Solution by Jeremy Watkins

Optimization of Employee Shuttles Stops

The framing of the problem is simple, given 2191 Employee Addresses and 119 potential bus stops find the 10 most efficient bus stop locations.

This is a first look at the data. The Employee addresses was found to need some cleaning which was done below.

```
import pandas as pd
import re

emp_addresses = pd.read_csv('Employee_Addresses.csv')
bus_stops = pd.read_csv('Potentail_Bust_Stops.csv')

print(emp_addresses.count())

print(bus_stops.count())

print('\n')

for index,row in emp_addresses.iterrows():
    #get steet address
    street=row['address'].split(',',1)[0]

    #if street address is missing number
    if not re.match("[0-9]",street):
        print(row['address'])
        emp_addresses.drop(index,inplace=True)

    #if street address is missing street name
    if not re.search("[a-z]",street):
        print(row['address'])
        emp_addresses.drop(index,inplace=True)

    #if address is not in San Francisco or Daly city
    if not re.search("San Francisco",row['address']):
         if not re.search("Daly City",row['address']):
            print(row['address'])
            emp_addresses.drop(index,inplace=True)

    #if address is not in USA
    if not re.search("USA",row['address']):
        print(row['address'])
        emp_addresses.drop(index,inplace=True)

    #if address is not in CA
```

```
    if not re.search("CA",row['address']):
        print(row['address'])
        emp_addresses.drop(index,inplace=True)



emp_addresses.to_csv('Employee_Addresses_Cleaned.csv')
```

Output:

```
address        2191
employee_id    2191
dtype: int64
Street_One     119
Street_Two     119
dtype: int64


80, San Francisco, CA 94105, USA
St. Luke's Hospital Garage, San Francisco, CA 94110, USA
St. Luke's Hospital Garage, San Francisco, CA 94110, USA
San Francisco War Memorial and Performing Arts Center, 301 Van Ness Ave, San Francisco, CA
94102, USA
Essex St, San Francisco, CA 94105, USA
Market Square, 1355 Market St, San Francisco, CA 94103, USA
80, San Francisco, CA 94105, USA
Market Square, 1355 Market St, San Francisco, CA 94103, USA
Alemany Blvd, San Francisco, CA 94112, USA
Twin Peaks Blvd, San Francisco, CA 94114, USA
St Mary's Park Footbridge, San Francisco, CA, USA
Trainor St, San Francisco, CA 94103, USA
San Jose Avenue, San Francisco, CA 94131, USA
101, San Francisco, CA 94124, USA
80, San Francisco, CA 94105, USA
Market Square, 1355 Market St, San Francisco, CA 94103, USA
San Francisco War Memorial and Performing Arts Center, 301 Van Ness Ave, San Francisco, CA
94102, USA
Treat St, San Francisco, CA 94103, USA
101, San Francisco, CA 94103, USA
101, San Francisco, CA 94103, USA
Twin Peaks Blvd, San Francisco, CA 94114, USA
 B Mission St, San Francisco, CA 94112, USA
Polk St, San Francisco, CA 94102, USA
Plum St, San Francisco, CA 94103, USA
Earl Warren Building, 350 McAllister St, San Francisco, CA 94102, USA
Earl Warren Building, 350 McAllister St, San Francisco, CA 94102, USA
```

Solution by Jeremy Watkins

Most of these places do not look like places where anyone would live so they were removed from the data set. The addresses with no street number were also removed because it would not be helpful with calculations for the bus stops.

Google API was used to get the geocodes from the addresses and cross streets.

