Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

An Investigation Into Drone Delivery Methods

Author:
Joshua Lister

Supervisor:
Professor Stephen Neethling

Email: Joshua.Lister20@imperial.ac.uk

Github: acse-jpl20

Contents

Acknowledgments

I would like express my appreciation to my supervisor Professor Stephen Neethling for his insightful guidance and patience throughout this project which he also volunteered to supervise. I would also like to thank the friends I made on the course.

LIST OF ABBREVIATIONS

DACSM	Drone and Charging Station method
TADM	Truck and Drone method
TOSL	Truck stop off location
TSP	Travelling Salesman problem
VRP	Vehicle Routing Problem
UAV	Unmanned aerial vehicle
KMC	K-Means Clustering
ALG	Algorithm
GA	Genetic Algorithm
CO	Crossover
MN	Mutation
ED	Euclidian Distance

1 Abstract

The delivery of parcels in a fast yet cost-effective manner has been thoroughly researched. Currently, deliveries are done door to door with a truck. This is expensive as many drivers are needed, each address has to be visited by the truck which is slow and consumes a lot of fuel. To reduce the associated costs with the last leg of the delivery, this software has been developed to reduce the costs associated with parcel delivery by implementing drone delivery. Each address is assigned to a cluster to which a drone would fly from the centre of that cluster. The route path through the clusters is determined and minimised to reduce fuel consumption, cutting costs. Two drone delivery methods were implemented and compared. If the drone is not constrained to one single address, then there is an electricity cost reduction of 36 percent. In addition to this, fewer drones are required to complete the delivery, for our test case, this saw a reduction of up to half the number of drones required.

2 Introduction

The most expensive step of logistic activities is the infamous 'last mile' (Boysen et al. 2020). The last mile refers to the final stage of the delivery process, where parcels are in transit from a distribution centre to a final destination. It is the most crucial step of the delivery chain and is responsible for most late deliveries (Wang, Zhang, Liu, Shen & Lee 2016).

The increased demand for online parcel delivery increases congestion and leads to more pollution, particularly in dense urban areas (Levy et al. 2010). Infrastructures such as roads and traffic measures hinder the delivery times of parcels by ground vehicles. UAVs or drones can reduce delivery time, cost of delivery, road traffic and pollution levels, proving in theory to be a viable alternative to traditional ground delivery methods (Agatz et al. 2015). A report from 2015 highlights the potential for cost-cutting were to deliver a parcel within 16.1km of any destination in the USA is $12.92 by UPS ground next-day delivery, but costs $1 delivering within 30 minutes using Amazon's drone delivery service PrimeAir (Keeney 2015).

UAVs also have potential in emergencies, delivering medicines, clean water and food parcels in rapid time over non-traversable terrain (Konert et al. 2019). There is a range of uses where optimisation of a modular drone delivery system: reduces the time of delivery, pollution, road congestion and cuts overall costs (Lee 2017).

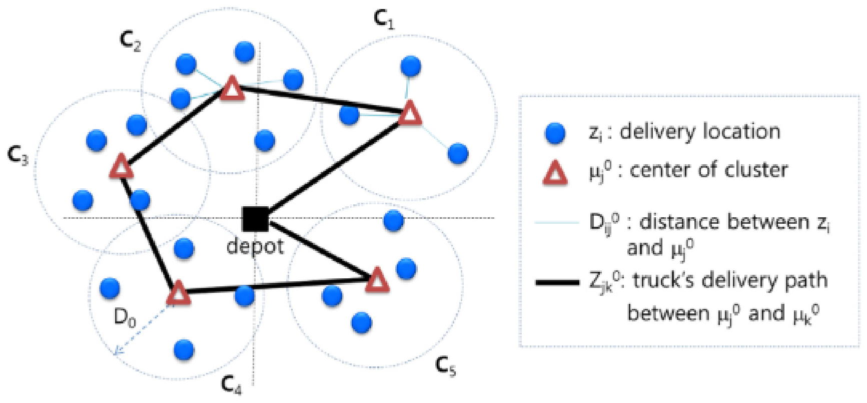
Over the past decade, there has been an increasing interest in using UAVs for delivery purposes. Large companies such as Amazon have already invested in drone delivery to replace the van to door delivery of parcels.

Previous studies into drone delivery generally focus on transportation and delivery in urban areas and are based on the travelling salesman problem (TSP) or the vehicle routing problem(VRP) (Khoufi et al. 2019, Dorling et al. 2016). Several challenges remain with drone delivery. The two leading principal limitations are payload and range (Hong et al. 2018). TSP is insufficient when used to optimise drone routes for door-to-door as it presumes a singular drone that could potentially only deliver to one address assuming a payload capacity limit of 5kg (86% of Amazon parcels delivered are under 2.3kg (Bamburry 2015)). When using lithium-ion batteries, commercial drones have a flight time of approximately 30 minutes, and a single drone takes more than one hour to recharge (Abdilla et al. 2015). Therefore, the number of trips a single drone can take before a recharge is severely limited. Particularly in rural areas where there is more likely to be a long-distance between delivery addresses. However, since in rural areas, there are fewer physical structures. Therefore, larger drones with a greater flight distance and payload capacity could be used and this would partially negate the issue of longer distances between addresses.

