Modelling Laboratory

Optimising garbage collection routing

April 10, 2022

Authors:
Jose Antonio Lorencio Abril
Julio Martínez Bastida
Pablo Miralles González
Sergio Muñoz Moñino
Pablo Tárraga Navarro

Professors:
José Fernández Hernández
Manuel Andrés Pulido Cayuela

Academic Year 2021/22

Contents

1 Introduction **5**

2 Formalising the problem **6**

3 Data **7**

4 Solution of the problem **7**

4.1 Formulation of the problem 7

4.1.1 Model 1 8

4.1.2 Model 2 9

4.2 Programming the solution 11

4.2.1 File *.mod* of model 1 11

4.2.2 File *.mod* of model 2 12

5 Testing the models **14**

6 Results **16**

7 Conclusions **18**

References **20**

List of Figures

1 Beautiful photograph of Cehegín 5

2 Salvador Dalí St. in the old town of Cehegín 6

3 Test case graph. 14

List of Tables

1 Comparison of the amount of variables and constraints of both models. 18

Listings

1 Parameters of model 1 . . . 11
2 Sets of model 1 . . . 11
3 Parameters of model 1 . . . 11
4 Variables of model 1 . . . 11
5 Objective function of model 1 . . . 11
6 Restrictions of model 1 . . . 12
7 Parameters and sets of model 2 . . . 12
8 Variable truck_load of model 2 . . . 12
9 Restrictions of model 2 . . . 13
10 Information about dumpsters and trucks . . . 14
11 Distance matrix . . . 14
12 Code to display the solution of model 1 . . . 15
13 Code to display the solution of model 2 . . . 15
14 Test output for model 1 . . . 15
15 Test output for model 2 . . . 16
16 Town Hall Solution . . . 16
17 Solution of model 1 . . . 17
18 Solution of model 2 . . . 18

1 Introduction

Cehegín is a small town in the north-west of Murcia, we can delight our sight with its beauty in 1. As it is usual in small communities, processes are rarely studied formally and optimised. Instead, it is usual to find intuitive plans and a learn-by-doing approach to problems.

Figure 1: Beautiful photograph of Cehegín

Knowing this situation, we contacted Pedro Lorencio, councillor for works and services of the town, and asked him for information on what kind of problems they were facing, with the intention of tackling them mathematically. Two main problems were discussed:

- **Timetables for the town hall workers**: Pedro thought that maybe the current timetables could be improved, but there was not much depth to the problem. It seemed more of a bureaucratic problem than a mathematical one.
- **Garbage collection routing**: the routes followed by the trash trucks were manually designed, and adaption to changes has been insufficient ever since. As the town grows the number of bins increases, but the routes are not redesigned. Instead, they are patched with the least amount of changes possibles. Therefore, we believe an analytical study of the problem will provide better solutions.

We decided to focus on the latter problem. We contacted the civil protection agency of Cehegín and arranged a meeting, in which the following information was given:

- There are three garbage trucks, and a designated zone for each one of them. One of the trucks is reserved for the old town, where streets are narrower and there is a bin for each house, such as in Figure 2. The other two trucks are responsible for the rest of the town and surrounding hamlets.

Figure 2: Salvador Dalí St. in the old town of Cehegín

- After filling out, the trucks must discharge the trash at the dump, located in the nearby town of Calasparra.
- The trucks start and end every workday in the civil protection centre. They must arrive there fully empty, and a worker must clean them after arrival. This is particularly important, as the trucks are also needed to collect plastic and other recyclables in other working shifts.
- Each truck has a capacity of 9 tons of garbage.
- They currently do not predict the amount of garbage to collect at each location. However, they estimate an average of 80 kg of trash per bin per day.
- Addresses for each location to visit was provided, along with the number of dumpsters to empty out.

2 Formalising the problem

The town hall has two trucks available for garbage collection, and it is responsible for the pickup of 431 dumpsters distributed over 334 locations around Cehegín and other districts. The trucks are initially stationed in the civil protection centre of Cehegín, and whenever they run out of space they must discharge the waste at Calasparra's dump. We introduce now certain notation:

- *(I)* denotes the initial location where trucks are parked, and *(V)* Calasparra's dump.
- Let $N = 334$ be the number of locations to visit. We will thus work with $N + 2$ nodes, where the first N will be locations to visit and the last two will be *(I)* and (V) respectively.
- Let $D = (d_{ij})$, where d_{ij} denotes the distance between the i-th and j-th node for $i, j \in \{1, 2, \ldots, N+2\}$.

- The l-th truck has capacity C_l. In the present situation, both trucks can take up to $9T$ each.
- We denote $P = (p_1, \ldots, p_N)$, where p_i is the amount of waste to collect at location i.

