# HH GPS Cluster Analysis: Ordering points to identify the clustering structure (OPTICS)

## Introduction:

Accurate customer location is the key in matching EOBR stops with SCO stops. We are sourcing customer location from four sources(NWK_NETWORK, GOOGLE API, SCO TRIP NODE and SCO handheld Cluster) to improve matching percentage. From our analysis we found that there are certain cases where current SCO Handheld cluster logic is not effective in matching with EOBR stops and there is an opportunity to improve the cluster logic by using more advanced algorithms like Ordering points to identify the clustering structure (OPTICS). I have developed a python code to implement the OPTICS algorithm and results are discussed in the below sections.

**Current SCO HH cluster logic:**

1. Get the history of all SCO stop completion LAT/LNG for each customer.
2. Define a boundary of 100m for each of these points and count the number of other GPS points fall into this boundary.
3. Choose the point which has highest count number as the centroid of the cluster. If more than one point has same count then select one randomly.

The Assumption in this approach is each customer will have only one cluster and calculating the centroid of that cluster may give the point where the chances of driver completing the stop close to it is high.

When the LAT/LNG calculated from this cluster logic is used in the EOBR SCO matching logic along with the LAT/LNGS from NWK NETWORK NODE, GOOGLE PLACES API and SCO TRIP NODE it improved the matching by 20%. If LAT/LNG from the cluster logic alone is used matching percentage is only 63%. Ideally, LAT/LNG from the clusters should be able to match more than 63% as these are the locations or GPS points where our drivers finishes their stops regularly. This rises the questions on the efficiency of the current Handheld cluster logic we implemented and this led to the further analysis on cluster logic.

One of the cases where EOBR stop cannot be matched to a SCO stop.

Ex: TRP_INSTC_NBR = 4226725609703
NODE_INST_ID = 810550801

From the Map below we can observe that for the customer 'Caterpilllar' there are multiple clusters of Handheld data(blue dots) where drivers had completed the stops(this data is from 2014). This is because the customer has different warehouses at the same location with the same address. In this case, the centroid calculated (green circle in bottom left) is very far from the EOBR stops(Blue circles at top right) and also NWK Network(red circle at top right), goole LAT/LNG doesn't match with EOBR stops. So these EOBR stops are showed as unscheduled EOBR stops in the report and may lead to unnecessary conversation between service center manager and driver even though it's a valid stop. This may affect the users confidence in the report.


## Ordering points to identify the clustering structure (OPTICS):

This proves that a simple clustering logic is not sufficient to identify the clusters for a customer. So I thought of implementing a supervised machine learning clustering algorithm to identify all the possible clusters and calculate the centroids for these clusters. I have chosen **Ordering points to identify the clustering structure (OPTICS)** algorithm to begin with. This supervised machine learning algorithm helps to find density based clusters in spatial data. You can read more about OPTICS here https://en.wikipedia.org/wiki/OPTICS_algorithm

To satisfy our requirements I have added an additional logic to calculate the centroid and region(centroid and the farthest point from the centroid) for each identified cluster and store them in a database table.

Pseudo logic:

```
OPTICS(DB, eps, MinPts)
    for each point p of DB
       p.reachability-distance = UNDEFINED
    for each unprocessed point p of DB
       N = getNeighbors(p, eps)
       mark p as processed
       output p to the ordered list
       if (core-distance(p, eps, Minpts) != UNDEFINED)
          Seeds = empty priority queue
          update(N, p, Seeds, eps, Minpts)
          for each next q in Seeds
             N' = getNeighbors(q, eps)
             mark q as processed
             output q to the ordered list
             if (core-distance(q, eps, Minpts) != UNDEFINED)
                update(N', q, Seeds, eps, Minpts)
                 coredist = core-distance(p, eps, MinPts)
    for each o in N
       if (o is not processed)
          new-reach-dist = max(coredist, dist(p,o))
          if (o.reachability-distance == UNDEFINED) // o is not in Seeds
              o.reachability-distance = new-reach-dist
              Seeds.insert(o, new-reach-dist)
          else               // o in Seeds, check for improvement
              if (new-reach-dist < o.reachability-distance)
                 o.reachability-distance = new-reach-dist
                 Seeds.move-up(o, new-reach-dist)
```

