Université de Lille **Master 1 Data Science**

Clustering trajectories project

Computer Science Refresher

Lucas DESPIN, Théo FAGNONI
Centrale Lille - Université de Lille, France

08/11/2020

Contents

I. Core idea

Considering a set of trajectories we want to cluster, the idea we chose to implement could be summarized as a dichotomy. By defining a certain scale of precision, the analysis of the trajectories is performed on elementary « boxes » on the map. The clustering method is applied to each of those boxes, thus the choice of the scale belongs to the user and the kind of trajectories he is looking at. In fact, if the scale is too small the result won't have any sense as too many information would be lost, and if the scale is too high the clustering method won't be efficient enough to simplify the input set of trajectories.

On each elementary box, the clustering consists in 3 steps :
(1) Identify which trajectories come in and out of the box by taking the impacts on each border of the box.

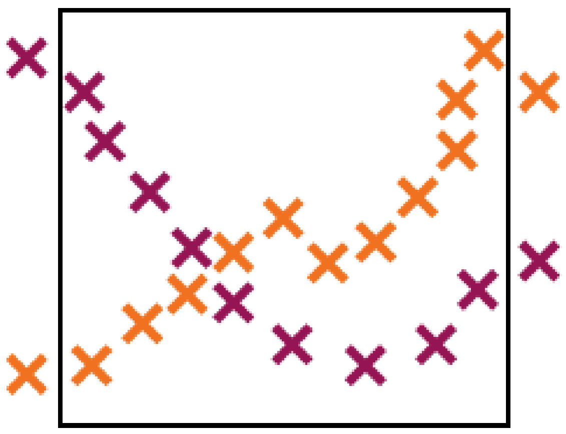


Figure 1: First step

(2) Calculate the impact points on each border of the box.

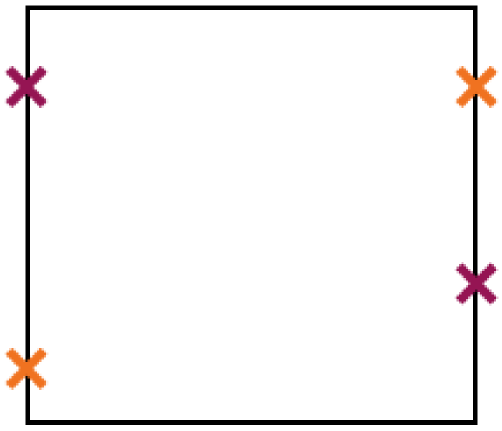


Figure 2: Second step

(3) Cluster the trajectories involved in the box, by replacing their subsets of points in the box by the pair(s) of average impact points corresponding.

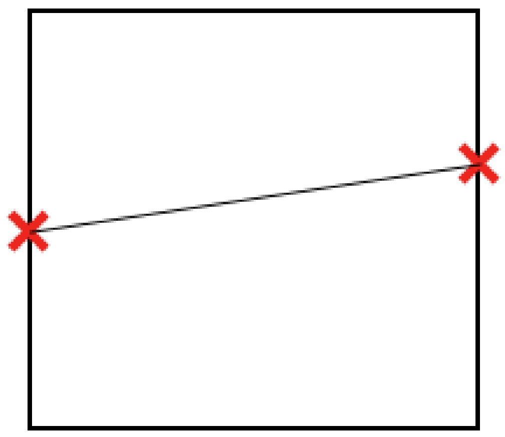


Figure 3: Third step

We can denote that the transitivity of the method is assured by the fact that the calculations of the average impact points are the same from a box to the successive box as they share one border.

The issue of the lack of points for a trajectory, compared to the other trajectories or to the scale chosen, is solved thanks to the introducing of indexes for points in a trajectory.

