

Creation of an Air Traffic Simulation Using Agent Based Modelling and Embedded Statistical Analysis

Sam Eberspacher
TJHSST Computer System Lab
2007-2008

June 10, 2008

Abstract

As the skies over the United States become increasingly crowded, airports in the United States are increasingly stressed to adapt to this increased demand. The goal of this project is to visually represent the strain on airports and passengers as a variety of problems generate record delays. By using agent based modelling, along with real air traffic information, this simulation may accurately predict the proliferation of delays through out airports in the United States.

1 Introduction

The purpose of this project is to visually represent the proliferation of a delay throughout a system of airports. By using techniques such as agent based modelling, the simulation will predict actual delays with decent accuracy. Additionally by repeating the simulation multiple times, the simulation generates increasingly accurate results as the number of trials approached infinity. While a simulation such as this would take a human enormous amounts of time, a computer may be able to run a simulation of 24 hours in a matter of minutes. Due to the scale of the problem, efficiency will be key for the computer to run the simulation in a timely matter.

2 Development

2.1 Agent Based Modelling

In order to simulate such a large system, this project will use a technique known as agent based modelling. The development of a system using agent based modelling is key for the success of the project. Each agent must interact with other agents in the system in the most realistic way possible in order to generate the most accurate results. One benefit of the agent based modelling is that parameters for interaction between agents define the overall behaviour of the system. This allows the programmer to work on much smaller problems with the agent in order to alter the overall system.

2.2 Embedded Statistical Analysis

Embedded statistical analysis is a done when real time statistics are needed in a simulation. The program uses data at each time step to readjust statistical values for the desired population. These statistics are useful when determining if the system is able to handle the introduction of new agents or constraints such as weather data.

2.3 Geocoding

Geocoding is a process by which a formatted address such as 6560 Braddock Rd. Alexandria, VA 22312 is converted to a longitude and latitude. This process is important when dealing with map information that is displayed on a computer. The computer is unable to relate formatted addresses so longitudes and latitudes are used to generate accurate relationships about location. This project uses the process to determine the location of each airport and accurately plot the distance between airports.

2.3.1 Google Geocoder

Geocoding is a complex process which involves a significant amount of computing power relative to web requests. Due to these requirements for geocoding many companies charge a small fee per request. Alternativeley, there are some companies which offer geocoding free of charge but wiht limitation on the number of geocoder requests per day. I found that Google offered free geocoding with a maximum of 5000 requests per day.

2.3.2 Request formatting

In order to interact with the Google geocoder, each request was done through an HTTP request sent to Google servers. These servers then interpret the parameters in the URL of the request and return the output specified by the user. The parameters in a request are as follows:

- q - The formatted address to be geocoded
- output - The desired output format (xml, kml, csv, or json)
- key - Google Maps API key

Sample Request (Key removed for privacy reasons)

`http://maps.google.com/maps/geo?q=BWI+airport&output=csv&key=API_KEY`

2.4 Software

Computer languages used in this project.

1. Java was used for the bulk of the project including all classes and the display of information.
2. Python was used for interacting with the Google geocoder.

2.5 Procedure

2.5.1 Static Information

To start the project, initial classes were written to display static information. The Airport class and Simulation class were drawn up and coded, leaving room for additional modifications which would take place at later stages of development. The first step was to normalize the data points in an effort to maximize used space and increase efficiency. A randomly generated set of coordinates was generated and used for this part of testing. The normalizing equation for used in this project is given by the following equations:

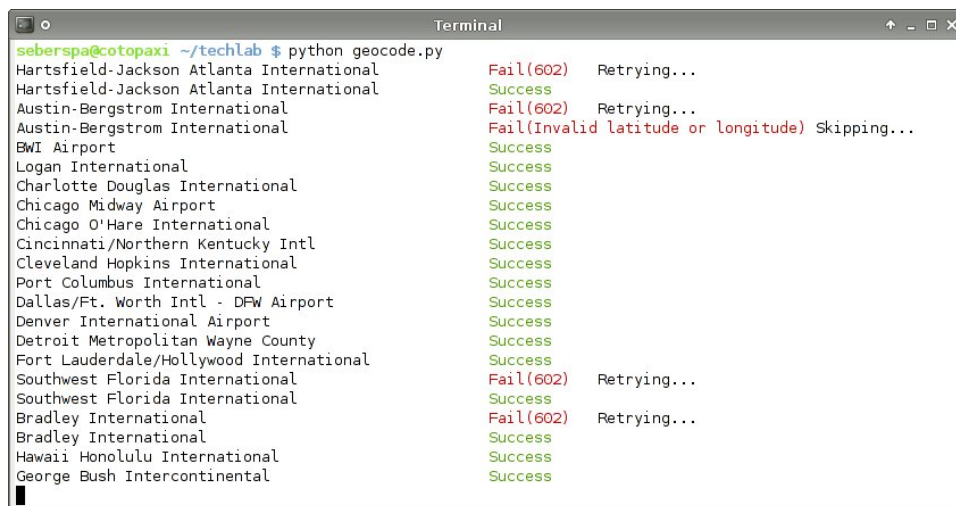
$$x_{norm} = \frac{x_i - x_{min}}{x_{range}} \cdot width_{screen}$$

$$y_{norm} = \frac{y_i - y_{min}}{y_{range}} \cdot height_{screen}$$

2.5.2 Geocoding

The second stage of the process was to geocode the airport locations and provide a list of latitudes and longitudes which would make up the airport map. In order to make the most robust program possible, a Python script was written to automate the geocoding process. The script reads from a list of airports and formats an address for geocoding. The script then sends the HTTP request to the Google geocoder and parses the returned file. The parsed data is then validated to verify that the location is in the United States and written to a file.