A proposition made by a few articles on the subject at hand is a combination of a swarm of drones with a moving truck to overcome these limitations (Chang & Lee 2018). Other articles suggest distance constrained single-flow approaches with recharge stations (Dorling et al. 2016). The number and locations of distribution centres and recharge stations were optimised to minimise the total costs of the system. These two methods are currently the most researched with slight adaptations in their scope and approach.

2.1 Travelling Salesman and Vehicle Routing Problem

Both TSP and VRP are combinatorial optimisation problems where the main difference being that VRP is also an integer programming problem. They are both NP-hard (non-deterministic polynomial-time) (Toth & Vigo 2002). Therefore, heuristic methods typically are used due to the size of real-world problems (Halim & Ismail 2019). VRP is a generalisation of TSP such that the drone doesn't have to return to the point of origin (Laporte & Nobert 1987).



C_1
C_2
C_3
C_4
C_5
D_0
depot
● z_i : delivery location
μ_j^0 : center of cluster
D_{ij}^0 : distance between z_i and μ_j^0
Z_{jk}^0: truck's delivery path between μ_j^0 and μ_k^0

Figure 1: TSP routing for a distribution centre and five clusters (Chang & Lee 2018).

2.2 Algorithm overview

Each address is assigned to a truck stop off location within a single drone reach by the KMC algorithm. The solution requires that each truck stop off location is visited once and only once. The use of random cross-overs and mutations will lead to many invalid circuits being generated. Therefore, problem-specific MO and CO operators were implemented to ensure no invalid circuits were produced. Finding an optimal truck route in a reasonable time is important as the calculation will be repeated for many postal address regions. Hence, the GA is suitable to return an optimal route in good time. Further explanation of the GA method is in section 5.2.

As stated previously, most studies presume a singular drone flight where each drone has been assigned to a single parcel. The truck/lorry is limited on the number of addresses it can deliver to by its volume capacity. By allowing for a singular drone to carry more than one parcel and to fly to more than one address, we then reduce the number of drones required to deliver to all addresses within a cluster as seen in 2. The removal of this limitation maximises the potential use of a singular drone by carrying multiple parcels and going from one address to its nearest neighbour and eventually back to its point of origin, the respective cluster centre. The reuse of a single drone not only reduces the total number of drones required but also reduces the total distance travelled by a collective of drones. Therefore, with fewer drones needed to successfully deliver all parcels to the addresses, fewer trucks will be required to drive on route storing the drones and parcels. By removing the single-use limitation, it would reduce the overall cost of the delivery system by not having to employ as many drivers, maintain as many trucks and purchase as many drones.

2.3 Project Objectives

The main costs associated with the approach are the truck(s) cost, cost per drone, the salary of truck driver fuel for the truck, electricity for drones, cost per drone and salary of the truck driver. The aim is to minimise: total distance travelled by the drones by use of the which corresponds to cheaper electricity costs, the number of drones required as a smaller initial investment is then required and fewer truck drivers would be required to transport the drones, hence fewer people on the payroll, and optimising the truck route to use less fuel.

3 Software Description

The software implements the GA, KMC and drone cluster on top of many sub-algorithms. Given the structure of the data, many of the algorithms have been implemented from scratch to solve problem-specific optimisations. The resultant clusters from KMC largely depend on the flight time of the drone, hence unlike in other KMC out there which do not have this constraint the algorithm continually adds to the cluster count until the constraint is satisfied. The GA uses appropriate crossover operators (CO) and mutation operators(MO) for the TSP problem which returns an optimal truck route. A number of drones fly from the stopping locations found by the KMC to deliver parcels. Each drone flies to as many addresses as possible given the user-defined or preset values for the payload capacity and maximum drone flight distance.

3.1 Architectural Design

As shown in figure 2 the data undergoes pre-processing. A python script reads in the data from the appropriate input file, which removes any rows from the data that are missing a column entry besides the postcode. The removed rows along with their postcodes are outputted to a text file so it is clear which postcodes need their information updated. The processed data is then formed into clusters using KMC until both the centroids of the clusters have converged and the distances from the centroids to addresses within a centroid are short enough for a drone to deliver to at least one address. The centroids are the input route for the GA, which then optimises the route to reduce the total distance. Drones with user-specified metadata are assigned to n addresses, where n is determined by the maximum flight distance and payload capacity of the drone.

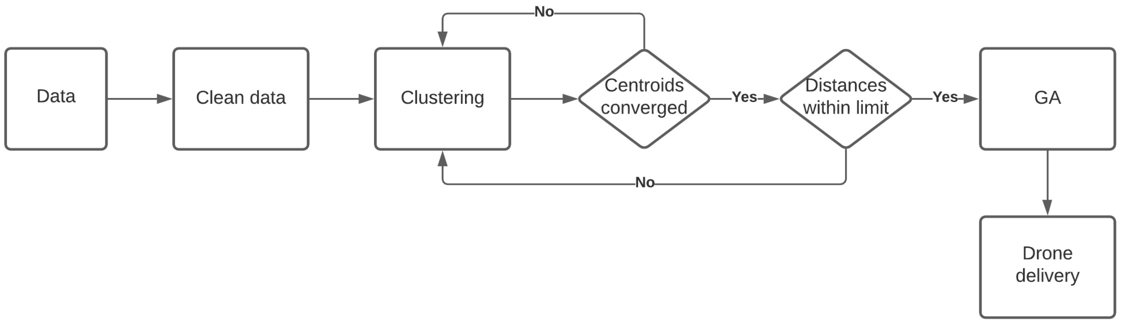


Figure 2: Simplified workflow for drone delivery.

