IEE 574 - Applied Deterministic Operations Research

Route Planning and Optimization – Boomerang Race California

Project Report

Ira A. Fulton Schools of Engineering

Arizona State University

Instructor: Dr. Adolfo Escobedo

Team Members:

Ashish Santha Kumar	**- 1217092981**
Karthikeyan Devaraj	**- 1217975122**
Prakash Sudhakar	**- 1217272901**

Table of Contents

INTRODUCTION: 3

Problem Statement 3

Data Collection and Pre-Processing: 3

Geographical Map: 5

MODELLING 6

Decision variables: 6

Parameters: 6

Objective function: 6

Constraints: 6

SOLUTION METHODS: 7

Off-the Shelf: Miller-Tucker-Zemlin (MTZ) Formulation: 7

Branch and cut Algorithm: 8

RESULTS: 9

Miller-Tucker-Zemlin (MTZ) Formulation: 9

Branch and cut Algorithm: 10

Iteration 0: 10

Iteration 1: 10

Iteration 2: 11

Iteration 3: 11

Iteration 4: 12

Iteration 5: 12

RECOMMENDATIONS: 14

REFERENCES: 14

INTRODUCTION:

Problem Statement

Adventurous drivers have concocted a new challenge that will allow them to race competitively but at a lower risk of violating federal law and attracting undesired attention. A newly proposed “boomerang” race entails driving in the fastest time possible from city ‘A’ in a particular U.S. state, visiting 14 additional “differently lettered” cities within the state in any order but without repetition, and ending back at city ‘A’.

The authors of the project propose a method using our domain knowledge in operations research to figure out the best of best circuit to drive in the race. We formulate of the project with the help of Traveling Salesman Problem using MTZ constraints, Branch and Cut algorithm – Integer programming, Add & Swap Heuristics and the programming part of the project is performed with the help of AMPL/ CPLEX Solver.

Data Collection and Pre-Processing:

The following steps are adopted to do the data collection and processing:

a. The state of California is chosen to design the racing trail.
b. We have chosen 15 cities in the California.
c. Each city corresponds to 15 unique letters in the English alphabet.
d. The data has been scraped from Wikipedia site using Beautiful Soup Python package and sorted based on the requirements.
e. The particular city is chosen for a particular letter by opting for the highest population. The population details can be found in table 1.
f. The distance between each city is found out by taking advantage of the google maps track feature.
g. We start and end with the city starting with ‘A’, which is Anaheim in our case.
h. The chosen cities and their distances between cities are shown below in table 2.

S.No	City	Population
1	Anaheim	336,265
2	Berkley	112,580
3	Chula Vista	243,916
4	Fresno	494,665
5	Glendale	191,719
6	Huntington Beach	189,992
7	Irvine	212,375
8	Los Angeles	3,792,621
9	Modesto	201,165
10	Norwalk	105,549
11	Oakland	390,724
12	Palmdale	152,750
13	Riverside	303,871
14	San Diego	1,301,617
15	Torrance	145,538

Table 1: Cities and its population

The distances between each city are shown below:

		Anaheim	Berkeley	Chula Vista	Fresno	Glendale	Huntington Beach	Irvine	Los Angeles	Modesto	Norwalk	Oakland	Palmdale	Riverside	San Diego	Torrance
	(i,j)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Anaheim	1	2500	401	102	245	34	22	15	27	338	12	397	78	37	95	29
Berkeley	2	401	2500	501	179	369	411	414	374	85	391	5	363	426	494	388
Chula Vista	3	102	501	2500	346	135	100	92	127	439	112	497	176	104	10	125
Fresno	4	245	179	346	2500	214	256	259	219	95	236	175	205	271	339	233
Glendale	5	34	369	135	214	2500	45	48	10	303	25	365	57	59	128	28
Huntington Beach	6	22	411	100	256	45	2500	13	37	348	23	406	98	52	93	30
Irvine	7	15	414	92	259	48	13	2500	40	352	26	410	102	39	85	36
Los Angeles	8	27	374	127	219	10	37	40	2500	309	17	370	62	54	120	21
Modesto	9	338	85	439	95	303	348	352	309	2500	329	78	298	364	432	326
Norwalk	10	12	391	112	236	25	23	26	17	329	2500	387	78	47	106	20
Oakland	11	397	5	497	175	365	406	410	370	78	387	2500	359	422	490	384
Palmdale	12	78	363	176	205	57	98	102	62	298	78	359	2500	75	170	72
Riverside	13	37	426	104	271	59	52	39	54	364	47	422	75	2500	99	63
San Diego	14	95	494	10	339	128	93	85	120	432	106	490	170	99	2500	118
Torrance	15	29	388	125	233	28	30	36	21	326	20	384	72	63	118	2500