The formatting of the address itself was very important to the geocoding process. Since the data obtained did not contain street addresses, a straight address request was not possible. However, it was determined that using the three letter code for the address yielded desirable results. Some airports though, still failed to meet validation standards (due to conflicts with other country codes) so a second format was needed. The airport code concatenated with “airport” gave the second best results and the Python code was then optimized to maximize the number of valid airports.



```
Terminal
seberspa@cotopaxi ~/techlab $ python geocode.py
Hartsfield-Jackson Atlanta International      Fail(602) Retrying...
Hartsfield-Jackson Atlanta International      Success
Austin-Bergstrom International               Fail(602) Retrying...
Austin-Bergstrom International               Fail(Invalid latitude or longitude) Skipping...
BWI Airport                                  Success
Logan International                          Success
Charlotte Douglas International              Success
Chicago Midway Airport                       Success
Chicago O'Hare International                 Success
Cincinnati/Northern Kentucky Intl           Success
Cleveland Hopkins International              Success
Port Columbus International                  Success
Dallas/Ft. Worth Intl - DFW Airport          Success
Denver International Airport                  Success
Detroit Metropolitan Wayne County             Success
Fort Lauderdale/Hollywood International      Success
Southwest Florida International              Fail(602) Retrying...
Southwest Florida International              Success
Bradley International                         Fail(602) Retrying...
Bradley International                         Success
Hawaii Honolulu International                Success
George Bush Intercontinental                 Success
```

Figure 1: Screenshot of geocoder script output

2.5.3 Waypoints, Collision Detection, and Collision Avoidance

Due to the large amount of air traffic which is needed to create an accurate model, a waypoint system needed to be implemented to minimize collisions of aircraft. Initially, each plane sets the next waypoint to the destination airport, which would be the shortest travel path. The plane will only modify it's waypoint stack if a collision is detected and a new waypoint is pushed on to the stack.

The process for deconfliction was based on the algorithm developed by students at the Czech Technical University and operates as follows:

- Plane A enters the alert range of Plane B
- Determine the type of collision and execute the appropriate action
 - Head on collision - both planes turn x degrees right and place a waypoint path.
 - Rear collision - Since the rear airplane is faster, the rear airplane turns to the right and places a waypoint path around the slower plane.
 - Side collision - If the angle of incidence is less than 90 degrees, treat similarly to head on collision otherwise treat similarly to rear collision. The difference is that Plane A must turn to the opposite direction of the flight path of Plane B to minimize delay.

In order to ensure that the planes were correctly avoiding each other, a simple test environment was written into the simulator. Two planes would be constructed and a collision would occur at a random angle. By watching the resulting actions of the airplanes, it was confirmed that the algorithm was working correctly, and it was implemented on the much larger simulation.

2.5.4 Path Tracing and Arc Rendering

In order to better visualize the data, both path tracing and arc rendering were included in the program. Path tracing is a process by which previous locations of the aircraft remains rendered to the screen. This allows the user too better visualize where the plane came from as well as the speed and direction of the airplane. Arc rendering supplements path tracing, by helping the user see where the plane is going. Arc rendering is done by falsely

rendering the location of the aircraft on the two dimensional viewing plane to give the illusion of altitude. The progression of the aircraft is based on an arc from the previous waypoint to the next waypoint on the stack. This illusion allows the user to better project where the plane will land.

2.5.5 Embedded Statistics

The implementation of embedded statistics was done mostly through new variables added to the Airport class. Due to the statistical properties of the mean and standard deviation, the mean and standard deviation of the whole simulation could be calculated without polling all of the agents for a second time. The mean and standard deviation were calculated using the following formulas.

$$\mu = \sum_{k=1}^n \mu + \mu_k$$
$$\sigma = \sum_{k=1}^n \sqrt{\sigma^2 + \sigma_k^2}$$

3 Results

The final version of the project is a functional representation of air traffic over the United States. Unfortunately due to time constraints, a few features of the project were not implemented. A glitch in the rendering process of the Java platform causes the planes to be rendered slightly off course, generally South-East of their true position. As a result, path tracing and arc rendering were not included in the final build and the statistics were not displayed real-time.