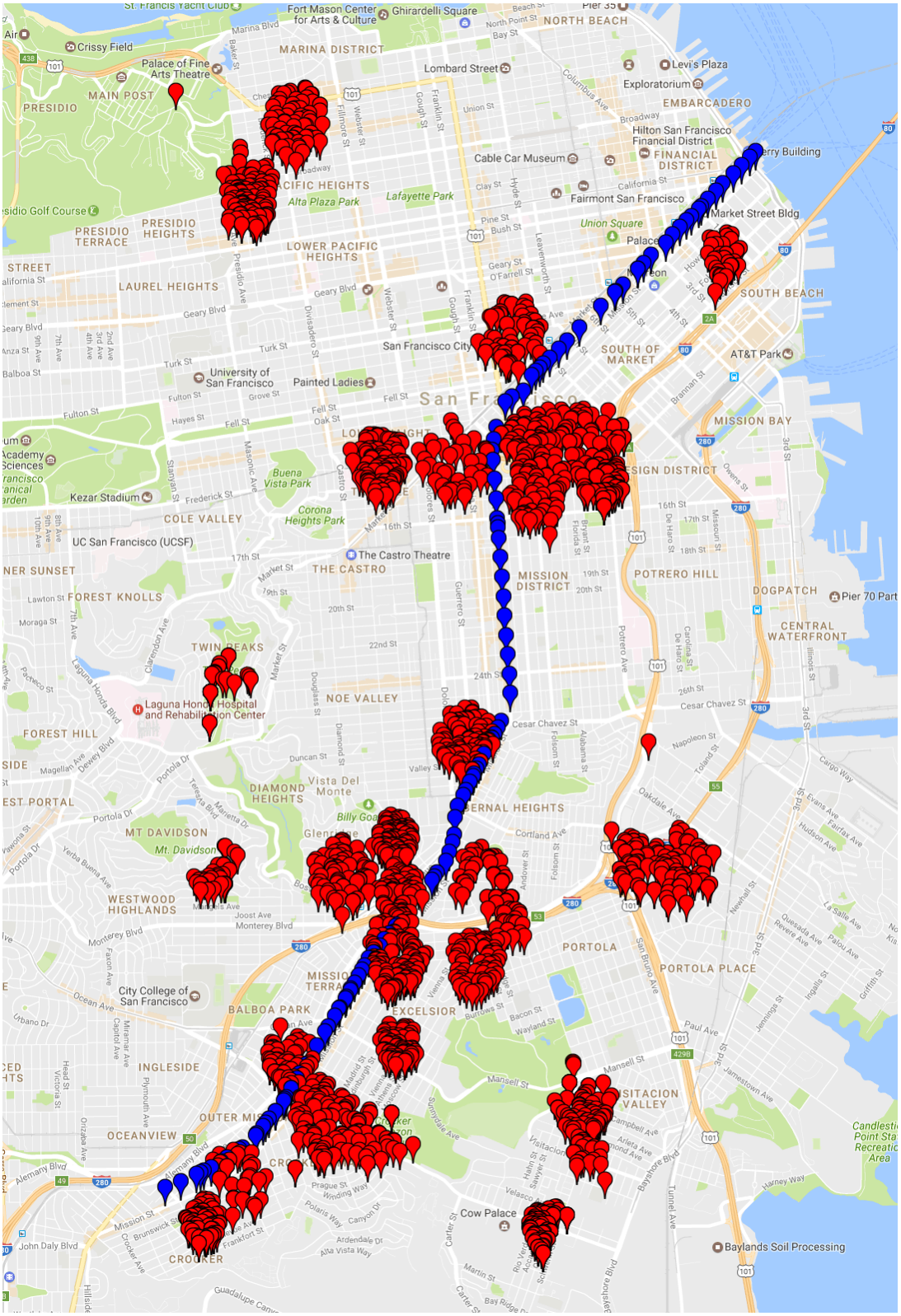
```
import pandas as pd
import requests
import json
import time

#read csv file into DataFrame
emp_addresses = pd.read_csv('Employee_Addresses.csv')

#function to grab geocode from google api
def geocode(location_string):
        geocode = requests.get(
            'http://maps.googleapis.com/maps/api/geocode/json?address="%s"' % location_string)
        geocode = json.loads(geocode.text)
        time.sleep(1)
        latlng_dict = geocode['results'][0]['geometry']['location']
        return latlng_dict['lat'], latlng_dict['lng']

#create geocode list
location = []
for index,row in emp_addresses.iterrows():
        location.append(geocode(row['address']))

#create lattitude and longitude list from location
lattitude=[]
longitude=[]
for each1,each2 in location:
    lattitude.append(each1)
    longitude.append(each2)

#add lattitude and longitude to DataFrame
emp_addresses[‘lattitude’]=lattitude
emp_addresses[‘longitude’]=longitude
emp_addresses.to_csv(“employees_with_geocode.csv”,index=False)
```

Solution by Jeremy Watkins

The gmplot library was used to make an interactive plot of the data.

```
import pandas as pd
import gmplot

#read data into DataFrames
emp_addresses = pd.read_csv('employees_with_geocode.csv')
bus_stops = pd.read_csv('bus_stops_with_geocode.csv')


#set  starting view point of map file
gmap = gmplot.GoogleMapPlotter(37.75, -122.427325, 13)

#plot employee addresses
for index,row in emp_addresses.iterrows():
	gmap.marker(row['lattitudes'],row['longitudes'],title=row['address']+' employee_id
'+str(row['employee_id']))

#plot potential bus stops
for index,row in bus_stops.iterrows():
	gmap.marker(row['lattitudes'],row['longitudes'],c='blue',title=str(row['lattitudes'])+'
'+str(row['longitudes']))

#create map
gmap.draw("Overview.html")
```