Note that we are measuring capacity in terms of weights and not volumes. We assume that on average waste.

We shall minimise the total distance covered by the trucks to reduce fuel costs, while fulfilling the following restrictions:

1. No truck exceeds its maximum load capacity. We will make the simplifying assumption that a truck collects every dumpster whenever it visits a location.
2. Trucks start their first round at node *(I)*.
3. Trucks start every round after the first one at node *(V)*.
4. Trucks end all rounds at node *(V)*. Note that independently of the route every truck will go back from *(V)* to *(I)* after the last round, so we may ignore this trip in our solution.
5. Every location must be visited exactly once, as we assumed there will be no partial collections.

3 Data

As described earlier, the problem requires the distance matrix, the trucks capacities and the waste volumes to pick up at each location.

The company provides addresses for every location, which need to be translated to coordinates and then to distances. By performing trivial transformations and using the ArcGIS API [1] we managed to obtain a fair amount of coordinates. The rest were taken manually via Google Maps.

We calculate distances by a naive approximation using the euclidean distance between every pair of nodes. It is clear that in order to obtain the best solution possible real coordinates and distances would need to be provided. This is out of the scope of the project as the town hall is responsible for that matter. If a proper solution is achieved, they could easily change the data input to improve it.

We also assume fully loaded dumpsters at every location. It is clear that more intelligent predictions of the volumes would improve the solutions obtained. Since it is not an optimisation problem we shall not concern ourselves with this matter.

4 Solution of the problem

4.1 Formulation of the problem

Let's start by introducing the popular Chinese Postman Problem (CPP), defined by Mei-Ko Kwan in 1962 to determine the most effective route for a postman to distribute the mail received from the post office and to return to the post office with the shortest walking distance during mail distribution [2]. In graph theory terms, the objective is to find an optimal Hamiltonian cycle, this is, a cycle that visits every node in the graph once except for the initial one, that is visited twice. This problem has been thoroughly studied by researchers, who are now more interested in a variety of extensions of traditional CPP problems. This is the case for our problem, which extends the original CPP in the following ways:

- There are two vehicles, instead of one. This kind of problem is called k-CPP [2].
- The starting node is different from the ending one.

- Location nodes must all be visited exactly once, but the special node (V) will probably be visited more than once.

These variations complicate particularly the avoidance of sub-cycles. We will use the MTZ (Miller, Tucker and Zemlin) formulation to solve this problem, although minor tweaks are required to adapt it to our models. This formulation is covered in the "Laboratorio de Modelización" course, but we include a brief overview.

Let G be a graph with n nodes numbered from 1 to n, where node 1 is the initial node of the cycle. Let X_{ij} be a binary variable denoting whether or not there is an edge between nodes i and j. Assume as well that the amount of incoming and outgoing edges for each node have both been restricted to the value of 1. The MTZ constraint goes as follows:

$$u_i - u_j + (n-1)X_{i,j} \leq n-2, \quad i,j = 1,\ldots,n : i \neq j \neq 1$$
$$u_i \in \mathbb{R}, \quad i = 1,\ldots,n,$$

where the additional real variables u_i takes the value t if the i-th node is the t-th visited in the Hamiltonian cycle in any feasible solution. This formulation is further explained and proved in the course's notes. We remark here that any upper bound of n works in these restrictions. This can be easily verified by carefully reading the proof in the course's notes.

We propose two different models for this problem, both in need of the following assumption. As the Calasparra's dump is quite far away from the city of Cehegín, we assume that loading each truck as much as possible is generally better. Given that the current amount of rounds both truck performs daily is between 2 and 3 in total, we reckon $K = 4$ a conservative upper bound for the number of rounds each truck performs.

4.1.1 Model 1

Our first model separates the rounds of each truck in different graphs, resulting in a Hamiltonian path in the first round and a Hamiltonian cycle in the rest. A Hamiltonian path is a graph path that traverses every node once. Aside from this difference, this alteration transforms the problem into a 2-CPP one.

