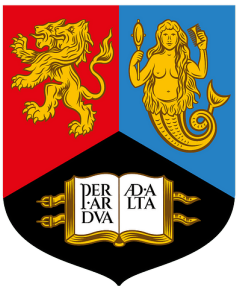

UNIVERSITY OF BIRMINGHAM

A Comparison of Approaches to Combinatorial Optimisation for Multi-Day Route Planning

Jacob Luck
2338114

B.Sc. Computer Science with a Year in Industry
Project Supervisor - Leandro Minku

April 2025
xxxx Words

Abstract

Write abstract

Contents

1 Introduction

1.1 Motivation

Write about reasoning for this project, can copy a little bit over from presentation slides. Explain problem informally. Link into aims and objectives.

1.2 Aims and Objectives

Explin goals of the project, link in to methodology.

1.3 Methodology

Briefly explain how the project will be carried out. A more thorough description can be provided later on, i.e., when describing algorithms.

Mention code being available from github. Include a code segment and explain how figure captions denote filepath within the repository.

1.4 Summary

Explain what is in the rest of the report. "In this report I shall...", cover each section, etc.

2 Problem Formulation

lstmisc

2.1 Problem Description

Given a positive integer d and a graph $G = (V, E)$, where V is set of locations including a designated starting point s and E is a set of weighted edges linking every location to every other location, find a route that:

1. Visits all nodes $V \setminus s$ once.
2. Starts and finishes at s, having visited it d times, without ever visiting consecutively.
3. Minimises both the cumulative edge weights in the route and the variance in cumulative weight between each visit to s.

2.2 Inputs, Outputs and Design Variables

Update so durations aren't in the graph, instead something like D, mapping a duration to each location.

Inputs:

- d: The number of times s should be visited in a route. Contextually, d represents the number of days a tourist will spend on their trip. $d \in \mathbb{Z}, d > 0$
- $G = (V, E)$: A pair comprising:
  - V: A set of nodes representing locations the tourist would like to visit. $v \in V, v = (x, y, t)$, a triple comprising:
    - x: Longitude, indicating the location's geographic east-west position on the earth $x \in \mathbb{Q}, -180 \leq x \leq 180$.
    - y: Latitude, indicating the location's geographic north-south position on the earth $y \in \mathbb{Q}, -90 \leq y \leq 90$.

- t: Duration, in minutes, indicating how much time to spend at this location.

  $t \in \mathbb{Z}, t > 0$.

- E: A set of edges $e \in E$ that connects every node to every other node, bidirectionally. $e = (v_1, v_2, w)$, a triple comprising:

  - v_1: A location representing the origin of the edge.

    $v_1 \in V$.

  - v_2: A location representing the destination of the edge.

    $v_2 \in V$.

  - w: A weight indicating the sum of the time it takes to travel from v_1 to v_2 and the time the tourist wishes to spent at v_2.

    $w \in Z, w > 0$.

- s: Starting point that should be visited d times. Contextually, s represents where the tourist is staying and will return to at the end of each day.

  $s \in V$.

Outputs:

- R: A valid route satisfying all constraints, represented as an ordered sequence of locations.

  $R = [r_1, r_2, \ldots, r_n], r_i \in V$.

2.3 Objective Function

As previously mentioned in the Problem Description, our goal is to find a route that minimises the cumulative weight and the variance in route weight between each visit to s. To accomplish this the following cost function is applied to each route:

$$Cost(R) = W/d \times (1 + \sigma^2) \tag{1}$$

Where W is the sum of the weights of all edges traversed in the route and σ^2 is the variance of the sum of weights between each visit to s:

$$W = \sum_{i=0}^{n-1} w(r_i, r_{i+1}), r_i \in R \tag{2}$$

Where $w(r_i, r_{i+1})$ is the weight of the edge between r_i and r_{i+1}.

$$\sigma^2 = \frac{\sum_{i=0}^{d}(x_i - \mu)}{d}, x_i \in X \tag{3}$$

Where R is divided into sections between each visit to s and X is a list of the sum of weights within these sections.
μ is the mean cumulative weight of each x_i.

2.4 Constraints

A valid solution must satisfy the following constraints:

- The route must visit every node $v \in \{V \setminus s\}$ exactly once:

  $\forall_{v \in \{V \setminus s\}}, |\{i \in \{1, \ldots, n\} : r_i = v\}| = 1$

- The route must visit s exactly d times:

  $|\{i \in \{1, \ldots, n\} : r_i = s\}| = d$

- The route must not visit s consecutively:

  $\forall_{i \in \{1, \ldots, n-1\}}, r_i \neq r_{i+1}$

- The route must end at s:

  $r_n = s$

3 Literature Review

Plan (and write) literature review

Remember to write about the strengths and weaknesses of existing work. At the end of this chapter you can then give a summary of the gaps that you'll be trying to improve with your work, and on the strengths that you will be maintaining in your work.

