CS581 Ride Sharing Project
Final Report

Siham Husein[1], Kostas Solomos[1], Michael Uhlarik[1] and Hoangminh Huynhnhuyen[1]

[1]Department of Computer Science, University of Illinois at Chicago

May 3, 2020

Contents

1 Introduction

Handling ride share requests in a timely and efficient manner is a crucial aspect of companies like Uber [1] and Lyft. A critical component of handling ride share requests is the ability to merge trips based on nearby pickup or drop-off locations, rider preferences and delay tolerances. This project presents one possible solution to this challenge.

The following trial utilizes data from the records of the New York City Taxi and Limousine Commision and isolates trips going to or from LaGuardia Airport. While certain assumptions and restrictions are conceded, this implementation presents significant and timely savings and could reasonably form the basis of a more complex merging mechanism in a real life environment.

2 Assumptions & Restrictions

The introduced methodology involves several assumptions and restrictions which allows a focus on the data handling and algorithmic components of the problem. First, as previously mentioned, the solution strictly handles trips going to or from the LaGuardia Airport. To apply the algorithm in our data, we set the following:

1. All pick ups and drop offs within a one mile radius of the airports coordinates will be considered airport trips.

2. Trips that have an exceptionally long distance or have a distance of zero, are removed from the dataset.

3. We filter out trips that have more than one passenger. On the dimension of total passengers, every trip can be merged with every other trip.

4. The passenger is not expected to walk any significant distance to reach their pick up or drop off location.

5. The social preferences of the passengers (non-smoking, similar interests, etc) are also not considered.

6. Traffic is not considered- the travel time is assumed to be constant.

3 Algorithms

3.1 Overview

The algorithm that is used to merge trips is the foundation of this solution. Once the assumptions and restrictions are in place and the data is cleaned and processed the merging process begins. First, the delay tolerance and pool size are specified. The delay tolerances represents that maximum addition to the total trip time of the passenger that is acceptable in a shared ride. The pool size is the time window during which the trips will be merged. For example, if the pool size is set to 5 minutes, then the only trips that can possibly be merged are trips that fall within the given 5 minute window.

Distance Formula The distance between two drop off or pick up locations is calculated by the following by the Euclidean distance i using the coordinates given in the dataset. Considering now a set of pairs `(p,q)` The mathematical equation that computes their distance is: $d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$ The distance is multiplied by a constant which represents the ratio between the Euclidean distance and the trip distance value from the dataset. This result is used as an approximation of real orld travel distance. The speed is assumed to be the average of the two trip speeds.

Shareable Trips. Two trips are considered to be shareable if they meet the following criteria: If the trip time from the source location to passenger As location plus the distance from passenger A's location to passenger B's location is less than the trip time from the source to passenger B's location plus passenger Bs tolerable delay, then those trips are shareable. This formula holds whether passenger A is dropped off first or passenger B is dropped off first, as well as if this is going to or coming from LaGuardia.

3.2 Algorithms in Depth

Here we present the pseudocode and details of our proposed algorithms. To be consistent we set the following variables :

1. LGA is denoted as H.

2. The average speed of each trip is calculated in advance: Speed = $distance \div (dropoff\_time - pickup\_time)$.

3. Delay ranges between 5,10,15,20 % of total time.

Algorithm 1: Algorithm to combine trips starting from LGA

```
Result: Average of distance saving, Trip saving and computation times.
Trips_Array,Distance_Array,Computation_Array=New_Array(); //init empty arrays for storing
Pools_Set= Data_Slice(window) // divide all trips into windows of x minutes
for pool in Pools_Set do
    start_time=current_time();
    G = New_Graph();
    G=add_trips(all);//Add all trips into nodes of graph G
    for each (tripA,tripB) in pool do
        / /For each pair of trips in this pool
        dist(A_B)= Euclidean(A,B); //Calculate the travel distance of 2 drop-off points
        dist(H_A)= Euclidean(H,A) ;// Calculate distance between LGA(H) and A
        dist(H_B)= Euclidean(H,B) ;// Calculate distance between LGA(H) and B
        //check for savings formula.
        if dist(A_B) ≤ dist(H_A) OR dist(A_B) ≤ dist(H_B) then
            delay_A =t(HA)*X% ;// calculate delay tolerance in time unit for each trip.
            delay_B =t(HB) *X% ;
            V (AB)= Avg(V (HA) + V (HB));// Calculate average speed to go from A to B
            t(AB) = d(AB)  V (AB) ; //Calculate time to travel between A and B

            Check if 2 trips are shareable
            if t(HA) + t(AB) ≤ t(HB) + delay_B OR t(HB) + t(AB) ≤t(HA) + delay_A then
                saving=MAX (dist(H_B)- dist(A_B),dist(H_A)- dist(A_B)) ;
                G.add_edge(H,A,B,saving) ; //add an edge with weight saving to graph G
            end
        end
        Max_Set= Max_Matching(G) ;// maximum weight matching algorithm on graph G to get
          matching set M
        end_time=current_time(); total_saving,total_distance =compute(Max_Set) ;// Calculate the
          total distance savings and alll trips of set .
        computation_time=end_time-start_time ; Trips_Array,Distance_Array,Computation_Array=
          total_saving, total_distance, computation_time; //store in arrays;
    end
    Calculate the average of distance saving, trip saving and
     compute time arrays
end
```

Algorithm 2: Algorithm to combine trips ending to LGA

```
Result: Average of distance saving, Trip saving and computation times.
Trips_Array,Distance_Array,Computation_Array=New_Array(); //init empty arrays for storing
Pools_Set= Data_Slice(window) // divide all trips into windows of x minutes
for pool in Pools_Set do
    start_time=current_time();
    G = New_Graph();
    G=add_trips(all);//Add all trips into nodes of graph G
    for each (tripA,tripB) in pool do
        / /For each pair of trips in this pool
        dist(A_B)= Euclidean(A,B); //Calculate the travel distance of 2 drop-off points
        dist(H_A)= Euclidean(H,A) ;// Calculate distance between LGA(H) and A
        dist(H_B)= Euclidean(H,B) ;// Calculate distance between LGA(H) and B
        //check for savings formula.
        if dist(A_B) ≤ dist(H_A) OR dist(A_B) ≤ dist(H_B) then
            delay_A =t(HA)*X% ;// calculate delay tolerance in time unit for each trip.
            delay_B =t(HB) *X% ;
            V (AB)= Avg(V (HA) + V (HB));// Calculate average speed to go from A to B
            t(AB) = d(AB)  V (AB) ; //Calculate time to travel between A and B
            //Check which trip started before the other and reverse the
             order
            delta = pickup_timeB - pickup_timeA

            Check if 2 trips are shareable
            if t(AB) ≤ delta + delay_B OR t(HB) + t(AB) ≤t(HA) + delay_A then
                saving= dist(H_B)- dist(A_B) ;
                G.add_edge(H,A,B,saving) ; //add an edge with weight saving to graph G
            end
        end
        Max_Set= Max_Matching(G) ;// maximum weight matching algorithm on graph G to get
         matching set M
        end_time=current_time(); total_saving,total_distance =compute(Max_Set) ;// Calculate the
         total distance savings and alll trips of set .
        computation_time=end_time-start_time ; Trips_Array,Distance_Array,Computation_Array=
         total_saving, total_distance, computation_time; //store in arrays;
    end
    Calculate the average of distance saving, trip saving and
     compute time arrays
end
```

4 Experiments

References

[1] UBER. Explore the Uber Platform — Uber United States. https://www.uber.com/, 2020. Online; accessed 2 May 2020.