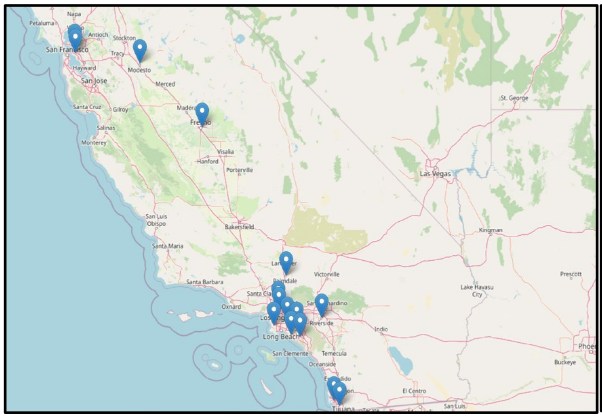
Table 2: Distance Matrix

Geographical Map:

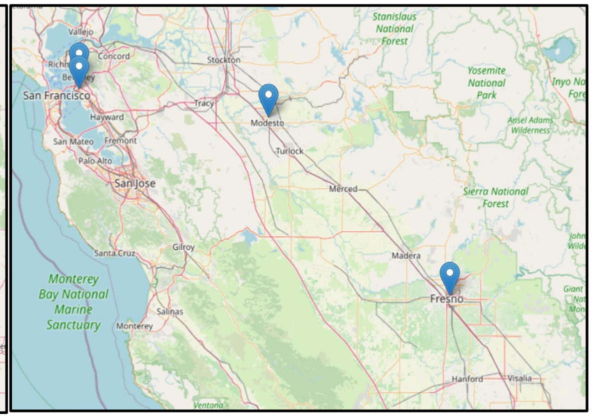
The below images represent the geographical location of the cities that are chosen.

1. The location of overall cities chosen in California.
2. The location of Berkeley, Oakland, Fresno, Modesto.
3. The location of San Diego, Chula Vista.
4. The location of Glendale, Palmdale, Los Angeles, Huntington Beach, Torrance, Irvine, Riverside, Anaheim, Norwalk.

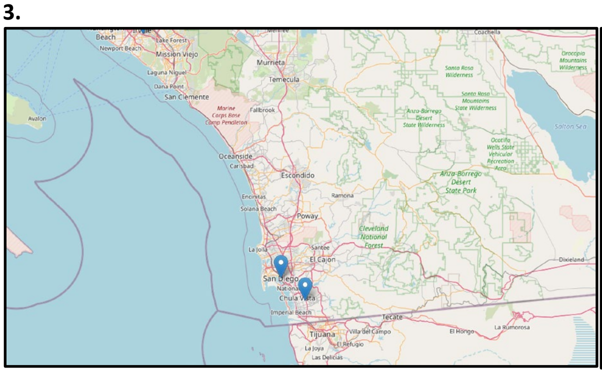
1.



