Flight Scheduling Optimiser

H446, 2022

CENTRE NUMBER:

Hollie Lumsden

Candidate number: 5118

CENTRE NAME: WYCOMBE HIGH SCHOOL

Analysis

# Problem Identification

Flight scheduling considers the arrival/departure times and block times for a specific flight. However, many other factors also affect a flight’s schedule, such as the specific airport, connecting flights, turnaround time, flight route, other airlines and destination. These flight schedules are created by software months in advance by the airline so that flights can run smoothly and with minimum disruptions.

However, unexpected events can happen, such as bad weather or a health problem holding a plane up. This can mean that the original scheduling done in advance needs to adapt and change. This is done by a human team of schedulers. This takes time and means the airlines must employ people to do this task, costing them money that could be used elsewhere. The airline will also further lose money for staying on the aircraft apron docked into a gate as it has exceeded the time it was allocated. Therefore, airlines want disruptions to be handled has quickly and as efficiently as possible, which can only be done to a limit with a human team.

Instead, a solution to this problem of dealing with unexpected events could be computational. This is because it will be faster to solve, meaning that scheduling changes can be made with minimum disruptions to other airlines. Also, this will reduce the long-term costs for the airlines, as they wont have to pay wages for people to do the scheduling, and also the costs for disruptions when at the airport.

# Stakeholders

The clients for this project will be airline owners and managers in charge of current flight scheduling. The stakeholders should represent the opinions of different airlines, and how they would want a computational solution to deal with different situations.

My stakeholders are Becky, Giacomo and William, who are all graduate airport planners for Atkins. Atkins is a British multinational engineering, design, planning, architectural design, project management and consulting services company. They will be helping me with the areas to include with the flight optimiser, as there are many factors I could include and will help me make choices in that area. The stakeholders I have chosen represent people who would be likely to use this software, as it is unique to a certain group of users.

# Why is it suitable to a computational solution?

The solution is suitable to become computational because of the implemented use of algorithms. The solution will be a software that will simulate an airport’s structure and use an algorithm to schedule the flights for the airport, considering different variables, which the algorithm can solve faster than a human.

# Computational methods that will be used

## Problem Decomposition

This problem can be divided into steps that then can further be divided into sub-problems (see Divide and Conquer). My original plan to decompose the problem is:

1. Take in the flight data
2. Schedule a flight with this data
3. Check for any external variables that have been added
4. Move any flights that are affected

This should mean that the program runs smoothly, and that each variable is checked and there are no clashing flights. It should also mean that the programs runs efficiently and smoothly.

## Divide and Conquer

This is when the larger problem is divided up into smaller sub-problems. These sub-problems are more solvable and means the project can become more manageable, as deadlines can be met by finishing certain sub-problems. The sub-problems then combine at the end to become the larger main program.

Some of the sub-problems will include:

* Creating a working button for the instructions menu
* Create a function that puts all the flights into a list
* Check the user interface in functional and working correctly
* Create an algorithm that can organise the flights

## Representational Abstraction

## What remains when unnecessary detail has been removed is a **representational abstraction**, i.e. a simpler version directed at solving a particular problem.

## Many real-world objects and situations are represented in computer systems. In a flight simulator, different planes will be represented in some way within the system. If object-oriented programming is used, the plane will be an object with a set of properties that are relevant to the features of the simulator. Some details will be essential, such as the weight of the plane (as this will affect its handling). Other details, such as the material used to upholster the seats, will be irrelevant, and these aspects will not need to be represented within the system or model.

## Information Hiding

**Information hiding** is a core concept in many aspects of computer systems. In fact, we couldn't comprehend their working if we didn't try to understand them using different layers of abstraction. This applies to both hardware and software, as ultimately, software written in very high-level languages is implemented through billions of state changes in logic gates constructed of semiconductor material.

Most systems hide their complexity behind an **interface**. An interface provides an abstraction of what lies behind it. For example, operating systems will usually have a graphical user interface. This allows users to interact with the computer system through icons and drop-down menus. Desktop shortcuts allow users to start programs or apps without any knowledge of the file name or location. Documents can be saved without the user having to understand the structure or operation of the storage device.

Operating systems are a good example of the fact that a common interface can be provided to very different machines. Whether you have a high-powered desktop computer or a small laptop, the same interface can be used. The interface also allows the workings of the machine to be changed without having any impact on the way that the user interacts with the machine. For example, the RAM or processor could be upgraded; this would improve machine performance but would not require the interface to be changed.

Interfaces are a fundamental principle of [object-oriented programming](https://isaaccomputerscience.org/concepts/prog_oop_encapsulation) (OOP). A class encapsulates (hides) all of its private attributes, and sometimes private methods, which are exposed only through its interface. The interface is provided in the form of public methods; the inner details and workings of classes are hidden from the user. For example, the interface of a class might specify that it has a method named sort, which you can use without giving any thought to the type of sorting algorithm that has been used. If you code in Python, you may have used a list's 'built-in' sort method. Unless you are very curious, you will not know that it uses a 'timsort' (which performs an insertion sort if a list has fewer than 64 items, or a modified version of a merge sort on larger data sets).

When you design a system, you should pay particular attention to the interfaces — both to the user interface, and also to the interfaces for the classes that you design or for the subroutines that you specify. Clean, clear interfaces allow implementation changes to be made without disrupting other parts of the system.

# Success Criteria First Draft

|  |  |
| --- | --- |
| **Criteria** | **How to Evidence** |
| Shows the flight schedule on main window | Screen shot of the main window with schedule on it through use of GUI |
| Visual representation of the airport | Screen shot of the airport visual representation through use of GUI |
| The option to change environmental/disaster variables | Screen shot of the buttons that can change the variables |
| The option to stop simulation | Screen shot of the buttons that can stop/pause the simulator |
| Able to add/delete flights | Screen shot of the buttons that can change the number of flights in the schedule |
| Shows that the software is fast and efficient | Show an efficient use of an algorithm (screen shot the code) |
| Instructions with how to use the software through a pop-up window | Screen shot the options to view the instructions and the instructions button |
| Settings menu were the changing variable and adding/deleting flights option will be stored | Screen shot of the menu and the settings button |
| Simple, usable design | Screen shot of the main window which shows a simple design were everything is clearly labelled and everything is explained |
| All functions are working | Screen shot evidence of testing, with all the inputs and outputs that are made |
| GUI is interactable | Screen shot of the simulation in use |

**UP TO HERE DUE NOVEMBER 15th!!!!! (2 weeks)**

# Interview

Needs to be done when email set up with stakeholders for initial meeting

# Research

See other word document done in summer

# Features of Proposed Solution

Ideas and then explain limitations

# Stakeholder Feedback

After meeting for interviews

# Requirements

Hardware and Software and then those proposed from the stakeholders

# Success Criteria

The criteria and how to evidence that in the project