INTRO

The purpose of my honours project was to create a piece of software to simulate a drone delivery scheduling system. Drones are an up and coming way for businesses to deliver items to customers over short distances. In a paper by Dorling et al, various companies are discussed as developing drone delivery systems, such as Amazon’s “Prime Air”, and DHL’s “Parcelcopter”.

There are a number of advantages for both customers and companies, encouraging growth of this technology such as:

- Speed of delivery. Drones are not affected by busy roads and rush hour traffic. They can fly directly from the depot to the customer, saving time and money.

- Cost of delivery will also likely decrease. A study performed by ARK Invest suggests that Amazon could afford to charge just $1 for delivery and still be profitable.

- Environmental damage will also be reduced. Research shows that co2 emissions produced by drones are lower than that of trucks when used for customers close to the depot.

These advantages in speed, cost and environmental damage are key driving factors in the increase in use of drones that we will see.

However, there are currently some drawbacks to drones, namely flight time and maximum weight they can carry. Amazon announced in 2019 that they expect their drones to be able to fly up to 15 miles and deliver packages under 5 pounds. This is a reasonable amount, but does require that depots are set up within relatively close proximity to one another in order to make this viable.

London for example covers an area of 1572 km squared, which, when using Amazons figures, would require at least 8 depots. When you apply this to every city, it quickly becomes an prohibitively expensive task.

Because of these restrictions, for the purpose of this project, it is assumed that drones in future will have improved capabilities in maximum flight time and weight capacity. This will allow a single drone to travel between multiple customers in per flight.

The issue here comes to the difficulty in finding efficient flight paths. There is a problem known as the travelling salesman problem. In the traditional model of this problem, a salesman wishes to find the most efficient route to visit a number of destinations, without revisiting any he has previously been to. While this sounds simple, the number of potential routes increases exponentially. If a route is to be found between 5 homes, it is a manageable 120 potential routes to compare. For 10 homes, there are 363,800 possibly routes, and for 15 homes there are 1.3e12 possible routes. There are no known techniques to solving the problem in a reasonable time frame.

This lead to the approach taken to solve the problem. I opted to use a combination of grouping customers together by geographic location, and then using a path finding method to get a good route between the smaller group.

For grouping customers, the “kMeans” clustering algorithm is used, which allows us to choose how many groups we would like.

Once the groups have been created, there are two options of path finding algorithms:

Genetic Algorithm is the first, which is an algorithm designed to model Darwins theory of evolution. It emulates nature by creating a ‘population’ of potential solutions and testing how good each one is. The best solutions ‘breed’, and switch some characteristics with one another. They are tested for fitness again, and the process repeats. The genetic algorithm allows a vast number of possible solutions to be tested and improved upon.

The second algorithm is “Greedy Best First”. This algorithm is simpler than the genetic algorithm. Here, the time taken to get from the drones current location to each other unvisited location is calculated. The location with the shortest travel time is selected, and the process repeats until all customers have been visited once.

APP DEMONSTRATION

A software application has been written in Python to model and solve the problem. I will demonstrate how the application would be used to generate a delivery schedule.

- Create new database and add locations

- Explain the input parameters

- Explain the clustering and path finding

- Perform a test, showing how the result changes as new locations are added

-Open Aberdeen DB, show example

- Show output .html file

- Explain why number of routes > number of drones.

REQUIREMENTS MET / NOT

|  |  |  |
| --- | --- | --- |
| **Concept** | **Reason for choice** | **Met** |
| **Must have:** | |  |
| A GUI | This is essential to make the end product usable for anyone. It would be extremely difficult to show the user what is happening without doing so visually. | Yes |
| Real city map | A map of a real city makes the system more applicable to the real world and shows a realistic outlook on how drones may operate | Yes |
| User input from GUI | Without this it becomes a difficult tool for the average user to use and understand | Yes |
| User controlled parameters from GUI | A final item of the GUI that will make the system intuitive to use. The user can directly see the effect that different parameters have on the system | Yes |
| Clustering technique | For a delivery system, speed is essential. The use of clustering allows the system to more quickly find suitable routes for drones. It also ensures that drones are not flying large distances in between customers where they can avoid it | Yes |
| Path finding algorithm | Possibly the most crucial step of the whole process is finding the routes. With no pathfinding algorithm the end product would not solve the problem of scheduling delivery. | Yes |
| **Should have:** | |  |
| Solution produced in a reasonable timeframe | The data should be analysed quickly, and a solution provided. May not be possible if there is a large amount of orders placed at once. | Yes |
| Simple to use GUI | It is important that anybody should be able to use the software without guidance. | Yes |
| Explore alternative path finding algorithms | Differing algorithms would be an interesting feature so that users can see how they vary in quality and speed of solution | Yes |
| Weather conditions considered | Makes the solution more realistic and interesting | Yes |
| Interactive user input | It is possible to restart the process every time a new order is added. However, this is far from ideal, as the routes would all need to be recalculated every time a new order was placed. | Yes |
| Delivered as a .exe file | This allows portability across windows machines. Additional formats could be created for macOS and linux. | No |
| **Could have:** | |  |
| Display current state of routes | It would be interesting for the user to see how routes are generated with different algorithms. | No |
| Show where drones are on their route | Provides a more interactive experience for the user. | No |