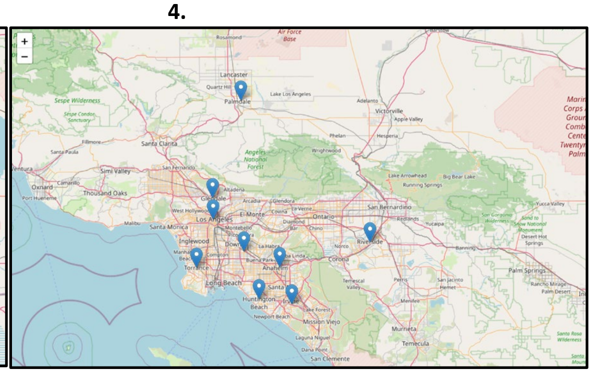
2.



3.



4.




```
!pip install geopy
from geopy.geocoders import Nominatim

import folium
import pandas as pd

data = {'Location': ['Anaheim','Berkeley','Chula Vista', 'Fresno', 'Glendale', 'Huntington Beach', 'Irvine', 'Los Angeles', 'Modesto', 'Norwalk',
                     'Oakland', 'Palmdale', 'Riverside', 'San Diego', 'Torrance'],
        'Latitude': [33.836111, 37.871667, 32.627778, 36.75, 34.146111, 33.692778, 33.669444, 34.05, 37.661389, 33.906944, 37.804444, 34.581111,
                     33.948056, 32.715, 33.834722],
        'Longitude': [-117.889722, -122.272778, -117.048056, -119.766667, -118.255, -118.000278, -117.823056, -118.25, -120.994444, -118.083333,
                      -122.270833, -118.100556, -117.396111, -117.1625, -118.341389]}

data = pd.DataFrame(data, columns = ['Location', 'Latitude', 'Longitude'])

locations = data[['Latitude', 'Longitude']]
locationlist = locations.values.tolist()

map = folium.Map(location=[36.77, -119.41], zoom_start=7)

for i in range(0, len(locationlist)):
  folium.Marker(locationlist[i], tooltip= data['Location'][i]).add_to(map)

map
```

MODELLING

The model is designed to determine the shortest distance that can be achieved by traveling to all the cities that are destined in our model. The decision variables are defined and the constraints are formulated to suit the requirements in our model. Here, we model the basic assignment problem relaxation for Travelling Salesman Problem.

Decision variables:

$X_{ij} = \begin{cases} 1, if\ the\ particular\ path\ is\ selected \\ \quad\quad\quad 0,\ otherwise \end{cases}$

Where, i → starting city and J → destination;

Parameters:

$Cost_{ij}$ = refers to the distance between the two cities

Where, i → start city and j → destination.

Objective function:

Minimize: $\sum_{i=1}^{m} \sum_{j=1}^{n} Cost[i,j] * x[i,j]$

Constraints:

1. $\sum_{i=1}^{m} x[i,j]$, for all j = 1..n (Leave only once from a particular city i)
2. $\sum_{j=1}^{n} x[i,j]$, for all i = 1..m (Arrive only once to a particular city j)
3. $\sum_{j=1}^{n} x[1,j]$, (Assigning the starting city as Aneheim)
4. $\sum_{j=1}^{n} x[j,1]$, (Assigning the destination as Aneheim)
5. Xij = {0,1} , Binary variable

SOLUTION METHODS:

Off-the Shelf: Miller-Tucker-Zemlin (MTZ) Formulation:

```
#mod file

set start; # starting city
set dest; # destination city

param cost{i in start, j in dest}; #distance from place i to j
param N; # number of cities

var x{i in start, j in dest} binary;
var u{i in dest}; # MTZ constraints to eliminate sub tours

#Objective Function
minimize tour_length: sum{i in start, j in dest} cost[i,j] * x[i,j];

#Constraints
subject to arriveonce{j in dest}: sum{i in start} x[i,j] =1;
subject to leaveonce{i in start}: sum{j in dest} x[i,j] = 1;

subject to departfrom: sum{j in dest} x[1,j] = 1; #Assigning the starting point as A
subject to returnto: sum{j in dest} x[j,1] = 1; #Assigning the end point as A

subject to subtour_elimination{i in 2..N, j in 2..N: i!=j}: u[i] - u[j] + N*x[i,j] <= N-1;
subject to u_nonneg{j in dest}: u[j] >= 0;
```

