) to the selected agent. However, if the selected agent is unable to resolve the entire conflict, he requests another agent to replan. This process continues until all conflicts are resolved or a solution cannot be found. In order to choose the agent that should be responsible for the planning the aircraft use only directly transmitted information such as position, heading and speed, as a result they do not manage to choose the most equipped one for this task. The chosen agent can also be the most spatially constrained one. Where the constraint factor is an aggregation of such considerations as the number of other nearby aircraft, fuel remaining, distance from destination, and message load. So in the end the agent with the most degree of freedom will be responsible to modify its plan. It is also possible to select the most knowledgeable agent, the one that has the most information about the other agents, so they can take into consideration the other agents intentions when replanning. The task sharing policy tries to improve the disadvantages of the centralized policy which only considers one agent for the actor and planner role as well. In the task-sharing policy two rounds of negotiation are necessary, one to determine the plan fixer and one to determine the actor. The planner might be the most knowledgeable agent while the actor could be the least constrained agent. This policy is communication-intensive and may be inappropriate when communication channels are unreliable or costly. From these policies the most performing has been the least constrained one because it was able to find solutions for the conflicts more easily. The most knowledgeable has outperformed the least constrained in very complex conflict resolutions, but its overall performance was intermediate. The task sharing policy which has the advantages of both least constrained and most knowledgeable agents, was found to be harder to implement when the sub tasks or the agents were not separable enough.

2.2.2 Autonomous Agents for Air-Traffic Deconfliction

While [Cammarata et al., 1988] is mostly concerned about providing a theoretical background of how this kind of system could be implemented, in contrast [Pěchouček et al., 2006] provides a working prototype of a successful multi agent system for air traffic deconfliction. They also take a step further in taking into consideration autonomous agents (aircraft run by auto-pilots).

The aircraft are modeled by agent containers hosting several agents. This agent is self-interested and it is responsible for creating a flight plan which contains the specified way-points and executes the plan by performing the flight. The aircrafts have multiple information specific zones: **Communication**, **Alert**, **Safety** and **Collision** zones.

The communication zone has the biggest radius, the aircraft can send and receive packages to any aircraft that has entered its Communication zone. The alert zone defines the operation range of the radar onboard the aircraft. When another aicraft enters the alert zone, the aircraft gets notified about its position and its flight code. The safety zone encapsulates the area around an aircraft that other aircraft are not allowed to enter in order to minimize the mutual influence of the aircraft movements, e.g. airspace turbulence. The collision zone defines the critical contact area.

The deconfliction can happen by cooperative or non-cooperative negotiation. When an Airplane A enters the alert zone of Airplane B, Airplane A tries to get a safe connection for communication with Airplane B, if possible. If the connection was established successfully, the agent A subscribes for a local area flight plan of the aircraft B. B sends an update to the subscriber every time it changes its own flight plan. When the pilot agent A receives an update from the pilot agent B, it executes the collision detection process on its own flight plan and the received one. If detection is detected agent A and B must modify their flight plan until deconfliction is achieved. The agents use a rule-based approach for modifying flight plans. When communication cannot be established the deconfliction happens in a non-cooperative manner. In this case the reasearchers used game theory to implements the agent's flight plan updates. The agent should change its flight plan in a way that guarantees minimal conflicts in the future, to determine the other aircraft's possible future position we can use information about the type of aircraft.

The collision of aircrafts can be modeled with the angle between their direction vectors at each timestamp. Generally we can have 4 types of collisions:

- head-on collision, in this case the airplanes avoid each other by both of them turning to the right.
- rear collision, there are two subcases: i) the front aircraft is faster airplanes do not change their current flight plans; ii) the rear airplane is faster it has to change its flight plan so that it turns to the right and passes the front airplane without endangering it.

- side collision (right), the airplane B has higher traffic priority. The aircraft A needs to slow down its speed so that it reaches the collision point later than plane B.
- side collision (left), the airplane B has lower traffic priority. The aircraft A changes its flight plan by increasing its flight speed so that it passes the collision point before the airplane B. The airplane A only accelerates as much as needed.

The agents use the collision types and rules to update their flight plans independently. After the update the other aircrafts are notified about the changes and the collision detection is verified again.

The flight plans can be described by waypoints and timestamps at which the airplane arrives to that specific waypoint. Using these the flight planner generates the path consisting of segments and elements. These elements can be straight, turns (vertical or horizontal) or spirals. When creating a flight path of these elements the planner considers the timestamps at which we should fly through each elements and the avoid overflying areas as well.

For more information about the implementation details refer to the original paper. [Pěchouček et al., 2006]

3 Conclusion

Intelligent air traffic flow management is one of the fundamental challenges facing the Federal Aviation Administration (FAA) today. On a typical day, more than 40,000 commercial flights operate within the US airspace, and the scheduling allows for very little room to accommodate deviations from the expected behaviour of the system. FAA estimates put weather, routing decisions and airport condition induced delays at 1,682,700 h in 2007, resulting in a staggering economic loss of over $41 billion, and in 740million gallons of fuel being wasted. [Agogino and Tumer, 2012]

The staggering numbers from above come from a study that has been published in 2012, which means that most likely we are facing even bigger problems in the present. As a result, investing into resolving this problem is still very important. The number of different elements that are affecting each decision that the system has to make and has to adapt to makes it very challenging to solve it in a centralised manner since we have to face the shortcomings of the currently available hardware as well.

It comes as no surprise that using decentralised systems is the most natural solution to the lack of more performant computational power. Multi agent systems offer several ways to adapt rapidly to changes and to model complex interactions between systems through different types of policies. Agents can decide by negotiation, coooperation or even through competition. Each of these policies have their own advantages and disadvantages which the literature still needs to explore in many ways. We only explored two problems from this domain in this paper, Air traffic flow management and Conflict resolution, but I think these are a good starting point in this domain so they serve as a good introduction for those trying to get introduced to multi agent systems in aviation in general.

References

[Agogino and Tumer, 2012] Agogino, A. K. and Tumer, K. (2012). A multiagent approach to managing air traffic flow. *Autonomous Agents and Multi-Agent Systems*, 24(1):1–25.

[Burmeister et al., 1997] Burmeister, B., Haddadi, A., and Matylis, G. (1997). Application of multi-agent systems in traffic and transportation. *IEE Proceedings-Software*, 144(1):51–60.

[Cammarata et al., 1988] Cammarata, S., McArthur, D., and Steeb, R. (1988). Strategies of cooperation in distributed problem solving. In *Readings in Distributed Artificial Intelligence*, pages 102–105. Elsevier.

[Pěchouček et al., 2006] Pěchouček, M., Šišlák, D., Pavlíček, D., and Uller, M. (2006). Autonomous agents for air-traffic deconfliction. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1498–1505. ACM.

[Tomlin et al., 1998] Tomlin, C., Pappas, G. J., and Sastry, S. (1998). Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Transactions on automatic control*, 43(4):509–521.

[Weiss, 1999] Weiss, G. (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press.