Cleaning Parks for a Safer Future

By Connor Mattes and Zachary Sorenson

Motivation and Abstract

With spring coming up, more and more people will be wanting to get outside. We can't think of a better place to spend that time at than our local parks. Having cleaner parks will give individuals safe place to engage in healthy activities as opposed to spending time on less desirable endeavors. Our project idea is simple: get people to parks in the best possible way so they clean them more efficiently.

Parks are a place for communities to come together. In fact, several studies have shown that a clean park can help to reduce the crime of the area in and around the park.[1] Our goal is to develop routes to clean parks in Denver. We will create these routes by solving a famous integer program called "The Traveling Salesmen Problem." We plan to implement Christofides' algorithm to find a route for city workers and volunteers to visit and clean our parks in a quick, efficient manner. An increase in cleaner parks will lead to safer neighborhoods around the parks. Park locations and size will be pulled from data sets provided on the Data to Policy website. Distances are calculated by using the driving distances between individual parks. Methods and processed that are applied were taken from MATH 7594 Integer Programming taught by Steffen Borgwardt.

Methods/Software Used:

- Excel
- Python
- Google Maps
- Sagemath

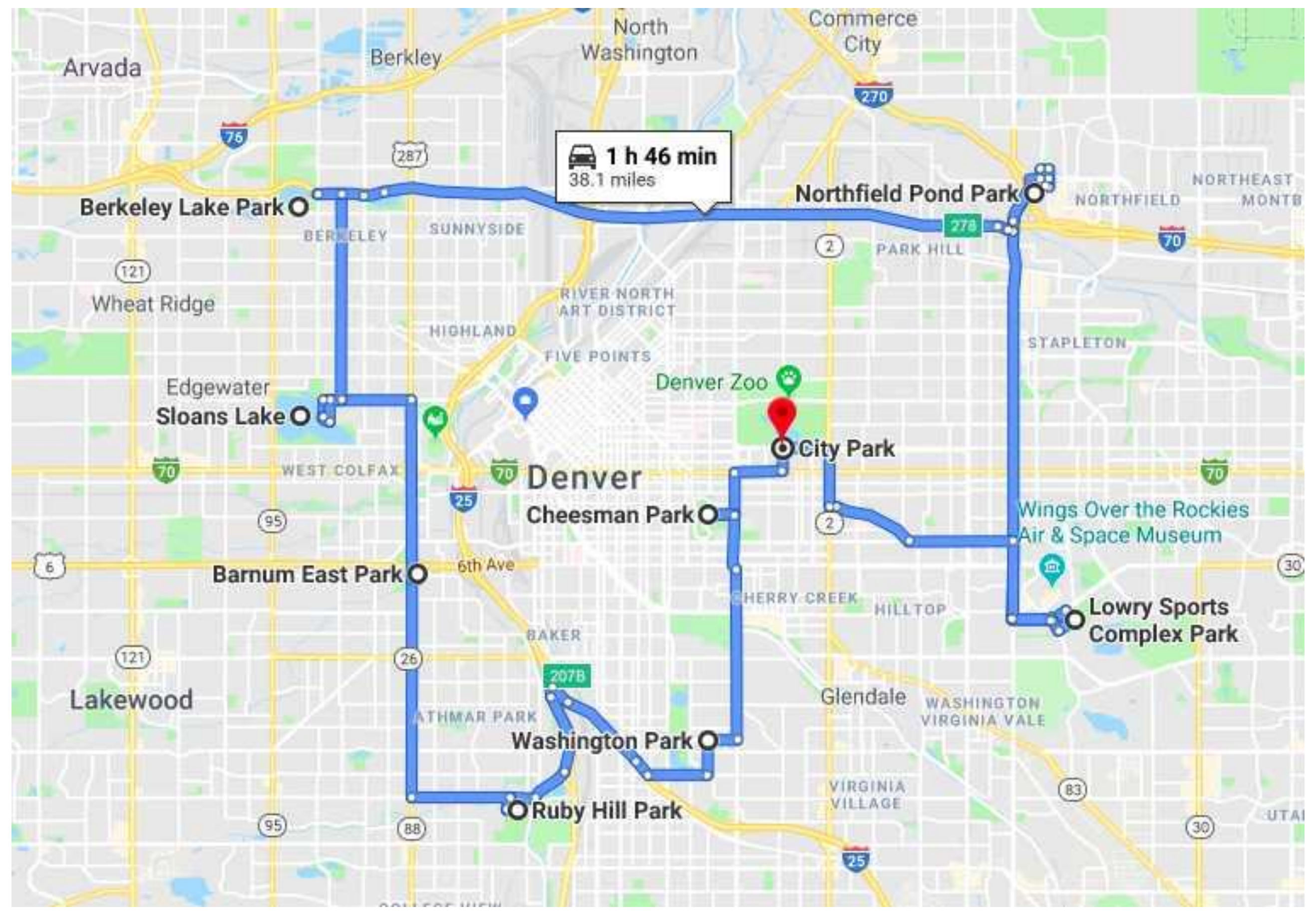
Data:

One of the challenges that our project faced was how to properly gather data. Our entire process revolved around that fact that the distance between parks had to be gathered. The simple strategy of "as the crow flies" measured the geographical distance between two locations. Although there are several pieces of software that can find these values, in reality, people use roads to travel from one location to another.

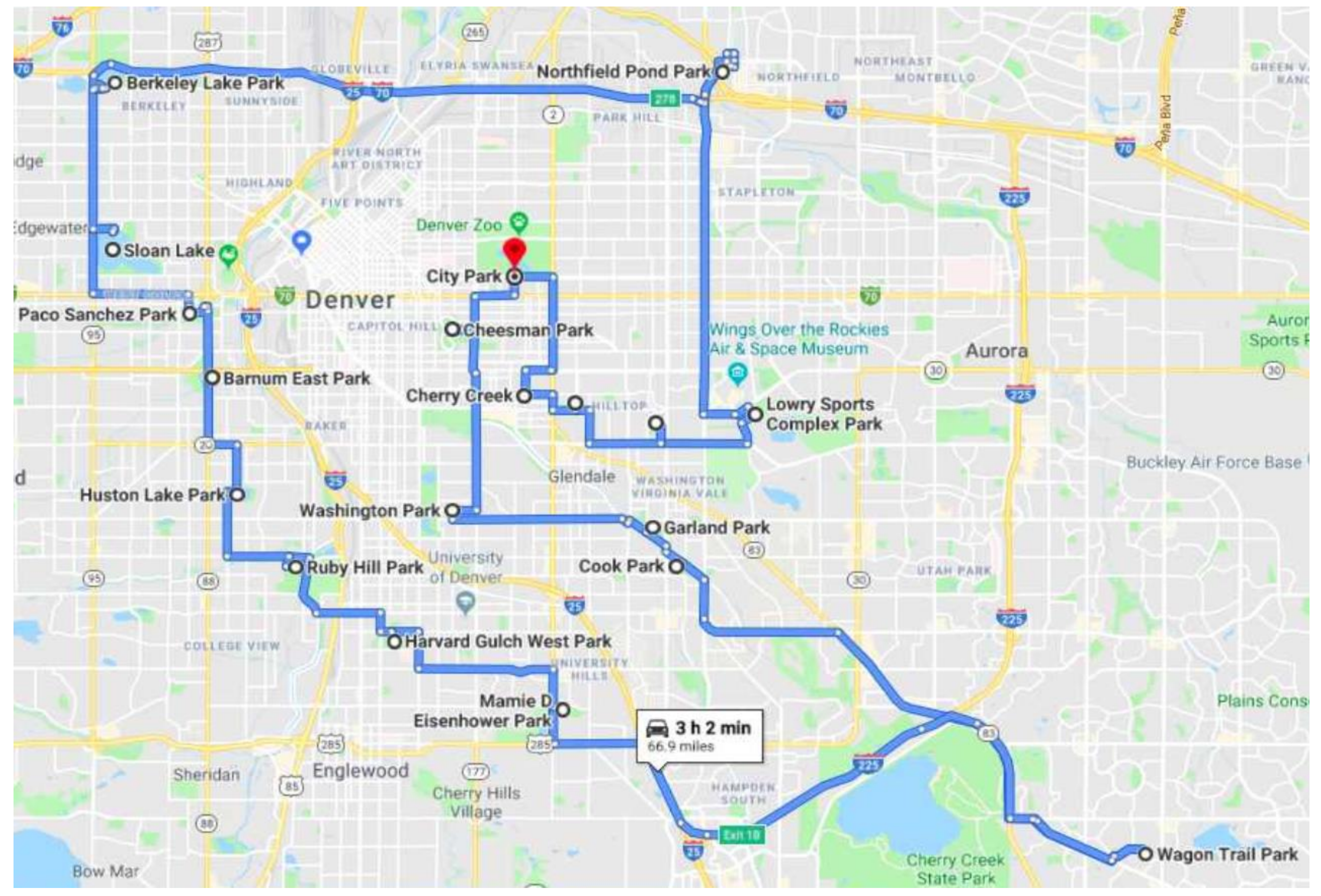
Distances between two locations as it pertains to traveling by car is called the "taxi cab metric'. Since this is more applicable than the previous strategy of "as the crow flies" this is the method that we built around our algorithm. The issue with choosing this method is that expensive software is required to find the distance between multiple locations. Instead, we decided to use a brute force method to find these distances. We decided to use distances rather than driving time because of the variability of Denver traffic.

