1. **Introduction to the problem**

The main objective of this project is to try to find the optimal way in which waste companies can send their different trucks considering that every day they can receive certain information of how the cans are now filled and some probability of how they will be the next day. We assume that companies have the opportunity to learn or have this kind of information. Finally, we must say that this problem is for not-quickly perishable goods, such as recycle goods, since the fact of having food that can be decomposed quickly can alter our decisions of not being necessary to go today but tomorrow.

We create an algorithm that every day takes the fulfilled cans information and split those cans in different clusters – clusters that depend on the number of observations violated -, to later apply the TSP algorithm finding the optimal route for one truck to do this route. Later on, we consider if it optimal to deviate from out path and collect the trash for the non-violated cans but with high potentiality to be fulfilled tomorrow. Given some fixed costs, we consider the one-step look ahead policy to decide if a can that is not currently violated should be introduced in the truck path or not. We will do this procedure every day, considering the information for the next day and considering that our current decision of to send or not to send a truck to a can will change its future fullness of the can.

1. **Current waste management system**

Currently, the Company that supplied us the data is sending all trucks they have every day to every can. We consider that this could be improved by sending strategically the trucks taking into account the structure of the city.

1. **Model**

The problem is to minimize the total cost that the system of sending or not sending trucks during a certain horizon has, given the fact that in every period we will have some dumpsters that are filled over a certain percentage threshold[[1]](#footnote-1) and some that will be optimal to also collect during our path to those already filled.

So, our minimization problem becomes:

***minimize Ew1,…,wN[gN(XNc) + gk(Xkc,Ukc,Wk+1c)]***

***{u0c,…,uN-1c}***

***s.t. Xk+1c=(1-Ukc)Xkc+Ek[Wk+1c]***

***Ukc ϵ {0,1}***

***Ukc = Ukv +Uknv (Tk) where Ukv=1 for Xkv>α***

Where c is the index for cans, X is a vector of the current percentage of fulfillment of the can (i.e., the current dump in the can divided by the capacity of the can), U is a binary decision of to collect or not to collect the trash from that c can, and W is the rate of rubbish generation that each can “experiences”, i.e., we will have a different Poisson distributions for each can that gives the number of trash that each can will receive in a day independently of all other cans. So, the expectation at time k of the generation of the rubbish that will happen between today that we need to plan and by the same time tomorrow is just the lambda from that Poisson distribution. *Ukc ϵ {0,1}* means that our decisions is to send or not to send a trunk to collect the rubbish of a certain can c. *Ukc = Ukv +Uknv (Tk)* means that our decision can be decomposed into the decision over the non-violated and the violated ones, and once we violate the threshold of fullness of the can we will collect them. For those that are not violated, we will decide to collect or not depending on a stopping criterion explained in the next paragraphs.

We have that our costs are represented as following:

***gk(Xk,Uk,Wk)=(1- Uknv)P(Xk+1nv>α | Xknv=xknv)F + Uknv(Ddev + SC) + Ukv (TD+SC)***

where *Uknv* is the decision vector in which we decide the elements for the non-violated cans (to go or not to go), and zeros otherwise, and *Ukv* is the decision vector for the violated cans, and then it is 1 for the position of the violated and zero otherwise. Ddev is the cost of deviating from our current path of servicing the already fulfilled, SC is the service cost and TD is the total distance from the first point to the last point that are already fulfilled. In next section we will explain how we calculate this distance.

*P(Xk+1nv>α | Xknv=xknv)* is the probability that with tomorrow we will be over the threshold alpha given that today we are at Xknv, and then*P(Xk+1nv>α | Xknv=xknv)= P((1-Uknv)Xknv+Ek[Wk+1nv]>α | Xknv=xknv),* if we do not go, we will face that our rubbish in the can will be a function of the current rubbish and the rate of throwing rubbish and then: *P(Ek[Wk+1nv]>α- xknv | Xknv=xknv),* which will be implemented as the probability of finding *Ek[Wk+1nv]= α- Xknv* given a lambda-distributed Poisson. And then:

*gk(Xk,Uk,Wk)=(1- Uknv) P(Ek[Wk+1nv]>α- xknv | Xknv=xknv) F + Uknv(Ddev + SC) + Ukv (TD+SC)*

Finally, the stopping criterion for the non-violated is as following:

*Uknv (Tk)=*

Which is the same as:

***Uknv (Tk)=***

1. **DP Implementation**

Then, our main objective is to construct our path of going with the violated points – that we assume we need to go – and then consider to deviate or not to deviate for those currently non-violated but potentially violated in the future.

Firstly, we randomly create a Poisson distribution per each can with three steps ahead. The numbers which are in the first column and are 1 are set into zero, and we add then the first and the second column in order to think that we do not start with zero rubbish in the first day. Secondly, we estimate those cans for which the % of rubbish in the can is over the threshold – which is 90% -. Those ones will be the violated ones and we will go for sure to those locations.

With those violated points we do different clusters (we assign one truck per cluster – the bigger the dataset of violated, the bigger the number of clusters), and later we apply the TSP per each cluster in order to consider the optimal path that ONE truck will make to collect rubbish from those cans.

1. **Results and conclusions**

1. Assumption: the ones which are already filled today will be collected [↑](#footnote-ref-1)