The above constraint marked by a red square is added to avoid the formation of sub tours during execution. The following image shows the original data file used in the execution. The value of the sub tour with the same city as the beginning and the destination is awarded a big value of 2500 to prevent the choice of the trip.

```
#dat file

set start:= 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
set dest:= 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;

param N:= 15;

param cost: 1    2    3    4    5    6    7    8    9    10   11   12   13   14   15:=
1           2500 401  102  245  34   22   15   27   338  12   397  78   37   95   29
2           401  2500 501  179  369  411  414  374  85   391  5    363  426  494  388
3           102  501  2500 346  135  100  92   127  439  112  497  176  104  10   125
4           245  179  346  2500 214  256  259  219  95   236  175  205  271  339  233
5           34   369  135  214  2500 45   48   10   303  25   365  57   59   128  28
6           22   411  100  256  45   2500 13   37   348  23   406  98   52   93   30
7           15   414  92   259  48   13   2500 40   352  26   410  102  39   85   36
8           27   374  127  219  10   37   40   2500 309  17   370  62   54   120  21
9           338  85   439  95   303  348  352  309  2500 329  78   298  364  432  326
10          12   391  112  236  25   23   26   17   329  2500 387  78   47   106  20
11          397  5    497  175  365  406  410  370  78   387  2500 359  422  490  384
12          78   363  176  205  57   98   102  62   298  78   359  2500 75   170  72
13          37   426  104  271  59   52   39   54   364  47   422  75   2500 99   63
14          95   494  10   339  128  93   85   120  432  106  490  170  99   2500 118
15          29   388  125  233  28   30   36   21   326  20   384  72   63   118  2500;
```

Branch and cut Algorithm:

The problem is executed with the initial constraints to obtain a relaxed model function. The mod file 'TSP_BC_iter_0.mod" is run with relaxed constraints and the original data file. After getting the optimal solution, the solution values are checked whether any of the city constraints are violated. If any of the city constraints are violated, then the appropriate constraint is added to the new model file " TSP_BC_iter_1.mod" in order to avoid that selection. The new model file is run and the optimal value is obtained. The previous steps are repeated until the city selections are met.

```
#Assignment Problem relaxation of the TSP Formulation

set start; # starting point
set dest; # destination point

param cost{i in start, j in dest}; # Distance between city i and j
param N; # number of cities

var x{i in start, j in dest} binary;

#Objective Function
minimize tour_length: sum{i in start, j in dest} cost[i,j] * x[i,j];

#Constraints
subject to arriveonce{j in dest}: sum{i in start} x[i,j] =1;
subject to leaveonce{i in start}: sum{j in dest} x[i,j] = 1;

subject to departfrom: sum{j in dest} x[1,j] = 1;
subject to returnto: sum{j in dest} x[j,1] = 1;
```

```
#dat file

set start:= 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
set dest:= 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;

param N:= 15;

param cost: 1    2    3    4    5    6    7    8    9    10   11   12   13   14   15:=
1           2500 401  102  245  34   22   15   27   338  12   397  78   37   95   29
2           401  2500 501  179  369  411  414  374  85   391  5    363  426  494  388
3           102  501  2500 346  135  100  92   127  439  112  497  176  104  10   125
4           245  179  346  2500 214  256  259  219  95   236  175  205  271  339  233
5           34   369  135  214  2500 45   48   10   303  25   365  57   59   128  28
6           22   411  100  256  45   2500 13   37   348  23   406  98   52   93   30
7           15   414  92   259  48   13   2500 40   352  26   410  102  39   85   36
8           27   374  127  219  10   37   40   2500 309  17   370  62   54   120  21
9           338  85   439  95   303  348  352  309  2500 329  78   298  364  432  326
10          12   391  112  236  25   23   26   17   329  2500 387  78   47   106  20
11          397  5    497  175  365  406  410  370  78   387  2500 359  422  490  384
12          78   363  176  205  57   98   102  62   298  78   359  2500 75   170  72
13          37   426  104  271  59   52   39   54   364  47   422  75   2500 99   63
14          95   494  10   339  128  93   85   120  432  106  490  170  99   2500 118
15          29   388  125  233  28   30   36   21   326  20   384  72   63   118  2500;
```