3.2 Development Methodology

The development of the code in this project was independent except for some contributions from my supervisor. Due to the project mostly being developed without external input, there was no predefined organisation strategy. Elements from the agile development method were most suitable. The design and requirements of the project changed throughout the project, although the aim of going beyond the state-of-the-art has been maintained throughout. Each substantial code section functionality was tested and verified quantitatively before moving onto the next stage of the development process. Due to the nature of the data being handled, some functions such as KMC could only be tested qualitatively, visualising the different clusters using the Python seaborn library.

3.3 Code Metadata

The standalone code has, for the most part, been developed in C++ and the use of external C++ libraries have been kept to a minimum. The C++ libraries used include but are not limited to: random, chrono, cmath, fstream. Functions from these libraries were used for specific operations (i.e. generate a uniform distribution, apply sqrt, etc). For standard libraries, the vector, unordered map and string were used for specific data structures(i.e. unordered map to count the number of addresses matched to each centroid, vector to store data, strings for the type of drone, etc). The C++ code was compiled with C++20 with Microsoft Visual Studio 2019. A lower-level programming language like C++ was chosen over Python due to its superior performance in speed and memory management. The processing, however, was done in Python due to ease of use. The libraries: pandas, NumPy and matplotlib were used due to their streamlined form of data representation, an extensive set of features and their efficiency in handling a large dataset.

GitHub Repository: https://github.com/acse-jpl20/uav
Documentation: https://uav-documentation.atlassian.net/wiki/spaces/UAV/overview

4 Main Data Structures

Each postcode has 2D position coordinates and a total parcel mass. The postcode and its metadata are stored in memory in the form of a struct. The standard template library (STL) vector stores each address. The STL vector is a dynamic sequential container that stores memory contiguously (Bai et al. 2011). Many of the algorithms access elements by position. The time complexity for random access is $\mathcal{O}(1)$. The speed of access is important as we read through the container sequentially via iteration. An associative container would be inappropriate given we are not searching for an element. Associate containers are used, however, to keep track of the number of addresses associated with each cluster.

After each address has been assigned to its nearest centroid. A vector of vectors of pointers points to the addresses where the column of the 2D vector points to all the addresses within the radius of that centroid. The size of each vector within the vector is determined by the values stored in the unordered map. By using pointers to point to the memory address of the metadata of the postcodes, we prevent unnecessary use of computer memory to store the address objectives themselves. The storing of memory addresses of postal addresses in separate clusters allows for faster and easier access when determining drone routes for each cluster.

5 Algorithms

The project provides solutions to the GA, KMC and other algorithms which will be discussed in more detail later. In the heuristic method, the KMC algorithm will determine the number of clusters required to ensure each address can be reached by a drone and which postal addresses correspond to which cluster by the use of an id number. The GA will return the stopping points for the truck which has the shortest total length in metres. Using the least distance truck route, the drone delivery algorithm calculates the total number of drones required for the entire route and the total distance travelled by the drones. The drone delivery algorithm allows for a drone to deliver to more than one address. The drone flies to as many addresses as possible when either the drone distance or payload capacity is met. The total previous total distance, total mass and addresses are assigned to that particular drone. The total cost of the journey including fuel for the truck the overall electricity cost from the drones is returned and displayed to the user.

5.1 K-Means Clustering

The KMC method is a clustering method which groups data together depending on their ED from each centroid and aims to minimise the objective function J (1) where $r_{nk} = 1$ if data point x_n belongs to cluster k else $r_{nk} = 0$. Finding the optimal solution is NP-hard and hence in this solution, the heuristic Lloyds algorithm has been implemented which has a time complexity of $\mathcal{O}$(nkdi) where n is the number of d-dimensional vectors, k the number of clusters and i the number of iterations required until convergence. However, in practice, Lloyds algorithm has a linear time complexity due to the small number of iterations required to meet convergence.

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}||x_n - \mu_k||^2 \tag{1}$$

The algorithm starts with randomly selecting k unique data points to be chosen as the initial centroids. Each data point is matched with its nearest centroid. Unlike in most KMC methods, the one implemented here doesn't require a specified number of clusters (MacKay & Mac Kay 2003). The algorithm starts with two centroids. The centroids are iterated upon until the convergence criteria have been met. Based on the drone maximum distance criterion, all the distances between the centroid and its respective matching addresses must be at least half the maximum drone distance. If any of the distances are longer than half the maximum drone distance, a new cluster is added. The process is repeated until there exists k number of clusters that satisfy the convergence criteria and all distances between centroids and addresses are within half the maximum distance a single drone can fly.

K-Means Clustering Validation

The distances between each centroid and its matching address are tested to see if all fall within the maximum drone distance constraint. If all the distance between the centroid and the address are less than the maximum drone distance then the KMC algorithm has been successful.