This literature review aims to explore existing research and approaches to other combinatorial optimisation problems. There is extensive previous research on various combinatorial problems, for example, the Travelling Salesman Problem, Vehicle Routing Problem and Tourist Trip Design Problem. It is important to understand how these problems are similar to the one presented in this report, as well as where those similarities end. By gaining an understanding of the strengths and limitations of existing approaches to similar problems, we can make better informed decisions regarding which approaches to investigate, how they may be adapted to suit our specific constraints and how they might be implemented in practice. While the approaches taken to these problems may not be directly applicable to our own, it is likely we can adapt their techniques to suit the specific constraints of this problem.

The Travelling Salesman Problem (TSP) is perhaps one of the most studied optimisation problems. This extensive research on the problem has acted as an 'engine of discovery for general-purpose techniques' offering large contributions across a wide range of mathematics.

cite tsp: a computational study, pg 40–41

The TSP can be described simply as: Given a set of locations and the cost of travel between them, find the shortest route that visits each location and returns to the start

cite tsp: a computational study, pg 1

.

There are clear similarities between the TSP and our own problem, both involve finding a route that visits every node, while attempting to minimise the time taken travelling said route. In fact, with an input of $d = 1$ our problem becomes the TSP, with greater values of d introducing additional complexity. With this in mind we are able to investigate existing approaches to the TSP and consider how they can be applied to our problem. The TSP is proven to be NP-hard

cite introduction to algorithms, pg 1096–1097

, meaning that there are no known algorithms capable of solving the problem in polynomial time. Considering this, and the aforementioned similarities to our own problem, we can conclude that our problem is also NP-hard.

Discuss how knowledge of NP-hardness can be used to decide which algorithms to investigate.

Discuss a couple of solutions/types of solution

sentence about TSP variants, leads into paragraph about mTSP

3.1 Classical Traveling Salesman Problem (TSP)

- Definition and mathematical formulation
- Complexity analysis and NP-hardness
- Key solution approaches
- Relevance and limitations in relation to our specific problem

Recommended Literature:

- Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2006). *The Traveling Salesman Problem: A Computational Study.* Princeton University Press.
- Laporte, G. (1992). "The traveling salesman problem: An overview of exact and approximate algorithms." *European Journal of Operational Research*, 59(2), 231-247.

- Lin, S., & Kernighan, B. W. (1973). "An effective heuristic algorithm for the traveling-salesman problem." *Operations Research*, 21(2), 498-516.

- Helsgaun, K. (2000). "An effective implementation of the Lin–Kernighan heuristic." *European Journal of Operational Research*, 126(1), 106-130.

3.2 Multiple Traveling Salesman Problem (mTSP)

- Extension of the TSP with multiple agents
- Mathematical formulation differences from TSP
- Application to multi-day planning scenarios
- Connection to our requirement of visiting the starting point d times

Recommended Literature:

- Bektas, T. (2006). "The multiple traveling salesman problem: an overview of formulations and solution procedures." *Omega*, 34(3), 209-219.

- Kara, I., & Bektas, T. (2006). "Integer linear programming formulations of multiple salesman problems and its variations." *European Journal of Operational Research*, 174(3), 1449-1458.

- Gavish, B., & Srikanth, K. (1986). "An optimal solution method for large-scale multiple traveling salesmen problems." *Operations Research*, 34(5), 698-717.

- Carter, A. E., & Ragsdale, C. T. (2006). "A new approach to solving the multiple traveling salesperson problem using genetic algorithms." *European Journal of Operational Research*, 175(1), 246-257.

3.3 Vehicle Routing Problem (VRP) and Variants

- Basic VRP definition and formulation
- Vehicle Routing Problem with Multiple Trips (VRPMT)

- Capacitated VRP and other variants
- Relevance to our balanced route planning requirement

Recommended Literature:

- Toth, P., & Vigo, D. (Eds.). (2002). *The Vehicle Routing Problem.* SIAM Monographs on Discrete Mathematics and Applications.
- Cattaruzza, D., Absi, N., & Feillet, D. (2016). "Vehicle routing problems with multiple trips." *4OR*, 14(3), 223-259.
- Brandão, J., & Mercer, A. (1997). "A tabu search algorithm for the multi-trip vehicle routing and scheduling problem." *European Journal of Operational Research*, 100(1), 180-191.
- Olivera, A., & Viera, O. (2007). "Adaptive memory programming for the vehicle routing problem with multiple trips." *Computers & Operations Research*, 34(1), 28-47.