**Inputs to the algorithm:**

1) Handheld GPS points data set
2) Parameter: Minimum points to be in the cluster
3) Parameter: Maximum radius to consider for the cluster
4) Parameter: Threshold distance between each cluster

After implementing the algorithm for the customer ‘810550801’

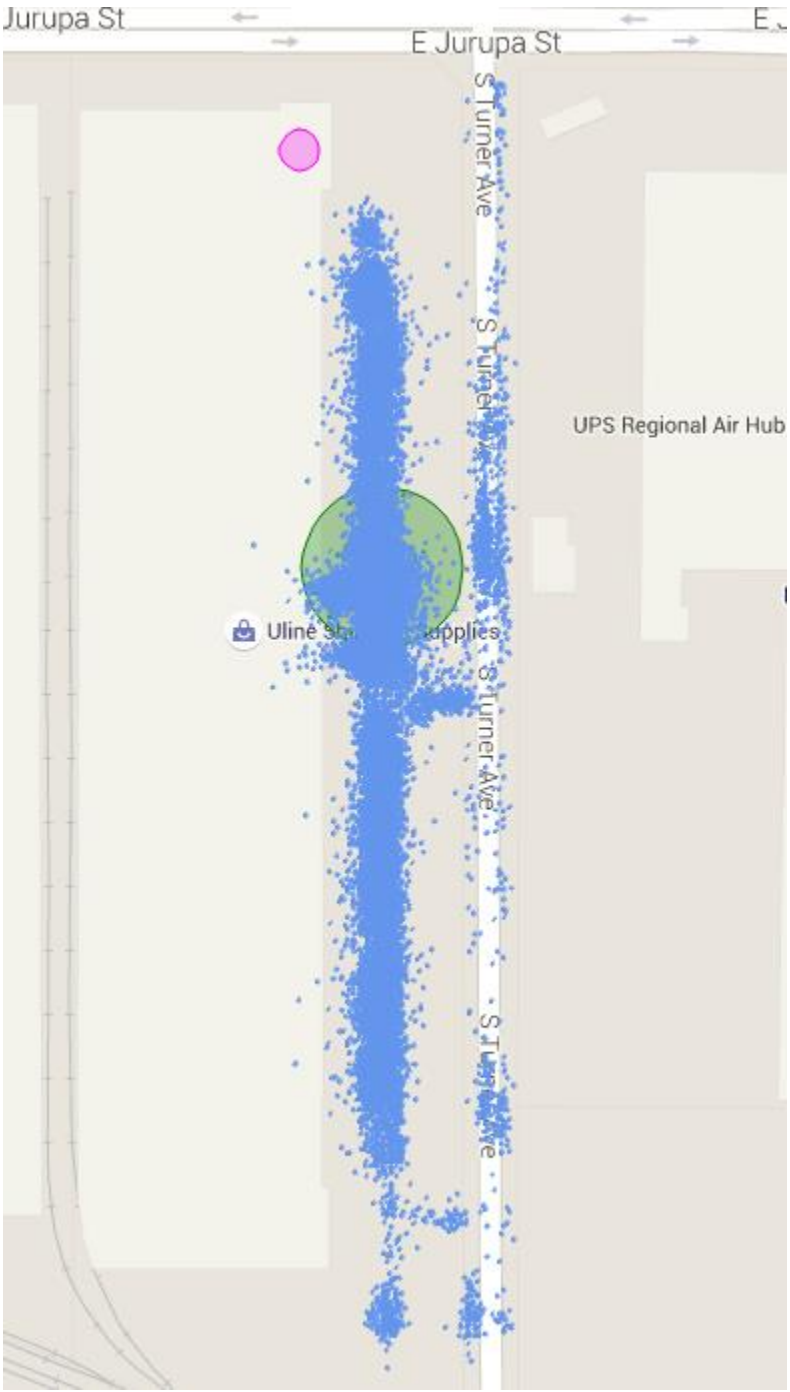

Before


After OPTICS (Min Points: 100, Max cluster Radius: 100, Cluster Threshold: 10)