5.2 Genetic Algorithm

The TSP problem is NP-hard with the worst-case running time increases superpolynomially and best case, exponentially. Therefore, for trucks with more stopping locations, the time taken to calculate an optimal route becomes impractical for real-life applications. The problem is discrete problem and thus gradient search methods which utilise the derivative of a continuous function are inappropriate for finding the optimal route. Hence meta-heuristic algorithms which can produce an optimal solution in good time are more appropriate.

The GA was chosen over another metaheuristic method such as simulated annealing (SA) because SA is a single solution based while the GA is population-based. The SA only exploits while the GA both explores and exploits (Mahfoud & Goldberg 1995). Given the multi-model nature of TSP, the exploration by use of population often results in a more optimal route.

The GA is an algorithm where the solution evolves to an optimal solution Deb & Spears (1997). The implementation of the GA is generic allowing for functions to be passed as parameters. The initial population consists of several chromosomes, where each chromosome represents a route for the truck to follow. The selection of chromosomes from a population compares the fitness values of the different chromosomes. Then based on a roulette wheel approach, a chromosome is selected from the population. The chromosome with the largest fitness value is passed onto the next generation left unchanged by CO or MO methods. By employing elitism, it allows us to show how the truck route path changes over generations. Elite parent chromosomes at most make up 10% to maintain diversity. The non-elite parents undergo CO and MO determined by user-specified probability rates for CO and MO to produce offspring. This process continues until the generation size limit has been reached. The algorithm continues until either the solution has converged or until the specified maximum number of generations, the threshold has been met.

Crossover Operators

Three different CO methods have been implemented, the ordered CO, the CO collision and the standard CO. The ordered CO takes a random slice from one parent chromosome and iterates over a different parent chromosome, adding elements that are not already in the new child chromosome (Ahmed 2010). Given the nature of the ordered crossover, there are no illegal path routes produced from performing the function and thus, there is no need to check the validity of the route. The fitness function which measures the distances between two truck stopping locations is sensitive to the relative positions instead of the absolute positions. Therefore, standard crossover functions which disrupt optimal subroutes perform poorly in comparison to the ordered crossover. In addition, given the high probability of illegal routes produced, the time taken to run the GA using the standard CO is exponentially long for large routes and does not finish. The ordered CO exploits the current solution to find the optimal route permutation in the local solution neighbourhood. It does not, however, and particularly so for smaller routes explore the solution space, unlike the MO operator.

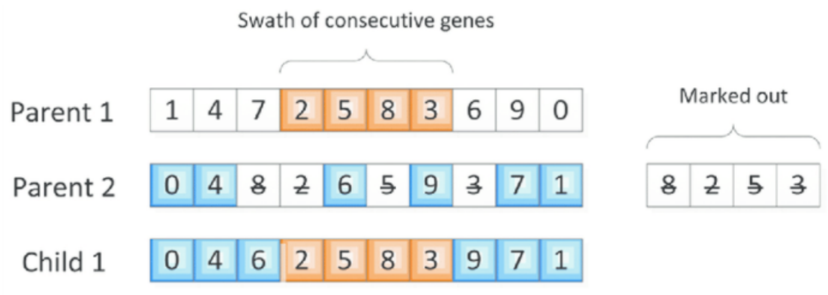


Figure 3: Example of the ordered crossover (Wang, Lu, Wei, Ji & Yang 2016).

Mutation Operators

Two MO methods have also been implemented. In previous studies, different mutation operators have been studied intensively. The MO operators reverse sequence mutation (RSM) and partial shuffle mutation (PSM) both showed great promise in comparison to the standard mutation where it swaps two locations of the route Hassanat et al. (2019). The RSM reverse the order of the locations of a segment of the route in which the start and endpoints for the segment are generated randomly. On the other hand, the PSM, the route is iterated through, swapping the index of the for loop increment and a randomly generated number. The mutation functions prevent the GA from getting stuck in a local minimum and therefore is responsible for the exploration of the solution space.

The CO and MO probabilities are pre-determined by the user, however, they do have default values. The default values promote a balance on improving diversity to prevent getting stuck in local minima but also to maintain chromosomes with a large fitness value.

GA Validation

The validation methods of the GA and sub algorithms associated with the GA mainly cover test and analysis. Validation of the ordered CO is performed by crossing over two known parents routes and comparing the new child route to the target route. The new child route and the target route are the same when tested and hence the ordered CO works as expected. In addition, the route after CO was tested to ensure no illegal routes (repeating truck stop locations) have been produced in the process. The validation of the mutation function compares the initial route and the route after mutation to ensure they are different. The GA itself has been validated by passing in a different fitness function as a parameter where the GA optimises to match a given route from an initial randomised route.

5.3 Multiple Address Drone Delivery

The Drone delivery method reduces cost by maximising the total distance a single drone can travel and carry. This minimises the total distance travelled by all of the drones and the number of drones required to reach all of the addresses. The ED from a truck stop off location (TSOL) and its closest address is calculated, if the total distance plus the distance required to fly back to the TSOL is below the drone distance constraint, then the ED between the address to the next nearest address is calculated and so on until the drone distance constraint is met or until the payload capacity has been reached. A drone object then points to the memory location of the original data structure containing the postal code addresses associated with the flight path of that drone. The total distance flown by all the drones is totalled as well as the mass of each parcel that each drone carries.