Let $X \in \mathcal{M}_{(N+2)\times(N+2)\times K\times 2}$, where

$$X_{i,j,k,l} = \begin{cases} 1 & \text{if the } l\text{-th truck goes from node } i \text{ to node } j \text{ in the } k\text{-round;} \\ 0 & \text{otherwise.} \end{cases}$$

Let $Du = \{1,\ldots,N\}$ be the vertices corresponding to dumpster locations, $Ro = \{1,\ldots,K\}$ the set of rounds, $I = N+1$ and $V = N+2$ the initial and dump vertices, respectively, and $Ver = Du \bigcup \{I, V\}$ the set of all vertices. Denoting the Kronecker delta by δ_{ij}, our model can be formulated as:

$$
\begin{aligned}
\min\quad & \textstyle\sum_{i\in Ver, j\in Ver, k\in Ro, l=1,2} X_{i,j,k,l} D_{ij} && && \text{(OBJ)}\\
\text{s.a.}\quad & \textstyle\sum_{i\in Ver} \sum_{j=1}^{N} X_{i,j,k,l} P_j \leq C_l && \forall l = 1,2, k \in Ro && \text{(M1-C1)}\\
& \textstyle\sum_{j\in Ver} X_{I,j,k,l} = \delta_{1k} && \forall l = 1,2, k \in Ro && \text{(M1-C2)}\\
& \textstyle\sum_{i\in Ver} X_{i,I,k,l} = 0 && \forall l = 1,2, k \in Ro && \text{(M1-C3)}\\
& \textstyle\sum_{j\in Ver} X_{V,j,k,l} = 1 - \delta_{1k} && \forall l = 1,2, k \in Ro && \text{(M1-C4)}\\
& \textstyle\sum_{i\in Ver} X_{i,V,k,l} = 1 && \forall l = 1,2, k \in Ro && \text{(M1-C5)}\\
& \textstyle\sum_{i\in Ver, k\in Ro, l=1,2} X_{i,x,k,l} = 1 && \forall x \in Du && \text{(M1-C6)}\\
& \textstyle\sum_{i\in Ver} X_{i,x,k,l} = \sum_{j\in Ver} X_{x,j,k,l} && \forall x \in Du, l = 1,2, k \in Ro && \text{(M1-C7)}\\
& u_{i,k,l} - u_{j,k,l} + (N+1) X_{i,j,k,l} \leq N && \forall i \in Ver, j \in Ver \setminus \{V\}, k \in Ro, l = 1,2 && \text{(M1-C8)}\\
& X_{i,j,k,l} \in \{0,1\} && \forall i,j = 1,\ldots,N+2, k \in Ro, l = 1,2 && \text{(M1-C9)}\\
& u_{i,k,l} \in \mathbb{R} && \forall i = 1,\ldots,N+2, k \in Ro, l = 1,2 && \text{(M1-C10)}
\end{aligned}
$$

The objective function is clear. We explain now each constraint.

(M1-C1) Restricts the load for each truck and round to the truck's capacity.

(M1-C2-5) Restrict the in-degrees and out-degrees of nodes (I) and (V) to model a Hamiltonian path or cycle depending on whether the round is the first one or not. Some of these restrictions are actually superfluous, as solutions not satisfying them would be sub-optimal. However, due to time and memory constraints on the solution searching process, we will only be able to find a sub-optimal solution. We believe these restrictions might aid the solver in finding a better solution, at the expense of the increased time and memory complexity.

(M1-C6) Forces vertices that include dumpsters to be visited exactly once, due to the initial assumption we made on partial collections.

(M1-C7) Restricts in-degrees and out-degrees of intermediate nodes to be equal.

(M1-C8) Corresponds to MTZ restrictions. There are two important observations to make here:

- In each Hamiltonian path/cycle less than N+2 nodes are included. As we remarked after the overview of MTZ restrictions, this is not an issue, as an upper bound suffices.
- The restriction works for the Hamiltonian path as well. In fact, we could even include the constraint for $j = N+2$ in this case, since node (V) will not appear twice. Again, careful study of the course's notes is enough to show this.

(M1-C9-10) Domain restrictions.

4.1.2 Model 2

Our second model tackles the problem of visiting the node (V) multiple times in a different way. We consider K different nodes to represent the dump, and each one of them is visited only once. The distance matrix is updated accordingly: the distance between two nodes representing the dump is 0, whereas the distance with other nodes is the same as that of the original (V) node. Therefore, our solution now consists of one Hamiltonian path for each truck, starting at node (I) and ending at node (V). We consider nodes $N+2,\ldots,N+1+K$ to represent the dump, and we may assume without loss of generality that Hamiltonian paths end at node $N+1+K$.

