Final Project: Air Traffic Controller AI Project

Sai Janjanam
100938621
COMP 4106
saijanjanam@cmail.carleton.ca

Air Traffic Controller AI Project

PROBLEM DOMAIN

The Air Traffic Controller AI project addresses the problem of applying artificial intelligence techniques to air traffic control. A fully developed ATC AI would need to carry out the following tasks:

1. Coordinating with other ATCs
2. Navigating aircrafts
3. Issuing altitude clearance
4. Maintaining aircraft separation

There are two types of air traffic controllers: en-route and airport. En-route ATCs are positioned along certain checkpoints to manage congested air ways. An air traffic controller maintains a region of airspace called a sector. ATCs coordinate the entry or exit of an aircraft with the neighbouring sectors. These sectors can be horizontally beside each other or vertically above or below each other. An aircraft is not allowed to leave a sector until the upcoming sector confirms responsibility. Navigation involves approving a pre-determined flight route and making any necessary diversions to avoid obstacles such as bad weather. An ATC would need to issue altitude clearances to allow the aircraft to climb or descend altitude. An altitude clearance by an en-route air traffic controller is usually issued to avoid obstacles or maintain separation. Apart from altitude change, separation is also maintained by issuing holding patterns or change in speed. An airport ATC controller needs to manage landing and takeoff of multiple planes. This can be difficult as many aircrafts can be scheduled to land and take off sequentially and obstacles such as aircrafts performing an intersecting holding patterns. The AI would need to issue landing clearances, attitude clearances or if there is a delay, holding patterns. This project will focus on applying artificial intelligence techniques to the en-route air traffic controllers.

MOTIVATION FOR THE PROBLEM

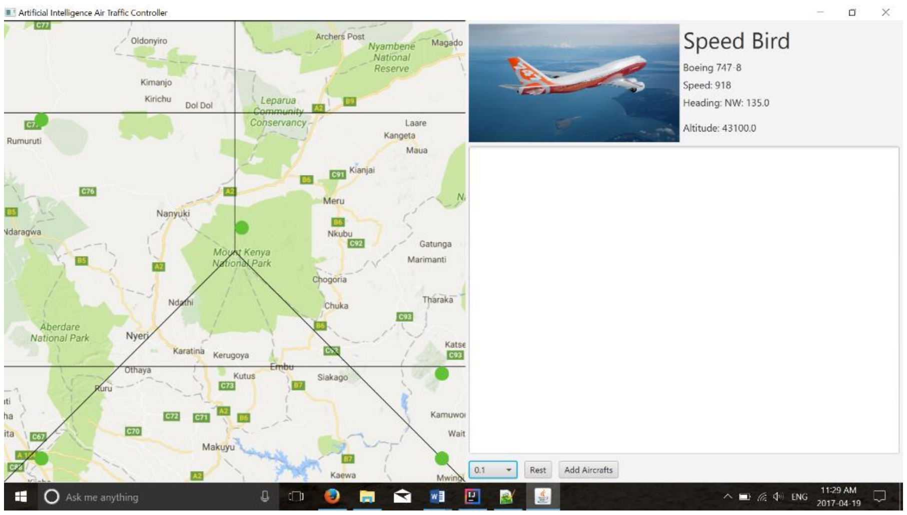
As the demand for air travel is becoming greater each year, the number of aircrafts will increase. Currently, most of the ATC duties are performed by human air traffic controllers. The increase in flights will put a great deal of stress on the system, and as we all know, humans are prone to make mistakes under heavy stress. These mistakes are very dangerous because they put passengers' lives in stake. Therefore, by implementing an AI to handle air traffic control automation, some of the stress can be reduced and help decrease the possibility of accidents.

PROJECT DESIGN

Radar Representation

The Air Traffic Controller AI project was built using Java and Javafx. To simulate air traffic, an artificial radar was created with aircrafts following arbitrary routes. These routes are predefined and intersect with other aeronautical paths. Each aircraft has distinct characteristics such as speed, altitude, fuel capacity, priority etc. Aircrafts are represented as a green circle on the radar.

These circles can be clicked to see the aircraft's characteristics. Along with the aircraft's information you can see all the commands issued to the aircraft by the ATC. Any commands made by the AI air traffic controller can be seen on the radar.


Artificial Intelligence Air Traffic Controller

Speed Bird

Boeing 747-8

Speed: 918

Heading: NW: 135.0

Altitude: 43100.0

Figure 1: Air traffic control radar representation

Artificial Intelligence

Sector Evaluation

The air traffic controller AI implements sector generation and analytical evaluation of the sector through various heuristics. A snapshot of a sector is the state. The model for state evaluation was proposed by D.A. Spencer who wrote the article "Applying Artificial Intelligence Techniques to Air Traffic Control Automation" The Lincoln Laboratory Journal, Volume 2, Number 3 (1989). The model is illustrated in *Figure 2*. The AI performs an evaluation for the current snapshot of a sector periodically. Each evaluation returns a score. The score is the summation of all the scores returned from the critics. A critic, as proposed by D.A. Spencer, is responsible for evaluating a certain aspect of the sector such as aircraft separation, fuel efficiency, weather, aircraft performance, arrival time, priority etc. Each critic as no knowledge of the other critics.