Drone Delivery Validation

The total distance is compared to the total distance flown when the constraint of a drone flying to one address and one only is enforced. After each address has been visited, a boolean value associated with the address is set equal to true. A simple check is used to ensure each address has been visited. Each address is checked to see whether the boolean ’visited’ is equal to true.

6 Results and Discussion

This section will introduce results from simulations of the documented algorithms. There are a few metrics which are used when evaluating the performance of the algorithms:

- *Truck Route Distance [m]* : The total distance travelled by the truck along the stop off locations.
- *Collective Drone Distance [m]* : The total distance travelled by all of the drones.

The maximum flight distance for the drone used in this investigation is 400m and the maximum payload capacity 5kg.

Minimising the truck route is essential as this is likely to be the most expensive part of the delivery as the petrol/diesel for the vehicle costs significantly more than electricity, and the driver of the truck is paid by the hour. Consequently, to reduce the total cost of the delivery, an optimal route permutation is required. Both the type of the MO and CO, and their associated probabilities can influence the optimal route solution found by the GA. The TSP benchmark BERLIN52 consisting of 52 locations was used to test the performance of the GA.

It is evident from figure 4 that the performance of the RSM is largely superior to the PSM and PM. The distance rapidly decreases for all three MO operators, however, the slope corresponding to RSM is steeper than that of PSM and PM. The RSM plateaus the earliest already at a significantly lower total distance route than the other two MO operators, but it does, however, gradually decrease as the mutation helps to stop the continual local search by promoting diversity, allowing the GA to explore a greater solution space for superior permutations. While the best permutation found by PSM is better than that of PM. The slope gradually decreases in comparison to PM at approximately 300 generations. Depending on the defined maximum generation and convergence count, the PM could prove to be a better alternative than the PSM which has been praised in previous literature for the TSP. The three MO operators do not finish on the same generation count as the GA exited at different points based on the convergence count. The PSM ran for almost the entirety of the max generation count. The RSM was used as the default MO method as a result of these findings.

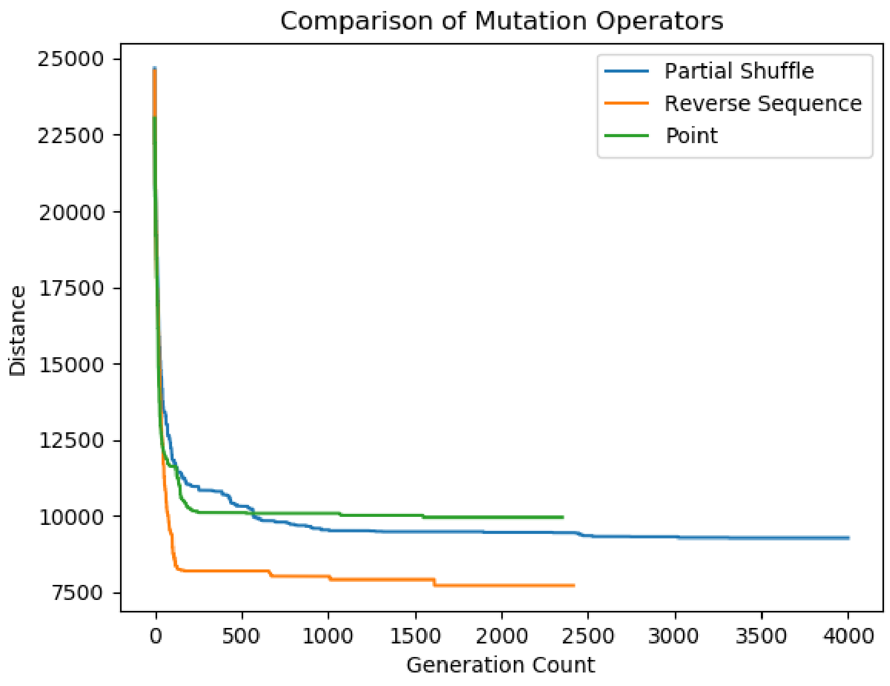


Figure 4: Optimisation of the BERLIN52 problem.

Given the optimal MO RSM, a grid search analysis was performed on the CO and MO probability values to try to determine the most suitable values in order to find solutions with smaller route sizes. A range of thirteen different values for CO and MO probabilities were tested using a generation size of 200 and a maximum generation count of 1500. At first glance, the graphs do not have a clear pattern suggesting that the MO and CO probabilities for larger TSP problems have little impact on determining the optimal solution. However, despite this, some conclusions can still be drawn. It is more easily seen on the heat map, that solutions with low CO and MO probability are less optimal. Solutions with a low MO probability perform worse than those with a low CO. This potentially suggests that for this benchmark test, the optimal solutions depend more on the MO. On average, the optimal permutations with the smallest distance centre between the CO probability of 0.7 - 0.9 and a MO probability of 0.2 - 0.45. The best result however was produced with a MO probability of 0.62 and CO probability of 1.0. Due to the large size of the problem, there are many local minimums in which the GA can get stuck. Using a large mutation rate of 0.62, this is more of a random search for the optimal solution than evolutionary. From this grid search analysis, the optimal probabilities were used again to investigate the elite parent routes from each generation.

