Introduction to Programming for Physicists – Final Assignment

Version 1.0:

The first version of the program is simply the standard problem set in the assignment. It will read in GBPlaces.csv and process the latitude and longitude values for each location into a 2D vector. It will then loop through the vector in order to find the minimum and maximum values for latitude and longitude. These will be used as bounds in the 'random_number' function to generate pseudo-random coordinates to test for hub suitability.

The program then utilises the hill-climbing algorithm to attempt to find the local minimum. The distance from the hub to each location in GBPlaces is found via the Great Circle formula. Once the local minimum has been found, if it is smaller than the smallest-known minimum, its value will be saved. The algorithm then iterates over with new randomly chosen coordinates. This is done 'N' times, and the smallest minimum found out of all the iterations is considered the 'global' minimum. The hub coordinates corresponding to this minimum are then printed out to the console, alongside the average distance of a town/city from this hub.

The resulting hub location that this program computes is always near to Birmingham. Comparing this both to Parcelforce's central hub location (Coventry [1]) and in particular to UPS's central hub location (Tamworth [2]), I am confident that the program returns a reasonable answer.

Version 1.1:

The second version of the program reads in the data and finds the minimum and maximum values of latitude and longitude in the same way as version 1.0. However, this time a 'fitness' function was created which was able to find the optimal location in the UK to place two separate hubs.

Further, the step length of the hill climbing algorithm is optimized by starting off with a relatively large step, finding the local minimum before decreasing the step size to determine a more accurate value for the minimum. This is done multiple times so that a very reliable value for the local minimum is calculated. This optimizes the hill climbing algorithm since starting with a larger step means that a good estimate for the minimum can be found, before then 'increasing the resolution' to determine a more accurate value.

In this case, the optimal locations for the two hubs are printed to the console alongside the average distance of a town/city from a hub. The program produces the result that one hub should be located near to London while the other hub should be placed near to Manchester/Sheffield. This corresponds well with the intuitive guess that it would be optimal to have one hub in the north and one in the south.

I also attempted to generalise two-hub version so that it could accommodate an arbitrary number of hubs; however, I did not see a clear way of doing this. Although having two hubs obviously decreases the average distance from a town/city to a hub, as can be seen in version 1.1 or 1.2 of the program when compared to 1.0, I wondered if there was a point of diminishing returns given that, after a certain number of hubs are built, the whole system of having a centralised sorting hub is then reduced to many sorting locations across the country, which is opposite of the goal of the hub system.

[1] https://www.parcelforce.com/help-and-advice/sending/national-hub
[2] https://postandparcel.info/23156/news/ups-opens-largest-uk-hub/

Version 1.2:

The third version of the program is identical to version 1.1 except that instead of using the Great Circle algorithm for finding the distance between two places on Earth, Vincenty's formula is used. This consists of an iterative method which is suited to calculating the distance between points on a non-perfect sphere, unlike the Great Circle method which assumes the planet to be a perfect sphere. Given that Earth is an oblate spheroid, the Great Circle method has an additional error which Vincenty's formula avoids. The resulting distances calculated using this algorithm are therefore more accurate, potentially to within 0.5mm of the correct value (although this is dependent upon the datum used and other factors [3]). The Great Circle/Haversine method, meanwhile, is typically accurate to around 0.3%. This accuracy improvement means that the average distance from the hub to any town/city is more reliable.

While Vincenty's method does require more computation time due to its iterative nature, the program is only slowed by a small amount compared to version 1.1. When tested over 100 iterations, version 1.2 took three additional seconds to print out the result compared to version 1.1; however, this value can obviously vary due to the random selection of coordinates.

Version 1.3:

A further extension was made to version 1.1 in the essence of the travelling salesman problem. The idea was to have each delivery truck drive to a location, find the nearest town/city, then drive to it before returning to the hub. The two locations that were visited are then placed on a 'blacklist' so that they are not visited again by a different truck. Running the program returns optimal hub locations near Rochdale and near Watford/High Wycombe, agreeing with the results from versions 1.1 and 1.2 and with intuition.

This extension obviously comes at the cost of greater run time or fewer iterations of different random coordinates. To lessen this impact, the search area was restricted to an area of +/- 0.5 degrees of latitude and longitude about the truck's current location (a box of approximately 70 miles squared with the current location at the centre) to avoid impractical & unnecessary distance evaluations. The idea behind this was that there is no use in a truck going to the nearest town/city if it is too far away from where it is currently. Having bounds of +/- 0.5 degrees means that the maximum distance the truck would have to travel from one location to another (as the crow flies) is when the nearest town/city is at the edge of the allowed box, giving a maximum additional travelling distance of $(0.5 * \pi/180) * 6371 * \sqrt{2} = 78.6$ km ≈ 49 miles, alongside the additional distance needed to be travelled to then get back to the hub.

An additional extension to the program could be to somehow find the distances from hub to city via (main) roads which would increase the accuracy and usefulness of the model, as well as allowing for a better determination of fuel costs and timescales for delivery.

In what turned out to be a less practical version of version 1.0, I made an extension whereby each location's distance from the hub was weighted by its (normalized) population value. I had thought that it would be advantageous to favour places like London and Birmingham as the hub would then be closer to the denser population centres, thereby creating a better service.

[3] https://www.movable-type.co.uk/scripts/latlong-vincenty.html#datums

The result from this test was that, due to London's overwhelming population when compared the other locations in GBPlaces, the hub was continuously being placed in near/within London.

Considering the idea further, I concluded that weighting distances according to population may not be a good idea in any case. This is because if the main benefit of a hub system is that every package can be shipped to the same place, sorted and then delivered, having a hub near to large population centres only makes sense if the inhabitants of London (say) were only sending and receiving parcels from other people in London. Otherwise, if somebody was sending a package from Manchester to Glasgow, or Brighton to Leeds, having a hub near to London is not the necessarily the optimal location to have the most efficient and cost-effective service.