**Literature Review**

The purpose of this review is to explore exactly what a drone delivery is, benefits of delivering via drone, and different methods of implementing a delivery scheduler. It will conclude with a recommendation on preferred techniques and environments to develop the scheduler in.

**Introduction**

**Drone Delivery**

A drone is a small, unmanned flying vehicle. In recent years many companies have unveiled plans to begin delivering packages via drone. In 2013 Amazon announced their ‘Prime Air’ service that will deliver packages to customers (Rose, 2013). DHL have announced their ‘Parcelcopter’ project, which has successfully delivered medicine to the island of Juist in the North Sea (Hern, 2015). There are many more companies who have unveiled plans to release their own delivery drone solution (Sacramento, 2019). This sudden upsurge in use has been brought on by advancements in technology used in construction of drones. The price of manufacturing carbon fibre has dropped from $25 to $10 per kg over the last 20 years. This combined with improvements in battery technology allows drones to fly faster and further (Dorling et al, 2017).

There is great demand from customers for a faster, more reliable option for delivery. This is a driving factor in the development of drones for last-mile delivery. A study conducted on over 4700 people from China, Germany and the USA showed that 23% of customers are willing to pay extra for the benefit of same-day delivery (McKinsey, 2016).

[image 1]

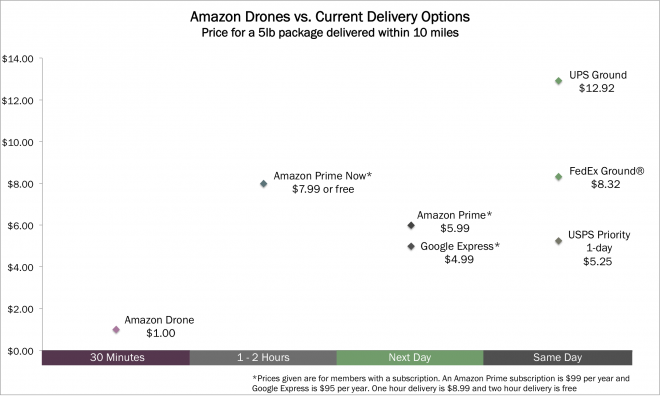
There are several advantages to using drones for so called last-mile delivery. The primary advantage of drone’s vs trucks is the speed and timing accuracy as drones are not affected by traffic or road layout of a city. This enables them to offer fast delivery and tell the customer to the minute when the parcel will arrive (Hau L. Lee et al, 2016).

For the company deploying the delivery solution, they will likely save money. A study performed by ARK Invest suggests that Amazons drone delivery service could be charged at just $1 per delivery and still be profitable (Keeney, 2015). [fig 1]

It is also suggested that delivery by drones will be environmentally beneficial. As drones are battery powered, they do not directly produce any diesel pollution. Research shows that carbon dioxide emissions produced by drones is lower than that of trucks, even taking into consideration the energy requirements of drones and the smaller service zone (Goodchild and Toy).

These are they key factors driving companies to invest in drones as a delivery solution.

[fig 1]



**Problem Specification & Boundaries**

What is going to be solved, what restraints will be relaxed (Flight time) Why?

The problem to be studied and solved here is the actual scheduling of delivery paths and times for drones. We are going to assume that a company of reasonable size has set up a depot within a reasonably sized city, and plan on using that for drones to use it as a base to collect parcels to deliver out to customers to. They will also use it as a charging station and while they are not in use they will stay here.

Current drone technology makes this a simple scheduling problem. Amazon announced on 5th June 2019 that they expect their drones to be able to fly up to 15 miles and delivery packages under 5 pounds (About Amazon Blog). The CEO Jeff Bezos has stated that 86% of items delivered by Amazon weigh 5 pounds or less (CBS News). This shows that Amazon are aiming to deliver one item per drone per trip.