RESULTS:

Miller-Tucker-Zemlin (MTZ) Formulation:

The following image shows the optimal solution for the conditions prescribed. The total distance of 1124 miles is the value of the optimal route that should be adopted by the racer for covering all the cities during the race.

```
CPLEX 12.10.0.0: optimal integer solution; objective 1124
598 MIP simplex iterations
56 branch-and-bound nodes
```

The following is the list of cities in the optimal route order. The interpretation of the order is Anaheim - Huntington beach - Irvine - San Diego - Chula Vista -Riverside- Palmdale - Fresno - Modesto - Oakland - Berkeley - Glendale - Los Angles - Torrance- Norwalk - Anaheim

```
ampl: # The driver should follow this path
ampl: # 1-6-7-14-3-13-12-4-9-11-2-5-8-15-10-1
ampl: # A-H-I-S-C-R-P-F-M-O-B-G-L-T-N-A
```

```
ampl: model TSP_MTZ.mod;
ampl: data TSP_MTZ.dat;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.10.0.0: optimal integer solution; objective 1124
598 MIP simplex iterations
56 branch-and-bound nodes
ampl: display x;
x [*,*]
:    1   2   3   4   5   6   7   8   9  10  11  12  13  14  15    :=
1    0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
2    0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
3    0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
4    0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
5    0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
6    0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
7    0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
8    0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
9    0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
10   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
11   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
12   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
13   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
14   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
15   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
;

ampl: # The driver should follow this path
ampl: # 1-6-7-14-3-13-12-4-9-11-2-5-8-15-10-1
ampl: # A-H-I-S-C-R-P-F-M-O-B-G-L-T-N-A
ampl: # Anaheim - Huntington beach - Irvine - San Diego - Chula Vista -Riverside- Palmdale - Fresno - Modesto - Oakland - Berkeley - Glendale - Los Angles - Torrance
ampl: #Torrance - Norwalk - Anaheim
ampl: #
ampl: #
ampl: #
ampl: #END#
ampl:
```

The above image shows the log file of the execution of “ TSP_MTZ.LOG”. The results of the execution are depicted in the image.

Branch and cut Algorithm:

The module file “TSP_BC_iter_0.mod” is executed and the results are as follows. The following iterations are performed to eliminate the sub tours that occur during the execution of each iteration by the addition of the list of constraints.

Iteration 0:

The objective value at the end of the iteration is 447 miles.

```
CPLEX 12.10.0.0: optimal integer solution; objective 477
13 MIP simplex iterations
0 branch-and-bound nodes
```

The following image shows the list of the sub tours that occur by the end of iteration 0.

```
ampl: #After the solving the relaxation model we have the following subtours
ampl: #1-15-10-1
ampl: #2-11-2
ampl: #3-14-3
ampl: #4-9-4
ampl: #5-8-5
ampl: #6-7-6
ampl: #12-13-12
ampl: #
ampl: #Contraints to prevent these subtours are added in iteration 1 mod file
ampl: #
```

Iteration 1:

The following constraints are added to the module file of the iteration 0 to prevent the sub tours in iteration 0.

```
subject to iter1a: x[1,15]+x[1,10]+x[15,1]+x[15,10]+x[10,1]+x[10,15]<= 2;
subject to iter1b: x[2,11]+x[11,2] <=1;
subject to iter1c: x[3,14]+x[14,3] <=1;
subject to iter1d: x[4,9]+x[9,4] <= 1;
subject to iter1e: x[5,8]+x[8,5] <= 1;
subject to iter1f: x[6,7]+x[7,6] <= 1;
subject to iter1g: x[12,13]+x[13,12] <= 1;
```