3.4 Tourist Trip Design Problem (TTDP)

- Problem definition focusing on tourist-specific constraints
- Time-dependent considerations and point-of-interest selection
- Personalization aspects in tourist routing
- Direct applicability to our problem's tourism context

Recommended Literature:

- Vansteenwegen, P., Souffriau, W., & Van Oudheusden, D. (2011). "The orienteering problem: A survey." *European Journal of Operational Research*, 209(1), 1-10.
- Gavalas, D., Konstantopoulos, C., Mastakas, K., & Pantziou, G. (2014). "A survey on algorithmic approaches for solving tourist trip design problems." *Journal of Heuristics*, 20(3), 291-328.

- Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., & Van Oudheusden, D. (2013). "The planning of cycle trips in the province of East Flanders." *Omega*, 41(3), 522-531.

- Garcia, A., Vansteenwegen, P., Arbelaitz, O., Souffriau, W., & Linaza, M. T. (2013). "Integrating public transportation in personalised electronic tourist guides." *Computers & Operations Research*, 40(3), 758-774.

3.5 Multi-Objective Optimization in Routing Problems

- Balancing competing objectives (like total weight vs. variance)
- Pareto optimality concepts
- Solution approaches for multi-objective routing
- Applicability to our dual-objective function

Recommended Literature:

- Jozefowiez, N., Semet, F., & Talbi, E. G. (2008). "Multi-objective vehicle routing problems." *European Journal of Operational Research*, 189(2), 293-309.

- Paquete, L., & Stützle, T. (2006). "A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices." *European Journal of Operational Research*, 169(3), 943-959.

- Laporte, G., Semet, F., Matl, P., & Voß, S. (2018). "Multi-objective vehicle routing problem." *Operations Research Perspectives*, 5, 50-57.

- Coello, C. A. C., Lamont, G. B., & Van Veldhuizen, D. A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer.

3.6 Balance-Oriented Routing Problems

- Problems focusing on workload balancing

- Min-max objectives in routing
- Variance minimization approaches
- Connection to our goal of minimizing variance between trips

Recommended Literature:

- Jozefowiez, N., Semet, F., & Talbi, E. G. (2009). "An evolutionary algorithm for the vehicle routing problem with route balancing." *European Journal of Operational Research*, 195(3), 761-769.
- Dell'Amico, M., Monaci, M., Pagani, C., & Vigo, D. (2007). "Heuristic approaches for the fleet size and mix vehicle routing problem with time windows." *Transportation Science*, 41(4), 516-526.
- Lee, T. R., & Ueng, J. H. (1999). "A study of vehicle routing problems with load-balancing." *International Journal of Physical Distribution & Logistics Management*, 29(10), 646-657.
- Liu, R., Xie, X., Augusto, V., & Rodriguez, C. (2013). "Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care." *European Journal of Operational Research*, 230(3), 475-486.

3.7 Time-Dependent Routing Problems

- Integration of visit durations into routing decisions
- Time windows and scheduling constraints
- Solution approaches for time-dependent problems
- Relevance to our edge weight definition that incorporates visit duration

Recommended Literature:

- Ichoua, S., Gendreau, M., & Potvin, J. Y. (2003). "Vehicle dispatching with time-dependent travel times." *European Journal of Operational Research*, 144(2), 379-396.

- Donati, A. V., Montemanni, R., Casagrande, N., Rizzoli, A. E., & Gambardella, L. M. (2008). "Time dependent vehicle routing problem with a multi ant colony system." *European Journal of Operational Research*, 185(3), 1174-1191.

- Hashimoto, H., Yagiura, M., & Ibaraki, T. (2008). "An iterated local search algorithm for the time-dependent vehicle routing problem with time windows." *Discrete Optimization*, 5(2), 434-456.

- Figliozzi, M. A. (2012). "The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics." *Transportation Research Part E: Logistics and Transportation Review*, 48(3), 616-636.

3.8 Synthesis and Research Gaps

- Comparison of problem characteristics across reviewed literature
- Key methodological approaches applicable to our problem
- Identification of research gaps in addressing our specific problem constraints
- Potential directions for adaptation of existing methodologies

Recommended Literature:

- Laporte, G. (2009). "Fifty years of vehicle routing." *Transportation Science*, 43(4), 408-416.

- Cordeau, J. F., Gendreau, M., Laporte, G., Potvin, J. Y., & Semet, F. (2002). "A guide to vehicle routing heuristics." *Journal of the Operational Research Society*, 53(5), 512-522.

