Pareto Optimization over Route Data

Paul Galatic
pdg6505@rit.edu

Sarthak Gupte
sg7179@rit.edu

ABSTRACT

Given a corpus of unlabeled GPS data representing routes, this paper describes an analytical approach to find the best route. Our approach uses multi-objective optimization of both the number of stops in a route as well as the total duration of a route. We prioritized de-duplication of routes so that we could get them into a state where they are easily compared, and then provide a visual representation of Pareto efficiency. This leaves the final decision as to the best route to the viewer.

1. OVERVIEW

We were provided roughly 1.77GB of unlabeled GPS data representing routes from home to work. Our objective was to determine which route, or set of routes, was most optimal. We decided to keep our implementation as simple as possible so that we could stay flexible and cover a lot of ground quickly. We also prioritized data cleaning over data mining.

There are two reasons for this. First, the dataset provided was extremely dirty and in a form that is difficult to use (GPS data in NMEA format). Simple approaches can work well on clean data, and if we invested too heavily in a complicated approach, we might have bit off more than we could chew. The second reason is that Pareto optimization works best with clean data.

Section 2 details the division of responsibilities. Section 3 discusses our data cleaning and anomaly detection procedures. Section 4 describes the design of our cost function. Section 5 describes how our algorithm detects stops. Section 6 provides further details of our algorithm. Section 7 reports our findings, including the visual components of our results. Section 8 mentions issues in our current implementation and trade-offs between simplicity and generalization. Finally, Section 9 wraps up our efforts and suggests possible improvements for future work.

2. TEAM COMPOSITION

Paul Galatic was responsible for Checkpoint 1, which put him in charge of developing the exterior structure of the program. He implemented functions to parse KML files, evaluate routes, and perform Pareto visualization on cleaned data. He was also responsible for writing the first draft of the report.

Sarthak Gupte was responsible for describing the cost function in Checkpoint 2. He also designed code to perform route de-duplication and anomaly detection to ensure that the data provided to Pareto visualization was as clean as possible.

Both team members reviewed all work before submission and collaborated to ensure it was completed to the highest reasonable standard.

3. CONVERTING GPS TO KML

The GPS data we were provided is extremely dirty. Many routes are parsed correctly by our library of choice, pynmea2[1]. However, a significant portion exhibit errors or anomalies. For instance, it is unlikely for someone to drive across the Atlantic ocean. This data was, unfortunately, unusable.

We performed several types of anomaly detection.

- Some lines were unparseable by pynmea2. We skipped these lines, assuming them to be the results of poorly formatted data.
- Some routes had a final timestamp that was earlier than its first timestamp. We assumed that the data was not collected with a DeLorean time machine, and thus discarded these routes.
- Some routes had waypoints that stretched far across the Atlantic ocean. We assumed that the data was not collected with an amphibious vehicle, and thus discarded these routes.
- Some routes had waypoints that jumped from one side of Rochester to the other between messages. These routes were also discarded.
- Routes that were longer than three hours or spanning more than 300km were also ignored, because they describe frankly unreasonable commutes. Routes that were shorter than 60 seconds or shorter than 100m were discarded as unreasonably optimistic.

[1] https://github.com/Knio/pynmea2

We define a "duplicate" route as one that has the same start point, end point, and general path. Our "de-duplication" process involved assessing all the routes, and then averaging the cost vector of all routes that we found were similar. This would account for the case in which two routes, sharing all other features, diverged sharply in the number of stops because of bad luck with street lights. The final cleaned dataset was provided to the Pareto visualization function.

4. COST FUNCTION

The cost function we chose was to minimize is the time taken to complete the trip and to minimize the number of stops. We expected that the overall fastest route will involve some number of stops, and that as the number of stops decreases, the duration of a route will increase. This is also known as Pareto efficiency/optimality.

This representation allows the user to choose the route that best fits their current situation. After all, the answer to "Which is best?" is almost always "It depends." As regularization, we applied a scaling reduction in cost based on the normalized distance traveled by a route. In our implementation, the cost of a route is reduced by 10% when distance is minimum, and unchanged otherwise.

The full equations can be seen below for both Pareto objectives C_i where α is the regularization constant[2] T is the duration of a route, S is the number of stops in a route, and D is the distance traveled by a route normalized between 0 and 1.

$$C_1 = T * ((1 - \alpha) + (D * \alpha)) \tag{1}$$

$$C_2 = S * ((1 - \alpha) + (D * \alpha)) \tag{2}$$

5. STOP DETECTION

Because our data is unlabeled, we did not use a classifier. Identifying stops or slowdowns is relatively easy as long as you are not looking for stoplights. We feel that falling below a certain speed is equally aggravating no matter when or where it occurs. As a result, we decided that if the car did not move at least 5 meters between messages, it was stopped. It was not moving again until it moved at least 10 meters between messages—this was to prevent noise in the GPS from making the car appear to start and stop multiple times.

