Snap2Route Breadcrumb Deviation Algorithm

For CS510 Data Explorations, Summer 2020

July 13, 2020

Deviation Algorithm Given an input set of timed breadcrumb records B along a single shape and a single set of coordinates S defining that shape, what is the projection of B onto S?

Note: The set B may contain breadcrumbs from multiple trips.

Principle 1 The projections onto S should be in chronological order.

Principle 2 If some point p has a large error but its neighbors do not, then p should be assigned a large deviation value and its neighbors should not be assigned large values. Especially, if p is out of order but its neighbors are not, then p should be the only point assigned a large deviation penalty.

COMPUTE-DEVIATIONS(S, B)

```
1  Sort S by pointOrder
2  for every record r_i ∈ S
3      if r_i.pointOrder ≠ 0
4          Set r_i.distTraveled = dist(r_i, r_{i−1})
5      else
6          Set r_i.distTraveled = 0
7  for every record r ∈ B
8      r.projection = nearest point on all segments in S
9      r.distTraveled = r.projection's distance along S
10 Sort B by tripID, time // Note we don't compare crumb records to each other until this point
11 for Each unique tripID t ∈ B
12     for Each record r_i ∈ B such that r.tripID = t
13         if r_i.distTraveled < r_{i−1}.distTraveled
14             Look backward in t to find correct projection halfway between new neighbors
15         elseif r_i.distTraveled > r_{i+1}.distTraveled
16             Look forward to find correct projection halfway between new neighbors
17 return B
```

A Note On Syntax In Python, time complexity is hard to determine due to the obfuscated nature of the abstract data structures. It appears that efficient operations do not loop through large record sets within the python code; instead, such loop control is delegated to libraries. Thus, I anticipate our algorithm will be implemented with function calls that do not syntactically resemble the above psuedocode.

Time Complexity Overall worst-case time complexity is given by the relation $O(|S|) + O(|B||S|) + O(|B|\log|B|) + O(|B||T_{max}|) \in O(|B|\log|B|) + O(|B||T_{max}|)$, where $|T_{max}|$ is the maximum number of breadcrumb records associated with any single trip. (Note that $O(|B||S|) \in O(|B||T_{max}|)$ because the number of breadcrumbs per trip far exceeds the number of points in a trip's shape.)

In the best case, no breadcrumb projections are out of chronological order and the time complexity becomes $O(|B|\log|B|)$. The best case more closely resembles the expected case, based on Sam Gomena's observation that the breadcrumbs rarely (if ever) run backward chronologically.