Let $X \in \mathcal{M}_{(N+1+K)\times(N+1+K)\times 2}$, where

$$
X_{i,j,l} = \begin{cases} 1 & \text{if the } l\text{-th truck goes from node } i \text{ to node } j;\\ 0 & \text{otherwise.} \end{cases}
$$

In order to model load limitations we need another set of variables. Let $TL \in \mathcal{M}_{2\times(N+1+K)}$, where

$$TL_{l,i} = \text{ amount of waste loaded onto truck } l \text{ after visiting node } i.$$

Let $Du = \{1, \ldots, N\}, I = N+1, V = \{N+2, \ldots, N+1+K\}$ and $Ver = Du \bigcup \{I\} \bigcup V$. The formulation is as follows:

min	$\sum_{i\in Ver, j\in Ver, k\in Ro, l=1,2} X_{i,j,k,l} D_{ij}$		(OBJ)
s.a.	$\sum_{j\in Ver} X_{N+1,j,l} = 1$	$\forall l = 1,2$	(M2-C1)
	$\sum_{i\in Ver} X_{i,N+1,l} = 0$	$\forall l = 1,2$	(M2-C2)
	$\sum_{j\in Ver} X_{N+1+K,j,l} = 0$	$\forall l = 1,2$	(M2-C3)
	$\sum_{i\in Ver} X_{i,N+1+K,l} = 1$	$\forall l = 1,2$	(M2-C4)
	$\sum_{i\in Ver, l=1,2} X_{i,x,l} = 1$	$\forall x \in Du$	(M2-C5)
	$\sum_{i\in Ver} X_{i,x,l} \leq 1$	$\forall x \in V \setminus N+1+K, l \in \{1,2\}$	(M2-C6)
	$\sum_{i\in Ver} X_{i,x,l} = \sum_{j\in Ver} X_{x,j,l}$	$\forall x \in Ver \setminus \{I, N+1+K\}, l = 1,2$	(M2-C7)
	$u_{i,l} - u_{j,l} + (N+1)X_{i,j,l} \leq N$	$\forall i,j \in Ver, l = 1,2$	(M2-C8)
	$TL_{l,x} = 0$	$\forall x \in Ver \setminus Du, l = 1,2$	(M2-C9)
	$TL_{l,x} \leq C_l$	$\forall x \in Ver, l = 1,2$	(M2-C10)
	$TL_{l,i} + P_j \leq TL_{l,j} + 10C_l \cdot (1 - X_{i,j,l})$	$\forall i \in Ver, j \in Du, l = 1,2$	(M2-C11)
	$X_{i,j,l} \in \{0,1\}$	$\forall i,j \in Ver, l = 1,2$	(M2-C12)
	$u_{i,l} \in \mathbb{R}$	$\forall i \in Ver, l = 1,2$	(M2-C13)
	$TL_{l,i} \in \mathbb{R}$	$\forall i \in Ver, l = 1,2$	(M2-C14)

(M2-C1-4) In-degrees and out-degrees of the initial and final nodes in the path.

(M2-C5) Every location must be visited exactly once.

(M2-C6) Dump nodes can be visited at most once. The last one is excluded because its degree has already been restricted.

(M2-C7) Intermediate nodes in the path must have the same number of inbound and outbound edges.

(M2-C8) MTZ restrictions. The observations from the first model are still relevant. Since we are looking for a Hamiltonian path we do not need to exclude any restriction, although the case $j = N+1+K$ is superfluous due to the restriction on its out-degree.

(M2-C9) The load after visiting (I) or (V) is 0, the trucks are empty.

(M2-C10) Loads cannot exceed the truck's capacity.

(M2-C11) This is the linearisation of the restriction $X_{i,j,l} \implies TL_{l,i} + P_j \leq TL_{l,j}$. It models the increase in load after visiting a node. We remark that the opposite inequality is not necessary and that this restriction is not true if the destination node j is the dump.

4.2 Programming the solution

To solve this problem we make use of *AMPL*. *AMPL* is an algebraic modeling language for mathematical programming, which offers an interactive command environment for setting up and solving mathematical programming problems. [3]

The typical folder structure of an *AMPL* problem consists of 3 files; the file *.mod*, in which the model is defined, the file *.dat*, where we specify the data values, and lastly, the file *.run*, which contains the commands that will be executed.

The file *.mod* includes the formulation of the problem, that is, the sets, parameters, variables, objective function and constraints. As we have already described, we approach this problem with 2 different models, and thus, each one is defined in a separate file *.mod*.

4.2.1 File *.mod* of model 1

First of all, we start by defining the parameters `N`, `NTRUCKS` and `max_rounds`, which correspond to the number of locations (N), the number of trucks and the upper bound of rounds that each truck performs (K), respectively.

```
param N >= 0;
param NTRUCKS >= 1;
param max_rounds >= 1;
```

Listing 1: Parameters of model 1

Secondly, we define the sets `TRUCKS`, `STOPS`, `DUMPSTERS` and `ROUNDS` which correspond to the set of trucks, the set of all vertices (Ver), the set of dumpster locations (Du) and the set of rounds (Ro), respectively.

```
set TRUCKS := 1..NTRUCKS;
set STOPS := 0..(N+1);
set DUMPSTERS = 0..(N-1);
set ROUNDS = 1..max_rounds;
```

