SECOND ASSIGNMENT

April 26, 2020

Decision Support Methods
(CC3003)

Francisco Gonçalves
201604505

Departamento de Ciência de Computadores
Faculdade de Ciências de Universide do Porto

0.1 Introduction

In order to answer the given questions in this second assignment, a mathematical optimization model was made using GLPK. Its implementation is somewhat similar for the two exercises that require it.

Both use the same *.dat* file which specifies all the data needed in order to solve these exercises, which is identical to the one used for the previous assignment. Given the number of locations used, resulting in quite a lengthy file the following transcript is merely demonstrative:

Listing 1: Data Specification (assignment.dat)

```
set LOCATION :=
Lisbon
Porto
(...)
Sao_Miguel_do_Couto
Galegos;

param:               population latitude   longitude :=
Lisbon               517802     38.71667   -9.13333
Porto                249633     41.14961   -8.61099
(...)
Sao_Miguel_do_Couto  5416       41.33167   -8.46185
Galegos              5404       41.56268   -8.57204 ;

end;
```

The rest of the implementation lies in a *.mod* file and can be changed depending on the decision variables and what this mathematical optimization model needs to achieve which is:

- minimizing sum of distribution costs to all locations given a limited number of a single DC.
- minimizing sum of distribution costs to all locations given a variable number of up to five Distribution Centers.

0.2 First Exercise

0.2.1 Minimization of total distribution cost with one DC

The program for the first question is as follows, with slight modifications for a better display and format:

Listing 2: Implementation (assignment1.mod)

```
set LOCATION;                                              // Set of Towns
param latitude {LOCATION};                                 // Town Latitude
param longitude {LOCATION};                                // Town Longitude
param population {LOCATION};                               // Town Population
param R := 6371.009;                                       // Earth Radius
param pi := 3.14159265359;                                 // Value of Pi
param dcCost := 25000;                                     // DC Opening Cost
param cost{i in LOCATION} := ceil(population[i] * 3 / 1000); // Town Cost

// MAIN VARIABLES
var dcLat; // DC Latitude
var dcLng; // DC Longitude

// AUXILIARY VARIABLES
var latDist{LOCATION} >= 0;  // L1 distance component in latitude
var lngDist{LOCATION} >= 0;  // L1 distance component in longitude
var dist{LOCATION} >= 0;     // L1 distance (sum of lat + lng components)

minimize totalDistCost: sum {i in LOCATION} (cost[i] * dist[i]) + dcCost;

subject to
latA {i in LOCATION}: latDist[i] >= 2 * pi * R * (dcLat - latitude[i]) / 360;
latB {i in LOCATION}: latDist[i] >= 2 * pi * R * (latitude[i] - dcLat) / 360;
lngA {i in LOCATION}: lngDist[i] >= 2 * pi * R * (dcLng - longitude[i]) / 360;
lngB {i in LOCATION}: lngDist[i] >= 2 * pi * R * (longitude[i] - dcLng) / 360;
totdist {i in LOCATION}: dist[i] = latDist[i] + lngDist[i];

solve;
printf "Location of the DC: %g, %g \n", dcLat, dcLng;
printf "Total distribution cost: %.10g \n", totalDistCost;
printf "Town with largest distribution costs: ";
printf {a in LOCATION: cost[a] * dist[a] = max {b in LOCATION} cost[b]*dist[b]} a;
printf "( = %.6g euros )" , max {b in LOCATION} cost[b]*dist[b];

end;
```

The *set LOCATION* as well as the three parameters that follow are used to obtain the data specified in the file previously mentioned and it refers to the list of all the towns to be considered. The remaining four parameters are constants: the cost of distribution to each town according to its population, the opening cost of each DC, the Earth's radius and the value of π. These last two constants are required to calculate the Manhattan Distance.

Afterwards, the two main decision variables, *dcLat* and *dcLng*, are self-explanatory, being the latitude and longitude of the Distribution Center (DC), respectively. These are required in order to obtain the L1 Norm to all the locations already defined.

The next three lines are considered auxiliary variables as they address the distance between a given town and the DC, by subtracting one's value from the other. This is the done for the latitude and longitude coordinates and then combined for a total distance.

