Assessment 2 : CCCS660 Computational Intelligence

Quentin Lao — Matricule : 261233739

HANDS-ON OPTION

1 Basic stage for the hands-on option

1.1 Problem statement

Each week, vehicles are in charge of collecting garbage from designated streets. In assessment 1, we found out how to compute quite optimal routes for each vehicle. However, each week, the designated streets change, so new routes must be computed.

Let's formalize the problem. We represent the city's streets as set $\mathcal{S}$ and $\mathcal{S}_i \subset \mathcal{S}$ the designated streets in week i. A route is simply a sequence of streets $s_1 \rightarrow s_2 \rightarrow ... \rightarrow s_p$ where $s_i \in \mathcal{S}$, taking the shortest path between successive streets. We define the length of a route as the total distance that needs to be traveled by a driver to achieve it. In the i-th week, k_i vehicles are deployed for garbage collection which requires the computation of k_i distinct routes for each vehicle :

— Route 1 : $R_{1,i} := s_{1,1} \rightarrow ... \rightarrow s_{1,p_1}$

— $\vdots$

— Route k_i : $R_{k_i,i} := s_{k_i,1} \rightarrow ... \rightarrow s_{k_i,p_{k_i}}$

where each $s_{m,n} \in \mathcal{S}_i$ is unique and $R_{l,i}$ is the l-th route computed at week i.

To drive the vehicles, there are K full-time workers. The set of workers is denoted as $\mathcal{W}$. We assume that $K \geq \max_i k_i$, that is there are always more workers than routes computed each week. Currently, k_i workers are randomly assigned to routes each week and the remaining workers handle maintenance. This is a quite bad strategy as routes may be of different lengths which leads to unequal workloads. Additionally, to prevent drivers like Marion from repeatedly covering the same streets, a better assignment strategy that promotes variety and exposure is needed. In the paper, we use the notation $R \equiv_i w$ or equivalently $w \equiv_i R$ to indicate that the worker $w \in \mathcal{W}$ is assigned to the route R at week i. A worker w assigned to maintenance at week i is represented by $w \equiv_i \emptyset$.

Our goal is to use decision trees to assign routes to workers for the next week, given route-assignment records from the q past weeks and the new computed routes for the next week $q+1$. We aim to take in account the workload balance among workers and the diversity in the streets they cover.

1.2 First tree : which workers will drive ?

As said before, we assume that the k_{q+1} routes for next week $q+1$ have been computed and we cannot change them. The only thing we can do is assign properly which workers will do the job. To do so, the first step is to properly select $K - k_{q+1}$ workers that will handle maintenance.

1.2.1 The *maintenance score*

We aim to measure a worker's recent maintenance activity. A worker that did not do a lot of maintenance in the near past would be more likely to perform maintenance in the next week. Thus, we define the *maintenance score* of a worker which is inversely related to the amount of recent maintenance work : a lower score indicates more maintenance done recently, and a higher score reflects more driving tasks. Formally, the *maintenance score* MS for worker w in week $q+1$ is calculated as :

$$MS(w, q+1) := \sum_{i=1}^{q} \mathbb{1}_{[w \not\equiv_i \emptyset]} \gamma^{q+1-i}$$

where $\gamma \in [0, 1]$ is a discount factor that quantifies how far in the past maintenance activities of the worker are considered.

Let's take an example. Two workers w_1 and w_2 have different activities over the past five weeks as shown in table 1. We take $\gamma = 0.9$. The *maintenance score* for w_1 for the week 6 is $\gamma + \gamma^2 + \gamma^4 = 2.37$ where as for w_2 is $\gamma + \gamma^4 + \gamma^5 = 2.13$. Worker w_2 has a lower *maintenance score* than worker w_1 meaning that he has engaged in more maintenance tasks recently.

TABLE 1 – Historical activty for workers w_1 and w_2

Week	Worker w_1	Worker w_2
1	Maintenance	Drove
2	Drove	Drove
3	Maintenance	Maintenance
4	Drove	Maintenance
5	Drove	Drove

1.2.2 Creation of the tree

The maintenance score for all workers is computed for the next week $q+1$. This score helps in deciding which workers will be assigned to maintenance duties next week. We then get the first tree illustrated in figure 1 that takes as input a worker.