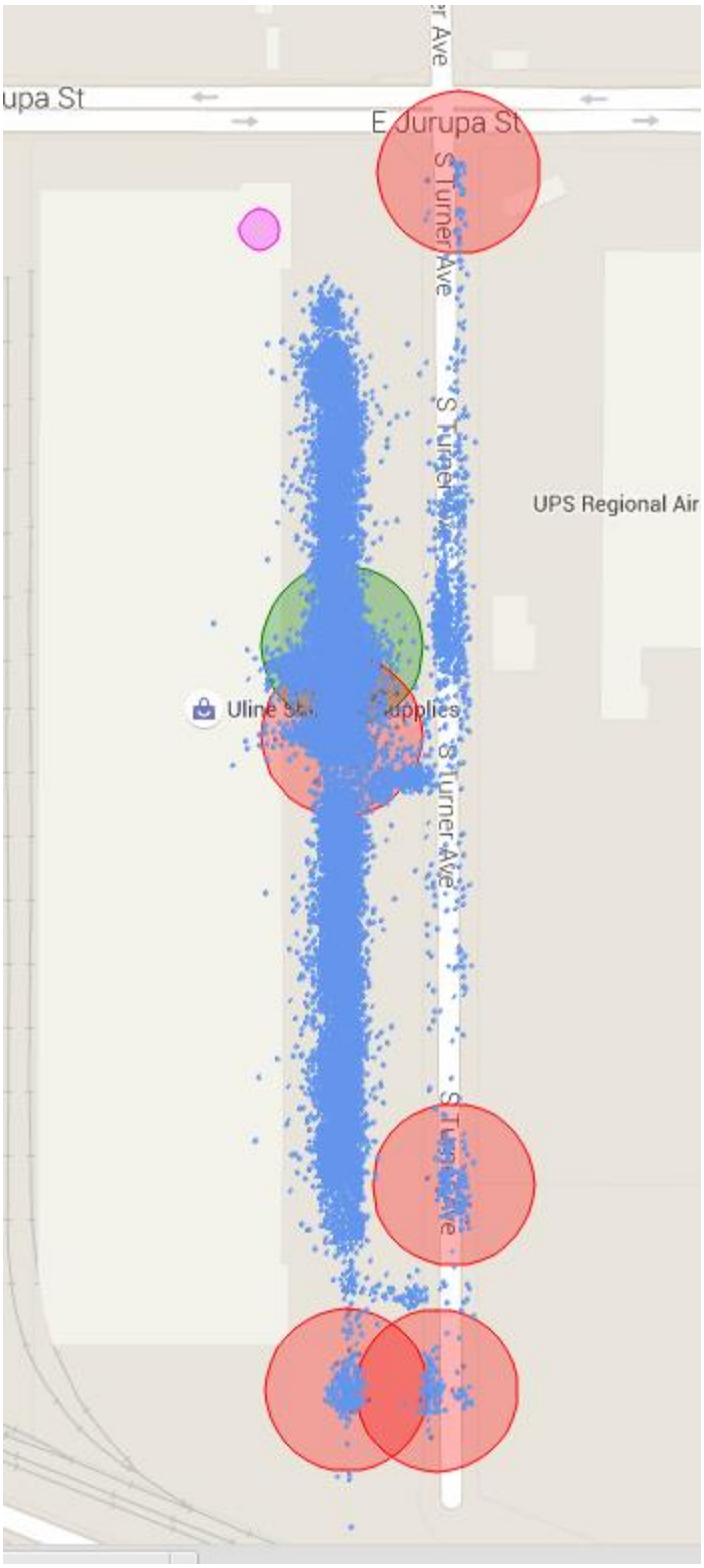
From the above picture it clearly shows that algorithm is able to identify the spatial clusters and calculated the centroids for each cluster (red circles). With these new LAT/LNGs associated to this customer we are able to match the EOBR stops (which we couldn't using the previous logic).

The characteristics of each cluster can be controlled using the parameters Min Points, Max cluster Radius and Cluster Threshold.

