

基因演算法期末報告

機車環島路徑之最佳化

組員:張智鈞、賴璟鍾、蔡宛秦、白欣怡



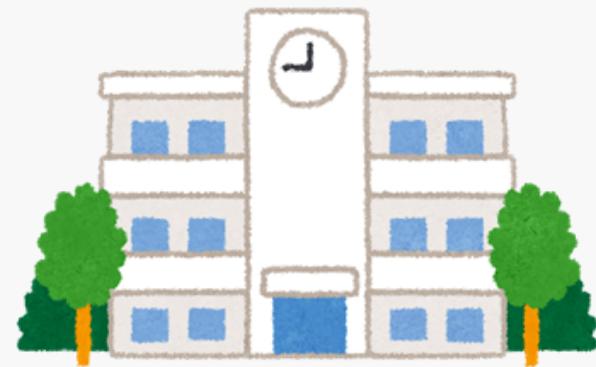
報告架構

1. 研究背景與目的
2. 研究方法
 - a. 建立景點資料庫
 - b. 生成距離矩陣
 - c. 產生最佳路徑
3. 研究結果
 - a. 成果示範
 - b. 方法比較與分析可行性
4. 結論與未來展望



1. 研究背景與目的

背景：很窮，只能騎上破舊的機車環島。在這趟路程我們需要用有效率的方式完成我們 11 天的環島之旅。



1. 環島出發地點：交大
(先往南騎)



3. 裝備：競戰4代
油耗量設定 $25\text{km/L} \sim 26\text{km/L}$



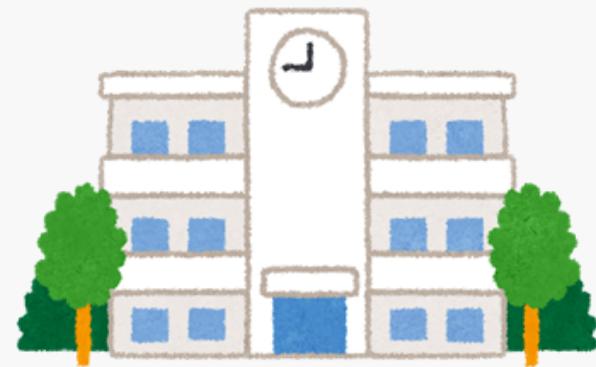
2. 環島時間：11天



4. 生活用品：
睡袋和紙板、科學麵 n 包

1. 研究背景與目的

背景：很窮，只能騎上破舊的機車環島。在這趟路程我們需要用有效率的方式完成我們 11 天的環島之旅。



1. 環島出發地點：交大
(先往南騎)