- Eksioglu, B., Vural, A. V., & Reisman, A. (2009). "The vehicle routing problem: A taxonomic review." *Computers & Industrial Engineering*, 57(4), 1472-1483.

- Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). "Heuristics for multi-attribute vehicle routing problems: A survey and synthesis." *European Journal of Operational Research*, 231(1), 1-21.

3.9 Conclusion

- Summary of most relevant approaches

- Recommendation for methodological direction

- Justification for selected approach based on literature findings

Recommended Literature:

- Gendreau, M., Potvin, J. Y., Bräumlaysy, O., Hasle, G., & Løkketangen, A. (2008). "Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography." In *The vehicle routing problem: Latest advances and new challenges* (pp. 143-169). Springer.

- Bräysy, O., & Gendreau, M. (2005). "Vehicle routing problem with time windows, Part I: Route construction and local search algorithms." *Transportation Science*, 39(1), 104-118.

- Glover, F., & Kochenberger, G. A. (Eds.). (2003). *Handbook of Metaheuristics*. Springer.

- Talbi, E. G. (2009). *Metaheuristics: From Design to Implementation*. John Wiley & Sons.

4 Algorithms Investigated

There should be a link either here or at end of literature which forms the basic for different methods (clustering, routing, trip generation).

Paragraph describing different types of algorithm used (Routing then cluster, Cluster then Routing, Genetic, etc.)

Remember to justify the choice of algorithms. You may also need to explain how to adopt these algorithms in your work. A figure showing the ralationship between different components of your work may also help.

Include code for evaluating a route here.

4.1 Clustering

Clustering as a concept can be described as 'the unsupervised classification of patterns (observation, data items, or feature vectors) into groups (clusters)'

cite Data Clustering: A review, A.K. Jain

. In our problem, clustering will be used to group locations together to form an itinerary for each day of the trip. These clusters, or days, will then be used as an input for some routing algorithm, which will try and find an optimal route for each day. These routes can then be combined to form a complete route for the trip.

```
# np.ix_([1,2,3], [1,2,3]) returns [[[1],[2],[3]],[1,2,3]]
# Which can they be used to easily form subgraphs for each cluster.
graphs = [graph[np.ix_(indexes, indexes)] for indexes in clusters]
durations_each_day = [durations[indexes] for indexes in clusters]

route = np.empty(0, dtype=int)
for sub_graph, cluster, sub_durations in zip(graphs, clusters,
                                             durations_each_day):
    num_locations = sub_graph.shape[0]
    num_days = 1
    sub_route = routing_function(num_locations, num_days, sub_graph,
                                 sub_durations)

    route = np.concatenate((route, cluster[sub_route]))
return route
```

Figure 1: From Clustering.find_route_from_clusters in algorithms\clustering.py

The goal of our clustering algorithms is to find a set of clusters that, when combined with some routing algorithm, will produce a route that minimises the cost function.

The clustering algorithms implemented in this project are: K-Means, Genetic Clustering and Genetic Centroid-based Clustering.

4.1.1 K-Means

K-Means is an iterative clustering algorithm that defines its clusters using a set of centroids (means) which are given a location in the input space, a given location is assigned to the cluster of the 'closest' centroid. The algorithm starts by initialising random centroids and iteratively improving the clustering from there, continuing until the algorithm converges (on a local optimum) or an iteration limit is reached. While the algorithm may not find the best solution, it is rather quick, with a time complexity of $O(mnki)$, where where m is the number of locations, n is the dimensionality of the input, k is the number of clusters, and i is the number of iterations iterations

cite Algorithm AS 136: A K-Means Clustering Algorithm

. Our inputs will always contain only two dimensions, and we will be setting a maximum number of iterations, this makes both n and i constants allowing us to simplify the time complexity to $O(mk)$.

In our implementation of K-Means we will initialise our centroids by randomly selecting unique locations from our input, and placing our centroids at their coordinates. A different approach was considered, which involved initialising the centroids with random coordinates in a similar area to the input, however this had the potential to create clusters with zero locations assigned to them resulting in invalid trips. By starting with locations from the input, we can be sure that all clusters have at least one location assigned to them. We will be using the coordinates of our locations to calculate the Euclidean distances between locations and centroids, these locations will be assigned to the cluster of the closest centroid:

```
# Gets a matrix of distances from each location to each centroid
distances = np.linalg.norm(
    coordinates[:, np.newaxis, :2] - centroids[:, :2], axis=2)

# For each location (index in distance matrix) gets the index for the
# centroid with the smallest distance
clusters = np.argmin(distances, axis=1)
return clusters
```