Listing 2: Sets of model 1

Subsequently, we define the parameters `tr_cap`, `distances` and `gar_qty`, which correspond to the load capacity of each truck (C_l), the distance matrix (D) and the amounts of waste (P) in each location, respectively.

```
param tr_cap {TRUCKS} >= 0;
param distances {STOPS, STOPS} >= 0;
param gar_qty {DUMPSTERS} >= 0;
```

Listing 3: Parameters of model 1

We also define the binary variable X and the real variable u from the MTZ constraint.

```
var X{STOPS, STOPS, ROUNDS, TRUCKS} binary;
var u{STOPS, ROUNDS, TRUCKS};
```

Listing 4: Variables of model 1

The objective function is named `Total_distance` and seeks to minimize the distance traveled by the trucks.

```
minimize Total_distance:
    sum{i in STOPS, j in STOPS, k in ROUNDS, l in TRUCKS} X[i,j,k,l] * distances[i,j];
```

Listing 5: Objective function of model 1

Finally, the restrictions which correspond to the restrictions M1-C1 to M1-C8:

```
subject to LoadRestriction{l in TRUCKS, k in ROUNDS}:
    sum{i in STOPS, j in DUMPSTERS} X[i,j,k,l] * gar_qty[j] <= tr_cap[l];

subject to NodeI_InDegree{l in TRUCKS, k in ROUNDS}:
    sum{i in STOPS} X[i,N,k,l] = 0;

subject to NodeI_OutDegree{l in TRUCKS, k in ROUNDS}:
    sum{j in STOPS} X[N,j,k,l] = (if k = 1 then 1 else 0);

subject to NodeV_InDegree{l in TRUCKS, k in ROUNDS}:
    sum{i in STOPS} X[i,N+1,k,l] = 1;

subject to NodeV_OutDegree{l in TRUCKS, k in ROUNDS}:
    sum{j in STOPS} X[N+1,j,k,l] = (if k = 1 then 0 else 1);

subject to VisitEachDumpNodeOnce{j in DUMPSTERS}:
    sum{i in STOPS, k in ROUNDS, l in TRUCKS} X[i,j,k,l] = 1;

subject to InOutDegreesEquality{j in DUMPSTERS, l in TRUCKS, k in ROUNDS}:
    sum{i in STOPS} X[i,j,k,l] = sum{i in STOPS} X[j,i,k,l];

subject to MTZ{i in STOPS, j in STOPS diff{N+1}, l in TRUCKS, k in ROUNDS}:
    u[i,k,l] - u[j,k,l] + (N+1)*X[i,j,k,l] <= N;
```

Listing 6: Restrictions of model 1

4.2.2 File *.mod* of model 2

This file includes the parameters previously defined as well as these new parameters and sets:

```
set TRUCKS := 1..NTRUCKS;
set DUMPSTERS = 0..(N-1);
param INIT = N;
set DUMPS = (N+1)..(N + 1 + max_rounds) ordered;
param Nnodes = (N + 1 + max_rounds);
set STOPS := (DUMPSTERS union {INIT}) union DUMPS;
```

Listing 7: Parameters and sets of model 2

We also add the variable `truck_load` that corresponds to the amount of waste loaded onto each truck after visiting a node (TL).

```
var truck_load{TRUCKS, STOPS};
```

Listing 8: Variable truck_load of model 2

Using the same objective function from model 1, we define these new restrictions which correspond to restrictions M2-C1 to M2-C11.

```
subject to StartAtINode {l in TRUCKS}:
    sum{j in STOPS} X[INIT, j, l] = 1;

subject to IInDegree{l in TRUCKS}:
    sum{i in STOPS} X[i, N, l] = 0;

subject to VisitLastVNodeOnce{l in TRUCKS}:
    sum{i in STOPS} X[i, last(DUMPS), l] = 1;

subject to LastVOutDegree{l in TRUCKS}:
    sum{j in STOPS} X[last(DUMPS), j, l] = 0;

subject to VInDegreeLimit{l in TRUCKS, x in (DUMPS diff {last(DUMPS)})}:
    sum{i in STOPS} X[i, x, l] <= 1;

subject to VisitEachContainerOnce{x in DUMPSTERS}:
    sum{i in STOPS, l in TRUCKS} X[i, x, l] = 1;

subject to IntermediateNodesInOutDegree{x in (STOPS diff {INIT, last(DUMPS)}), l in TRUCKS}:
    sum{i in STOPS} X[i, x, l] = sum{j in STOPS} X[x, j, l];

# MTZ: no cycles, hamiltonian path
subject to MTZ{i in STOPS, j in STOPS, l in TRUCKS}:
    u[i, l] - u[j, l] + (Nnodes + 1) * X[i, j, l] <= Nnodes;

# Load restrictions
subject to Reload{x in ({INIT} union DUMPS), l in TRUCKS}:
    truck_load[l, x] = 0;

subject to LoadLimit{x in STOPS, l in TRUCKS}:
    truck_load[l, x] <= tr_cap[l];

# X[i,j,l] => truck_load[l,j] >= truck_load[l,i] + gar_qty[j]
# We take 10 * tr_cap[l] as upper bound
subject to LoadRelations{i in STOPS, j in (STOPS diff (DUMPS union {INIT})), l in TRUCKS}:
    truck_load[l,i] + gar_qty[j] <= truck_load[l,j] + 10 * tr_cap[l] * (1 - X[i, j, l]);
```