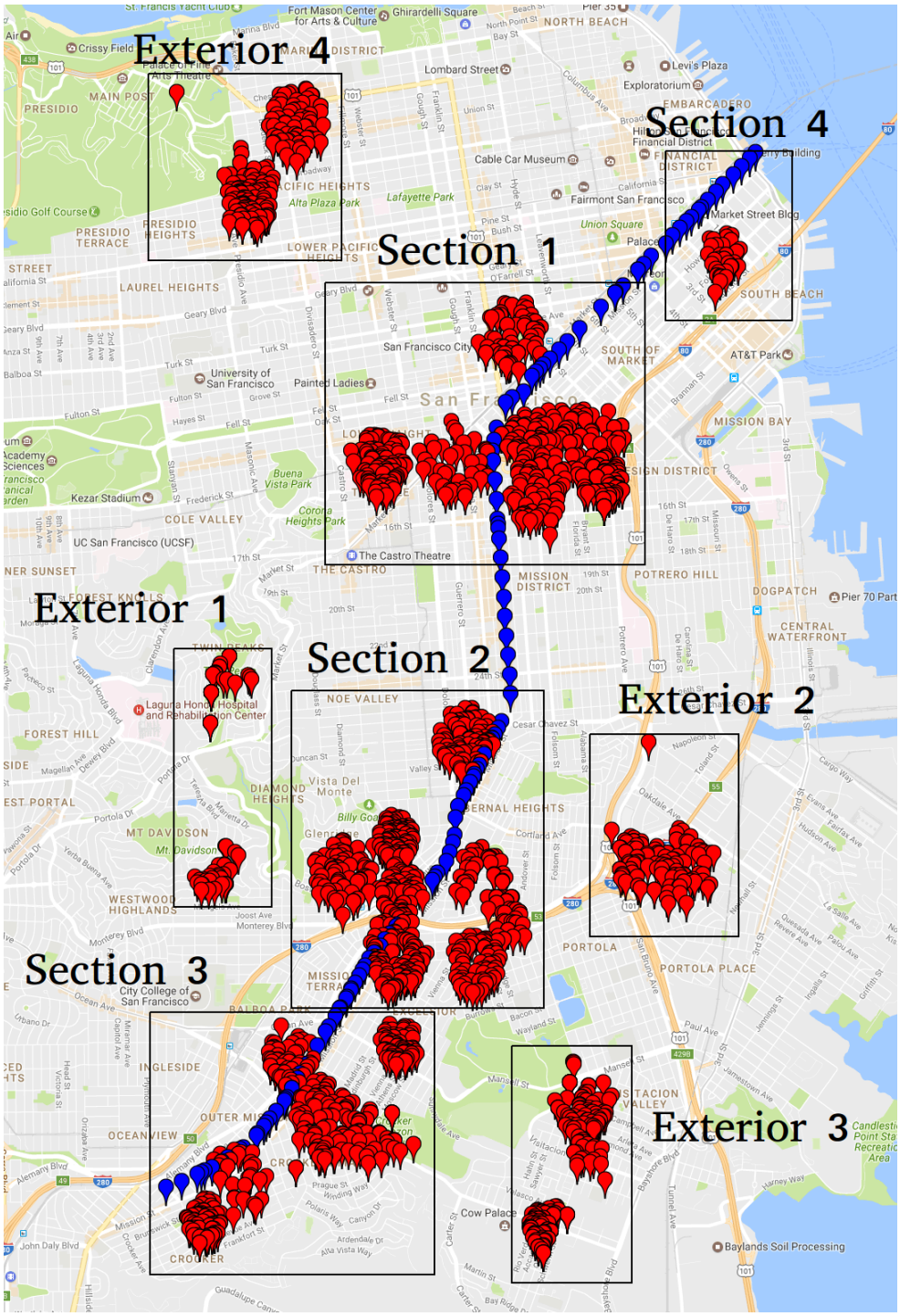
Solution by Jeremy Watkins



In the .html files employee ID and address can be viewed by hovering the mouse over the marker. Employee addresses are the red markers and potential bus stops are blue markers.

Solution by Jeremy Watkins

Simply computing the combination of bus stops that yields the lowest overall distance from each employee address is rather intensive. There are 119 choose 10 combinations which comes out to about 1.06e+14. To find the most efficient bus stops the map was split into manageable sections where it was possible to calculate the best combination of stops to yield the lowest overall distance from each of the employees to the nearest stop. All employees that were greater than a 25 minute walk from Mission Street to be able to walk were removed from the calculation as I figured that further that this was unreasonable to walk. These employees were assigned to "Exterior" areas that will be analyzed later as they will need a separate means of transportation to Mission street. For this, public bus routes can be utilized, which have somewhat flexible drop off locations near Mission Street.



Solution by Jeremy Watkins

Calculations

The Euclidean distance was used to estimate the efficiency of bus stop locations. This provided a quick and simple algorithm to apply to each section. The distance was calculated from each employee address to the nearest bus stop and each distance for each employee address was added together to get the total distance. It was calculated exhaustively for every combination of bus stops within each section and then the combination that yielded the lowest total distance was selected. Different numbers of bus stops in each section was used to determine what number of bus stops in each section would yield the optimal overall results. The code for calculating the distances for each section can be viewed in sector[1-4].py.

Section 1

number of bus stops: 2
CPU run time ~0.5 seconds
total distance: 4.443288894436137

number of bus stops: 3
CPU run time: ~5 seconds
total distance: 4.32550657196003

number of bus stops: 4
CPU run time ~33 seconds
total distance: 4.24510185476948

Section 2

number of bus stops: 2
CPU run time: ~2 seconds
total distance: 3.4775575720267273

number of bus stops: 3
CPU run time: ~35 seconds
total distance: 2.9888868253362606

number of bus stops: 4
CPU run time: ~7 min 30 seconds
total distance: 2.8638046862278705

Section 3

number of bus stops:2
CPU run time ~0.5 seconds
total distance: 2.123775614486866

number of bus stops:3

CPU run time ~8 seconds
total distance: 1.868219917112954

number of bus stops:4
CPU run time: ~1min 10 seconds
total distance: 1.8092163367658658

Section 4

number of bus stops: 1
CPU run time ~0.001 seconds
total distance: 0.19379909084842276

number of bus stops: 2
CPU run time ~0.008 seconds
total distance: 0.19036118812729588