Here is another interesting case NODE_INST_ID = 432962948




Before

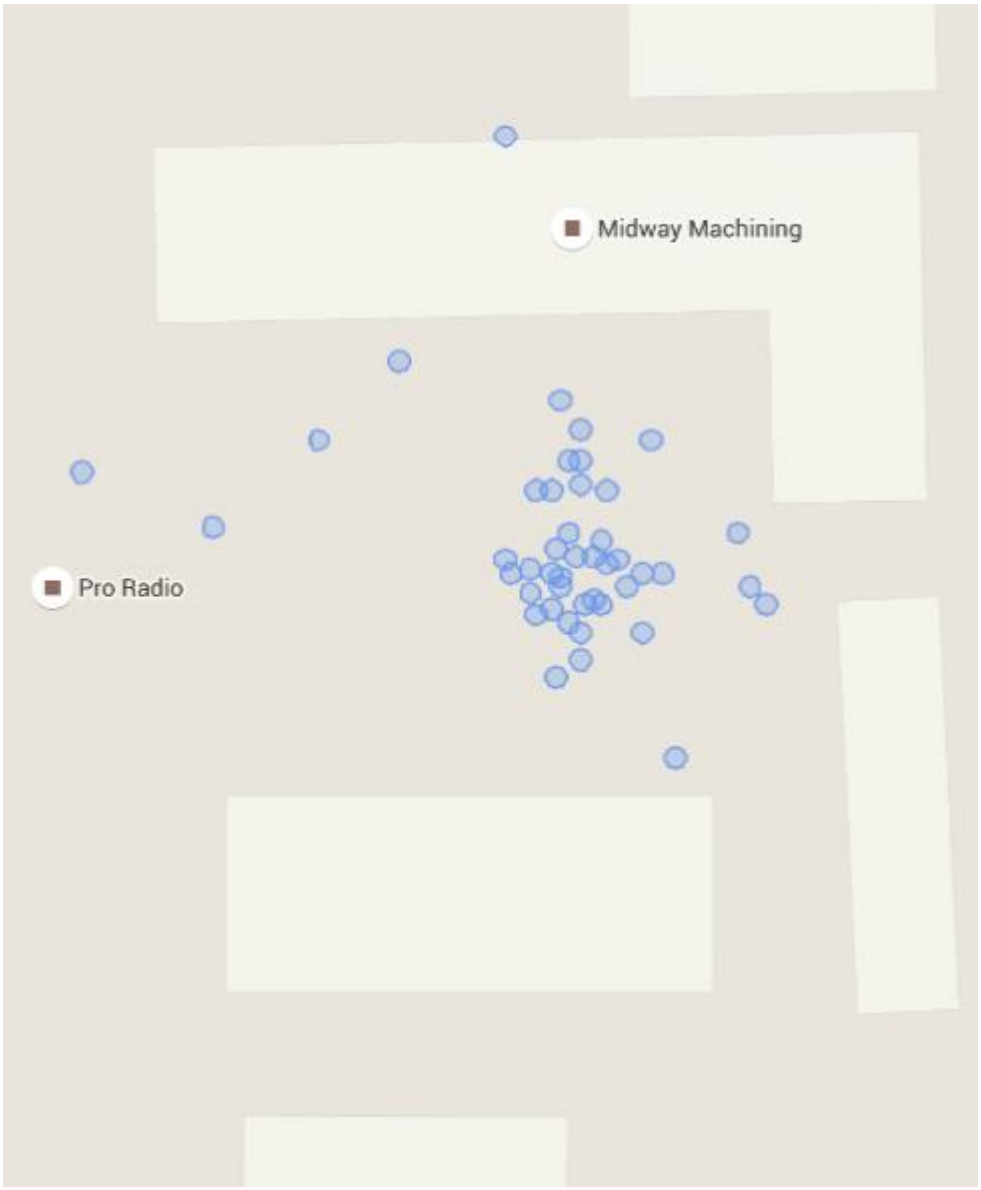



After OPTICS (Min Points: 100, Max cluster Radius: 100, Cluster Threshold: 10)

