Multi-agent Path Finding Air Traffic Control User Manual

Joshua Argent

April 23, 2018

Contents

1 Installation **3**

2 Using The System **3**

2.1 Launching a Scenario . . . 3

2.2 Adding an Aircraft . . . 3

2.3 Loading a Schedule . . . 3

2.4 Enable Logging . . . 4

2.5 Launching Command Line Options . . . 4

3 User Interface **5**

4 Source Code **6**

4.1 Aircraft Simulator . . . 6

4.2 ATC Automation . . . 6

1 Installation

The program distribution contains a bin folder which comes with a pre-compiled .jar file called 'AI.jar' and a scenario file which contains airspace data for Scotland. The bin folder also contains a couple of example schedule files which you can use.

To install the program, simply extract the bin folder to any suitable location on your hard drive.

2 Using The System

2.1 Launching a Scenario

To launch the program with the provided scenario file;

1. Launch a new instance of Windows command line.
2. Execute the following command: *java -jar AI.jar scotland.json*

This will launch the program and load the scenario file for Scotland. The program will not launch unless a scenario file path is provided. The system has it's own command line interface which provides basic help for controlling the system.

2.2 Adding an Aircraft

With the program running, use the *ADD* command in the command line interface to add an aircraft. The command looks like this:

```
ADD [callsign] [type] [start] [goal] [altitude]
```

Callsign a unique identifier for the aircraft (e.g. 'BAW001')

Type The type of the aircraft (see list of aircraft types)

Start The name of the start waypoint for the aircraft

Goal The name of the goal waypoint for the aircraft

Altitude The start and goal altitude for the aircraft to fly at, in thousands of feet

Be aware that sometimes there can be a delay of up to 10 seconds when adding new aircraft to a busy scenario.

2.3 Loading a Schedule

A schedule defines a list of aircraft along with a schedule start time. This allows for repeatable tests to be carried out, using the same schedules. The schedules are written in JSON and are fairly self explanatory (example schedules are included). To load a schedule, use the following command:

```
SCHEDULE -load [filename]
```

Filename the path to the schedule file

2.4 Enable Logging

The system has an ability to record flight data including aircraft tracks, the changes in agent priority and logging details of any conflicts. To enable logging, use the following command:

```
LOGGING -on folder
```

Folder the path to a folder which will contain all the log files

2.5 Launching Command Line Options

The system has various optional command line arguments which can be used to automate tests. The full command looks like this:

```
java -jar AI.jar [scenario] [heuristic] [priority] [intermediate node spacing] [reservation time] [window size] [schedule] [logging] [-exit]
```

Scenario the scenario file to load

Heuristic the heuristic system to use:

- manhattan - Manhattan distance heuristic
- straight - Straight line distance heuristic
- true - True distance heuristic

Priority the priority system to use:

- agitation - Agent agitation priority system
- fifo - First-in first-out priority system
- furthest - Furthest distance to goal priority system
- closest - Shortest job first priority system

Intermediate Node Spacing the spacing (in NM) between intermediate nodes

Reservation Time The amount of time agents reserve nodes for (in minutes)

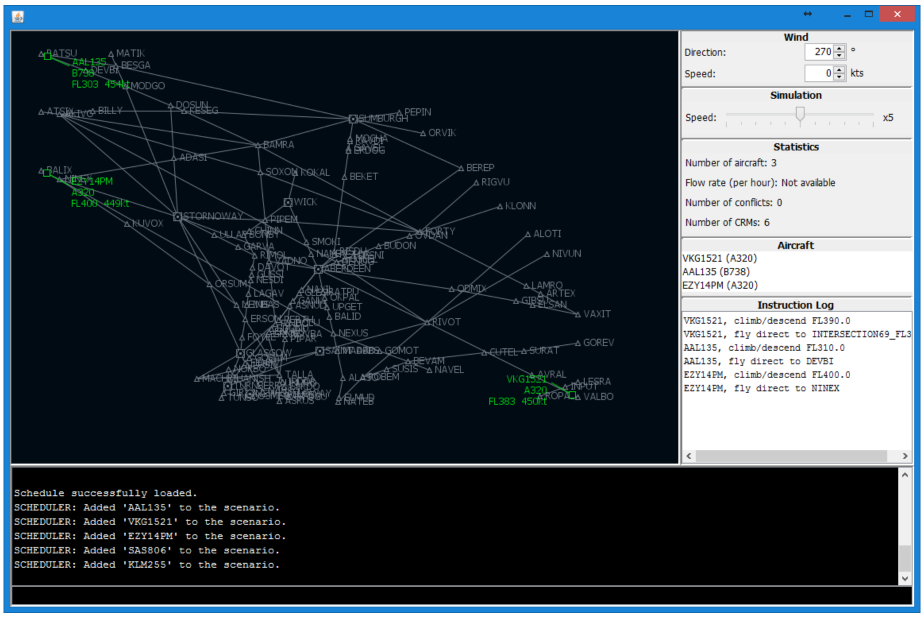
Window Size The size of the WHCA* window (in minutes)

