Data Analysis of Mileage Savings via Ride Sharing

CS480 Project

**Background and Objective**

Taxi rides are expensive and the mileage driven contributes to climate change. Ride-sharing can mitigate both negative impacts. In a ride sharing mechanism, the passenger submits a ride request shortly before departure specifying some parameters such as pickup time, trip origin and destination, willingness to ride-share, the maximum delay tolerated due to ride-sharing. For the purpose of this project you can assume that all trips are willing to be shared.

There are several open data sets available that record taxi operation in major cities. An example is NYC taxi cab data [a]. The dataset generally includes the taxi pick-up and drop-off locations with time stamps.

This project evaluates the potential of ride-sharing using a relational database representing nearly 700 million trips in New York City.

The goal of the project is to combine trips into ride sharing pairs given various constraints. This project will analyze the number of miles driven that could have been saved by ridesharing in the above database.

**Task**

Query the trips database in SQL and determine how trips can be combined in an optimal way, i.e. in a way that maximizes the number of miles saved. At most 2 original trips can be combined into a single ridesharing trip. The taxi maximum occupancy is assumed to be 3.

You need to query the trips database, merge trips (allocate *at most 3* passengers in a taxi), and create a ridesharing-trips relation. In deciding whether 2 trips can be merged you can assume that a taxi is available immediately at the origin of your choice. Each tuple in the ridesharing-trips relation must provide the keys of the two trips that are merged, and the order in which the passengers are picked up and dropped off.

Analyze what distance will be saved if:

1. Each ride in the combined trip can be picked up at most y minutes before the pickup time of the single (non-ridesharing) trip, and

2. Each ride in the merged trip can be dropped off at most y minutes after the drop-off of the single (non-ridesharing) trip.

y is called the *delay* due to ridesharing.

A possible query strategy is the following: if the delay is 5 minutes, read consecutive 5-minutes pools, where a *pool* consists of all the trips that started within the 5 minutes interval. Within each pool determine:

1. what pairs of trips can be merged (an example of an algorithm of doing so is shown in the Appendix), and

2. what trips should actually be combined in order to maximize mileage saved. You can use an off-the-shelf maximum matching algorithm

Another strategy is to query pools of 10 minutes.

Use MySQL for working with the NYC taxi dataset. Feel free to use any tools and languages you are comfortable with.

**Outcome**

The project will be summarized in a final report that will be submitted and also be presented in class using prerecorded video. The video will be followed by questions and answers from the instructors. There will also be a presentation (using the same format of video + Q&A) of preliminary results in mid-semester.

The project will be evaluated as a whole; also, each student will be evaluated for individual contribution to the project.

The overall project will be evaluated based on the following criteria, which must be included in the project report:

* the database querying strategy (what are your queries),
* the performance (the faster the running time per analyzed month, the better), and how it was determined
* the effectiveness (the more miles saved the better), and how the miles-saved were calculated.
* the period of time for which rides were analyzed (the longer the better, i.e. a year is better than 6 months)
* the values of y (the delay) that were considered (the more, the better)
* teamwork retrospective analysis: how the collaboration was organized, which elements of the organization worked well, and which didn’t.

Extra credit: consider the % of trips that are willing to share.

The intermediate presentation should present partial results along the above lines, and restrict queries to trips originating and ending in Mahattan.

Within each project, each team member must have assigned individual responsibilities. Examples: data cleaning, evaluation of the performance of different queries, code to determine if 2 trips can be combined, writing section x of the final report, output validation and verification. These responsibilities must be identified and documented in the project report. Each student must also document individual contributions to the project (edits and comments to other team-members’ products and code, bug discovery, report edits, etc.).

**References:**

[a] <https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>

**Team size**

There will be four students in each team. At most one graduate student per team.

Appendix:

Algorithm to determine if trip 1 with origin o1(x1,y1) and destination d1(z1,w1) is mergeable with trip 2 with origin o2(x2,y2) and destination d2(z2,w2).

1. Determine the average speed of the taxi that serviced Trip1, assuming that it traveled along the straight line o1\_\_\_\_\_\_d1 (this line has length called distance(T1)); do so by considering the start-time and end-time of the trip. Call this average speed s1.
2. Determine the average speed of the taxi that serviced Trip2, assuming that it traveled along the straight line o2\_\_\_\_\_\_d2 (this line has length called distance(T2)); do so by considering the start-time and end-time of the trip. Call this average speed s2.
3. Assume that the taxi X that will service the merged trip (T1,T2) has a speed s3 which is the average of s1 and s2.
4. Consider the following sequences of straight line segments:

o1\_\_\_o2\_\_\_\_d1\_\_\_\_\_\_d2

o1\_\_\_o2\_\_\_\_d2\_\_\_\_\_\_d1

o2\_\_\_o1\_\_\_\_d1\_\_\_\_\_\_d2

o2\_\_\_o1\_\_\_\_d2\_\_\_\_\_\_d1

If one of the sequences has a total distance D that is lower than [distance(T1)+distance(T2)], and can be satisfied with the given delay[[1]](#footnote-1), then T1 and T2 are mergeable with a savings S: S = [distance(T1)+distance(T2)] - D. If more than one sequence can be satisfied with the given delay, then pick the one with the maximum savings. Otherwise T1 and T2 are not mergeable.

1. Consider for example the sequence o2\_\_\_o1\_\_\_\_d1\_\_\_\_\_\_d2 with delay 5 minutes. The sequence can be satisfied with the given delay if: a taxi with speed s3 can start at o2 at the time Trip2 started, arrive at o1 no more than 5 minutes before Trip1 actually started (in other words, due to ridesharing Trip1 starts no earlier than 5 minutes before its actual start time in the database) then arrive at d1 no later than 5 mins after Trip1 ended, and then arrive at d2 no later than 5 mins after Trip2 ended, then the sequence can be satisfied with the given delay. [↑](#footnote-ref-1)