Delivery Hub Optimiser

Introduction

The Delivery Hub Optimiser program has been designed as per the specification provided for the assignment, with some assumptions changed or removed entirely. Some steps to optimise the process were taken, however some were left out for flexibilities sake, and no doubt some were missed due to no rigorous profiling.

Assumptions

The assumptions made in the creation of this program are as follows:

1. The cost of travel from A → B is proportional to the great circle distance.
2. The cost of travel from A → B is inversely proportional to the population of B.
    a) The demand from a location for goods is proportional to population.
3. Trucks only travel to one location from the hub at a time.
4. Only one hub will serve a location.

Justification:

(1) The distance to a point introduces a cost of fuel and driver time, both can be roughly approximated to be proportional to the distance travelled. Many other associated costs (i.e. the goods, warehouse costs etc.) are assumed to exist regardless, given the company wishes to operate.

(2) The population of a location is assumed to be proportional to the demand for goods from that location. As such population would be proportional to number of trucks needed to service that location per unit time. As such one can see an inverse proportionality to the optimal distance of the hub, derived from (1).

(3) Given the population of a city, and the likelihood for the need to have local distributors as a result (e.g. the Amazon pick-up service many newsagents offer) it seems reasonable for a truck to service only one city or town in a trip – at least for more popular services – when considering the capacity limits that exist.

(4) The hubs are intended to be large distribution centres, as such it is reasonable that for most locations one hub could handle demand from the location.

Code Design

An MVC architecture was utilised for the program as it suits the nature of the program very well: a set of data to be read, an algorithm or two to be ran, and an output to be displayed. The benefits being both to readability and the ease of changing out, or modifying, components of the code.

Hill Climb Algorithm

The hill climbing algorithm was implemented in a general sense, allowing it to be used with arbitrary fitness calculations provided by a function pointer—arguments, beyond the x and y coordinates in the hill's space,

for which can be supplied via a variadic template argument list. In this assignment, this functionality was used to allow the hill climbing algorithm to be used over a space in which both physical distance and population of locations affected the fitness value of a given point.

K-Means Clustering Algorithm

The k-means clustering algorithm was utilised to construct regions in an arbitrary space that separate points into clusters in that space. Like the Hill Climb implementation, this algorithm was implemented generally, with the program then utilising it in its specific case with a function pointer for calculating distance between points. The initialisation method used is called k-means++, as an improvement to the initialisation used normally in k-means clustering implementations – both to the chance of finding most (/more) optimal clustering and to the time taken to reach convergence.

The specific use case of the algorithm calculated distance in its space to be a value of:

$$distance = worldDistance * \sqrt{clusterPopulation + locationPopulation}$$

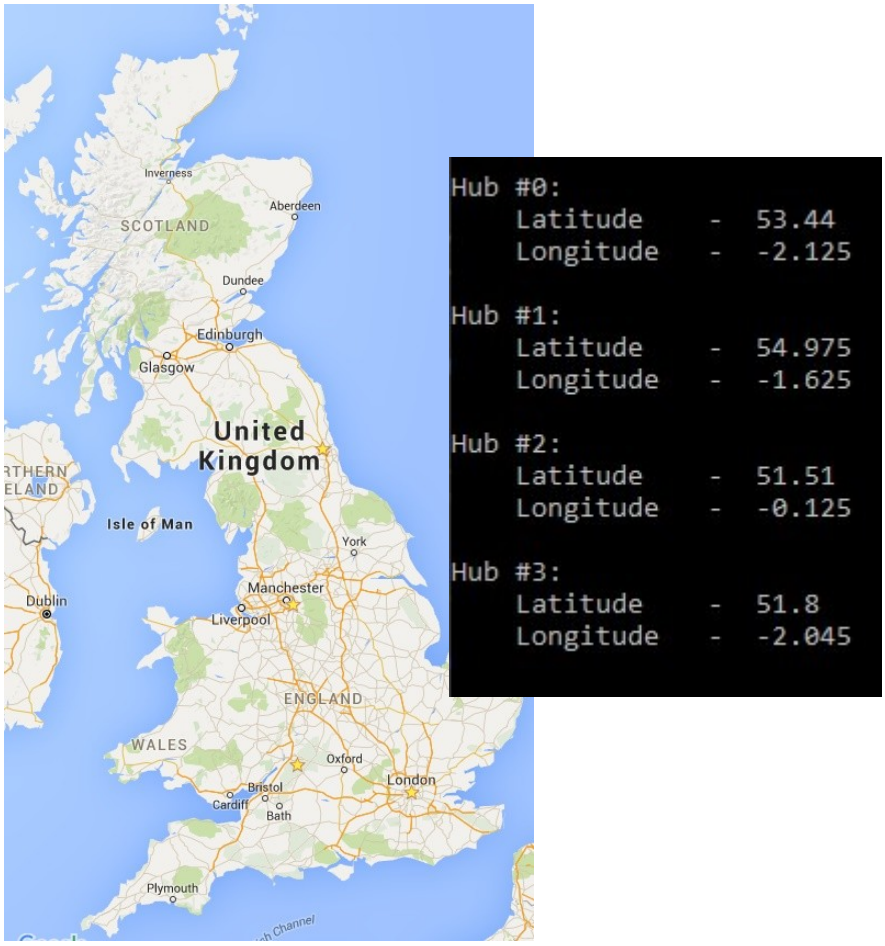
The justification being that it was undesirable for dense locations such as London to be matched up solely spatially with other nearby locations given assumption (4). The factor the physical distance is multiplied by ensures that all points are more likely to favour the clusters that are both near them and smaller over similarly close but much larger clusters. The square of population was used as the distance seemed to be the most important factor, but some clustering due to high populations was desired.