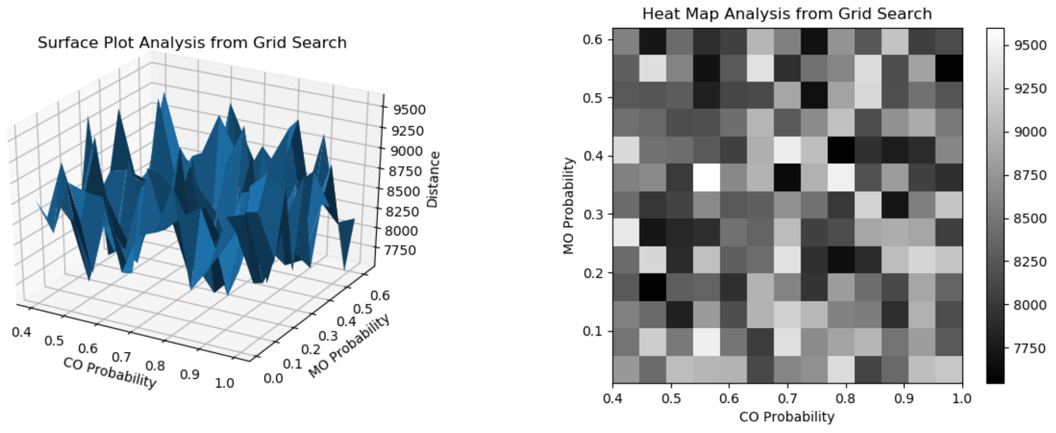


Figure 5: Grid search of MO and CO probabilities using ordered CO and RSM on the BERLIN52 test problem.

Using a population size of 2000 and a maximum generation count of 4000 an optimal solution of 7728.28m was found after 1614 generations using the RSM operator and the ordered CO operators. This result demonstrates the exceptional performance of the GA using ordered CO, RSM operator and the optimal CO and MO probabilities previously determined by the grid search analysis even on large routing problems. However, due to a limited volume capacity for a truck, realistic truck routes would have much fewer nodes than the BERLIN52 problem, and therefore an optimal solution would be found in a much quicker time. The test problem was used as an extreme case to explore the performance of the GA separately from the route produced by KMC. Given the evidence of the GA performance, the truck routes will be optimised, minimising the total distance travelled by the truck and the number of hours required by the driver to complete the route, in turn cutting overall costs.

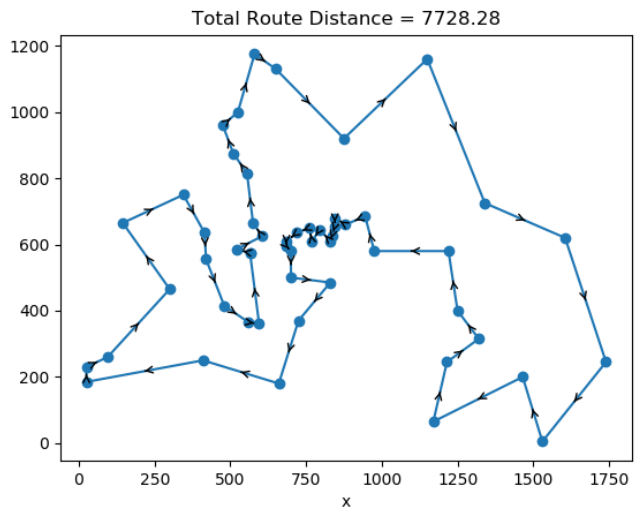


Figure 6: Visual representation of the optimised route for the BERLIN52 problem.

6.1 Drone Delivery Analysis

To further decrease the total costs of the 'last mile' problem, two different methods of drone delivery were investigated. A number of 35 random addresses with a range from 0 - 1200 for both the x and y coordinates of each address would be used for this investigation. The multi-address drone delivery method shows promising results with a fairly significant reduction in the electricity cost as seen in figure 7 method 1. In addition, the number of drones required is significantly less with a reduction of more than half when using the multi-address method over the single address one. Therefore, for larger deliveries, fewer trucks and fewer drivers will be required, consequently, this cuts costs and raises profits.

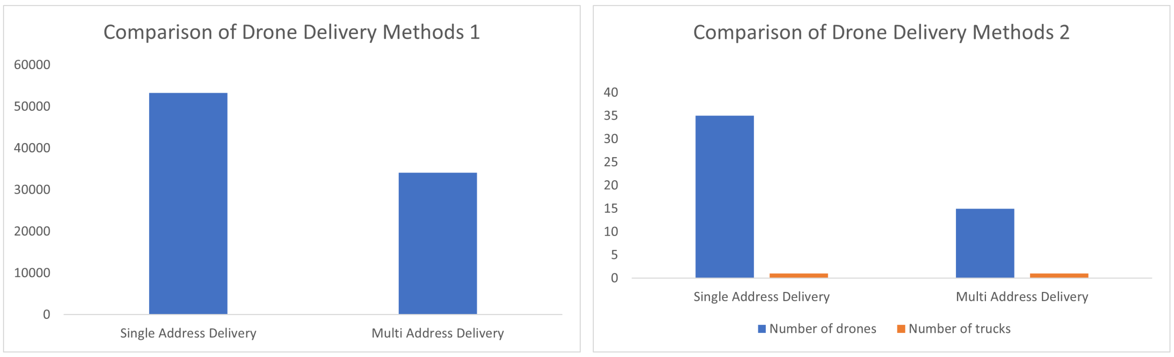


Figure 7: Grid search of MO and CO probabilities using ordered CO and RSM on the BERLIN52 test problem.