We included logic to account for situations where the car stopped for very long periods of time, presumably outside a grocery store or a gas station. A stop is only added if we have been stopped less than that 4 minutes.

6. ASSIMILATION ACROSS TRACKS

Our design was intentionally kept simplistic so that results could be generated and iterated on as soon as possible. As the dataset was a relatively small 1.77GB, there was no need to invest in a database instance or other advanced data manipulation techniques; the entire dataset could easily fit in the memory of an average computer. Because we assumed each route was independently and identically distributed, we could test our algorithm for bugs on a small subset of the gps files before running it on the full dataset.

[2]Our implementation used $\alpha = 0.1$.

Our main Python program ended up being around 500 lines long, including comments and functions that served only to write the headers and footers of KML files. We used an external module to store constants. Our programming approach was as "top-down" as possible, where top-level functions recursively assume the presence of helper functions. By implementing these helpers, we keep the size of functions small and their scope easily explained.

7. RESULTS

Our anomaly detection was so thorough that all the routes that did not get discarded had some merit. They were then easily de-duplicated and compared using our cost function. Ultimately we found two Pareto optimal routes, both of them only slightly different from each other.

Our assumption that the number of stops may not correlate with the duration of a route did not end up being true. This makes intuitive sense; after all, a route that has many stops will naturally take longer than a route with few stops. As a result, our Pareto frontier ended up very small, with only a few results. However, this in and of itself provides strong evidence as to the optimality of these routes, given our cost function. Choosing the truly optimal route of this pair is left as an exercise to the reader.

A screenshot of our results can be seen in Figures 1 and 2. The Pareto frontier generated by our program can be seen in Figure 3.

8. DISCUSSION

There were many difficulties with our algorithm. The first is that it is inflexible. Unlike a clustering algorithm or decision tree, our approach may not generalize to other datasets, even with similar objectives. It was created with many assumptions, like the start and end points of each route being relatively static, and for routes to not be longer or shorter than certain durations or distances.

The prevalence of many thresholds throughout our program create rigidity and make debugging difficult. This is a trade-off in exchange for sheer simplicity and ease of prototyping. If we had focused on more complex or robust solutions, we may not have had enough time for data cleaning.

9. CONCLUSION

Given the corpus of roughly 1.77 GB of noisy and unlabeled GPS data, we were able to find the Pareto optimal routes. Our objectives were to minimize the number of stops and slowdowns and also to minimize the time taken to complete the route. Pareto Optimization helped us find the best route possible without compromising user choice.

De-duplication of routes was performed using a combination of a similarity metric and coordinate checking to make sure that every route assessed for Pareto optimality was as unique as possible. Instead of using a complex data mining algorithm, we decided to focus on data cleaning and preparation. This helped us develop a deep understanding of the data and identify the vast variety of anomalous GPS routes and handle them appropriately.

Future work in this area could include the detection of turns. There are many reasons to avoid left turns in a route, and that could serve as an alternative, or additional, regularization factor.