Listing 9: Restrictions of model 2

5 Testing the models

In order to test both models, we create a fictitious problem with 6 nodes in addition to the initial and the dump vertices. We include two trucks with a capacity of 9000kg, and they have to pick up 4500kg at every location, meaning they can only visit two nodes with dumpsters every round.

The graph for the test problem is constructed by dividing a circumference into eight equal arcs, where the dump and the initial nodes are opposite of each other (Figure 3). The graph is completed with the distances the circumference induces.

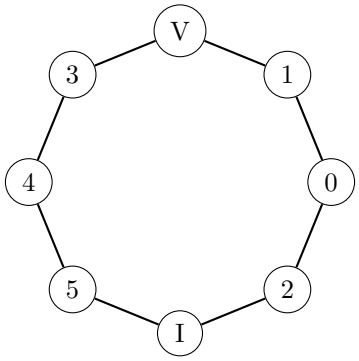

Figure 3: Test case graph.

The *.dat* file and distance matrix for this test are as follows.

```
data;
param N := 6;
param NTRUCKS := 2;

param max_trips := 2;

param tr_cap :=
1 9000
2 9000
;
param gar_qty :=
  0 4500
  1 4500
  2 4500
  3 4500
  4 4500
  5 4500
;
```

Listing 10: Information about dumpsters and trucks

```
param distances
: 0 1 2 3 4 5 6 7 :=
0 0 1 1 3 4 3 2 2
1 1 0 2 2 3 4 3 1
2 1 2 0 4 3 2 1 3
3 3 2 4 0 1 2 3 1
4 4 3 3 1 0 1 2 2
5 3 4 2 2 1 0 1 3
6 2 3 1 3 2 1 0 4
7 2 1 3 1 2 3 4 0
;
```

Listing 11: Distance matrix

The files with extension *.run* contain the commands that will be executed in order to solve the problem and display the solution achieved. In these files, we include commands to set up the model and data, to choose the solver and to select options that may improve solver performance.

Once the solver reaches a solution for model 1, and for the purpose of displaying that solution, we execute the next chunk of code:

```
param Y;
if Total_distance > 0 then {
    for {l in TRUCKS}{
        printf "Camion %u\n", l;
        let Y := N;
        for {k in ROUNDS}{
            printf "VUELTA %u --> ", k;
            if Y = N+1 then printf "(V)";
            else printf "(I)";
            repeat {
                for {j in STOPS}{
                    if X[Y,j,k,l] = 1 then {
                        if j = N+1 then {
                            if Y != N+1 then printf " (V)";
                        } else printf " %u", j;
                        let Y := j;
                        break;
                    };
                };
            } until Y = N+1;
            printf " \n" ;
        };
    };
};
```

Listing 12: Code to display the solution of model 1

With this code, for each truck and for each round, we print the nodes each truck travels through. Note that `Y` is parameter that functions as a standard programming variable and stores the current node visited by the truck.

The next chunk does a similiar job displaying the solution of model 2:

```
param Y;
if Total_distance > 0 then {
    for {l in TRUCKS}{
        let Y := N;
        printf "Camion %u --> (I)", l;
        repeat {
            for {j in STOPS}{
                if X[Y, j, l] = 1 then {
                    if j in DUMPS then {
                        if Y not in DUMPS then printf " (V)";
                    } else printf " %u", j;
                    let Y := j;
                    break;
                };
            };
        } until Y = last(DUMPS);
        printf " \n";

    };
};
```