7 Conclusion

The software constructed determines the optimal: clusters, truck route and drone delivery method. The GA is shown to perform well on the benchmark test BERLIN52 almost coming within the known exact optimal route(7544m) at a total distance of 7728m. The multi-address drone delivery method has been shown to dramatically reduce costs in terms of the total electricity used by the drones, the number of drones needed and in turn, the number of trucks required. In conclusion, by using employing the use of the software, we reduce the costs of the 'last mile' to which 50% of the delivery costs are down to. It offers an alternative cheaper method to the traditional delivery methods of using just a truck.

8 Future Work

The software does not take into account battery capacity and how adding more parcels to a drone, even if it's within its payload capacity would consume more electricity than if a drone was carrying only one parcel. There are however some functions implemented which model drone power consumption that can be improved upon to make the approach more realistic. Currently, it is presumed the input data format is ordered such that the first address has the smallest x and y coordinates and the last address the largest. A pre-processing script to sort the address would reduce the total flight distance with the multi-address drone delivery method.

Bibliography

Abdilla, A., Richards, A. & Burrow, S. (2015), Power and endurance modelling of battery-powered rotorcraft, *in* '2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 675–680. pages 3

Agatz, N., Bouman, P. & Schmidt, M. (2015), 'Optimization approaches for the traveling salesman problem with drone, erim report series research in management', *Erasmus Research Institute of Management, Erasmus University Rotterdam* . pages 3

Ahmed, Z. H. (2010), 'Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator', *International Journal of Biometrics & Bioinformatics (IJBB)* **3**(6), 96. pages 8

Bai, K., Lu, D. & Shrivastava, A. (2011), Vector class on limited local memory (llm) multi-core processors, *in* '2011 Proceedings of the 14th International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)', IEEE, pp. 215–224. pages 7

Bamburry, D. (2015), 'Drones: Designed for product delivery', *Design Management Review* **26**(1), 40–48. pages 3

Boysen, N., Fedtke, S. & Schwerdfeger, S. (2020), 'Last-mile delivery concepts: a survey from an operational research perspective', *OR Spectrum* pp. 1–58. pages 3

Chang, Y. S. & Lee, H. J. (2018), 'Optimal delivery routing with wider drone-delivery areas along a shorter truck-route', *Expert Systems with Applications* **104**, 307–317. pages 4

Deb, K. & Spears, W. M. (1997), 'C6. 2: Speciation methods', *Handbook of Evolutionary Computation. Institute of Physics Publishing* . pages 8

Dorling, K., Heinrichs, J., Messier, G. G. & Magierowski, S. (2016), 'Vehicle routing problems for drone delivery', *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **47**(1), 70–85. pages 3, 4

Halim, A. H. & Ismail, I. (2019), 'Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem', *Archives of Computational Methods in Engineering* **26**(2), 367–380. pages 4

Hassanat, A., Almohammadi, K., Alkafaween, E., Abunawas, E., Hammouri, A. & Prasath, V. (2019), 'Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach', *Information* **10**(12), 390. pages 9

Hong, I., Kuby, M. & Murray, A. T. (2018), 'A range-restricted recharging station coverage model for drone delivery service planning', *Transportation Research Part C: Emerging Technologies* **90**, 198–212. pages 3

Keeney, T. (2015), 'Amazon drones could deliver a package in under thirty minutes for one dollar', *ARK INVEST* pp. 12–01. pages 3

Khoufi, I., Laouiti, A. & Adjih, C. (2019), 'A survey of recent extended variants of the traveling salesman and vehicle routing problems for unmanned aerial vehicles', *Drones* **3**(3), 66. pages 3

Konert, A., Smereka, J. & Szarpak, L. (2019), 'The use of drones in emergency medicine: practical and legal aspects', *Emergency medicine international* **2019**. pages 3

Laporte, G. & Nobert, Y. (1987), Exact algorithms for the vehicle routing problem, *in* 'North-Holland Mathematics Studies', Vol. 132, Elsevier, pp. 147–184. pages 4

Lee, J. (2017), Optimization of a modular drone delivery system, *in* '2017 Annual IEEE International Systems Conference (SysCon)', IEEE, pp. 1–8. pages 3

Levy, J. I., Buonocore, J. J. & Von Stackelberg, K. (2010), 'Evaluation of the public health impacts of traffic congestion: a health risk assessment', *Environmental health* **9**(1), 1–12. pages 3

MacKay, D. J. & Mac Kay, D. J. (2003), *Information theory, inference and learning algorithms*, Cambridge university press. pages 7

Mahfoud, S. W. & Goldberg, D. E. (1995), 'Parallel recombinative simulated annealing: A genetic algorithm', *Parallel computing* **21**(1), 1–28. pages 8