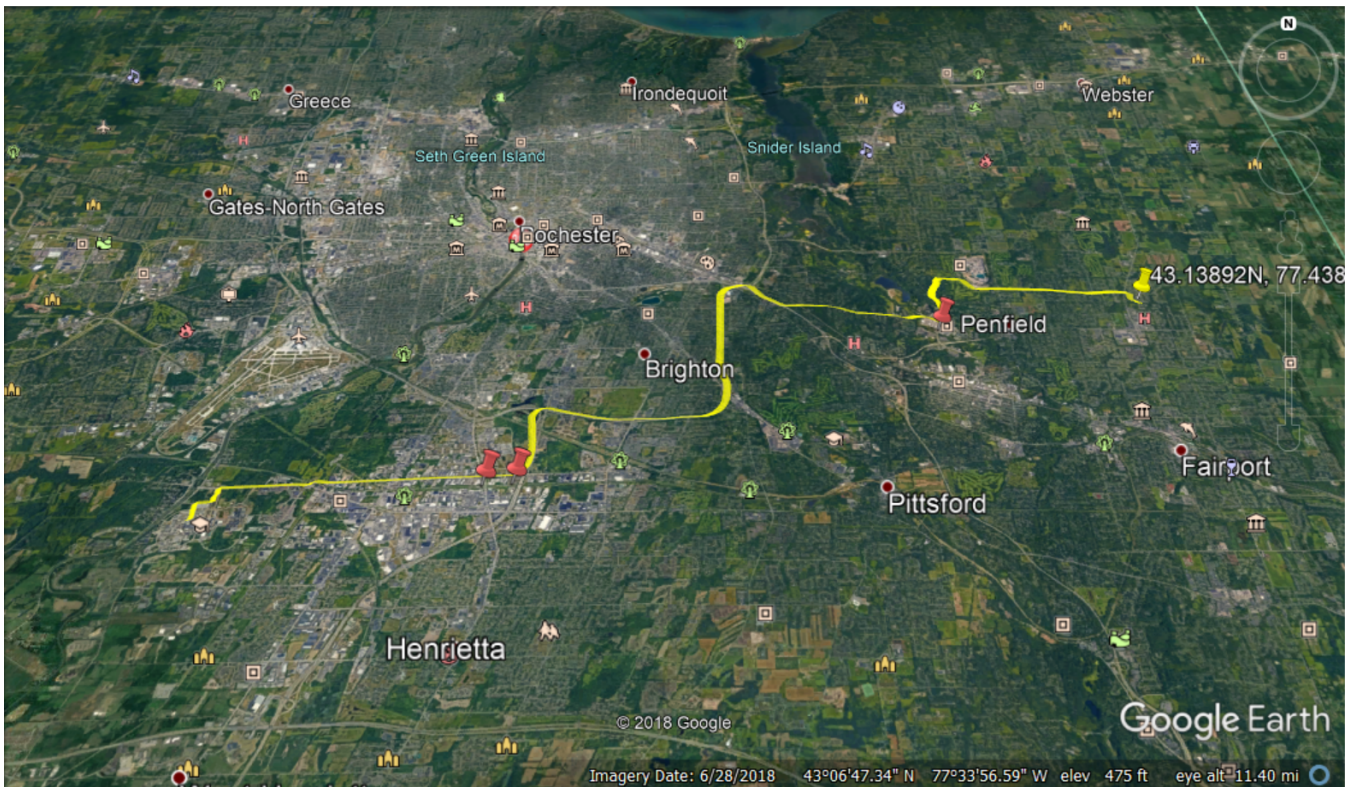

Figure 1: The first Pareto optimal route. Its label was 2019_04_04__1422_18.