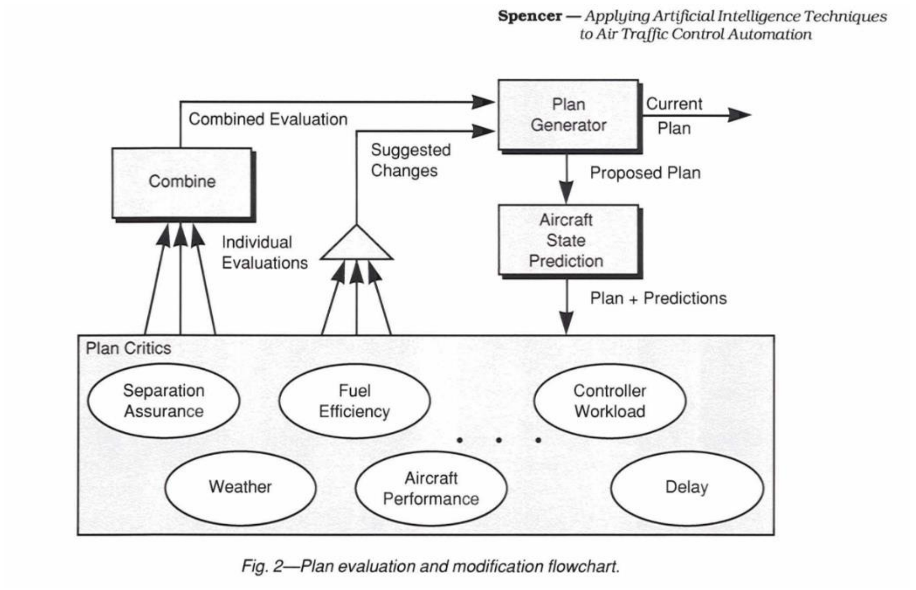


Figure 2: Sector evaluation model

This model was used because it can incorporate various variables into its evaluation. Different ATC locations have different variables that affect their sector and some variables would be given higher priority than others. Therefore, this model of evaluation provides a modular way for developers to add or modify critics based on the location.

These are the following critics that were implemented into the project:

Separation Critic

The separation critic evaluates the distance between each aircraft in x, y and z directions. According to ATC standards, at least 1000ft of separation is required between aircrafts below 29000ft. This rule was represented in the code by giving each aircraft a personal airspace of 1000ft width, height, and depth. Upon evaluation, each aircraft is given a temporary evaluation airspace of 10km in all directions. The evaluation airspaces acts as a buffer zone for the aircraft to make any necessary corrections. When these evaluation airspaces intersect, it signals a possible collision of two aircrafts. By considering the aircrafts' magnitude (speed) and heading (direction), the precise point of possible collision is determined. The critic generates a list of all aircrafts that will be involved in possible collisions and returns a score based on the size of the list.

Arrival Time Critic

Each aircraft has a certain arrival time, therefore a greater priority is given some aircrafts more than others. When an aircraft of higher priority is given a lower speed than other aircrafts with lower priority the score is incremented.

Intelligent Search Tree

Once an anomaly (change in evaluation score) in the sector is detected, A* search is used to find the most optimal sector with the least amount of changes. The state space used in the search tree consists of the sector. The only varying parts of the state are the aircrafts, more specifically the aircrafts' characteristics. Currently the only possible changes to the aircrafts include increasing speed, decreasing speed, increasing altitude and decreasing altitude. This is its production system. The production system is only applied to the aircrafts returned from the getConflictAircrafts() method call from the critic objects. The heuristic used to evaluate each state is the same evaluation method used to detect anomalies in the sector. The goal state is achieved when the evaluation returns a 0. Once a goal state is achieved, a set of commands are created to apply to the currents sector. The A* search was used because it finds the optimal solution with the least changes. Therefore, in real world scenarios, the pilot doesn't need to follow a series of commands to correct the situation. As mentioned in the article D.A. Spencer, one of limitation of his search implementation is that it only had a depth of 1. Therefore, some solution that require multiple changes are lost. By using A* search the AI can think like an ATC by building upon intermediate solutions.