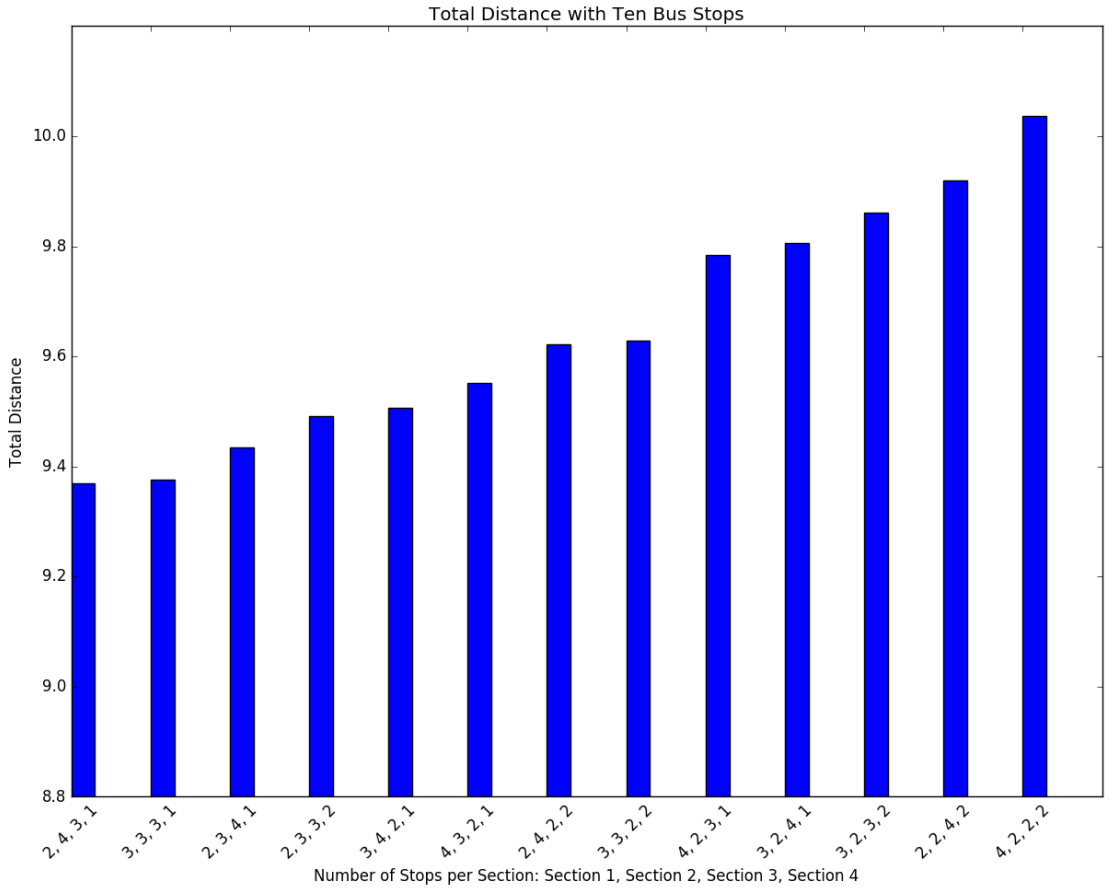
I created a bar graph to summarize my findings:

```
import itertools
import matplotlib.pyplot as plt

#input number of stops and total distance per section
Section_1=[(2,4.443288894436137),
        (3,4.32550657196003),
        (4,4.24510185476948)]
Section_2=[(2,3.4775575720267273),
        (3,2.9888868253362606),
        (4,2.8638046862278705)]
Section_3=[(2,2.123775614486866),
        (3,1.868219917112954),
        (4,1.8092163367658658)]
Section_4=[(1,0.19379909084842276),
        (2,0.19036118812729588),]

#get all combinations with 10 total stops
combination_10_stops=[]
for combination in itertools.product(Section_1,Section_2,Section_3,Section_4):
    total_stops=0
    for num_stops,distance in combination:
        total_stops+=num_stops
    if total_stops==10:
        combination_10_stops.append(combination)
```

Solution by Jeremy Watkins

```
#create tuple
tuple_comb=[]
for sec1,sec2,sec3,sec4 in combination_10_stops:
    x=(str(sec1[0]) + ', ' + str(sec2[0])+', '+ str(sec3[0])+', ' + str(sec4[0]))
    y=sec1[1]+sec2[1]+sec3[1]+sec4[1]
    tuple_temp = (x,y)
    tuple_comb.append(tuple_temp)

#sort tuple based on total distance
tuple_comb=sorted(tuple_comb,key=lambda x: x[1])

#create x and y for plot
y=[]
x=[]
for sections,total_distance in tuple_comb:
    y.append(total_distance)
    x.append(sections)

#create plot
fig = plt.figure()
ax = fig.add_subplot(111)
ax.set_xticks(range(len(y)), minor=False)
ax.set_xticklabels(x, rotation=45)
ax.axis([0,13,8.8,10.2])
ax.bar(range(len(y)), y, width=.3, color="blue")
ax.set_ylabel('Total Distance')
ax.set_xlabel('Number of Stops per Section: Section 1, Section 2, Section 3, Section 4')
ax.set_title('Total Distance with Ten Bus Stops')
plt.show()
```