Figure 2: from Clustering._assign_nodes_to_centroid in algorithms\clustering.py

After assignment, the centroids are recalculated such that their coordinates are the average of all locations assigned to their cluster:

```
computed_means = np.empty((num_days, 2))

for i in range(num_days):
    cluster = coordinates[coordinates[:, 2] == i, :2]
    computed_means[i] = cluster.mean(axis=0)
return computed_means
```

Figure 3: from KMeans._compute_means in algorithms\clustering.py

These steps of cluster assignment and centroid recalculation are repeated until either a maximum allowed number of iterations is reached, or until the algorithm converges on an optimum solution. Our convergence criterion is that the centroids stop changing between iterations, i.e., the centroids are the same as the previous iteration:

```
for _ in range(self.maximum_iterations):
    cluster_assignments = self._assign_nodes_to_centroid(coordinates,
                                                         means)
    coordinates[:, 2] = cluster_assignments

    if np.array_equal(cluster_assignments, previous_clusters):
        break
    previous_clusters = np.array(cluster_assignments, copy=True)

    means = self._compute_means(coordinates, num_days)

return cluster_assignments
```

Figure 4: from KMeans.find_clusters in algorithms\clustering.py

Below is an example of the K-Means algorithm run on an input with 25 points of interest around London over seven days:

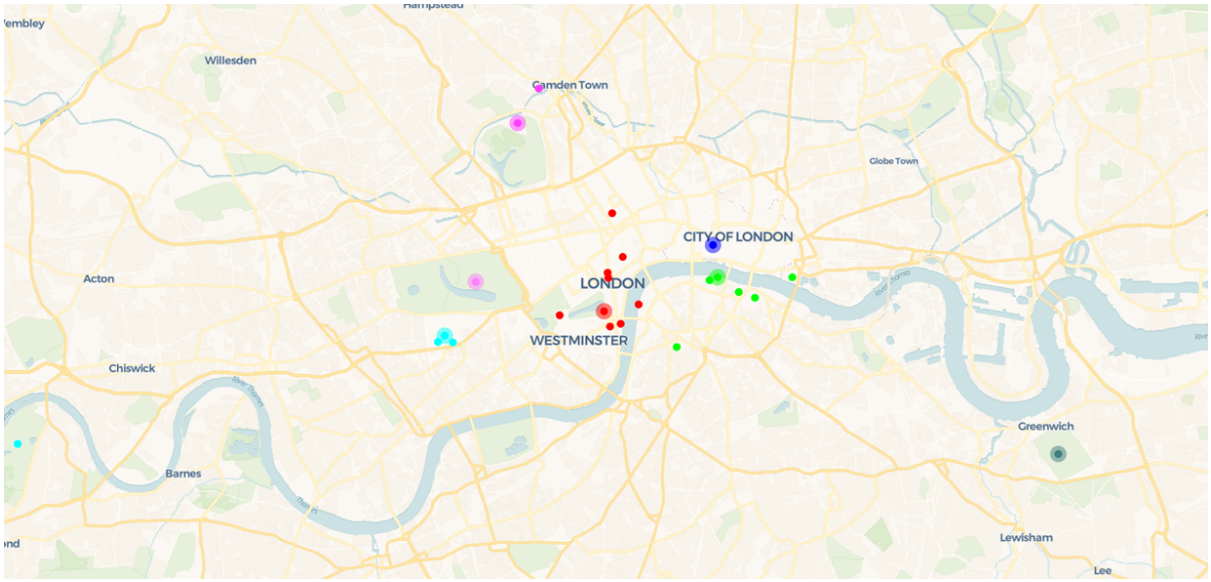

Figure 5a: Step 1, Initial centroid positions and cluster assignments.