We will be relaxing some of these restrictions to enable us to look to the future. If we image that soon the drones will carry 15lbs worth of items up to 15 miles, we can implement a solution to solve this future development. This is a reasonable assumption for Amazon to roll out in future iterations of their delivery system, as drones already exist that can carry much more then this, with the Alta 8 from FreeFly Systems able to carry up to 18kg (FreeFlySystems.com).

These assumptions of the future of drone tech and their use within industry turn this problem from simple delivery time scheduling, to a travelling salesman problem. The travelling salesperson problem is an old one, and variations appear as early as the 1800s. In the 1930s it began being studied by mathematicians in Vienna and Harvard (math.utwaterloo.ca). It is described as finding the optimal route for a salesperson to take to visit several cities. They start and end the journey in the same location, and they also cannot visit a city more than once per trip. The optimality can be defined by various measures, such as time taken or cost.

The problem is easy to understand, and the method of solving it isn’t difficult, all we must do is find every possible route there is for the salesperson to take, and then pick the shortest one. The issue is that adding new cities to the journey very quickly increases the number of possible routes.

If we name the number of cities n, the number of possible routes is the factorial of n - 1(ibm.com), since you don’t need to take into account travelling to the city you start at. That is that if we have 5 cities to visit, we multiply 4 by every number below it down to 1. If the salesperson has to visit 5 cities, there are 24 possible routes. If we increase this to 10 cities, there are 362,800 possible routes, and 15 cities gives us 87,178,291,200 possible routes. The problem is known as ‘NP Hard’ (ibm.com), which means that we don’t know if we can solve it in a reasonable amount of time.

If we apply this to the drone delivery domain, where we could have hundreds of customers per day, it quickly becomes an unmanageable problem to solve.

**Techniques**

Here we will review several techniques applicable to solving the travelling salesperson problem (herein TSP).

First, we must consider what type of search we will use. An informed search is a type of search that contains an array of knowledge about the search space, such as cost so far and distance from the target. There are a range of algorithms that fall under informed searches, but we will focus on A\* here. A\* has several benefits, but it is mainly used due to being optimal and complete. What this means is that on any given search space, if there is a solution, A\* is guaranteed to find the best one (geeksforgeeks.org).

There are two key issues with using this type of search on this domain. Primarily, we don’t know what the target is for the search to find, other than to have visited every location and have the shortest route possible. A\* searches are useful for finding the route to a target and showing how to get there, in our case we don’t care about how to get there, we just want the final route. The second issue we have is hardware limitations. As A\* searches keep track of where they have been until now, if the route we must take is extremely long, it may be unfeasible to run this type of search on our hardware (Stackabuse.com). These limitations add up and make A\* an unsuitable method of solving our problem.

This leads us on to local searches. A local search is unlike an informed search in that it does not keep track of where it has been. Because of this, the issue of hardware limitation is eliminated, as only the current point the search has got to is stored in memory. A local search doesn’t know what the final target it is aiming towards looks like, if we knew the optimal route to take there would be no need to do a search at all. Unlike A\*, local searches carry no guarantee of finding a solution, and if they do find one, it may not be the best solution there is.

They are named local searches because they look at what moves are possible from where they are now (locally) and select one of these options. For these reasons, we will review several local search algorithms and find which is the most suitable for solving our problem.

**Hill Climbing**

The first of our local search algorithms is known as hill climbing. The most basic of local searches where it looks to each of its neighbouring states and selects the one that seems the best. The search is complete and returns the route that it has found once there are no neighbours around with a better solution. The issue with this type of search is that it can be stuck in what’s known as a local maximum. This is where there are no ways immediately surrounding the current route that are better, however elsewhere in the domain there are. The algorithm can’t go backwards to a worse solution in order to look for a better one elsewhere.

[local maximum image]

* **Nearest Neighbour/Convex Hull/Cheapest Insertion**
* **Genetic Algorithm**
* **TSP grouping clusters**

**Environments**

Libraries available, available resources for this domain, creating UI and displaying output, familiarity

* **Java**
* **Python**
* **Web Based?**