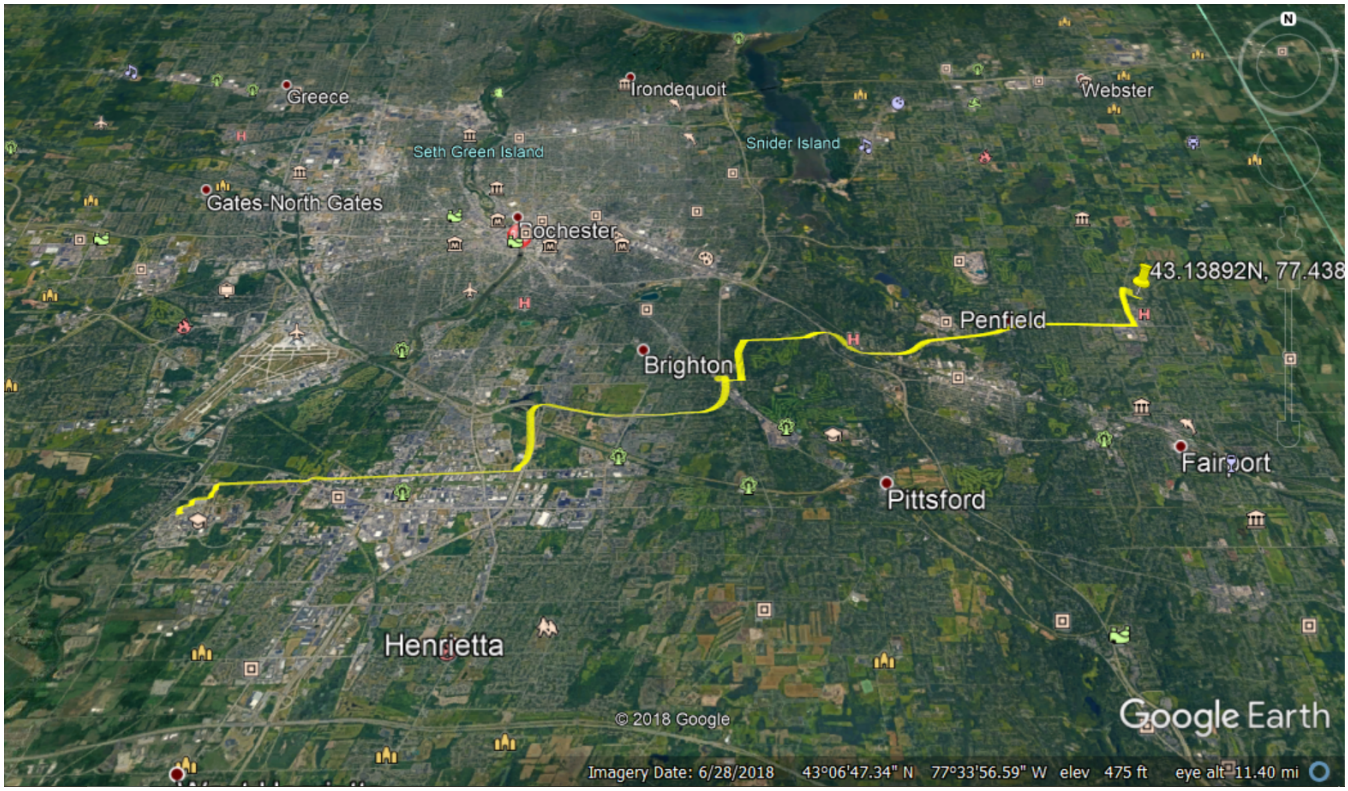

Figure 2: The second Pareto optimal route. Its label was 2019_05_04__1230_23.

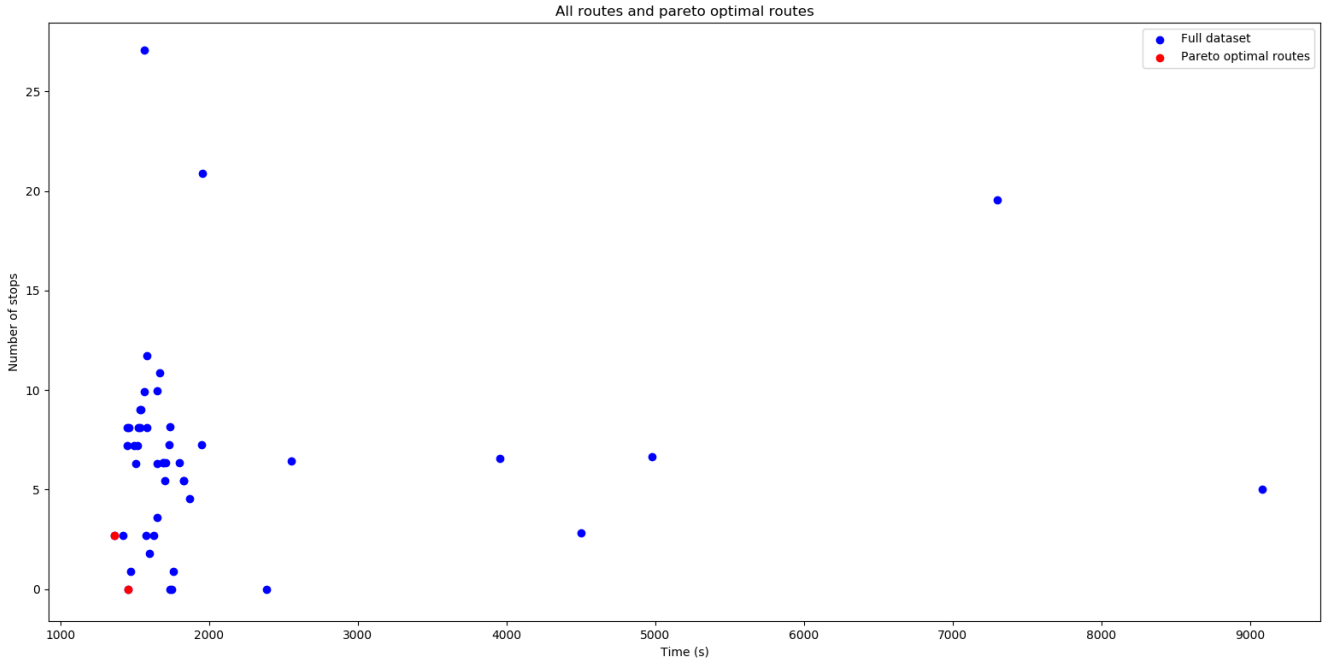

Figure 3: The Pareto optimization results. Each dot represents a route. The red dots represent Pareto optimal routes.