II. POO : classes definition

We used object oriented programming : points, trajectories and boxes are objects we can manipulate and associate attributes to.// // Here are the classes we introduce :// //

- Point :
  - .x : coordinate along x
  - .y : coordinate along y
  - .ind : indice in the trajectory
- PointImpact :
  - .x : coordinate along x
  - .y : coordinate along y .ind : indice in the trajectory
  - .box : indice of the box it belongs to
  - .border : the border of the box it is on
- Trajectory :
  - .traj : list of points that composed the trajectory
  - .pairs : list of pairs of points, along with a marker to know where the trajectory is getting in (marker = 1) or out (marker = -1) of a box
  - .impactpoints : list of calculated points corresponding to the impacts of the trajectory on the boxes it crosses
  - .num : indice of the trajectory Trajectories :
  - .trajectories : list of all the trajectories

- AVGPoints : the points used to reconstruct our trajectories
  - .AVGPoints : list of all average points, calculated on each border of each box, by taking the average of the coordinates of the corresponding impact points
- Box :
  - .xmin : left border
  - .xmax : right border
  - .ymin : bottom border
  - .ymax : upper border
  - .ind : indice of the box
  - .impactPoints : 4 lists of points corresponding to the impact points from every trajectories on each border (u : up, b : bottom, l : left, r : right)

III. Functions

To perform the three steps, the first function we implemented was **crossingPoints(boxes,Traj**. Taking a box and a trajectory, it returns the pairs of points corresponding to the in and out of the trajectory we consider.

Then, **defImpactPoints(boxes,Traj** allows us to conclude the second step, by creating the impact points where the trajectory collides with the box.

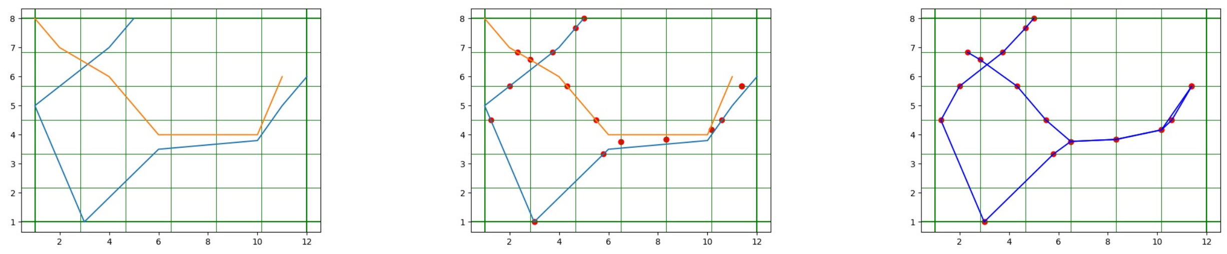
The function **AvgPoints(boxes, K)** perform the third step, for every boxes and a given scale K. K corresponds to the degree of subdivision we're gonna deep into, a subdivision consisting into splitting a given box into 4 boxes of the same area. Having, for each box, its impact points on its borders, we calculate the average impact point for each border of the box.

The last step consists into the construction of the simplified maps, composed of clustered trajectories. The function **consTraj(boxes,Traj,K)**, looks for each trajectory to its ordered impact points, and compare each of them to the average impact points relative to the set of boxes. To do so, we refer to the box and border attributes of those points. When all the matches are found for each trajectory, when can successively construct the clustered trajectories made of average impact points.

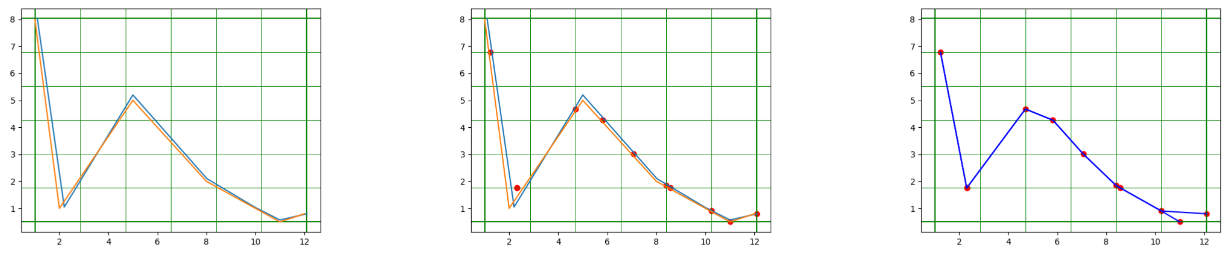
IV. Results

This last section aims to present our results for several test cases.// For each cases, from left to right, we show (i) the trajectories to cluster, (ii) we add the calculated average points, (iii) the clustered trajectories.