In the specific implementation of the k-means clustering algorithm that was used in this project, it was noted that it wasn't always possible to get the clusters to converge fully for certain numbers of hubs (in particular, 6 or 8 hubs) and so a warning to denote this was added. The data is still reasonably close to the local maxima it was approaching and so the hubs are still calculated and displayed, for the discretion of the user to determine if the algorithm should be ran again.

Another issue this particular implementation has brought up is that the clustering algorithm can not always successfully construct the specified number of clusters (never more, but some times less) this is also accounted for and a similar warning displayed for the discretion of the user. Both this and the former issue seem to be edge cases in the initialisation method for the clustering algorithm – the occurrence rate of these edge cases was reduced with the implementation of k-means++ initialisation.

Result

The resulting locations for hubs seem to be very reasonable. For four hubs, the locations are close to the pictured locations (below). These include a location in the Newcastle area, in the North East, where Amazon just recently opened up a new hub of their own.



Another interesting result was if the hub count was set to a high number (i.e. 20). This yielded clustering similar to that seen by Amazon's locating of its fulfilment centres, as a consequence of population densities (images below).

```
Hub #0:                    Hub #10:
    Latitude    -  51.51       Latitude    -  55.865
    Longitude   -  -0.125      Longitude   -  -4.26

Hub #1:                    Hub #11:
    Latitude    -  50.88       Latitude    -  51.795
    Longitude   -  -1.42       Longitude   -  0.77

Hub #2:                    Hub #12:
    Latitude    -  53.745      Latitude    -  50.375
    Longitude   -  -0.335      Longitude   -  -4.14

Hub #3:                    Hub #13:
    Latitude    -  50.895      Latitude    -  52.12
    Longitude   -  -0.025      Longitude   -  -0.49

Hub #4:                    Hub #14:
    Latitude    -  52.48       Latitude    -  52.73
    Longitude   -  -1.895      Longitude   -  -2.08

Hub #5:                    Hub #15:
    Latitude    -  53.48       Latitude    -  53.77
    Longitude   -  -2.24       Longitude   -  -1.63

Hub #6:                    Hub #16:
    Latitude    -  57.14       Latitude    -  51.54
    Longitude   -  -2.1        Longitude   -  -0.755

Hub #7:                    Hub #17:
    Latitude    -  54.89       Latitude    -  53.41
    Longitude   -  -1.47       Longitude   -  -2.975

Hub #8:                    Hub #18:
    Latitude    -  51.48       Latitude    -  51.46
    Longitude   -  -3.18       Longitude   -  -2.585

Hub #9:                    Hub #19:
    Latitude    -  55.95       Latitude    -  53.38
    Longitude   -  -3.195      Longitude   -  -1.46
```

Discussion

There is definitely room for improvement in the above solution. The most obvious that comes to mind is to use a more accurate model of physical distance than the great circle distance (i.e. use an algorithm to work out the most optimal route by road between two locations). Another direction for improvement would be to use accurate, specialised models for demand driven by locations – as some rural locations may be less likely to desire deliveries for certain goods than urban locations, and vice versa. One last improvement considered would be to split up locations into sub locations that may be served by different hubs, this could be done at a macro-scale of just population, or in greater detail using an algorithm to determine which streets to best be handled by different hubs – this last improvement being specific to direct delivery, though could be adapted for supplying different stores in a city.