Toth, P. & Vigo, D. (2002), *The vehicle routing problem*, SIAM. pages 4

Wang, S., Lu, Z., Wei, L., Ji, G. & Yang, J. (2016), 'Fitness-scaling adaptive genetic algorithm with local search for solving the multiple depot vehicle routing problem', *Simulation* **92**(7), 601–616. pages 9

Wang, Y., Zhang, D., Liu, Q., Shen, F. & Lee, L. H. (2016), 'Towards enhancing the last-mile delivery: An effective crowd-tasking model with scalable solutions', *Transportation Research Part E: Logistics and Transportation Review* **93**, 279–293. pages 3

9 Supporting Information

Algorithm 1: K-Means Clustering

```
while All addresses within flight range do
    for cluster no $\in [1, no.\ of\ clusters]$ do
        idx = random number
        centroids[cluster no] = address-list[idx]
        id of address-list = cluster no
    while unconverged = true do
        previous-centroids-list = centroid-list
        for adr-no $\in [1, no - of - addresses]$ do
            minimum distance = FLT-MAX
            for cluster no $\in [1, no.\ of\ clusters$ do
                Calculate euclidian distance between adr-no and cluster no
            Match id of cluster no to id of adr-no with smallest distance
        for i $\in [1, no - of - addresses]$ do
            initialise no-of-ids, easting-sum and northing-sum to 0
        for adr-no $\in [1, no.\ of\ clusters]$ do
            Add +1 to no-of-points to relevant id position
            Add easting sum to relevant id position
            Add northing sum to relevant id position
        for cluster-no $\in [1, no.\ of\ clusters]$ do
            centroids[cluster-no]-¿easting value = easting sum[cluster-no] /
              no-of-points[cluster-no]
            centroids[cluster-no]-¿northing value = northing sum[cluster-no] /
              no-of-points[cluster-no]
        if converge = true then
            if all addresses within flight range then
                break
        else
            no of clusters += 1
```

Algorithm 3: Genetic Algorithm

```
Input: (int) conv count, (func) fitness, (func) initialise generations, (func) evaluate
       circuit
Output: (Result) result
while generation count $\leq$ maximum generation count do
    for $i \in [1, population\ \ size]$ do
        Initialise population vectors with randomised circuit vectors
    for $chromosome \in generation$ do
        Calculate fitness
    Store the index of the vector with the best fitness vector
    if |max − previous max| < tol then
        conv count + = 1
    else
        conv  cnt = 0
    previous max = max
    if conv count > max conv or generation count = maximum generation count then
        Return best circuit-vector and fitness value
        break
    n = 1
    while n < population size do
        Select two different chromosomes
        if random number 1 < user given CO probability then
            CO
        for $i \in range\ 2$ do
            if random number 2 < user given MN probability then
                MN
        for $i \in range\ 2$ do
            evaluate chromosome if evaluation = true then
                $n+ = 1$
        generation count + = 1
        swap(generation, new generation)
    return result
```

Algorithm 5: Multi Address Drone Delivery

```
Input: (int) k, (vector(vector(address-metadata*))) cl-data,
       (vector(address-metadata)) optimal truck route, (double) payload capacity,
       (double), max drone distance
Output: tuple(double, double, int, int, int) multi adr drone
for cluster $\in$ $clusters$ do
    centroid id = optimal-route[cluster].id, j = 0
    while j ¡ cl-data[centroid id.size do
        if cl-data[centroid id].size == 1 then
            distance sum = 2 * distance(optimal route[cluster], cl-data[centroid id][0]
            mass sum = cl-data[centroid id][0]$-$ $>$parcel mass
            total distance += distance sum, total mass += mass sum
            break
        double d1, d2, m1, distance sum = 0, mass sum = 0
        distance sum += distance(optimal route[cluster], cl-data[centroid id][j]
        distance-tracker.pushback(distance sum)
        mass sum += cl-data[centroid-id][j]$-$ $>$parcel-mass
        mass-tracker.pushback(mass sum), j++
        bool constrained = false
        while !constrained do
            if j == cl-data[centroid-id].size - 1 then
                distance sum += distance tracker[cnt-distance]
                mass sum = sum(mass tracker[0], mass tracker[cnt-mass])
                break
            d1 = distance(cl-data[centroid id][j], cl-data[centroid id][j + 1]
            d2 = distance( cl-data[centroid id][j + 1], optimal route[cluster])
            distance-tracker.pushback(d2)
            if distance sum + d1 + d2 $>$ max drone distance then
                distance sum += distance tracker[cnt-distance]
                mass sum = sum(mass tracker[0], mass tracker[cnt-mass])
                constrained = true, j++
                break
            m1 = cl-data[centroid id][j + 1]$-$ $>$parcel-mass
            mass-tracker.pushback(m1)
            if mass sum + m1 $>$ payload capacity then
                distance sum += track distance[cnt-distance]
                mass sum = sum(mass tracker[0], mass tracker[cnt-mass])
                constrained = true, j++
                break
        distance sum += d1, mass sum += m1, j++
    all distances += distance sum, all masses += mass sum , cnt-distance = 0,
     cnt-mass = 0
```