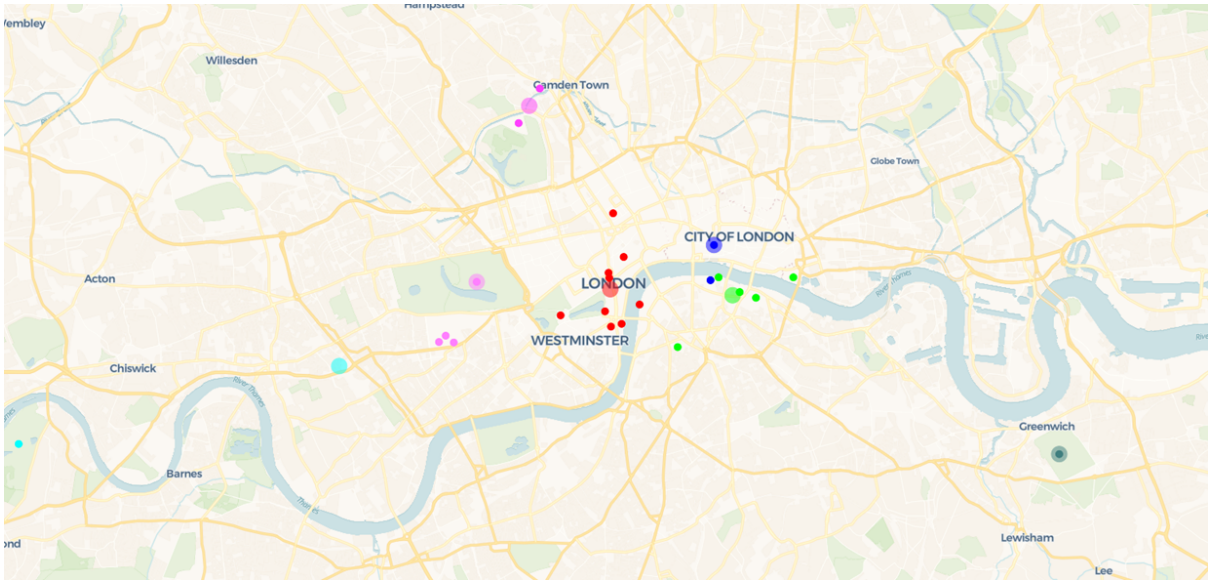

Figure 5b: Step2, Centroids have been updated, locations are reassigned to their closest centroid.

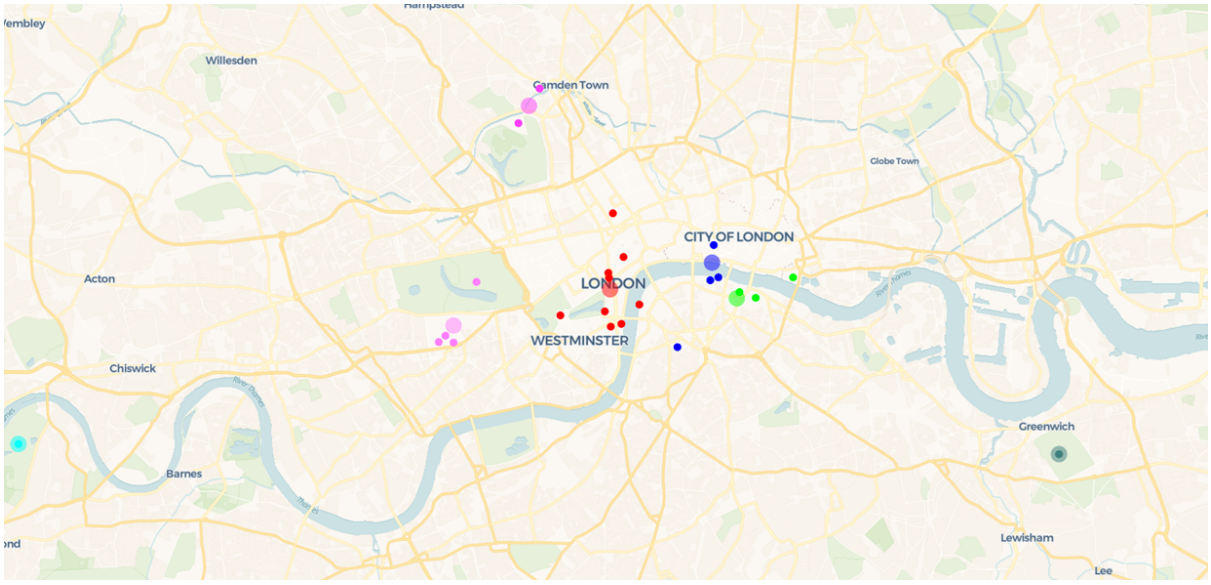

Figure 5c: Step3, Centroids updated again and locations updated.