However, the simulation is still functional and several discernable patterns were noticed by simply watching the simulation run. The presence of clusters (as seen in the following screenshot) shows that the system is not yet at capacity and that any delays should be minimized on average. Airports launch and recover planes at a constant rate, so the presence of clustering shows that some airports have launched all of the planes under their control. Another possible explanation for the clustering is the collision detection algorithm. Since the algorithm does not take into account final destination,

it is possible for a plane to be pushed slightly off course until the angle of incidence to the next waypoint incurs a different action.

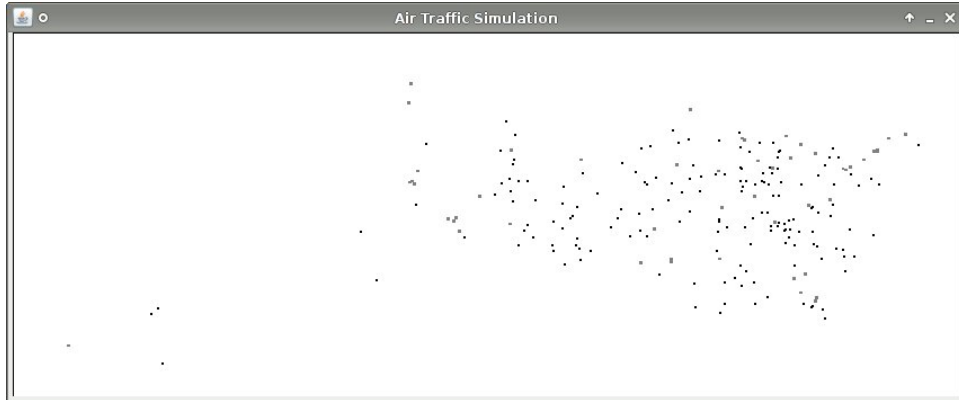


Figure 2: Screenshot of the final simulation interface

Another pattern that was noticed is the low concentration of planes which travel across the North-West. This pattern shows that some planes may reach their destination faster by traveling across this relatively unused space. By avoiding highly trafficked areas of the system, the plane reduces the probability of using the collision avoidance system, which can result in a more efficient flight path.

4 Future Work

In general projects like this involve several people working full time for a number of years. Given the amount of time allotted for work, the progress made on this project is about normal but also has a lot of room for additional work. A few tests and modifications that would work well with this simulation include modification of the collision avoidance algorithm, implementation of multiple agent types, and weighted destination selection.

Modification of the collision avoidance algorithm would be a terrific idea for future work. As the system approaches capacity the collision avoidance algorithm becomes very important. By taking the statistics from one run of the simulation and comparing them to the modified version, it is possible to

determine which algorithm is more efficient given the air traffic infrastructure of the United States.

Implementation of multiple agent types is also another way which this project could be adjusted in the future. By implementing multiple agent types, the simulation can more accurately model real traffic over the United States. A few possible agent types that could be implemented are military aircraft, freight aircraft, and personal aircraft. These three types of aircraft all represent large portions of the total air traffic in the United States, but all three of these aircraft types operate under different protocols than commercial passenger aircraft.

Another area for future work on this project would be to weight the airports when they are selected for destinations. This greatly increases the realism of the model because some airports are travelled to more frequently than others. Additionally, some airports are used as stepping stones on the way to other airports in a route. In order to implement this, significant research would need to be done into the routing that major airlines use to maximize efficiency. However, the benefits the simulation would gain are quite important to creating a realistic simulation.

References

- [1] Koblin, Aaron, comps. Flight Patterns. University of California at Los Angeles. 21 Nov. 2007 <<http://users.design.ucla.edu/~akoblin/work/faa/>>.
- [2] Tumer, Kagan, and Adrian Agogino, comps. Distributed Agent-Based Air Traffic Flow Management. University of Oregon, UCSC, NASA Ames Research Center. 18 Jan. 2008 <<http://web.engr.oregonstate.edu/~ktumer/ktumer-aamas07.pdf>>.
- [3] Benson, Kirk C., David Goldman, and Amy R. Pritchett, comps. Applying Statistical Control Techniques to Air Traffic Simulations. Georgia Institute of Technology. 18 Jan. 2008 <<http://portal.acm.org/citation.cfm?id=1161734.1161979>>.
- [4] Michal Pěchouček and David Šišlák and Dušan Pavlíček and Miroslav Uller Autonomous agents for air-traffic deconfliction. Czech Technical University. 30 March 2008 <<http://portal.acm.org/citation.cfm?id=1160633.1160925>>.

- [5] Seungman Lee, Amy Pritchett, David Goldsman. Hybrid agent-based simulation for analyzing the national airspace system. Proceedings of the 33rd conference on Winter simulation, December 09-12, 2001, Arlington, Virginia. 18 May 2008 <<http://portal.acm.org/citation.cfm?id=564272>>.
- [6] Pritchett, A. R., van Paassen, M. M., Wieland, F. P., and Johnson, E. N. 2003. Aerospace vehicle and air traffic simulation. In Applied System Simulation: Methodologies and Applications, M. S. Obaidat and G. I. Papadimitriou, Eds. Kluwer Academic Publishers, Norwell, MA, 365-389. 18 May 2008 <<http://portal.acm.org/citation.cfm?id=966031>>.