Schedule File path of a schedule file to load on start-up

Logging The directory to enable logging to

-exit Optionally include this flag to close the application once the schedule has complete

3 User Interface



Wind

Direction: 270 °

Speed: 0 kts

Simulation

Speed: x5

Statistics

Number of aircraft: 3

Flow rate (per hour): Not available

Number of conflicts: 0

Number of CRMs: 6

Aircraft

VKG1521 (A320)
AAL135 (B738)
EZY14PM (A320)

Instruction Log

```
VKG1521, climb/descend FL390.0
VKG1521, fly direct to INTERSECTION69_FL3
AAL135, climb/descend FL310.0
AAL135, fly direct to DEVBI
EZY14PM, climb/descend FL400.0
EZY14PM, fly direct to NINEX
```

```
Schedule successfully loaded.
SCHEDULER: Added 'AAL135' to the scenario.
SCHEDULER: Added 'VKG1521' to the scenario.
SCHEDULER: Added 'EZY14PM' to the scenario.
SCHEDULER: Added 'SAS806' to the scenario.
SCHEDULER: Added 'KLM255' to the scenario.
```

Figure 1: User interface

Figure 1 shows the user interface for the system. The UI has the following components:

- The command line terminal at the bottom of the screen. This is where commands for adding aircraft, loading schedules and turning logging on can be entered. Use the HELP command to see a full list of commands.
- The instruction log. This shows a running log of instructions sent to aircraft. The phrasing is similar to how an air traffic controller might speak to an aircraft.
- Aircraft list. This shows a list of all the aircraft currently in the scenario.
- Statistics. Shows how many aircraft are in the scenario, the flow rate (how many aircraft enter/leave per hour), the number of conflicts, the number of conflict resolution manurers (CRMs).
- Simulation speed control. Dragging the slider changes the rate at which the simulation runs. Setting it to zero pauses the simulator. Be careful not to run the simulator at a speed which the AI can not keep up with, a speed of x5 is suitable for most modern computers.
- The wind panel lets you alter the wind direction (in degrees) and the wind speed (in knots).
- The radar display. This can be dragged using the mouse pointer and zoomed in/out by using the scroll wheel.

4 Source Code

The source code is include in the src folder. It is formed of two separate Eclipse projects: ATCAutomation and AircraftSimulator. Both projects have a dependency for json-simple-1.1.1.jar which is also included in the src folder. The AircraftSimulator project needs to be compiled to a library (Simulator.jar) which ATCAutomation depends on.

4.1 Aircraft Simulator

The AircraftSimulator project has multiple run configurations. All of these configurations are included in the 'tests' package which is not required by ATCAutomation.

- *TestDisplay* - this launches the simulator as a standalone application and provides a basic command-line user interface to control aircraft.
- *SimulatorUnitTests* - a JUnit test suite for testing the simulator module.

4.2 ATC Automation

The ATCAutomation project has three different run configurations:

- *ATCAutomationTests* - a JUnit test suite for testing the A* path finding algorithm and testing the graph building unit.
- *GridTester* - a GUI for testing WHCA* using a 10x10 grid.
- *Main* - the main entry point to launch the system.