Optimizing Travel Routes in Order to Maximize the Number of Walkers and Minimize the Amount of Walking Done Through a "Walking School Bus" Program

Kristen Bystrom, Matthew Reyers, Brad Smallwood, Barinder Thind

EduHacks: Hackathon

Abstract

We looked to solve an optimization problem that some schools around B.C will inevitably face with the advent of a "walking school bus" program. The problem is how to optimize travel routes while also trying to efficiently cluster children together that will be a part of each of these routes. We implemented a combination of a K-Nearest Neighbors algorithm incorporated in an optimization strategy that used the Google's API to find optimal travel distances. These distances included stops for each student that would end up in each of these clusters. On top of this, we began a website that takes in registrants, provides parenting schedules for each of the clusters, and provides information about the routes to take.

Keywords: Optimization, Geocoding, K-Nearest Neighbors, Education

I. Introduction

Our solution utilizes machine learning and optimization mechanics in order to create a hybrid solution to a real world problem. Both teachers and parents are attempting to create environmental impact, active living, and community engagement through an initiative known as a "Walking School-bus" but are unable to implement it due to logistical difficulties and constraints. Our solution solves this problem by creating an easy to use, accurate, and low maintenance web application that enables parents to sign up, and be given cohort of children in their geographical area who are also on a short distance road route to the local elementary school. We have used K-Mean clustering, a machine learning algorithm, to group together local children and then using optimization methods, we've created the shortest walking distance between the children's homes and their school. With our web application the barriers to implementing this in demand community program have been greatly lowered and local communities can begin to create environmental, healthy, and community impact.

II. Methodology

0.1. Dataset

The data used on our analysis was randomly sampled from a specified geographical fenced area. While our data was randomly samples, the real data would be acquired through a sign-up form on our web application. The data used includes: the family name, address, city, and province, latitude, and longitude. By used the Google Maps application programming interface (API) we were able to acquire the latitudinal and longitudinal data from the address of the houses. Using this information, we randomly assigned houses to be designated as "houses with children that go to an elementary school". This was based on averages about the number of children at a particular elementary school. Further manipulation and cleaning of the data was performed but it remained largely similar to the base structure though out our process.

0.2. K-Mean Clustering

K-Mean is a unsupervised machine learning model, popular for applying to Big Data. The algorithm partitions data into k distinct groups,

$$C_1 \cup C_2 \cup \dots C_k = 1, \dots, n \text{ where } C_k \cap C'_k = \emptyset, \forall k! = k'$$

A good cluster is one which the within-cluster variation is minimized. This minimization is done through squared Euclidean distance.

$$\underset{min}{C_1, ..., C_k} [\sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2]$$

Put simply, this algorithm operates by assigning a number from 1 to K, which is the initial cluster for the set of observations. Then it iterates until the clusters stop changing. This process is demonstrated below in Figure 1.

For the purposes of this project, K-NN was used as an in-between step to get to our final results[1]. In particular, imagine that each dot in figure 1 is an observation around the school (the $\star$). Then, our goal was to cluster student in such a way that all the clusters (which are

[1].e in conjunction with the optimization algorithm outlined in the next section

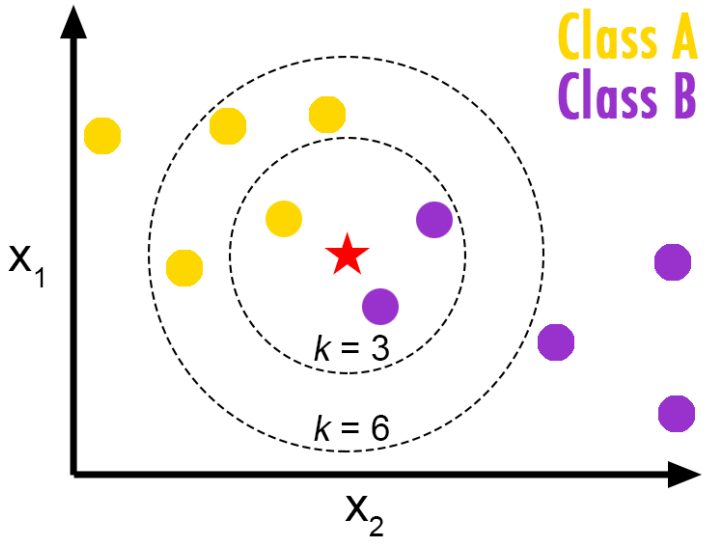

Figure 1: K-NN Example

reasonable with respect to distance) could walk to school without too much time being taken off. This then becomes a minimization problem subject to the two constraints: time (or distance traveled) and optimality of the cluster sizes.