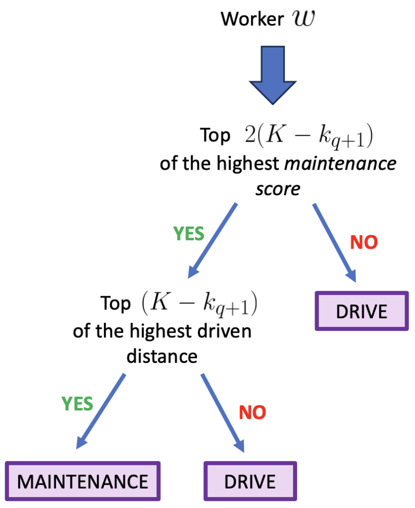

Worker w
Top $2(K - k_{q+1})$ of the highest *maintenance score*
YES
NO
DRIVE
Top $(K - k_{q+1})$ of the highest driven distance
YES
NO
MAINTENANCE
DRIVE

FIGURE 1 – First tree to choose the k_{q+1} drivers

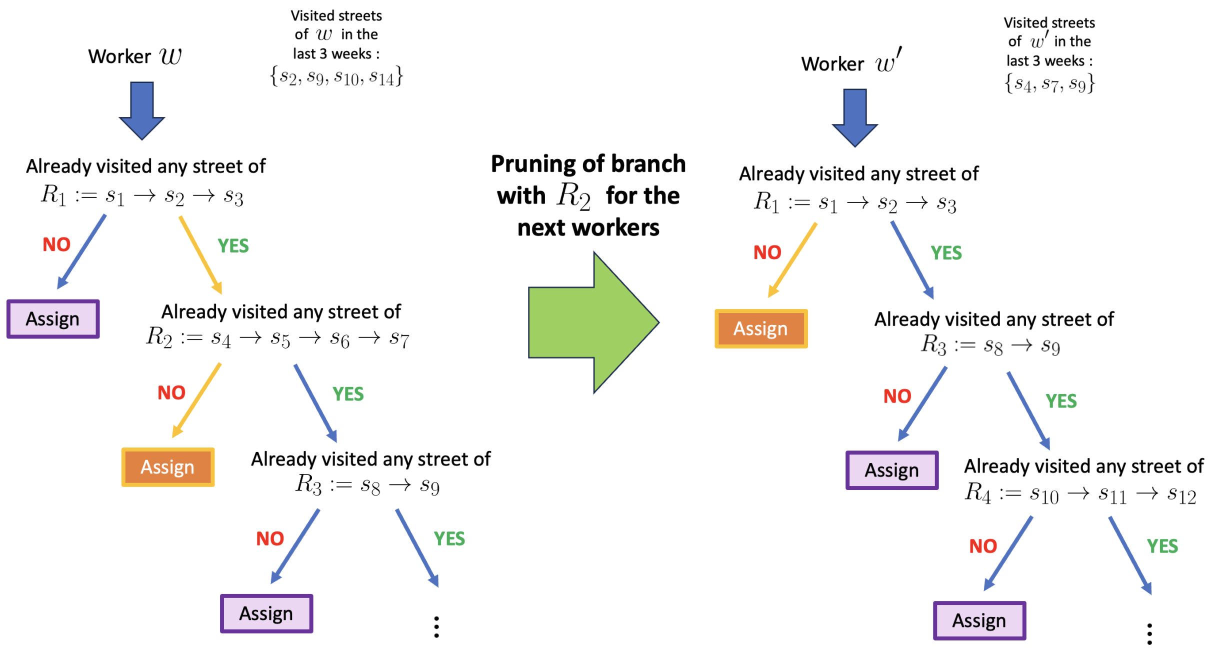
First of all, we test if the worker belongs to the top $2(K - k_{q+1})$ with the highest maintenance scores among all the workers. The purpose is to apply a first filter to keep only those workers who have not performed much maintenance recently. Following this, workers are split based on the total distance they have driven since week 1. The idea behind is that those who have driven more should be more likely to undertake maintenance tasks. So, the workers doing maintenance are simply the top $K - k_{q+1}$ workers who have the highest total driving distance from the $2(K - k_{q+1})$ previous workers.

And *voilà* ! Therefore, we get the k_{q+1} drivers. The remaining task involves assigning each of these k_{q+1} workers to one of the k_{q+1} routes.

1.3 Next trees : Assigning the drivers to routes

Recall that our goal is to ensure that each worker covers a diverse range of streets. To determine if a route R can be assigned to a worker w, the basic idea is that we check whether any street in route R has already been covered by worker w in the last three weeks. If worker w has already visited any street in route R during this period, then we should not assign route R to worker w.

The next trees will take as input a worker w and assign them to a route. Each node in these trees is a question like "Has worker w visited any street in route R during the past three weeks ?". If the answer is no, then route R is assigned to worker w : $R \equiv_{q+1} w$. We then remove this node for future workers as the route R has been assigned. Figure 2 clearly illustrates this tree structure and the pruning process.



Worker w
Visited streets of w in the last 3 weeks : $\{s_2, s_9, s_{10}, s_{14}\}$
Already visited any street of $R_1 := s_1 \to s_2 \to s_3$
NO
YES
Assign
Already visited any street of $R_2 := s_4 \to s_5 \to s_6 \to s_7$
NO
YES
Assign
Already visited any street of $R_3 := s_8 \to s_9$
NO
YES
Assign
Pruning of branch with R_2 for the next workers
Worker w'
Visited streets of w' in the last 3 weeks : $\{s_4, s_7, s_9\}$
Already visited any street of $R_1 := s_1 \to s_2 \to s_3$
NO
YES
Assign
Already visited any street of $R_3 := s_8 \to s_9$
NO
YES
Assign
Already visited any street of $R_4 := s_{10} \to s_{11} \to s_{12}$
NO
YES
Assign

FIGURE 2 – Process of the next trees

In Figure 2, R_i denotes the routes planned for the next week $q + 1$. These trees are built sequentially after each worker's assignment. Starting with a tree containing all routes R_i, it is pruned by one branch after each worker's assignment. This pruned tree is then used for the next worker, and the process repeats.

