# Travelling Salesman Problem
CS164 - Spring 2020

*Chenshuo Ma*
*Dennis Kageni*
*Taha Bouhoun*

## Contents

**1 Problem description:** **1**

**2 Touristic sites in the city:** **1**

**3 Approach:** **2**

**4 Results:** **3**

**5 Appendix:** **4**
5.1 Contributions: . . . . . 4
5.2 Code source: Google Colab link . . . . . 5

OPTIMIZING BUENOS AIRES TOUR

# 1 Problem description:

In this report, we present an approach to solving the Traveling Salesmen Problem using Convex Problem Solver in Python. The aim is to minimize the distance between all the touristic attractions in Buenos Aires starting from the Minerva residence hall (Hotel CH Le Petit Suites) located in 933 Esmeralda street, Buenos Aires.

# 2 Touristic sites in the city:

0. Plaza de Mayo (*19 min walk from the Residence Hall*):
   This is the oldest public square in Buenos Aires. Around it, are famous buildings such as the seat of the national government (Casa Rosada), the Metropolitan Cathedral (Pope Francis used to conduct mass here) and Cabildo which hosts the National Museum of the Cabildo and the May Revolution.

1. San Telmo (*17 min walk from the Plaza de Mayo*):
   The oldest neighborhood (barrio) in Buenos Aires that features many art galleries, art spaces and museums. It's characterized by it's colonial-style buildings and is considered by many to be the destination for contemporary art lovers.

2. Palacio Borolo:
   This was once South America's tallest building when it was completed in 1923. The 22 story building's unique design is inspired by Dante's Divine Comedy.

3. Recoleta Cemetery (*58 min walk/ 16 min car ride from the San Telmo*):
   A cemetery, right? Located in the Recoleta neighborhood, it contains the remains of famous Argentine figures such as a granddaughter of Napoleon, former presidents, first ladies nobel prize winners, among others. It contains about 5000 vaults, all above ground, with a wide variety of architectural styles such as neo-Gothic, Baroque and Art Deco.

4. La Boca Caminito (*91 min walk/ 25 min car ride from the Recoleta Cemetery*):
   A street museum of colorful painted houses. It features works by Argentine artists and it a tourism hot-spot. Should you want to learn tango while sipping coffee, then you'd be pleased to know that several restaurants offer tango and folk dance shows

5. Reserva Ecologica (*49 min walk/ 13 min car ride from La Boca*):
   Should you prefer being in nature, the Costanera Sur Ecological Reserve is the biggest and most bio-diverse green space in Buenos Aires. The best thing is that it's in close proximity to the downtown area so you can still enjoy it's trails either on foot or on a bike

6. Bosques Palermo (93 min walk/ 16 min car ride from Reserva Ecologica_):
   The oldest park on Buenos Aires that features multiple green spaces, three artificial lakes and a wide variety of biodiversity. It's a fantastic space for picnics, nature photography or exercise

7. El Ateneo Grand Splendid (*47 min walk/ 8 min car ride from Bosques Palermo*):
   Perhaps the most famous bookstore in Argentina. Build in 1919, it is a converted theatre

where one can immerse themselves into books and coffee before making a purchase. Despite being re-purposed into a bookstore, it still retains the feeling of being a theatre.

8. La Bombonera Stadium (*83 min walk/ 18 min car ride from El Ateneo* ):
   For all soccer fans, this is the home to Boca Junior's; once of Argentina's most popular football clubs (with a fan-base of over 16 million fans)

9. Espacio Memoria y Derechos Humanos:
   This is a former naval academy that was a detention center between 1976 to 1983. It's now a museum dedicated to documenting the torture and disappearance of political dissidents in this period.

10. Corte Comedor: Did anyone say steak? What better place to enjoy Argentina's world famous beef other than in this modern steakhouse; without breaking the back, of course.

11. Teatro Colón:
    This imposing structure that spans an entire block is Buenos Aires' main performing arts venue. If you are into the world of theatre, ballet or opera then take this opportunity for a 50 minute long backstage tour to revel in the magnificent interior, architecture and acoustics.