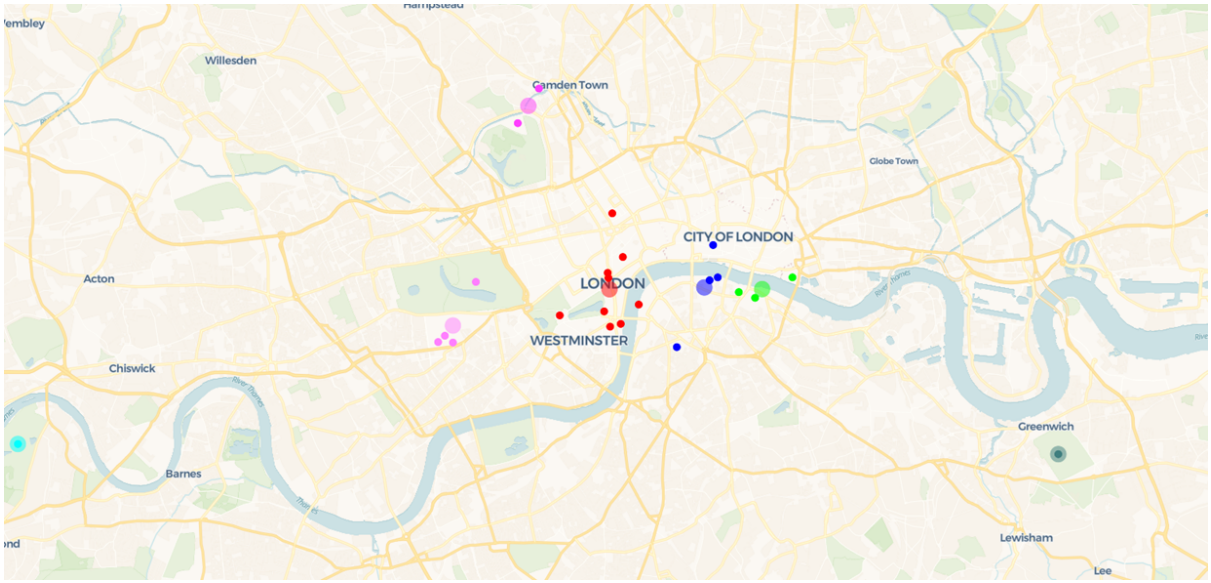

Figure 5d: Step4, Centroids have updated and no locations have changed cluster, a solution has been found.

It is worth noting that K-Means only forms these clusters based on Euclidean distances, grouping together locations that are close geographically. However, as formally described in the Objective Function section of the Problem Formulation, a good trip will minimise both the route length of the trip and the variance between time spent each day. K-Means does not aim to optimise for the variance between days, it doesn't even consider the time spent at each location. Furthermore, while each cluster might be optimised for distance, how close two locations are may not reflect the travel time between them. While K-Means

does not intentionally optimise for variance between days or consider travel time between locations, it was still chosen for this project out of curiosity as to how effective a heuristic it might provide. Hopefully it will offer a simple and computationally efficient baseline for comparison with more complex algorithms.

4.1.2 Genetic Clustering

Explain genetic clustering

Genetic clustering applies genetic algorithms to attempt and find the best assignment of locations to clusters. Genetic algorithms are a type of evolutionary algorithm that aims to mimic biological evolution to find an optimal solution. They involve creating a population of potential solutions (individuals) and iteratively improving the population through selection (keeping the best individuals in the population, akin to natural selection), crossover (combining individuals to create offspring, akin to sexual reproduction), and mutation (randomly altering the genomes of individuals in the population, akin to biological mutation).

For us to perform selection, and find the best solutions in a population, we need to assign some fitness to each individual. To do this we will combine the cluster assignments with a chosen routing algorithm, and then apply our cost function to the route found. This cost will be used to rank our population, helping us to find clusters that help produce a good trip.

The performance of Genetic algorithms is highly dependent on its hyperparameters: mutation rate, determining how common mutation is; crossover rate, determining how often offspring are created via crossover, as opposed to new additions to the population; population size, determining how many individuals there are per generation; number of generations, determining how many generations will be evolved to reach a solution; crossover method used, determining how crossover is performed to create offspring; and in our case, the routing algorithm used, which may impact how clusters are used to form routes. These hyperparameters impact both the runtime of the algorithm and the exploration of the search space, indirectly impacting the quality of the solution. The

time complexity of genetic clustering largely depends on the chosen routing algorithm, b Genetic clustering has a time complexity of $O(gpn)$, with g being the number of generations, p being the population size and n being the number of locations.

For this genetic clustering, an individual is represented by a genome, which will provide a mapping that assigns each location to a cluster, for example:

Genome: [0, 0, 0, 1, 1, 1, 2, 2, 2]