There is data available on the Data to Policy website that has the locations of various parks in Colorado. The data was sorted to focus on the 50 largest parks in Denver. Then, google maps was used iteratively to find the distance between each park in relation to the other 49 parks. This data was then entered into Python in order to solve the Traveling Salesman Problem.

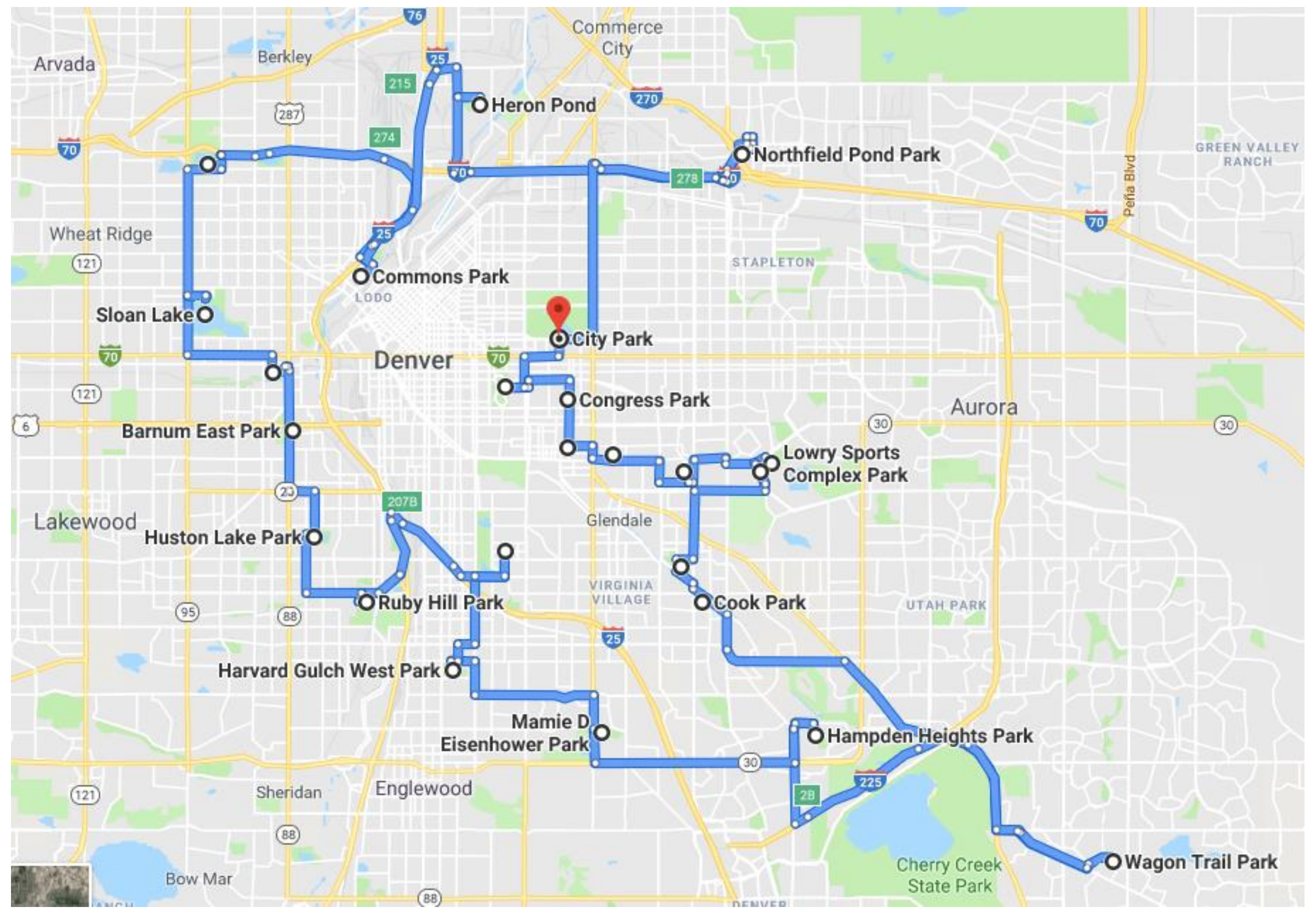
Results



9 Parks



19 Parks



24 Parks

Mathematical Formulation

$$\begin{aligned} \min \quad & \sum_{e\in E} c_e x_e & \\ & \sum_{e\in\delta(i)} x_e = 2 & \text{for } i \in V \\ & \sum_{e\in\delta(S)} x_e \geq 2 & \text{for } \emptyset \subset S \subset V \\ & x_e \in \{0,1\} & \text{for } e \in E. \end{aligned}$$

- c_e – Distance between 2 parks
- δ(i) – All edges leaving vertex i
- Minimize Total Distance
- Enter and exit every park
- One big loop
- Either pick a path between two parks or don't

Implementation

This is a difficult (NP-hard) problem to implement. To solve it we used Sagemath, which has built in commercial mixed integer program solver (CPLEX). This is good for up to approximately 30 parks. It does not however take into account we are in a metric space (which allows for quick and good approximations uses Christofides' algorithm) which could make the program significantly faster. Furthermore it does not allow for many of the extensions listed below such as subtours. Therefore implementation in a more programmable language such as AMPL could be used for some of the extensions below, and as a way to make the program more efficient.

Extensions

- Subtours to allow for multiple crews or cleanup over multiple days (Hard)
- Implementation of Christofides' algorithm to allow for more parks
- Automation of both getting distances from Google Maps, and using distances in algorithm
- Visualization with Google Maps using more than 24 parks
- Look at different cities
- Comparison with tour from coordinates for ease of implementation
- Driving times rather than distances

References

- [1] "Community Clean Up" https://www.ncjrs.gov/pdffiles1/171690.pdf. July 1999
- Integer Programming . Michele Conforti, G´erard Cornu´ejols, Giacomo Zambelli. Springer International Publishing Switzerland 2014