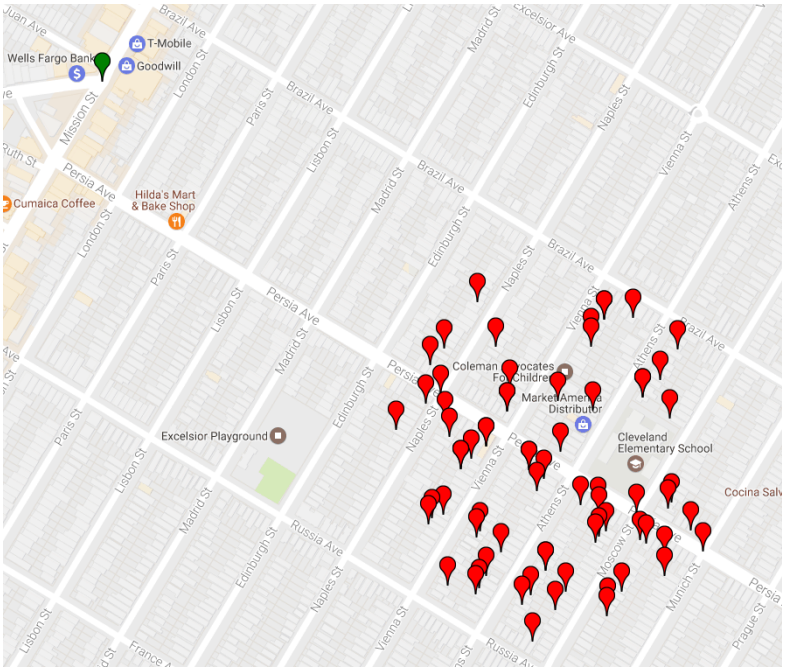
Solution by Jeremy Watkins



Number of stops yielding lowest overall distance:
Section 1: 2
Section 2: 4
Section 3: 3
Section 4: 1

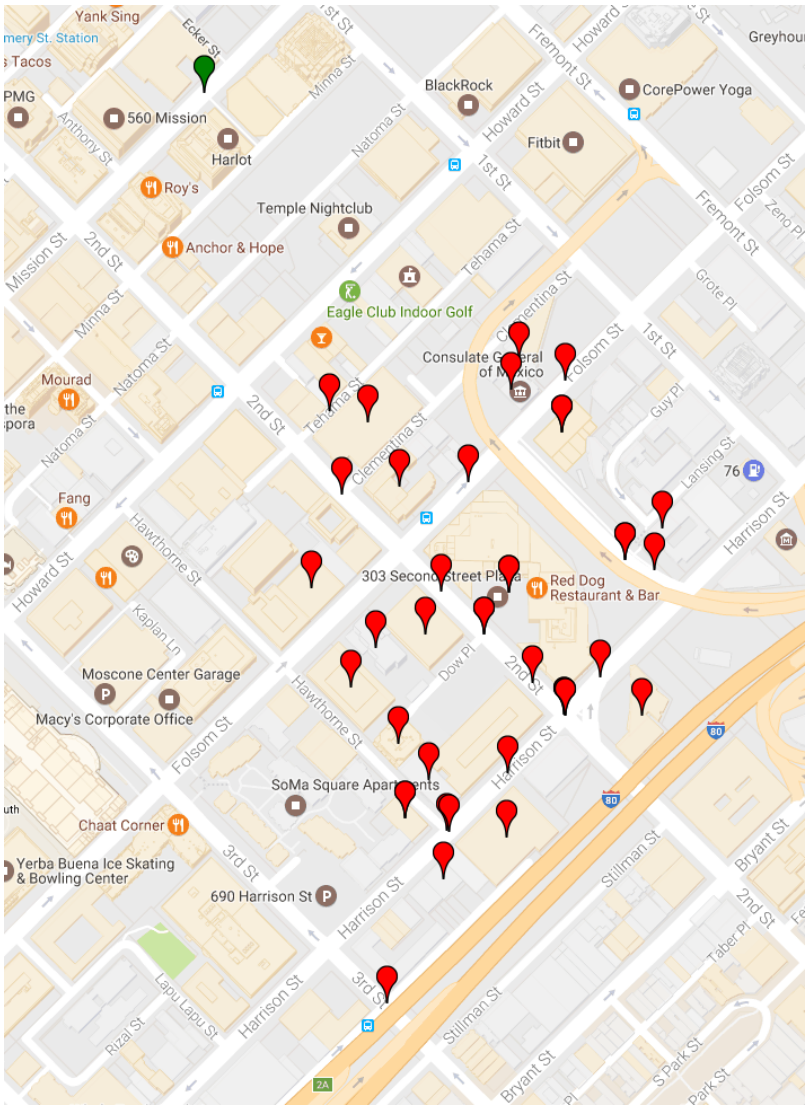
Solution by Jeremy Watkins

From Section 2



Red markers are the employee addresses and green markers are the calculated bus stops.