After the additions of the constrains, the results obtained in the iteration 1: The objective value at the end of the iteration is 786 miles.

```
CPLEX 12.10.0.0: optimal integer solution; objective 786
33 MIP simplex iterations
0 branch-and-bound nodes
```

The following image shows the list of the sub tours that occur by the end of iteration 1.

```
ampl: #After the solving the Iteration 1 we have the following subtours
ampl: #1-6-7-14-3-13-12-5-8-15-10-1
ampl: #2-11-9-4-2
ampl: #
```

Iteration 2:

The following constraints are added to the module file of the iteration 1 to prevent the sub tours in iteration 1.

```
subject to iter2a: x[1,6]+x[1,7]+x[1,14]+x[1,3]+x[1,13]+x[1,12]+x[1,5]+x[1,8]+x[1,15]+x[1,10]+
                   x[6,1]+x[6,7]+x[6,14]+x[6,3]+x[6,13]+x[6,12]+x[6,5]+x[6,8]+x[6,15]+x[6,10]+
                   x[7,1]+x[7,6]+x[7,14]+x[7,3]+x[7,13]+x[7,12]+x[7,5]+x[7,8]+x[7,15]+x[7,10]+
                   x[14,1]+x[14,6]+x[14,7]+x[14,3]+x[14,13]+x[14,12]+x[14,5]+x[14,8]+x[14,15]+
                   x[14,10]+x[3,1]+x[3,6]+x[3,7]+x[3,14]+x[3,13]+x[3,12]+x[3,5]+x[3,8]+x[3,15]+
                   x[3,10]+x[13,1]+x[13,6]+x[13,7]+x[13,14]+x[13,3]+x[13,12]+x[13,5]+x[13,8]+
                   x[13,15]+x[13,10]+x[12,1]+x[12,6]+x[12,7]+x[12,14]+x[12,3]+x[12,13]+x[12,5]+
                   x[12,8]+x[12,15]+x[12,10]+x[5,1]+x[5,6]+x[5,7]+x[5,14]+x[5,3]+x[5,13]+x[5,12]+
                   x[5,8]+x[5,15]+x[5,10]+x[8,1]+x[8,6]+x[8,7]+x[8,14]+x[8,3]+x[8,13]+x[8,12]+x[8,5]+
                   x[8,15]+x[8,10]+x[15,1]+x[15,6]+x[15,7]+x[15,14]+x[15,3]+x[15,13]+x[15,12]+x[15,5]+
                   x[15,8]+x[15,10]+x[10,1]+x[10,6]+x[10,7]+x[10,14]+x[10,3]+x[10,13]+x[10,12]
                   +x[10,5]+x[10,8]+x[10,15] <= 11;

subject to iter2b: x[2,11]+x[2,9]+x[2,4]+x[11,2]+x[11,9]+x[11,4]+x[9,2]+x[9,11]+x[9,4]+x[4,2]+x[4,11]+x[4,9] <= 3;
```

After the additions of the constrains, the results obtained in iteration 2 are,
The objective value at the end of the iteration is 913 miles.

```
CPLEX 12.10.0.0: optimal integer solution; objective 913
35 MIP simplex iterations
0 branch-and-bound nodes
```

The following image shows the list of the sub tours that occur by the end of iteration 2.

```
ampl: #After the solving the Iteration 2 we have the following subtours
ampl: #1-7-6-10-1
ampl: #2-9-11-2
ampl: #3-13-14-3
ampl: #4-12-4
ampl: #5-15-8-5
ampl: #
```

Iteration 3:

The following constraints are added to the module file of the iteration 3 to prevent the sub tours in iteration 2.

```
#The following cut constraints are added to prevent the above mentioned subtours

subject to iter3a:
x[1,7]+x[1,6]+x[1,10]+x[7,1]+x[7,6]+x[7,10]+x[6,1]+x[6,7]+x[6,10]+x[10,1]+x[10,7]+x[10,6] <= 3;

subject to iter3b:
x[2,9]+x[2,11]+x[9,2]+x[9,11]+x[11,2]+x[11,9]<= 2;

subject to iter3c:
x[3,13]+x[3,14]+x[13,3]+x[13,14]+x[14,3]+x[14,13] <= 2;

subject to iter3d:
x[4,12]+x[12,4] <= 1;

subject to iter3e:
x[5,15]+x[5,8]+x[15,5]+x[15,8]+x[8,5]+x[8,15] <= 2;
```

After the additions of the constrains, the results obtained in iteration 3 are,
The objective value at the end of the iteration is 1092 miles.

```
CPLEX 12.10.0.0: optimal integer solution; objective 1092
42 MIP simplex iterations
0 branch-and-bound nodes
```

The following image shows the list of the sub tours that occur by the end of iteration 3.

```
ampl: #After the solving the Iteration 3 we have the following subtours
ampl: #1-13-3-14-7-6-1
ampl: #2-11-9-4-12-2
ampl: #5-8-10-15-5
ampl: #
```

Iteration 4:

The following constraints are added to the module file of the iteration 4 to prevent the sub tours in iteration 3.

```
subject to iter4a:  x[1,13]+x[1,3]+x[1,14]+x[1,7]+x[1,6]+x[13,1]+x[13,3]+x[13,14]+x[13,7]+x[13,6]+
x[3,1]+x[3,13]+x[3,14]+x[3,7]+x[3,6]+x[14,1]+x[14,13]+x[14,3]+x[14,7]+x[14,6]+x[7,1]+x[7,13]+x[7,3]+
x[7,14]+x[7,6]+x[6,1]+x[6,13]+x[6,3]+x[6,14]+x[6,7] <= 5;

subject to iter4b: x[2,11]+x[2,9]+x[2,4]+x[2,12]+x[11,2]+x[11,9]+x[11,4]+x[11,12]+x[9,2]+x[9,11]+
x[9,4]+x[9,12]+x[4,2]+x[4,11]+x[4,9]+x[4,12]+x[12,2]+x[12,11]+x[12,9]+x[12,4] <= 4;

subject to iter4c: x[5,8]+x[5,10]+x[5,15]+x[8,5]+x[8,10]+x[8,15]+x[10,5]+x[10,8]+x[10,15]+x[15,5]+x[15,8]+x[15,10] <= 3;
```

After the additions of the constrains, the results obtained in iteration 3 are,
The objective value at the end of the iteration is 1124 miles.

```
CPLEX 12.10.0.0: optimal integer solution; objective 1124
35 MIP simplex iterations
0 branch-and-bound nodes
```

The following image shows the list of the sub tours that occur by the end of iteration 3.

```
ampl: #After the solving the Iteration 4 we have the following subtours
ampl: #1-10-1
ampl: #2-11-9-4-12-13-3-14-7-6-15-8-5-2
ampl: #
```

Iteration 5:

The following constraints are added to the module file of the iteration 5 to prevent the sub tours in
After the additions of the constrains, the results obtained in iteration 3 are,

```
subject to iter5a: x[1,10]+x[10,1]<= 1;

subject to iter5b: x[2,11]+x[2,9]+x[2,4]+x[2,12]+x[2,13]+x[2,3]+x[2,14]+x[2,7]+x[2,6]+x[2,15]+
                    x[2,8]+x[2,5]+x[11,2]+x[11,9]+x[11,4]+x[11,12]+x[11,13]+x[11,3]+x[11,14]+x[11,7]+
                    x[11,6]+x[11,15]+x[11,8]+x[11,5]+x[9,2]+x[9,11]+x[9,4]+x[9,12]+x[9,13]+x[9,3]+x[9,14]+
                    x[9,7]+x[9,6]+x[9,15]+x[9,8]+x[9,5]+x[4,2]+x[4,11]+x[4,9]+x[4,12]+x[4,13]+x[4,3]+x[4,14]+
                    x[4,7]+x[4,6]+x[4,15]+x[4,8]+x[4,5]+x[12,2]+x[12,11]+x[12,9]+x[12,4]+x[12,13]+x[12,3]+
                    x[12,14]+x[12,7]+x[12,6]+x[12,15]+x[12,8]+x[12,5]+x[13,2]+x[13,11]+x[13,9]+x[13,4]+
                    x[13,12]+x[13,3]+x[13,14]+x[13,7]+x[13,6]+x[13,15]+x[13,8]+x[13,5]+x[3,2]+x[3,11]+
                    x[3,9]+x[3,4]+x[3,12]+x[3,13]+x[3,14]+x[3,7]+x[3,6]+x[3,15]+x[3,8]+x[3,5]+x[14,2]+
                    x[14,11]+x[14,9]+x[14,4]+x[14,12]+x[14,13]+x[14,3]+x[14,7]+x[14,6]+x[14,15]+x[14,8]+
                    x[14,5]+x[7,2]+x[7,11]+x[7,9]+x[7,4]+x[7,12]+x[7,13]+x[7,3]+x[7,14]+x[7,6]+x[7,15]+
                    x[7,8]+x[7,5]+ x[6,2]+x[6,11]+x[6,9]+x[6,4]+x[6,12]+x[6,13]+x[6,3]+x[6,14]+x[6,7]+x[6,15]+
                    x[6,8]+x[6,5]+x[15,2]+x[15,11]+x[15,9]+x[15,4]+x[15,12]+x[15,13]+x[15,3]+x[15,14]+x[15,7]+
                    x[15,6]+x[15,8]+x[15,5]+x[8,2]+x[8,11]+x[8,9]+x[8,4]+x[8,12]+x[8,13]+x[8,3]+x[8,14]+
                    x[8,7]+x[8,6]+x[8,15]+x[8,5]+x[5,2]+x[5,11]+x[5,9]+x[5,4]+x[5,12]+x[5,13]+x[5,3]+x[5,14]+
                    x[5,7]+x[5,6]+x[5,15]+x[5,8] <= 12;
```

The objective value at the end of the iteration is 1124 miles.

```
CPLEX 12.10.0.0: optimal integer solution; objective 1124
37 MIP simplex iterations
0 branch-and-bound nodes
```

The resulting solution shows that the iteration in gives the optimal solution no sub tours. The optimal route that should be adopted by the driver is shown below.

```
ampl: #No Subtours are found
ampl: #The Solution is
ampl: #1-10-15-8-5-2-11-9-4-12-13-3-14-7-6-1
ampl: #The driver should take the following path
ampl: #Anaheim-Norwalk-Torrance-Los Angeles-Glendale-Berkeley-Oakland
ampl: #-Modesto-Fresno-Palmdale-Riverside-Chula Vista-San Diego-Irvine
ampl: #-Huntington Beach-Anaheim
```

RECOMMENDATIONS:

From the above two methods of analysis for the optimal solution, we have found that the optimal travel distance is 1124 miles and we have two different routes to recommend the driver for participating in the race.

MTZ Constraints → 1-6-7-14-3-13-12-4-9-11-2-5-8-15-10-1

Branch and Cut → 1-10-15-8-5-2-11-9-4-12-13-3-14-7-6-1

For future purposes, when there are high computation needs, it is always better to perform sub-tour elimination rather than using MTZ constraints to avoid unnecessary computational space and time. This is evident from our solution, MTZ constraints gives out results in 598 iterations, while Branch and cut using sub tour elimination gives out results in 37 iterations. The variables and constraints being generated in MTZ is way large when compared to sub tour elimination.

REFERENCES:

1. List of cities and towns in California – Wikipedia

2. A branch-and-cut algorithm for traveling salesman problem with pickup and delivery - Elsevier

3. AMPL Google groups

4. Introduction to Mathematical Programming – Wayne L Winston

5. AMPL – A modeling Language for Mathematical Programming – Robert Fourer