12. El Zanjon:
    An archaeological site located in San Telmo for all those who want to immerse themselves in the city's history. It is a series of old tunnels, sewers and cisterns that date back to the 1700s and provide context into one of Buenos Aires' oldest settlements

# 3 Approach:

- First, formulating the problem mathematically to minimize the distance that must be travelled to cover all the destinations and return to the residence hall:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\forall j \in [1,\ n] \rightarrow \sum_{j=0}^{n} x_{ij} = 2$$

$c_{ij} = c_{ji}$ : distance from city i to city j (symmetric version)

$x_{ij} = 1$ if the traveler visits i then j, and 0 otherwise (binary)

The set of constraints to prevents subtours:

$$\forall i, j \in [2,\ n] \ \rightarrow \ t_i - t_j + n x_{ij} \leq n - 1$$

- Second, we input the list of popular attractions in the city and the extracted the distances relative to each other using the Google Maps API to build the following symmetrical matrix: (distances are expressed in meters)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | 1000000.0 | 2566.5 | 4397.0 | 2307.0 | 2298.0 | 7001.0 | 2558.5 | 5743.0 |
| **1** | 2566.5 | 1000000.0 | 2289.0 | 1624.5 | 4279.0 | 4536.5 | 2204.5 | 7257.0 |
| **2** | 4397.0 | 2289.0 | 1000000.0 | 3110.5 | 5813.0 | 3526.5 | 2777.0 | 8600.5 |
| **3** | 2307.0 | 1624.5 | 3110.5 | 1000000.0 | 3586.5 | 5898.5 | 3467.5 | 6537.5 |
| **4** | 2298.0 | 4279.0 | 5813.0 | 3586.5 | 1000000.0 | 8714.0 | 4206.5 | 3738.5 |
| **5** | 7001.0 | 4536.5 | 3526.5 | 5898.5 | 8714.0 | 1000000.0 | 4111.5 | 11692.0 |
| **6** | 2558.5 | 2204.5 | 2777.0 | 3467.5 | 4206.5 | 4111.5 | 1000000.0 | 6315.5 |
| **7** | 5743.0 | 7257.0 | 8600.5 | 6537.5 | 3738.5 | 11692.0 | 6315.5 | 1000000.0 |

- Third, we formulated the problem in Python and used the CVXPY package to single out the pattern that satisfies the constraint and output the minimum distance.

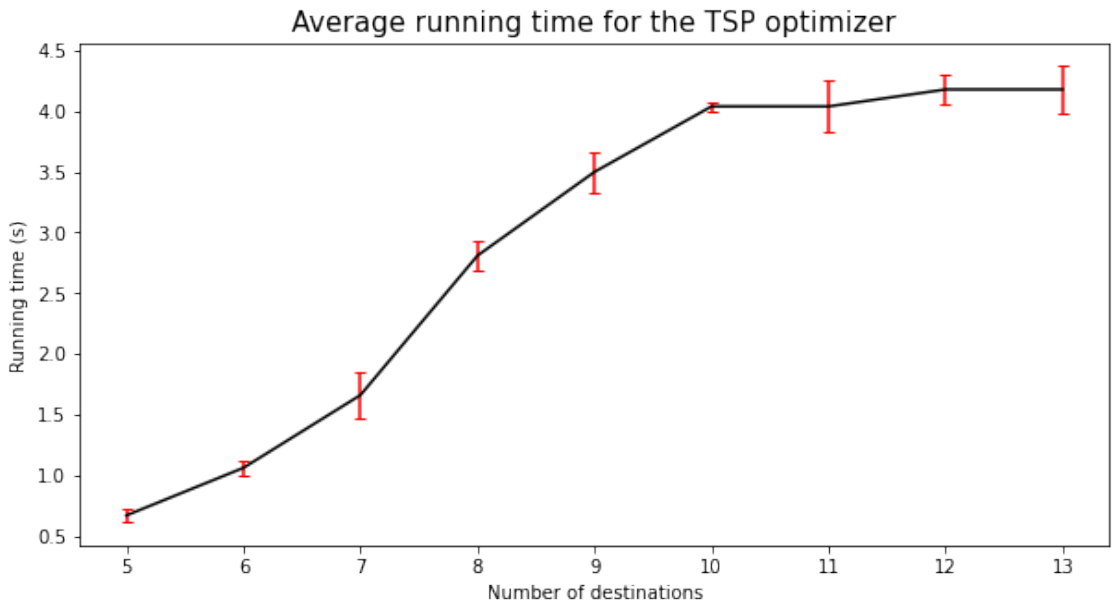
## 4 Results:

The shortest trip through all 13 destinations starting from the residence hall at Esmeralda 933 then eventually going back is:

```
The shortest path:  [0, 8, 4, 11, 10, 7, 6, 9, 5, 2, 13, 1, 3, 12]
The length of the shortest path: 42883.5 meters
```

Furthermore, testing the running time of the TSP using `%timeit` proved that the cost for running the algorithm increases in a logarithmic manner as shown below:

```
05 locations: 668 ms ± 29.6 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
06 locations: 1.06 s ± 28.3 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
07 locations: 1.66 s ± 95.2 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
08 locations: 2.81 s ± 64.2 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
09 locations: 3.5 s ± 84.7 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
10 locations: 4.04 s ± 17.8 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
11 locations: 4.04 s ± 106 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
12 locations: 4.18 s ± 64.4 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
13 locations: 4.18 s ± 96.5 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
```




# 5 Appendix:

## 5.1 Contributions:

- Chenshuo Ma: Implemented a salable version of the problem in Python and solved for the optimal distance.
- Dennis Kageni: Compiled a list of touristic attractions in Buenos Aires as well as their descriptions
- Taha Bouhoun: Compiled the matrix of distances using Google API, Turned the sequence of the solution into a graph to visualize it on a map, and run-time analysis.

### 5.2 Code source: Google Colab link

```
In [0]: !pip install googlemaps

In [0]: !pip install mplleaflet

In [2]: import matplotlib.pyplot as plt
        import cvxpy as cvx
        import numpy as np
        import pandas as pd
        import networkx as nx
        import mplleaflet
        import googlemaps
        import random
        import json

In [4]: # GET THE DISTANCES BETWEEN PLACES USING GOOGLE MAPS API
        gmaps = googlemaps.Client(key='AIzaSyBTz-a4viHh6g9xYwWNvfMWxFBmOPk2dm4')

In [5]: matrix = []
        destinations = ['ch_le_petit_suites', 'plaza_de_mayo',
                        'san_telmo', 'palacio_borolo',
                        'recoleta_cemetery', 'la_boca_caminito',
                        'reserva_ecologica', 'bosques_palermo',
                        'grand_splendid', 'la_bombonera',
                        'espacio_memoria_y_derechos_humanos',
                        'corte_comedor', 'teatro_colón', 'el_zanjon' ]

        for i in destinations:
            for j in destinations:
                matrix.append(gmaps.directions(i, j)[0]\
                              ['legs'][0]['distance']['value'])

In [6]: # PUTTING THE DISTANCES INTO A MATRIX (IN METERS)
        matrix = np.array(matrix).reshape(len(destinations),len(destinations))

        # SYMMETRICAL MATRIX
        matrix = (matrix + matrix.T) / 2
        np.fill_diagonal(matrix, 1e6)
        pd.DataFrame(matrix)
```

Out[6]:

| | 0 | 1 | 2 | 3 | 4 | 5 | \ |
|---|---|---|---|---|---|---|---|
| 0 | 1000000.0 | 2566.5 | 4397.0 | 2307.0 | 2298.0 | 7001.0 | |
| 1 | 2566.5 | 1000000.0 | 2289.0 | 1624.5 | 4279.0 | 4536.5 | |
| 2 | 4397.0 | 2289.0 | 1000000.0 | 3110.5 | 5813.0 | 3526.5 | |
| 3 | 2307.0 | 1624.5 | 3110.5 | 1000000.0 | 3586.5 | 5898.5 | |
| 4 | 2298.0 | 4279.0 | 5813.0 | 3586.5 | 1000000.0 | 8714.0 | |
| 5 | 7001.0 | 4536.5 | 3526.5 | 5898.5 | 8714.0 | 1000000.0 | |
| 6 | 2558.5 | 2204.5 | 2777.0 | 3467.5 | 4206.5 | 4111.5 | |

```
7      5743.0     7257.0     8600.5     6537.5     3738.5    11692.0
8      1918.0     3845.5     5435.0     2768.0     1590.5     8324.0
9      6451.0     4100.5     3090.5     5092.5     7845.5      965.0
10    12439.0    14274.0    15587.0    13882.5    11100.5    18372.5
11    11227.5    12108.0    14375.5    11417.5     7995.0    16543.0
12     1047.0     2418.5     4121.0     1801.0     2341.0     6868.0
13     3931.0     1628.5     1269.0     2484.0     5831.0     3517.0

             6          7          8          9         10         11  \
0       2558.5     5743.0     1918.0     6451.0    12439.0    11227.5
1       2204.5     7257.0     3845.5     4100.5    14274.0    12108.0
2       2777.0     8600.5     5435.0     3090.5    15587.0    14375.5
3       3467.5     6537.5     2768.0     5092.5    13882.5    11417.5
4       4206.5     3738.5     1590.5     7845.5    11100.5     7995.0
5       4111.5    11692.0     8324.0      965.0    18372.5    16543.0
6    1000000.0     6315.5     4138.5     3634.5    13571.0    11741.5
7       6315.5  1000000.0     4605.0    11142.0     6869.5     4023.0
8       4138.5     4605.0  1000000.0     7422.0    13550.0     8861.5
9       3634.5    11142.0     7422.0  1000000.0    17913.5    16702.0
10     13571.0     6869.5    13550.0    17913.5  1000000.0     2699.0
11     11741.5     4023.0     8861.5    16702.0     2699.0  1000000.0
12      3022.0     5939.0     2192.5     6062.0    12646.5    11435.0
13      1935.0     7940.0     5565.0     3081.0    15195.5    13366.0

             12         13
0       1047.0     3931.0
1       2418.5     1628.5
2       4121.0     1269.0
3       1801.0     2484.0
4       2341.0     5831.0
5       6868.0     3517.0
6       3022.0     1935.0
7       5939.0     7940.0
8       2192.5     5565.0
9       6062.0     3081.0
10     12646.5    15195.5
11     11435.0    13366.0
12   1000000.0     3520.0
13      3520.0  1000000.0
```

In [17]:
```
# CHECK IF THE MATRIX CONSTRUCTED IS INDEED SYMMETRIC
(matrix.transpose() == matrix).all()
```

Out[17]: True

In [18]:
```
N = len(destinations)
V = cvx.Variable((N, N), boolean=True)
```

In [19]:
```
obj = cvx.Minimize(sum([matrix[i,:] @ V[:,i] for i in range(N)]))
```

In [20]:
```
constr = [cvx.sum(V[i,:])==1 for i in range(N)]+\
          [cvx.sum(V[:,i]) == 1 for i in range(N)]

cons1 = [V[i]>=0 for i in range(N)]
cons2 = constr + cons1
```

In [21]:
```
prob = cvx.Problem(obj, cons2)
print("Optimal value", prob.solve())
print("Optimal var")
print(V.value)
```

```
Optimal value 28586.0
Optimal var
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]]
```

In [22]:
```
'''
Aux function which gives sequence of cities visited given a adj. matrix.
returns a list less than 9 if subtours contained.
'''
def print_tour(B):
    N = B.shape[0]
    tour = [0]
    current_city = 0
    for i in range(N):
        for j in range(N):
            if B[current_city,j]!=0 and j not in tour:
                current_city = j
                tour.append(j)
    return tour
```

In [23]:
```
### First iteration has subtours, it seems.
print_tour(V.value)
```

Out[23]:
```
[0, 12]
```

In [24]:
```
### Putting whatever I had above into a function.

def full_tour(dist_matrix):
    if dist_matrix.shape[0]!= dist_matrix.shape[1]:
        return "bad dist matrix, check again!"
    else:
        N = dist_matrix.shape[0]
        V = cvx.Variable((N,N),boolean=True)
        obj = cvx.Minimize(sum([dist_matrix[i,:]@V[:,i] for i in range(N)]))
        constr = [cvx.sum(V[i,:])==1 for i in range(N)]+\
                 [cvx.sum(V[:,i]) == 1 for i in range(N)] +\
                 [V[i]>=0 for i in range(N)]

        # Initialize this to be every single city
        not_visited = [i for i in range(N)]
        # While subtours exist:
        # not_visited will be updated to be cities which are unreachable from city 0
        while not_visited:
            problem = cvx.Problem(obj,constr)
            opt = problem.solve()
            tour_mat = V.value
            visited = print_tour(tour_mat)
            not_visited = [i for i in range(N) if i not in visited]
            if not_visited:
                constr = constr + [sum([V[i,j] for i in visited
                                        for j in not_visited]) >= 1]

        return print_tour(tour_mat), opt
```