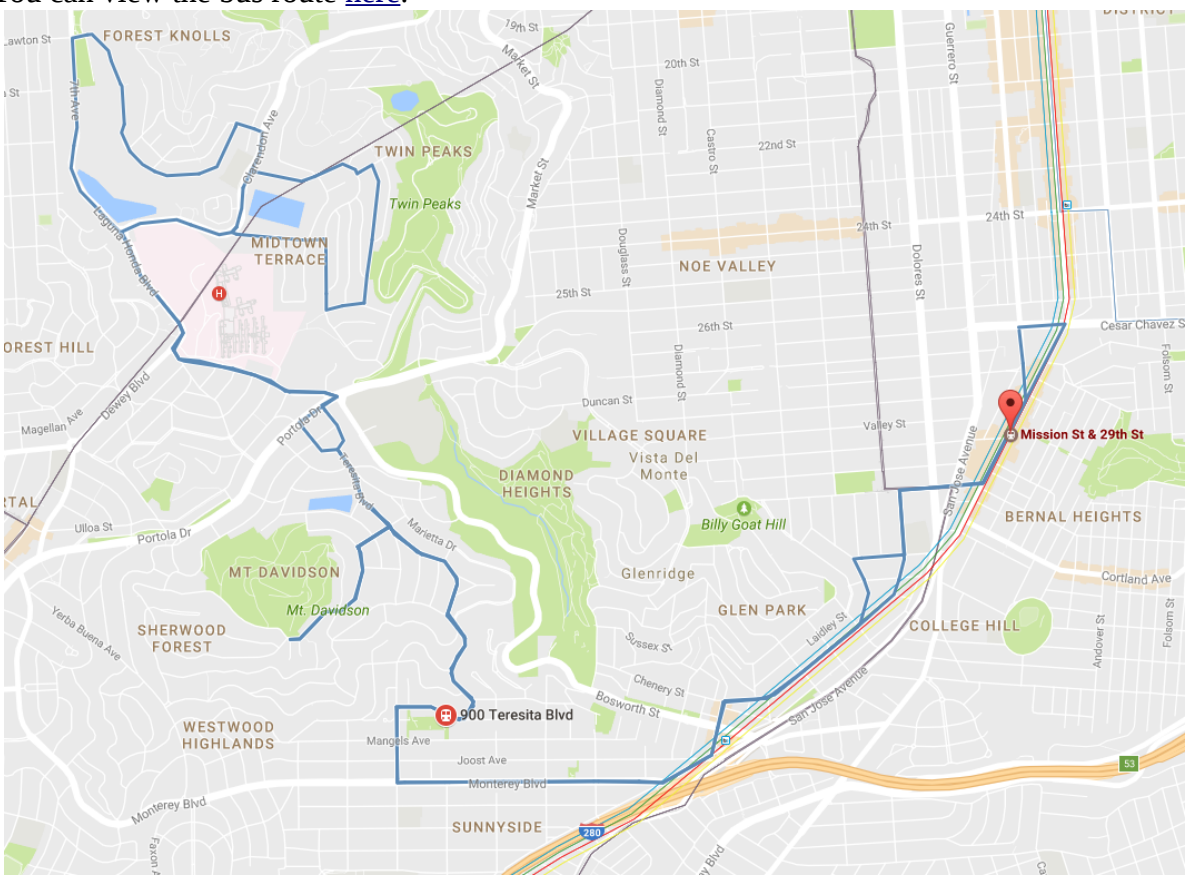
From Section 4



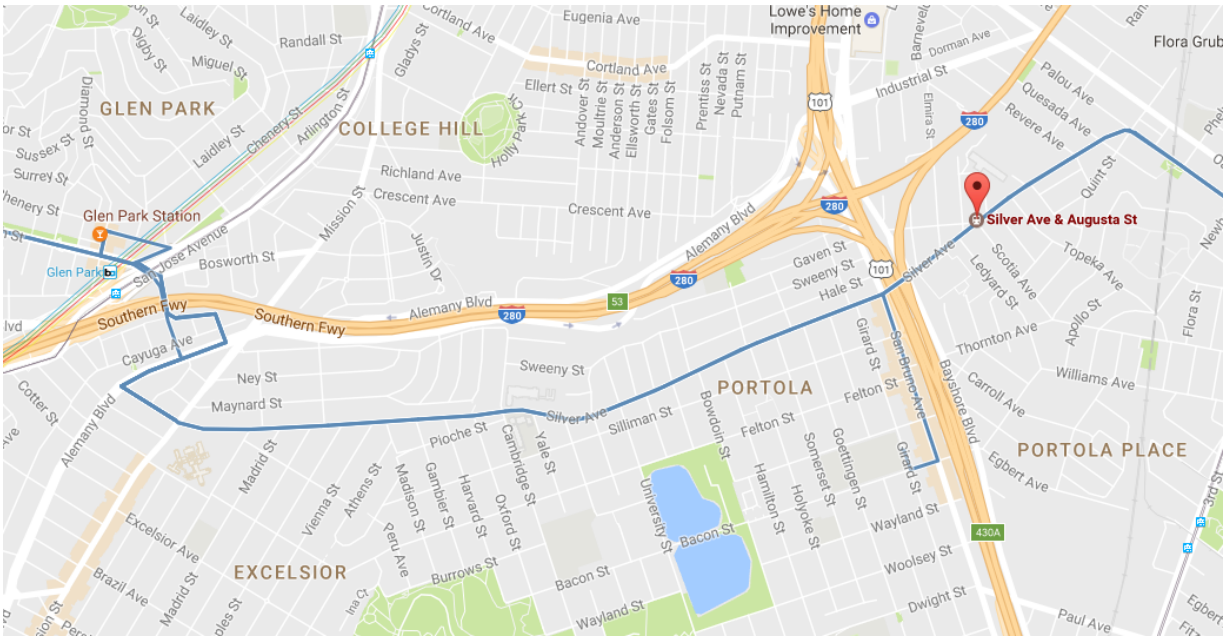
In Section 2 and 4 there were interesting cases. From visual inspection you can see that Mission St and Persia Ave and Mission St and 2nd St should be the correct stop locations. This is due to a discrepancy between the direct distance between the points (Euclidean distance) and the distance the person would actually have to walk (Manhattan distance).

