**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]

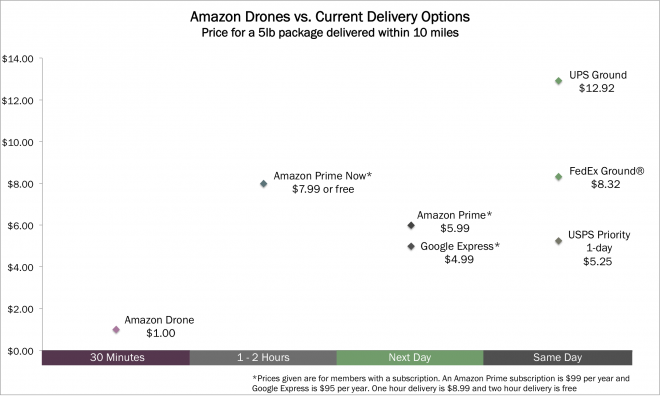
There are several advantages to using drones for so called last-mile delivery. The primary advantage of drone’s vs trucks is the speed and timing accuracy as drones are not affected by traffic or road layout of a city. This enables them to offer fast delivery and tell the customer to the minute when 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**

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**

The travelling salesperson problem is an old one, and variations appear as early as 1759 (P.Larran AGA)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 consider travelling to the city you start at. 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 NP-hard which means that there are no known techniques to solve it in polynomial time. (Kylie Bryant)

If we were able to find and evaluate 1 million routes per second, it would take over a day to evaluate all the routes in a 15-point route. If we apply this to the drone delivery domain, we may have thousands of orders a day and the amount of processing time and power to brute force the best route is unrealistic. For this reason, we need to find alternative methods to solve the TSP.

**Clustering**

Because of the difficulty of the Travelling Salesperson Problem (herein TSP), as well as the limitations of drone technology now and in the near future, it is useful to break down our problem into clusters. If we take the above example of a set of 15 cities taking over a day to find and evaluate every route and we group locations into sets of 5, our computation time drops dramatically. Assuming the same computation speed this would take 0.000072 seconds to find and evaluate the best routes. While 1 million routes per second is an unrealistic number, it illustrates the potential benefits to computing and time resources needed with clusters. Additionally, clustering allows us to model a delivery system with multiple drones delivering simultaneously when we have clusters.

**K-means**

The k-means technique takes a parameter of k, and randomly selects that many locations to begin. These locations are set as initial centroids, or exemplars. From here the algorithms assigns each location to a cluster depending on which centroid it is closest to. It then recalculates the centroid by taking the mean of all the locations per cluster. Finally, it reassigns locations to their nearest centroid again. This process repeats until no locations change cluster. The model can be adapted slightly to assign the closest location to the mean as the centroid. This is known as K-mediod. (Courtney-Marie Bruggeman)

While this technique is widely used, it has two key issues. Primarily we need to define the number of clusters ourselves. This can cause problems both if we use too many or too few clusters. If we specify too many clusters, we may end up with drones flying when they don’t need to be, wasting money. If we specify too few, we will have an increased amount of processing time to calculate our route and customers will have to wait longer for their delivery. Additionally, k-means can give poor results if a poor spread of initial locations are selected. Because the locations are randomly selected, they may end up all in proximity. The solution to this is to run the algorithm several times until a good solution is found, but again this wastes time and resources.