In [25]:
```
solution = full_tour(matrix)
print('The shortest path: ', solution[0],
      '\nThe length of the shortest path: {} meters'.format(solution[1]))
```

```
The shortest path:  [0, 8, 4, 11, 10, 7, 6, 9, 5, 2, 13, 1, 3, 12]
The length of the shortest path: 42883.5 meters
```

In [26]:
```
path = solution[0]
path.append(path[0])
graph = []
for dist in destinations:
    graph.append([dist, list(gmaps.geocode(dist)[0]\
                             ['geometry']['location'].values())[::-1]])

graph = pd.DataFrame(graph, columns=['places', 'coordinates'])
edges = list(map(tuple, zip(path, path[1:])))
graph
```

Out[26]:

| | places | coordinates |
|---|---|---|
| 0 | ch_le_petit_suites | [-58.37829499999999, -34.5972512] |

```
1                          plaza_de_mayo          [-58.3722832, -34.6083667]
2                              san_telmo          [-58.3713942, -34.6218351]
3                           palacio_borolo          [-58.3858428, -34.6095914]
4                         recoleta_cemetery          [-58.3934409, -34.5874834]
5                          la_boca_caminito          [-58.3626839, -34.6393399]
6                         reserva_ecologica          [-58.3509217, -34.6081059]
7                           bosques_palermo  [-58.41729640000001, -34.5712762]
8                           grand_splendid          [-58.3942285, -34.5959833]
9                             la_bombonera  [-58.36475629999999, -34.6356109]
10  espacio_memoria_y_derechos_humanos   [-58.46329679999999, -34.538097]
11                          corte_comedor          [-58.4481816, -34.5540677]
12                           teatro_colón          [-58.3830786, -34.6010406]
13                              el_zanjon  [-58.37180799999999, -34.6167218]
```

In [27]:
```
graph_nx = nx.Graph()
graph_nx.add_nodes_from(map(int, graph.index))
graph_nx.add_edges_from(edges)
print(nx.info(graph_nx))
```

```
Name:
Type: Graph
Number of nodes: 14
Number of edges: 14
Average degree:   2.0000
```

In [28]:
```
plt.figure(figsize=(10, 8))
nx.draw(graph_nx, pos = graph.coordinates, edgelist = edges,
        node_size = 100, node_color='red')
mplleaflet.display()
```

```
C:\Users\Taha\Anaconda3\Anaconda4\lib\site-packages\IPython\core\display.py:689: UserWarning: Co
  warnings.warn("Consider using IPython.display.IFrame instead")
```

Out[28]: <IPython.core.display.HTML object>

In [59]:
```
# TIMING THE TSP OPTIMIZER RUNNING TIME
for i in range(5, len(destinations)):
    %timeit -n 5 full_tour(matrix[:4+i,:4+i])
```

```
668 ms ± 29.6 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
1.06 s ± 28.3 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
1.66 s ± 95.2 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
2.81 s ± 64.2 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
3.5 s ± 84.7 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
4.04 s ± 17.8 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
4.04 s ± 106 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
4.18 s ± 64.4 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
4.18 s ± 96.5 ms per loop (mean ± std. dev. of 7 runs, 5 loops each)
```

In [16]:
```
# PLOTTING THE MEASUREMENT OF RUNNING TIME AND ERROR ESTIMATES
meas = np.array([.668, 1.06, 1.66, 2.81, 3.5, 4.04, 4.04, 4.18, 4.18])
err = np.array([29.6, 28.3, 95.2, 64.2, 84.7, 17.8, 106, 64.4, 96.5])*2
plt.figure(figsize=(10, 5))
plt.errorbar(range(5, len(destinations)), meas, yerr=(err)/1000,
             capsize=3, ecolor='red', color='black')
plt.xlabel('Number of destinations'); plt.ylabel('Running time (s)')
plt.title('Average running time for the TSP optimizer', fontsize=15)
plt.show()
```