The orders of the R_i routes in the initial tree and the sequence in which workers pass through the trees are crucial. We want to pair longer routes with workers who have less cumulative driving distance. To do so, we sort the R_i routes in ascending order of their length and place routes with shorter lengths nearer to the root of the initial tree. Consequently, R_1 represents the route with the shortest length in the initial tree. Workers are then processed in descending order of their total driven distance. These orders ensure that workers who have covered more distance are given priority to be assign to the shorter routes. This aims to distribute the workload among the workers.

2 In-depth stage for the hands-on option

2.1 Quick code explanations

The code can be found here : https://github.com/soniquentin/CCCS660_McGill_Assignment2.

I implemented my solution in Python in the file `main.ipynb`. The notebook is divided in 5 parts :
— Part 1 : Import of libraries and definition of some global variables
— Part 2 : Definition of important functions to create a simulated city (where streets are represented by random 2D-coordinates) with streets to clean each week
— Part 3 & Part 4 : Generates route-assignment records, implements the decision trees described above and makes a route-assignment for the next week
— Part 5 : Integrates the functionalities of Parts 3 and 4 into a single function and conducts a simulation of the decision-making process over several weeks, not just for the one week.

In the implementation, there are 20 workers. The number of streets of the virtual city in total is 100. The city is a mere square and the streets are randomly placed in it. Each week, the number of streets to clean varies between 50 and 70. The discount factor γ is set to 0.9. The length of a route is determined by summing up the Euclidean distances between consecutive streets of the route.

2.2 Result

Route-assignment records were produced for an initial period of 15 weeks during which routes were assigned to workers randomly. Then, my decision-making tool was employed to handle route assignments for the next 15 weeks. Let's see how well it performs in figure 3. The dotted red line delimits the transition from random assignments (the first 15 weeks) to the assignments via the proposed decision tree method (the next 15 weeks).

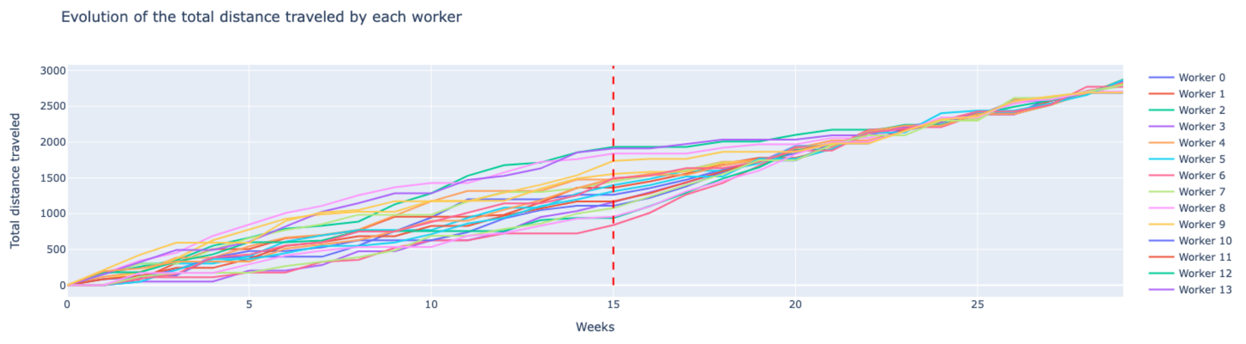


FIGURE 3 – Evolution of the total driven distance of each worker

As we can see, during the initial 15 weeks, there is an imbalance in workload occurred due to a bad assignment strategy : some workers travel significantly more than others. This issue was effectively addressed by our method which made all the total distances converging toward a line : **our decision-process tool helps to decrease workload imbalance !**

Now, let's look at the garbage collection on random street during the first 15 weeks :

$$\times \to \times \to 14 \to 10 \to 7 \to 14 \to \times \to 18 \to 5 \to \times \to 10 \to 3 \to 10 \to \times \to \times$$

Here, $\times$ indicates no cleaning was needed. As we can see, worker 10 visited this street two times within 3 weeks. In contrast, with our model, the worker sequence of the next 15 weeks is :

$$1 \to 3 \to \times \to 18 \to 8 \to 11 \to \times \to 1 \to 13 \to 14 \to \times \to 4 \to \times \to 18 \to \times$$

This demonstrates that our model promotes diversity and decreases the risk of workers covering the same streets within a period of three weeks. Many residents will be happy to meet Marion !