Listing 13: Code to display the solution of model 2

After running both models using the Gurobi solver, we get the following outputs.

```
Presolve eliminates 36 constraints and 84 variables.
Adjusted problem:
204 variables:
    172 binary variables
    32 linear variables
238 constraints, all linear; 1168 nonzeros
    38 equality constraints
    200 inequality constraints
1 linear objective; 170 nonzeros.

Gurobi 9.5.0: optimal solution; objective 12
```

```
247 simplex iterations
1 branch-and-cut nodes
plus 60 simplex iterations for intbasis
Camion 1
VUELTA 1 --> (I) 5 4 (V)
VUELTA 2 --> (V) 3 (V)
Camion 2
VUELTA 1 --> (I) 2 0 (V)
VUELTA 2 --> (V) 1 (V)
```

Listing 14: Test output for model 1

```
Presolve eliminates 76 constraints and 62 variables.
Adjusted problem:
178 variables:
    146 binary variables
    32 linear variables
306 constraints, all linear; 1178 nonzeros
    26 equality constraints
    280 inequality constraints
1 linear objective; 138 nonzeros.

Gurobi 9.5.0: optimal solution; objective 12
428 simplex iterations
1 branch-and-cut nodes
plus 57 simplex iterations for intbasis
Total_distance = 12

Camion 1 --> (I) 2 0 (V) 1 (V)
Camion 2 --> (I) 5 4 (V) 3 (V)
```

Listing 15: Test output for model 2

As we can see, both models reach the same solution. This is intuitively the optimal solution, but we still tested it against a brute force solution in Python, ensuring its optimality.

6 Results

In this section we analyse the results obtained from Models 1 and 2, and we compare them with the currently implemented routes.

Firstly, it must be remarked that the estimations of the amounts of garbage to pickup at each location are extremely pessimistic. The routes given by the town hall do not satisfy those constraints by a large margin. In order to make the comparison fair, we adapt the town hall's routes by visiting the dump vertex when the truck becomes full under the assumption of fully loaded dumpsters. The resulting routes add up to a total of 613km, and is described below.

```
Camion 1
VUELTA 1 --> (I) 0 1 2 ... 78 (V)
VUELTA 2 --> (V) 79 80 ... 165 (V)
VUELTA 3 --> (V) 166 167 ... 180 (V)
Camion 2
VUELTA 1 --> (I) 181 182 ... 274 (V)
VUELTA 2 --> (V) 275 276 ... 333 (V)
```

Listing 16: Town Hall Solution

We focus now on the results of our models. Due to the time complexity of our problem we use the NEOS server. After some testing we decided to use the Gurobi solver, which is one of the world's fastest and most powerful optimizers [4]. The submissions are performed via web interface due to limits on input data using *kestrel*.

The NEOS server presents several limitations. Particularly relevant are the time limit of 8 hours and the RAM memory limit of $3GB$, as our models require larger amounts of both resources. Exceeding any of these limits results in the program crashing without showing solutions found up until that point. To prevent this, we use the following Gurobi's options:

timelim. We set this self-explanatory option to 27000 seconds or, equivalently, 7.5 hours.

nodefilestart. This option limits the amount of RAM used for the Branch-and-Bound tree. Nodes are stored in disk files after exceeding it. We set this option conservatively to $1GB$.

threads. This option limits the amount of threads running our model concurrently. We set a limit of 2.

Running the first model results in the following output.

```
You are using the solver gurobi_ampl.

%% COMMENTS %%%%%%%%%%%%%%%%%%%%%%%%%%
Gurobi Model 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Checking ampl.mod for gurobi_options...
Checking ampl.com for gurobi_options...
Executing AMPL.
processing data.
processing commands.
Executing on prod-exec-2.neos-server.org

Presolve eliminates 3370 constraints and 10050 variables.
Adjusted problem:
1122270 variables:
    1118910 binary variables
    3360 linear variables
1125954 constraints, all linear; 7829000 nonzeros
    3694 equality constraints
    1122260 inequality constraints
1 linear objective; 1118902 nonzeros.

Gurobi 9.1.2: nodefilestart=1
threads=2
timelim=27000
Gurobi 9.1.2: time limit with a feasible solution; objective 557.8405985
262329 simplex iterations
2797 branch-and-cut nodes
No basis.
No dual variables returned.
Total_distance = 557.841

Camion 1
VUELTA 1 --> (I) 240 242 243 244 245 249 246 248 247 241 331 332 216 217 78 182 179 280 283 274 273 284
     286 (V)
VUELTA 2 --> (V)
VUELTA 3 --> (V) 46 13 11 14 47 56 77 50 58 76 55 94 93 82 84 83 19 85 86 87 88 239 317 254 170 171 173
     158 151 152 153 160 163 165 164 167 168 166 178 169 175 172 154 174 64 63 238 233 232 231 230 225
     227 228 72 98 91 90 89 99 226 18 79 81 80 20 21 10 16 48 49 95 54 52 53 57 51 8 41 35 (V)
VUELTA 4 --> (V) 289 281 279 22 24 23 257 261 260 252 259 258 313 303 302 309 320 (V)
VUELTA 5 --> (V) 176 177 32 33 133 132 112 111 108 113 110 39 131 125 124 123 134 38 40 126 114 109 115
     116 105 129 130 119 118 104 101 147 5 196 4 1 6 195 7 120 121 122 318 103 102 117 100 2 3 135 140
     136 141 219 220 218 222 145 144 137 221 142 139 138 143 146 149 150 213 212 210 206 209 207 205
     204 203 61 201 200 215 214 208 211 223 62 66 65 236 68 67 224 69 237 234 235 278 (V)
Camion 2
VUELTA 1 --> (I) 181 190 43 31 27 26 29 45 0 12 9 15 25 17 253 333 255 316 262 315 263 266 300 301 299
     298 297 296 295 293 294 292 291 288 290 287 330 304 328 (V)
VUELTA 2 --> (V) 329 321 322 327 325 326 305 324 323 250 306 307 (V)
VUELTA 3 --> (V)
VUELTA 4 --> (V) 282 277 272 70 229 148 155 162 161 157 156 199 198 197 106 107 128 127 37 36 28 30 192
     265 193 194 189 183 185 184 44 191 187 188 159 180 186 42 34 59 74 75 92 96 97 73 60 202 71 256
     319 271 270 269 264 268 267 314 276 275 285 (V)
VUELTA 5 --> (V) 310 251 308 311 312 (V)
```