Solution by Jeremy Watkins

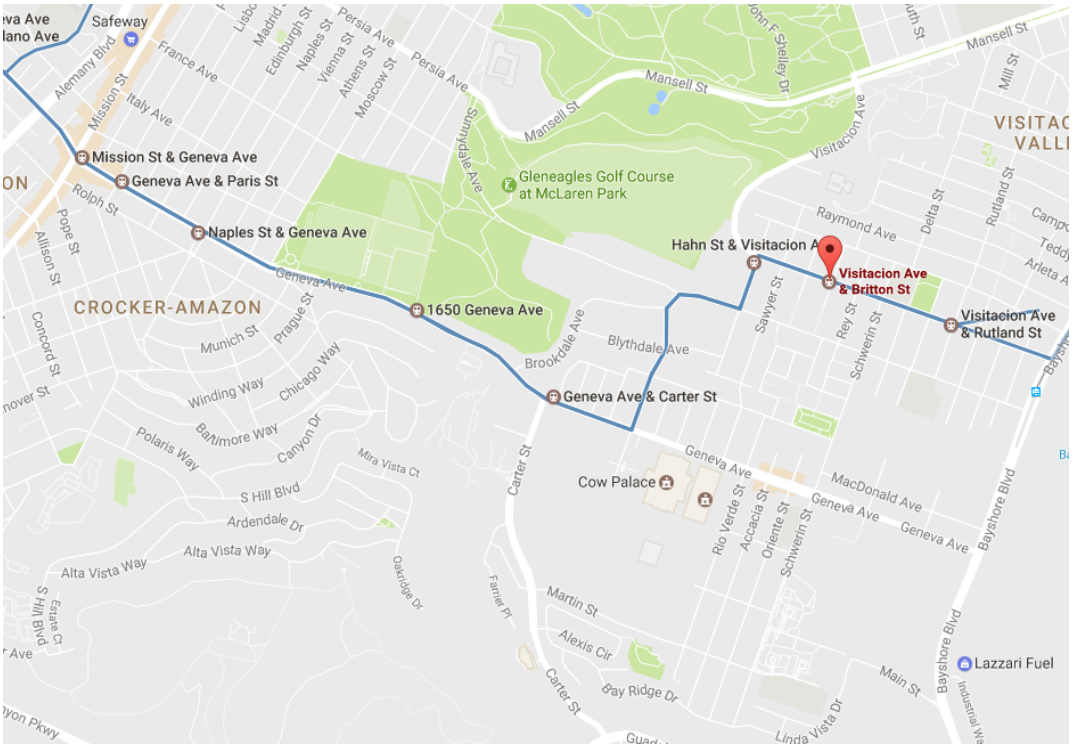
Exterior 1:
This bus route covers all employees that are within Exterior 1 and it will take them directly to a bus stop. You can view the bus route here.



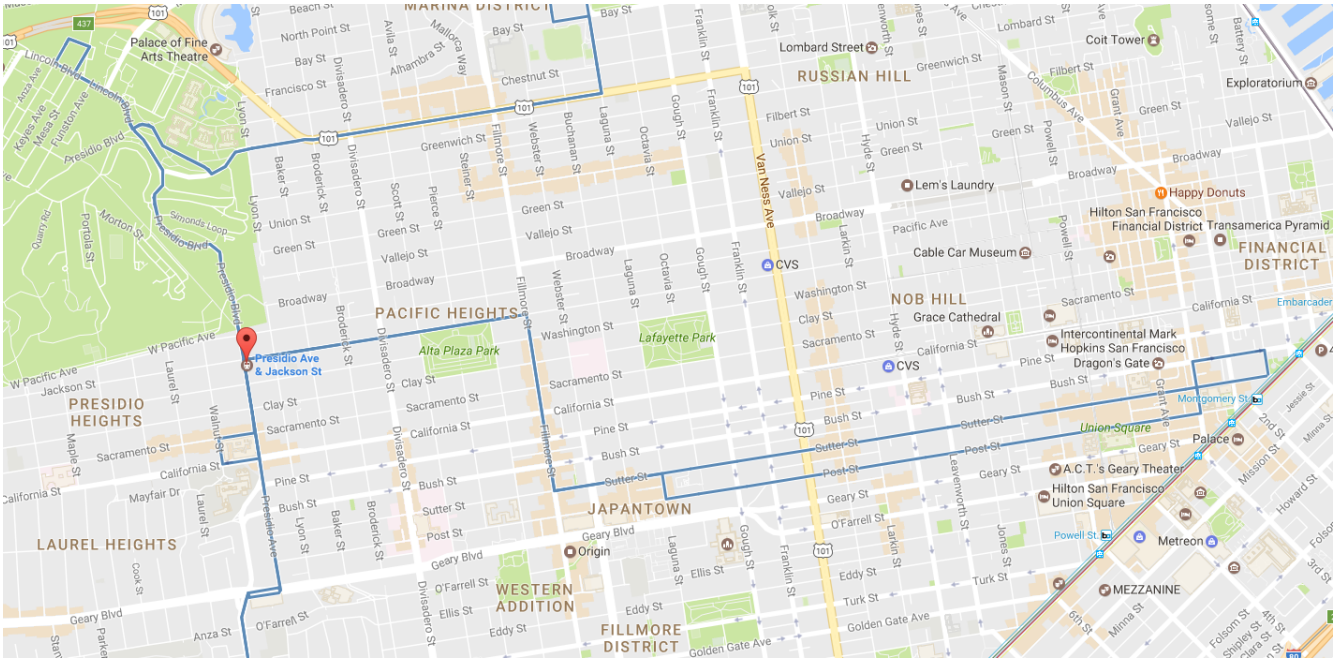
Exterior 2:
This bus route covers all employees within Exterior 2. The route goes within one block of a bus stop. You can view the bus route here.



Exterior 3:
This bus route covers all employees within Exterior 3. The route stops within one block of a bus stop. You can view the bus route here.



Exterior 4:
This bus route covers all employees within Exterior 4. The route stops within one block of a bust stop. You can view the bus route here.