Referring back to figure 1, you can imagine that each of the classes is an optimized cluster produced by our algorithm, then, now that we have these clusters[2], we can provide a pathway (using Google's API) to students to form the "bus" and get to school. This clustering process can be generalized to any school conditioned on enough participation.

0.3. Optimization

The optimization component of our application plays a key role in creating a tool that produces a system that users will want to utilize. By searching for the optimal route to pick of children and deliver them to school in the shortest time and shortest distance within clusters we implemented a permutation optimization algorithm. This algorithm connects to the Google Maps API and allows us to calculate the shortest road distance between nodes (the children's homes) which allows the supervising adults to minimize the time commitment to the program. The problem we tackled is similar to the Traveling Salesman Problem, albeit in a much more scaled down version where we are able to find a optimal solution.

[2]Although, the optimization step is crucial in getting that specific cluster

While we were able to find a solution, we have to note that our solution was also subject to computational constraints. So, while we have local optima, we cannot run our code so that it can obtain the global optima. However, the way we have written out our algorithms, this can easily be remedied as technological advancements are made[3]. These restrictions come in the form of the number of permutations: ${}^{n}P_{k} = \frac{n!}{(n-k)!}$ required to get to the optimal. For example, if the number of observations in a cluster was 35, we would be looking at $2^{31!} > 999,999,999...$ which is just not feasible for a computer today.

We created a distance matrix modeled after a Cartesian product:

$$\mathbf{X} = \begin{bmatrix} House_{1,1} & House_{1,2} & ... & House_{1,n} \\ House_{2,1} & House_{2,2} & ... & House_{2,n} \\ House_{3,1} & House_{3,2} & ... & ... \\ House_{4,1} & House_{4,2} & ... & House_{n,n} \end{bmatrix}$$

Where $House_{1,2}$ and $House_{2,1}$ hold the same value[4]. This value is pulled from Google Maps Directions API and allows us to find the optimal routes between each of the houses as well as the actual distance. Once we had this information, we used the following algorithm[5]:

```
# df = The data set for each school compiled of the registrants
# maxD = The maximum distance that is reasonable to walk
# firstRun = a TRUE/FALSE variable used for convenience

opt_func2 <- function(df, maxD, firstRun){ # Creating function
  if(firstRun == TRUE){ # For convenience
    if (nrow(df) <= 11){ # Computational constraint, k needs to be of
    ↪  clusters of 11 or less
      k = 1
```

[3]Our code is generalized
[4]This holds for all of these "symmetric" relationships
[5]Comments within the code explain every line

```
    for (i in 1:ncol(df)){ # This entire chunk is used to make our data set
     ↪ compatible with Google's API. The location variable below turns
     ↪ into a string all the relevant required variables for the API and
     ↪ we store it into the data set.
    df[, i] <- gsub(" ", df[,i], replacement = "+") }
    location <- NULL
    for(i in 1:nrow(df)){
      location[i] = paste0(df$address[i],"+", df$city[i], ",BC")
    }} else{ # This deals with any clusters of size greater than 11
    k = floor(nrow(df)/12)
    # Change data strings to be readable by optimization functions
    for (i in 1:ncol(df)){
      df[, i] <- gsub(" ", df[,i], replacement = "+") }
    location <- NULL
    for(i in 1:nrow(df)){
      location[i] = paste0(df$address[i],"+", df$city[i], ",BC")
    }
    df2 <- cbind(df, location)
    }
  }
  # So, at this point, we essentially have the right values for k that
   ↪ will allow for the correct maximum sized clusters and we also have
   ↪ data that can easily be inputted into the optimization functions

testSplit = pathSplit(df2, k) # Here, we run our K-NN function on our data
 ↪ set so that it actually clusters the data
busPath = list() # This will be the list with all of our distances
```

```
for (i in 1:length(unique(testSplit$Cluster))){

  temp <- testSplit />/ filter(Cluster == i) # Here, we filter out our data
   ↪  set so we can focus on the results for each of the k values
  if(nrow(temp) <= 1){ # This is error handling code
    return()
  }
  # The code below calls upon the optimization functions
  holder = callAPI(temp[1,7], temp[2,7])
  holder2 = holder[[2]]
  for(j in 3:nrow(df2)){
    holder2 = callAPI(holder2, df2[j,7])[[2]]
  }

  busPath[i] = bestPath(holder2, "Bothwell+Elementary+School+Surrey,BC",
   ↪  "17070 102 Ave") # This is where we actually get the combinations of
   ↪  the best ways for each of the clusters. This is based on google's
   ↪  directions for each of the clusters.
}

# Now, we look at each of the "busPaths" or bestPaths for each cluster and
 ↪  see whether or not they are below our "maxD" threshold. If not, we
 ↪  re-begin the process with k = k + 1 splits. These iterations will
 ↪  allow us to minimize to the distance of walking we require while also
 ↪  having the optimal directions for each of the clusters
for (i in 1:length(unique(testSplit$Cluster))){
  if(busPath[i][[1]] >= maxD){
    k = k + 1
```

```
      testSplit = pathSplit(df, k)
      opt_func(testSplit, maxD, FALSE)
    } else{
      return(as.list(testSplit, busPath, k)) # Return all relevant
       ↪ information
    }

  }
}
```