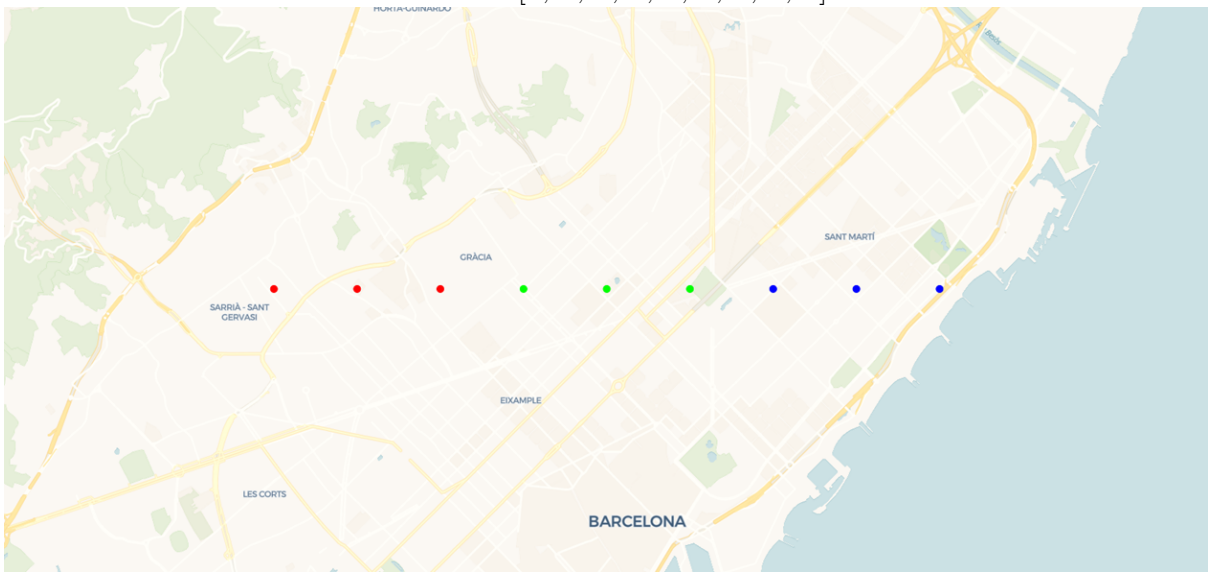

Figure 6: An example of how a genome corresponds to assigning clusters.

These genomes are our target for performing crossover and mutations. We begin our evolution process by randomly generating an initial population of individuals. From there, we repeat the following steps until we reach a maximum number of generations:

1. Evaluate the fitness of each individual in the population.
2. Select individuals to be parents, based on their fitness.
3. Create offspring from the selected parents, using crossover and mutation.
4. Replace the current population with the new population of offspring.

4.1.3 Genetic Centroid-based Clustering

Explain genetic centroid-based clustering and how it differs from general clustering.

4.2 Routing

Explain purpose of routing/goal of algorithms.

4.2.1 Brute Force

Write brute force explanation

The brute force algorithm is an exhaustive algorithm that tries every possible route to find one with the least cost. By checking every route it is guaranteed to find the optimal route, however, its computational cost becomes impractical as input size grows, having a time complexity of $O(n!)$.

Maybe cite time complexity of brute force?

Considering we will be comparing algorithms based on their speed and the quality of their results, brute force is a useful benchmark, providing a lower bound for speed and an upper bound for quality.

In our brute force implementation, where n is the number of locations in the route, we will be generating all $n-1!$ permutations of the set $\{1, 2, \ldots, n-1\}$, with each permutation representing the order of locations visited in a route. Each route will be evaluated according to our optimisation function, and the route with the lowest cost will be returned. We only need to consider $n-1!$ permutations because all our routes will start and end at the same location.

4.2.2 Greedy Routing

Explain greedy routing algorithm

Greedy routing

4.2.3 Gift Wrapping

Explain gift wrapping algorithm

Something like: ”Once gift wrapping has found a convex hull, a greedy insertion algorithm is used to find the optimal route within the convex hull.”

4.2.4 Genetic Routing

Explain genetic routing

4.3 Route Insertion

Explain route insertion, how it is used in route planning and the goal of our algorithms.

4.3.1 Brute Force

Explain how brute force algorithm can be modified for route insertion.

4.3.2 Greedy Insertion

Explain how greedy algorithm can be modified for route insertion.

4.4 Trip Generation

Explain trip generation, how it is used in route planning and the goal of our algorithms.

4.4.1 Brute Force

Explain how brute force algorithm can be modified for trip generation.

4.4.2 Genetic Trip Generation

Explain genetic trip generation

5 Evaluation and Comparison

5.1 Methodology

5.1.1 Constraints

5.2 Results & Analysis

Gather data for London and Birmingham with POI

Write paragraph about experiment process. Comparison based on computation time and route evaluation. Describe how route is evaluated. Describe data being tested on.

Present comparison of different combinations of algorithms on different inputs.

Reflect about the questionns you are trying to answer with your evaluation. You can have onne subsection for each question that you are trying to answer. It's also important to justify your choices when it comes to the methodology used for evaluation.

6 Conclusion and Future Work

Write conclusion, discuss results, comparison of algorithms, etc.

6.1 Project Reflection

Reflect on the project, what went well, what didn't go well, what I would do differently. This should lead into future work.

6.2 Future work

Discuss further work, what I will be doing to improve the project

Discuss spectral clustering

Discuss Neural Networks

7 Bibliography