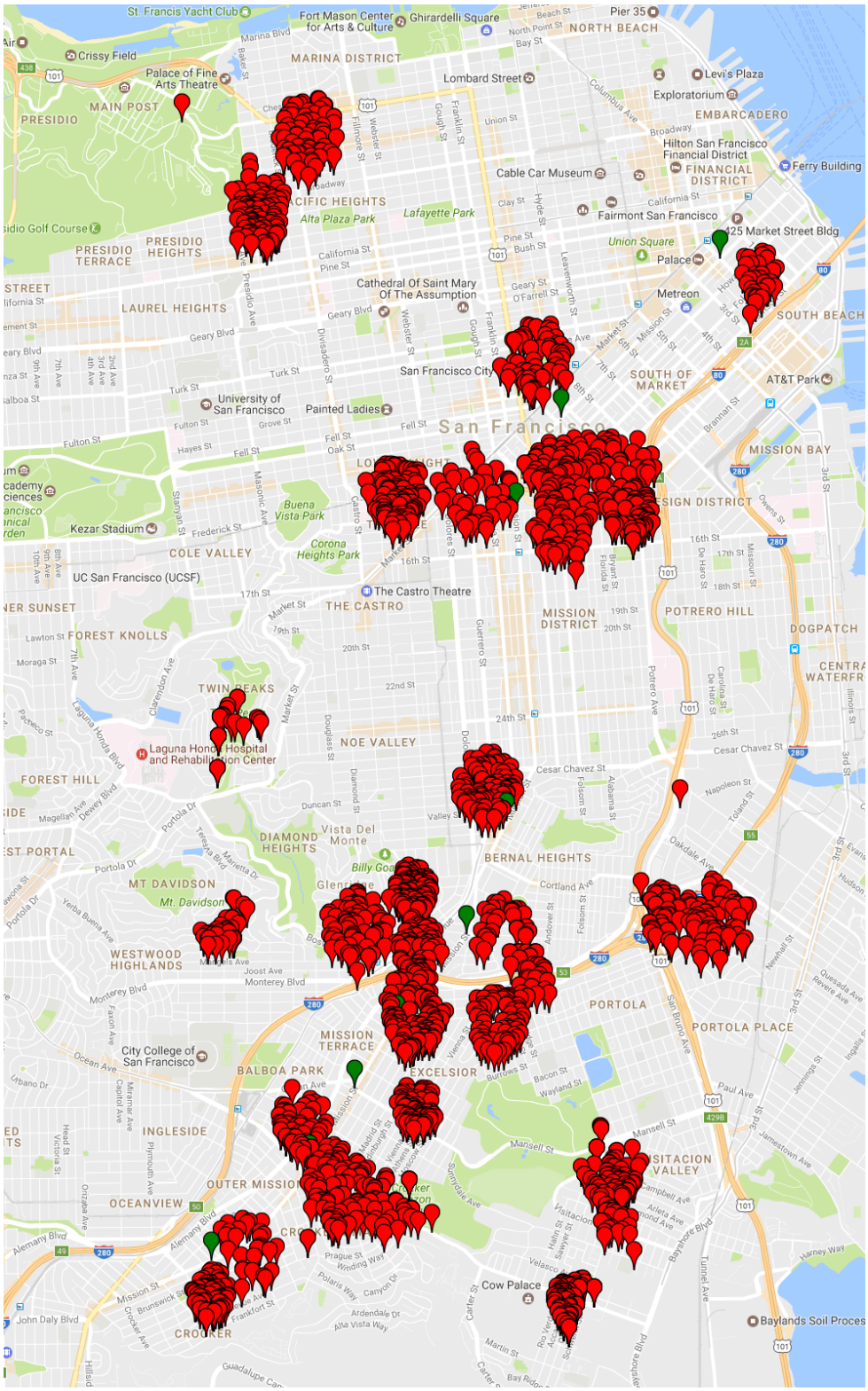


Solution by Jeremy Watkins

Had there not been bus routes that went by the stops that were previously calculated new points would have been added into the calculation. For each member in each respective exterior section a new employee address would be added, located at the public bus drop off point that would be nearest to Mission Street.
Since this was not the case as all exterior regions had bus routes that went to the employee bus stops, keeping the employee bus stops as calculated previously is sufficient. Another thing to consider is that the employees may have other means of transportation to the bus stops that is more flexible than public bus routes. From the data provided there is no way to account for this.

Final Results:

MISSION ST, LAURA ST
MISSION ST, AMAZON AVE
MISSION ST, PERSIA AVE
MISSION ST, TINGLEY ST
MISSION ST, TRUMBULL ST
MISSION ST, COLLEGE AVE
MISSION ST, 29TH ST
MISSION ST, 14TH ST
MISSION ST, GRACE ST
MISSION ST, 02ND ST



Conclusion and Further Work

The model used although not perfect provides a good estimation for the ten most efficient bus stops based on minimizing the overall walking distance for each employee to the nearest bus stop. Each employee is accounted for and provided reasonable access to the employee shuttle. Greater efficiency can be achieved if employees that are too far to be able to walk to Mission Street were able to provide a preferred means of transportation to Mission Street. Also, the algorithm could be optimized if the Manhattan distance between each bus stop and the employee address could be calculated but that would be much more difficult. It would likely require a query of the geocode for every intersection in the area. Then it would require a path finding algorithm and take much more time to compute. This was a very interesting problem and I enjoyed my time solving it.