Listing 17: Solution of model 1

It is obvious that the second round of truck 1 and the third round of truck 2 are irrelevant to the solution of the problem. The distance covered in total is 557.841km, which improves the current solution by 10%.

On the other hand, using the second model leads to the following output.

```
You are using the solver gurobi_ampl.
Checking ampl.mod for gurobi_options...
Checking ampl.com for gurobi_options...
Executing AMPL.
processing data.
processing commands.
Executing on prod-exec-1.neos-server.org

Presolve eliminates 2718 constraints and 2054 variables.
Adjusted problem:
231872 variables:
    230522 binary variables
    1350 linear variables
459358 constraints, all linear; 2060516 nonzeros
    1016 equality constraints
    458342 inequality constraints
1 linear objective; 230472 nonzeros.

Gurobi 9.1.2: nodefilestart=1
threads=2
timelim=27000
Gurobi 9.1.2: time limit without a feasible solution
1977669 simplex iterations
12982 branch-and-cut nodes
No basis.
No primal or dual variables returned.
Total_distance = 0
```

Listing 18: Solution of model 2

As we can see, the solver does not find a feasible solution in the given time. This is particularly interesting due to the amount of constraints and restrictions (Table 1). In spite of having a much smaller amount of both, the other model is the one to find a feasible solution. We have not been able to find a satisfactory explanation. In the next iteration of this report we will try to use the solution found by the first model as an starting point of both, hopefully improving said solution.

	Model 1	Model 2
Total variables	1122270	231872
Binary variables	1118910	230522
Linear variables	3360	1350
Constraints	1125954	459358

Table 1: Comparison of the amount of variables and constraints of both models.

7 Conclusions

This project can hardly be called a success. Even though we did manage to get a good solution to our problem, it was under many false assumptions, such as the use of approximated locations, euclidean distances and pessimistic estimations of garbage amounts at each location. It looks very unlikely that our models will be implemented in the future at all, as providing decent data would be too much of a hassle to feel worth it for the town hall.

The assignment can be criticised from many angles:

- It is very difficult to divide the project in concurrent tasks, which means that coordinating five people is too costly.
- Experience gained from interacting with companies is hardly valuable. Since the stakes are non existent for the companies, students tend to end up solving fictitious problems that are of no use to anyone.
- Decent problems such as ours require large amounts of computing power. Students do not have

access to such resources, and the faculty does not provide them neither. The NEOS server is clearly not a sufficient solution due to the mentioned limits.

On the bright side, we did gain some experience in solving this specific type of optimisation problem. Overall, we did not find this project to be very satisfactory.

References

[1] "ArcGIS developer" ArcGIS Developer. () https://developers.arcgis.com/ (visited on 03/23/2022).

[2] S. Liu, S. J. Liu, N. Harris, and H. M. La "A genetic algorithm for minmax k-chinese postman problem with applications to bridge inspection" 2019.

[3] R. Fourer, D. M. Gay, and B. W. Kernighan *"AMPL: a modeling language for mathematical programming"*. Boyd & Fraser, 2001.

[4] B. Bixby "The gurobi optimizer" *Transp. Re-search Part B* vol. 41 no. 2, pp. 159–178 2007.