RESULTS / FURTHER IMPROVEMENTS

By examining the results from moderate testing, majority of the conflicts are resolved by altitude changes. This solution fixes the problems but is very undesirable because changing altitude requires more work rather than changing the speed. A possible improvement would be to have priority for different commands. For example, a speed change would have a greater priority than altitude change. Due to possible restrictions in some areas of air traffic control, not all possible commands might be valid. Thus, the production system could be suggested by the critics. This will allow developers to create custom production systems based on the local ATC requirements. The critics were able to asses the sectors as expected and returned accurate scores. A future improvement would be to increase the number of critics and actually base the AI upon real-world ATC locations. In terms of testing, a possible improvement is to obtain and use real world data. By obtaining real world data of aircraft positions and paths, the AI can be tested and improved based upon its intended environment of use.

REFERENCES

"Applying Artificial Intelligence Techniques to Air Traffic Control Automation" The Lincoln Laboratory Journal, Volume 2, Number 3 (1989). D.A. Spencer

https://www.ll.mit.edu/publications/journal/pdf/vol02_no3/2.3.13.artificialintelligence.pdf

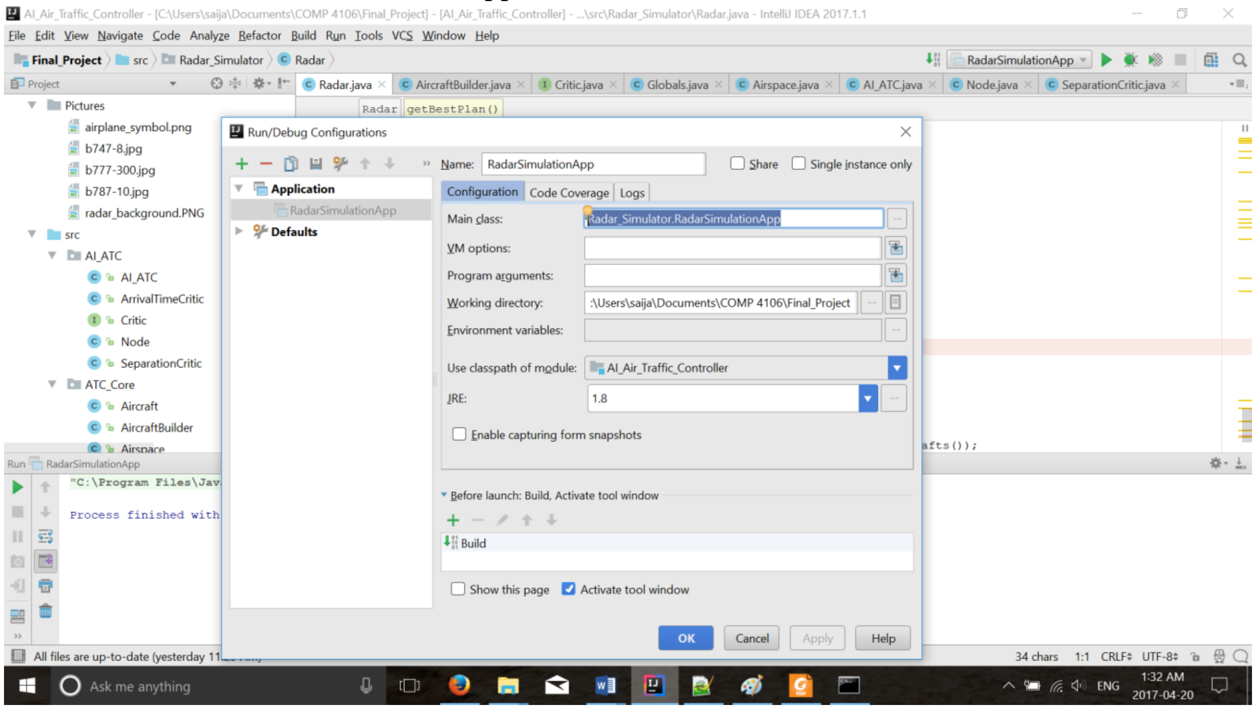
APPENDIX

Requirements to run the code

1. Java SDK 1.8
2. IntelliJ IDE

Instructions:

1. Import the source code to the IntelliJ IDE.
2. Make sure the run/debug configurations are set properly where the Main class should be "Radar_Simulator.RadarSimulationApp"



Run/Debug Configurations

Name: RadarSimulationApp [ ] Share [ ] Single instance only

Application > RadarSimulationApp

Configuration

Main class: Radar_Simulator.RadarSimulationApp

VM options:

Program arguments:

Working directory: :\Users\saija\Documents\COMP 4106\Final_Project

Environment variables:

Use classpath of module: AI_Air_Traffic_Controller

JRE: 1.8

[ ] Enable capturing form snapshots

Before launch: Build, Activate tool window

Build

[ ] Show this page [✓] Activate tool window

3. Then run the project.