Finally, the objective function is defined to suit whatever needs to be obtained, which is for the first exercise, the least possible distribution cost.

After running the program, a *.sol* file is created with the requested result. In this case, given the overwhelming amount of variables, the said file is quite extensive so the important information is specified in the desired output, made possible with the three existent *printf* lines at the end of this mathematical optimization model:

Listing 3: Results (assignment1.sol)

```
> minimizing total distribution costs...
Location of the DC: 39.3373, -8.67422
Total distribution cost: 3830025.616 €
Town with largest distribution costs: Lisbon ( ≈186579€ )
```

Figure 1: Geographical Location of the Distribution Center

0.3 Second Exercise

0.3.1 Minimization of total distribution cost with up to five DCs

Listing 4: Programa (GLPK)

```
set LOCATION;  // Set of Towns

param latitude {LOCATION};   // Town Latitude
param longitude {LOCATION};  // Town Longitude
param population {LOCATION}; // Town Population

param cost{i in LOCATION} := ceil(population[i] * 3 / 1000); // Town Cost
param R := 6371.009;                                          // Earth Radius
param pi := 3.14159265359;                                    // Value of Pi
param dcCost := 25000;                                        // DC Opening Cost

// MAIN VARIABLES
var nDC >= 1;      // Number of DCs
var dcLat{1..5};  // DC Latitude
var dcLng{1..5};  // DC Longitude

// AUXILIARY VARIABLES
var latDist{LOCATION,1..5} >= 0;    // L1 distance component in latitude
var lngDist{LOCATION,1..5} >= 0;    // L1 distance component in longitude
var dist{LOCATION,1..5} >= 0;       // L1 distance (sum of lat + lng components)

minimize totDistCost: sum{i in LOCATION,n in 1..5} (cost[i]*dist[i,n]) + (dcCost*nDC);

subject to
numDC: nDC <= 5;
latA {i in LOCATION, n in 1..5}: latDist[i,n] >= 2 * pi * R * (dcLat[n] -
    latitude[i]) / 360;
latB {i in LOCATION, n in 1..5}: latDist[i,n] >= 2 * pi * R * (latitude[i] -
    dcLat[n]) / 360;
lngA {i in LOCATION, n in 1..5}: lngDist[i,n] >= 2 * pi * R * (dcLng[n] -
    longitude[i]) / 360;
lngB {i in LOCATION, n in 1..5}: lngDist[i,n] >= 2 * pi * R * (longitude[i] -
    dcLng[n]) / 360;
totdist {i in LOCATION, n in 1..5}: dist[i,n] = latDist[i,n] + lngDist[i,n];

solve;

end;
```

Most of the implementation remains the same as the only major difference lies in the number of Distribution Centers that can be opened, which is now up to five.

Therefore, the decision variables remain the same as we still need to calculate the existing distances. The same thought process is valid for the restrictions which are related to the distance limits. However, the distances from a town to each DC must be determined and then, the lowest distance is the one that needs to be used to obtain delivery costs. The cost of opening each DC must also be obtained accordingly.

Unfortunately, this particular implementation could not be properly done in order to minimize the total distribution cost for different DC coordinates, ending up in always obtaining the same location, defeating its purpose.

It is also worth noting that despite having declared a variable for the number of DCs that can exist and restricting it to 5, the final result would always only consider one DC, perhaps hinting that the most financially efficient method is to have only one DC given its opening cost. If the number of DCs happened to be manually forced to be higher than 1, then they wiuld all have the same coordinates. After running the program, a *.sol* file is created with the requested result. Given the overwhelming amount of variables, the said file is quite extensive so the important information was to be specified in the desired output, just like in the previous implementation, however most of the results seen here are being miscalculated and should not be considered correct nor approximate:

Listing 5: Results (assignment1.sol)

```
> minimizing total distribution costs...
Number of DCs: 1
Location of the DC #1: 39.3373, -8.67422
Total distribution cost: 19150128.08 €
Town with largest distribution costs: Lisbon ( ≈186579€ )
```