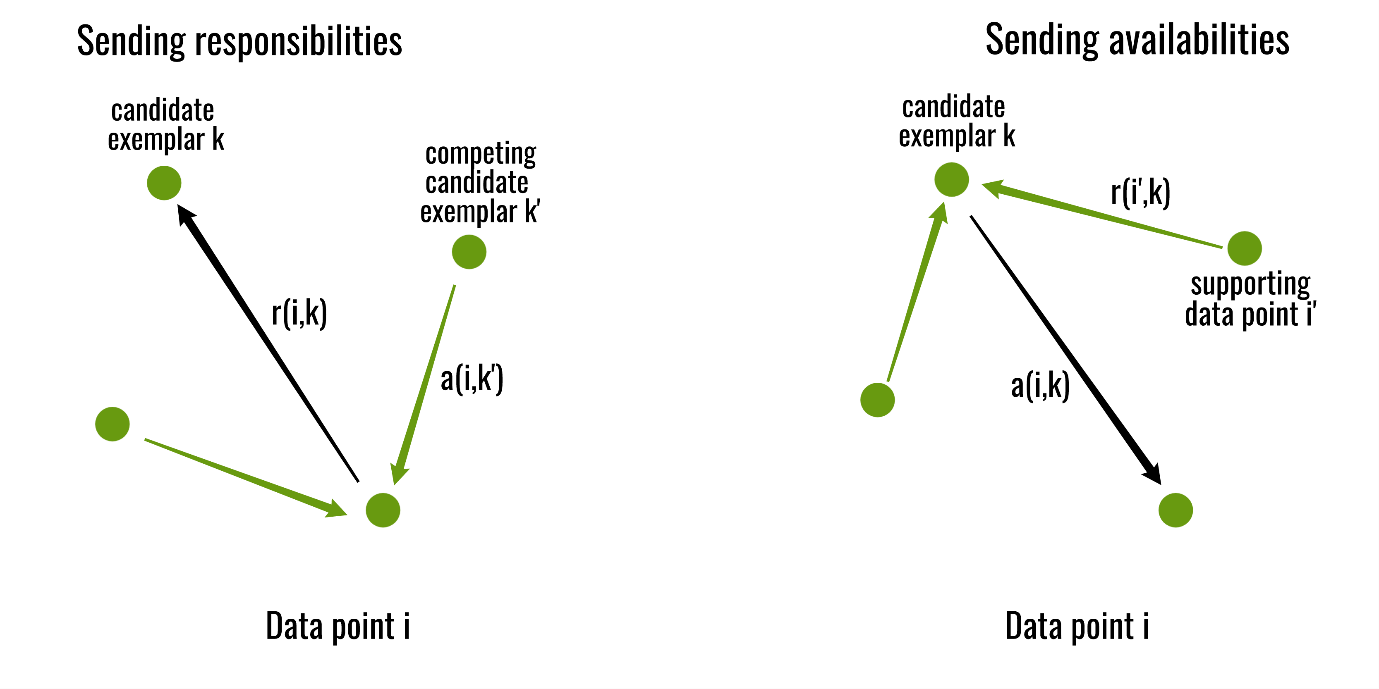
**Affinity Propagation**

Affinity Propagation solves these key issues that k-means faces. It tackles the issue of number of clusters by defining its own, with no input from the user. Because of this is also does not fall victim to a poor initial selection of locations.

The process works by alternating between two steps:

* Responsibility. This shows how well fitted a location is to be the exemplar for another, relative to all other locations. Low similarity means a low chance of two points becoming part of the same cluster. This is sent by each location to each other location.
* Availability. This shows how appropriate it would be for a location to pick another location as its exemplar, considering the responsibility score it received from each other location. This is sent back as a reply from each location to each other location.

The process iterates until there is agreement between all locations for where is the best exemplar for where, and no further changes are needed. (Tan L).



(Geeks for geeks) Figure 2 – Affinity propogation steps

Lizhuang Tan et al performed a comparative study of k-means and affinity propagation for clustering with a travelling salesperson problem. They surmised that both algorithms showed an improvement in computational cost then when solving the same problem without clustering. Furthermore, they concluded that they prefer affinity propagation because of the sensitivity of k-means to poor initial centroids, as well as a requirement for a pre-set number of clusters.

For this reason, we will be using affinity propagation as our clustering technique.

**Travelling Salesperson Problem**

Here we will review several techniques applicable to solving the Travelling Salesperson Problem.

First, we must consider what method we will use to solve the problem. The simplest method of solving the problem is brute force. The method is to run through every possible route and picking the shortest one. This method is possible for a small number of destinations, however as previously mentioned, this is an unrealistic method as the number of possible routes increases so dramatically for each extra item that has to be delivered. For our uses, this is not a feasible method to use. If we have even 15 customers at once, finding the best route is unreasonable.

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. If we knew this target, there would be no need to perform a search at all. 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 is at is stored in memory. A local search doesn’t know what the final target it is aiming towards looks like. 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.

**Nearest Neighbour**

Nearest Neighbour is a basic search algorithm where it looks to each of its neighbouring states and selects the one that seems the best. In this domain it would look to every location it hasn’t visited and pick the one that is closest. The search is complete and returns the route that it has found once there are no new places to visit. The issue with this type of search is that it can be stuck in what’s known as a local maximum. This is where the route the algorithm has found cannot be improved upon by the algorithm. It will always find this same route. (Amanur Rahman Saiyed)

**Convex Hull & Cheapest Insertion**

Next there is a combination of techniques that are viable to solve the TSP. The convex hull algorithm is used to create an outside boundary that all locations lie within. It begins the search at an extreme point, such at topmost. Started facing away from the rest of the locations, the search looks clockwise and stops when it finds another point. The process is repeated until it returns to the start point. This gives us our outside boundary. (Engineering book)

From here we use a method called cheapest insertion to visit all the locations that lie within this boundary. This finds every remaining location, and every way to get to this location. It then calculates a ‘penalty’ for travelling to this location. This penalty is the difference between the distance travelled for the new route, and the distance travelled for the old route. The algorithm then selects the route with the lowest penalty and moves on until all locations have been visited.

**Genetic Algorithms**

Genetic algorithms were first suggested in the 1960s (Bremermann et al). They are intended to simulate evolution as it occurs in nature. They mimic natural selection by selecting only the best individuals to go on to produce more individuals in the next generation. Each individual has a set of characteristics, and they pass this on to their offspring, so strong genes are maintained through generations.

The search begins with a randomly generated set of individuals, or in the case of TSP, routes. These individuals are characterised by a set of parameters and should all be different. From here a fitness score is calculated for each individual. This score determines good the solution is compared to others. Here our fitness score would be length of route. Next, selection occurs. Selection is the method of passing on genes to the next generation. Two pairs of individuals are selected based on their fitness score and move forward.

A process called crossover then occurs. For each pair selected to produce offspring, a crossover point is selected. This is a random point somewhere in the genes. Offspring are generated by exchanging genes within this crossover point. The offspring are added to the population. When offspring are formed, there is a low probability that mutation will occur, meaning that certain properties of the new individual are changed in some way. For TSP this may be a pair of cities switching places randomly within the route.

The process continues until the population has converged. Conversion here means that the offspring being created are not significantly different from the generation creating them. The size of the population does not grow. Once new generations have been formed, individuals that have the lowest score are removed from the search space. (P. Larranaga)



Fig 3 – Flowchart of a genetic algorithm

**Environments**

**Python**

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