Drive With Friends

Chris Spalding, Kyle Dybdal & Davis Shurbert

Summary

Our project will be a mobile application for both Android and iOS, the core functionality of which is to track the user as they drive their daily commute, and match them up with users who have a similar commute so that they can carpool together.

The Algorithm

The core of our project is the algorithm to match users with similar routes and schedules. We will maintain a dictionary that keeps track of users, who they are matched with, and the routes that they take. This data structure will be used to find groups of users with similar routes. In order to create and maintain this structure, we need to find an efficient way of storing and comparing routes.

The most naive approach to this problem would be to store three data points for each route, start location, end location, and departure time. Using this information, we would search for clusters of data points in three dimensional space. However, the problem with this approach is that we lose the ability to match potential partners where one route is a subset of another. As a result, we need to store more information about the route for each user. While this approach is not in line with our vision for the project, we will implement it as both a control to compare against our more robust algorithm, and as part of our backup plan in case our project turns out to be more ambitious than we have planned.

The specifics of how we conduct this search will depend highly on the structure of our route data. For example, we have several options for how to handle the departure time variable. Option one would be to store this information as part of the route and consequently treat it as a variable in our cluster process. This is a viable option as we could accommodate more or less time sensitive schedules as a parameter in our search, meaning someone who needs to arrive at work at a precise time could be matched with someone who can arrive an hour early. Alternatively, we could break up the day into 30 or 60 minute chunks, only comparing two users if they depart for work in the same time period. This approach simplifies computation, but we may lose potential matches and make unwanted matches if a person cannot leave for work 20 minutes earlier than usual. This is just one example of the complications and choices involved in the organization of our data and behavior of our algorithm.

Once we decide how to organize our data, we have a number of factors to consider when creating/choosing a clustering algorithm. One major area of concern is precision versus

computation time. We envision our app to have little time constraints. That is, a new user doesn't need to be matched with a group right away, rather hours or days later. While the ability to have our computations take longer should allow us to be more precise in our matching, the sheer size and dimension of our data set will still require us to be quite efficient in our calculations.

There are multiple options for the design of our algorithm. When we come across difficult decisions such as explained in the above examples, our plan is to create rough implementations of each option. For example, we would implement both approaches to our time variable and compare the efficiency and outputs of each choice.

Challenges

One of the largest challenges of this project will be creating an efficient and reasonable matching algorithm. The algorithm will need to prioritize who to match based on many factors, including the time of day that each user drives a certain route as well as the similarity between routes among many users. Another issue will be figuring out which users' routes to search through in order to find potential matches. If the application gains a large following, it will quickly become implausible to search through all routes to find a match for a single user, especially if there are users who live very far away from one another.

Along with this, we will need to create a database that stores all users' routes driven while using the app, which clearly will not be a trivial task. No one in the group has taken databases, so this task will be almost entirely new material for all of us. The information will be stored in a central server that will do the bulk of the computational work for the application. Furthermore, we will ideally use some concepts of networking in order to allow users to message each other directly through the application after they have matched.

Of course we will need a sleek looking application in order to entice mobile users to download it in the first place. We plan on developing for both Android and iOS systems. None of us has done any formal development with either of these systems. However, Swift and Java, the main development languages for iOS and Android, respectively, both have straightforward syntax and the concepts of mobile development should follow relatively easily from what we have all learned during our computer science classes at the school.

Aside from development problems, we will need to figure out how to divide work equally and efficiently throughout the semester in order to chip away at the project consistently. Early on in the project while two of us are creating the front ends of the mobile app, the third member of the group will be able to do research on how to create an interface between mobile applications and the Google Maps API. After we have this information, one person can start to tackle the issue of the phone tracking routes, another can start to set up a basic database, and the third member can start to lay the ground work for the algorithm. After creating the foundation for these three aspects of the project, since the algorithm is such an integral part of the project, we feel that it is important for the entire group to work together to finish creating the algorithm. Thus, we will spend at least two weeks as a group working together on this one goal. After implementing the algorithm, we will work on connecting the application to the server and finalizing how users can communicate after they have matched one another. These aspects of the project, as well as making any tweaks to

the interface will take up the final weeks of the project. We all feel that this division of work gives each member of the team a reasonable amount of work to do at all times throughout the project, but also allows everyone to be involved with each piece of the project and gain an understanding of each aspect of the problem that we will be solving.

Necessary Skills

We will be working deeply with algorithms to figure out the best way to match people with one another. We will be working specifically with graphs and graph traversal algorithms, a topic covered deeply in 361. This will encompass analyzing time and space complexity of algorithms that already exist as well as considering the efficiency of any algorithms that we create ourselves. We will also continue to use efficient coding practices; for example using the most efficient data structures for a given task and thinking critically about how to implement each step of the project.

The project will also involve app development skills, which will vary depending on the OS we decided to develop for (Swift/XCode for iOS, Java/OpenGL for Android). These skills will be mostly new to everyone in the group, so while app development languages and programs are not very hard to figure out, there may be a steep learning curve that will require a significant initial investment of time and thought. That being said, reading tutorials about Swift, XCode, and Android development can be at least partially completed over winter break, so that by the time we return to school we are almost immediately ready to jump in to the project.

We will need to use a database in order to store users' route and profile information. None of us has taken this class so it will be an entirely novel challenge. That being said, we should be able to consult Professor David Chiu as well as many online resources in order to gain the necessary background to create a database that can store many users' information.

Time line

Week	Person 1	Person 2	Person 3
Jan 20	Designing the app	App Design	Learning Google API
Jan 26	Front End (iOS)	Front End (Android)	Porting Applications
Feb 2	Database Groundwork	Algorithm Groundwork	How to Track Routes
Feb 9	Algorithm (Database)	Algorithm	Porting Applications
Feb 16	Algorithm	Algorithm (testing)	Algorithm
Feb 23	Algorithm	Algorithm	Algorithm
Mar 2	Networking	Algorithm (testing)	App Development
Mar 9	Networking	Algorithm	App Development
Mar 16	Networking	Algorithm (testing)	Connecting Modules
Mar 23	Break	Break	Break
Mar 30	App functionality (iOS)	App Functionality (Android)	Server Work
Apr 6	UI Tweaks	Connecting Modules	Testing
Apr 13	Tweaks	Connecting Modules	Tweaks
Apr 20	Beta	Beta	Beta
Apr 27	Testing	Testing	Testing
May 4	Final Tweaks and Tests	Final Tweaks and Tests	Final Tweaks and Tests

Backup Plan

A huge benefit to this project is the ability to simplify our product without losing core functionality or significant usability. If we begin falling behind in our time line, we can simply cut down the complexity of our data, pushing more and more responsibility onto the user.

For example, we may decide to eliminate all or the majority of our consideration of time. In this scenario, the app would only be responsible for matching users based on physical route information, leaving the task of scheduling to the users. Another example of such simplification would be to ignore the subset route problem, in which one user's route is a subset of another. However, as data structure organization is so crucial to our project and this must be decided early in the project, we will have to choose a design strategy and algorithm which works easily with this simplification.

As a last resort, if we do not have enough time to finish our project, we will still have a significant amount of information to write about and present to the public. We will have spent the entire semester creating and analyzing graph algorithms, and thus this should provide ample opportunity for presentation. We may be able to leave our code in a Git repository or share it with others if the research is worth continuing.