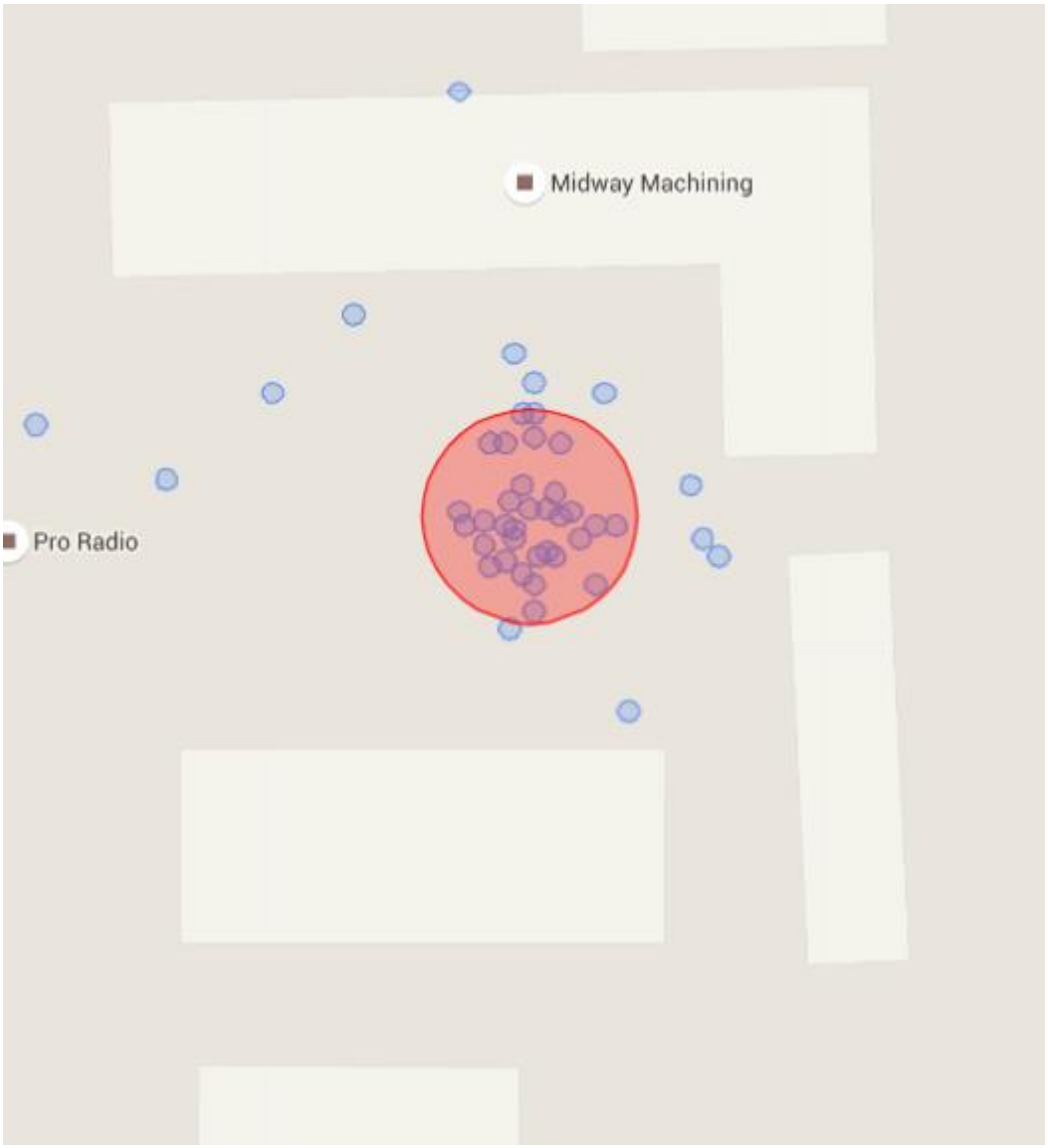
For this customer,  points are uniformly distributed and based on the old logic we were able to identify only one centroid(green). It clearly shows EOBR stops can happen very far from  the centroid point and google point(Pink circle).

But the Optics was able identify multiple clusters and calculated the centroids for them. With this we have new and more accurate knowledge on where a stop could be completed and increases the chances of matching the EOBR stops with SCO.

Cases where there is very less data NODE_INST_ID = 633453061




Before




After OPTICS (Min Points: 50, Max cluster Radius: 10, Cluster Threshold: 10)

## Conclusion:

With OPTICS it is possible to identify most of the possible locations near a customer where a stop could be completed in Handheld and thus increases the chances of matching EOBR stop with SCO. This helps to avoid to show valid EOBR stops as unscheduled stops.

Further, I plan to integrate this with Google Tensorflow to improve the performance of the algorithm.