This algorithm will return us the best paths to take to school for each of the clusters. On top of that, we can easily update the data set for new registrants, run it through the algorithm again, and put the individual into the appropriate cluster.

III. Results & Discussion

Now, we see that in figure 2 below, the image on the left is a plot of our registrants and where their houses are.

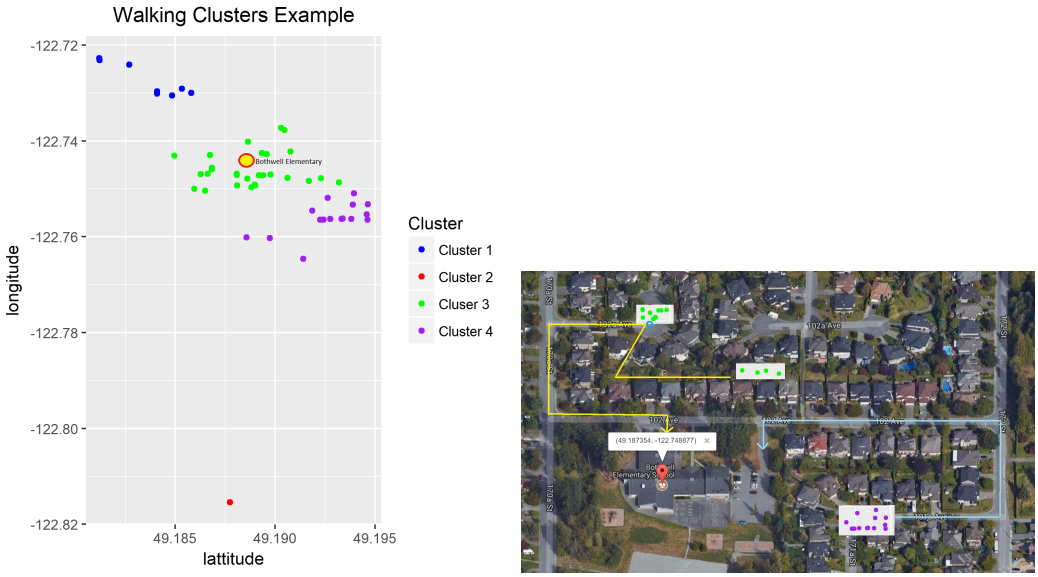

Figure 2: Comparison of Google Maps with K-NN Optimized Cluster Plots

We also highlight the location of the school. The clusters are color coded. The image on the right highlights the location of some of these clusters on the actual map of the location around Bottwell Elementary School. The colored paths indicate the result that we would get from the $Optim_func2$ function above which calls our K-NN and optimization functions.

...

IV. Conclusion

Parents and teachers both desire to have a program like this put into place but logistically, it is difficult and costly for them to do so. Our web application eliminates these costs by being simplistic for users and low cost to maintain, while still finding a solution that will drastically reduce the time commitment required by users. The implementation of our joint optimization and K-Mean clustering algorithm produces a sophisticated solution that generates substantial value for the user. Through utilization of our web application we are able to generate environment impact by reducing reliance on vehicles, enable healthier and more active children through morning exercise, and nurture community bonds through partnerships with the local elementary school and parents of the students. With the logistical and managerial constraints of implementing a program like this removes, teachers can focus more on students, parents can focus on parenting, and kids can enjoy the benefits of living in a more impactful, environmentally friendly, and vibrant community.

V. References

- James, Gareth, et al. An Introduction to Statistical Learning: with Applications in R. Springer, 2017.
- Google's API