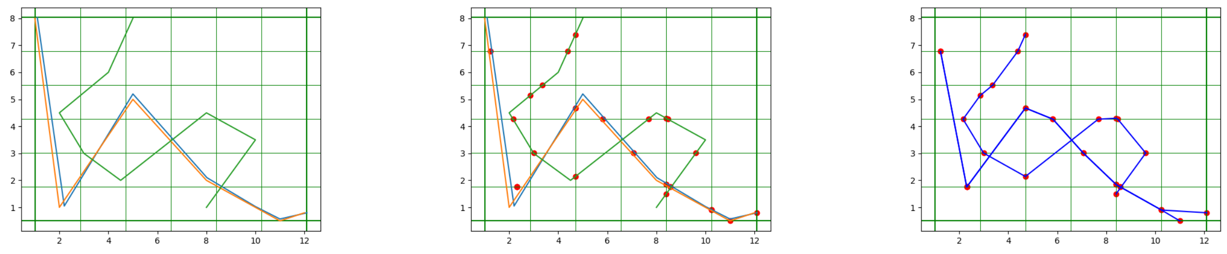
i. Test i



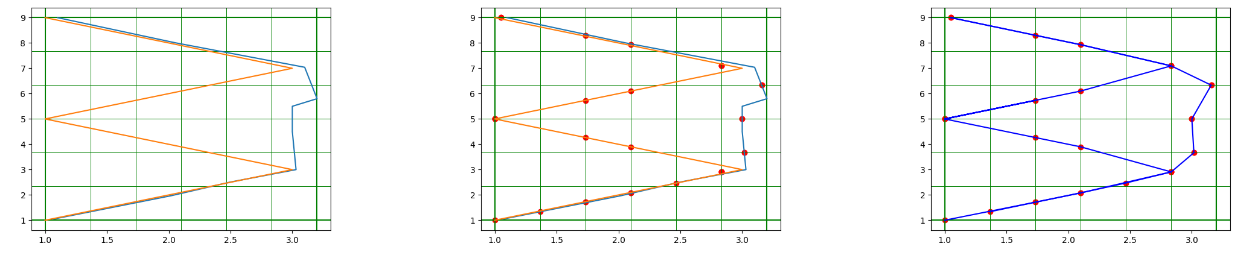
ii. Test ii



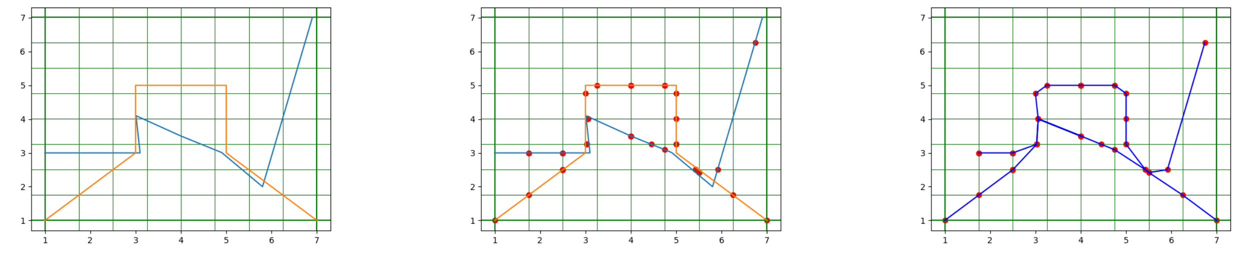
iii. Test iii



iv. Test iv



v. Test v



vi. Test DB cabs