3. 裝備：競戰4代
油耗量設定 $25\text{km/L} \sim 26\text{km/L}$



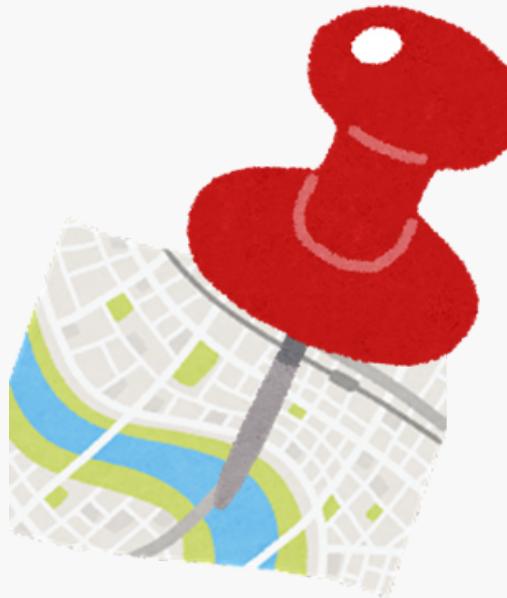
2. 環島時間：11天



4. 生活用品：
睡袋和紙板、科學麵 n 包

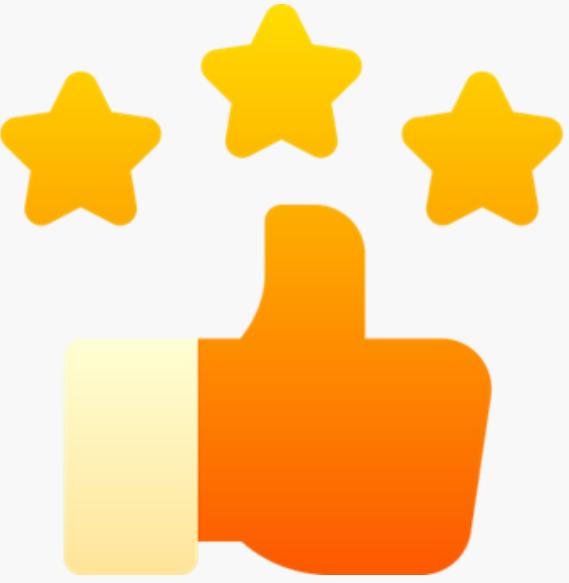
1. 研究背景與目的

目的：使用Google API 與基因演算法算出最佳優化路徑，
達成客製化、路途短、滿意度高的目標。



1. 中心景點

考慮到每位騎士的偏好，
選擇想去或是靠近省道的景點
作為各縣市旅途的中心



2. 景點評分

在路徑規劃中，將考慮到騎士的遊玩滿意度，優先選擇中心附近評分最高的景點遊玩，
把景點評分作為一個重要的參數



3. 距離

將每個景點之間機車所需騎行的實際距離作為一個重要的參數

2. 研究方法

STEP1. 建立景點資料庫

利用google api找出中心位置
利用map.py 生成 .CSV檔



2. 研究方法

STEP1. 建立景點資料庫

利用google api找出中心位置
利用map.py 生成 .CSV檔



STEP2. 生成距離矩陣

CSV檔 + distsquare.py
->生成10景點的距離矩陣



2. 研究方法

STEP1. 建立景點資料庫

利用google api找出中心位置
利用map.py 生成 .CSV檔



STEP2. 生成距離矩陣

CSV檔 + distsquare.py
->生成10景點的距離矩陣



STEP3. 產生最佳路徑

.CSV 檔 + 距離矩陣 + tour.py
->生成最佳路徑



2. 研究方法

STEP1. 建立景點資料庫



- 怎麼用google api找出中心位置

```
1 import requests
2
3 def get_coordinates(address):
4     url = f"https://maps.googleapis.com/maps/api/geocode/json?address={address}&key=AIzaSyA65CeniB10
5     response = requests.get(url)
6     data = response.json()
7
8     if data['status'] == 'OK':
9         location = data['results'][0]['geometry']['location']
10        latitude = location['lat']
11        longitude = location['lng']
12        return latitude, longitude
13    else:
14        return None
15
16 address = "猴硐貓村"
17 coordinates = get_coordinates(address)
18
19 if coordinates:
20     latitude, longitude = coordinates
21     print(f"Latitude: {latitude}")
22     print(f"Longitude: {longitude}")
23 else:
24     print("Unable to find coordinates for the given address.")
```

得出位置:

```
PS C:\Users\jason\OneDrive\桌面\基因演算法\final project> & C:/Users/jason/AppData/Local/Micros
oft/WindowsApps/python3.11.exe "c:/Users/jason/OneDrive/桌面/基因演算法/final project/TSP_prob.
py"
Latitude: 25.0869527
Longitude: 121.8274974
```

STEP2. 生成距離矩陣

STEP3. 產生最佳路徑

2. 研究方法

STEP1. 建立景點資料庫



- 怎麼用google api找出中心位置
- 怎麼用中心點找出附近評分高且評論不小於100的景點

定義中心點的座標、半徑大小、評論數：

```
# Define the center point and radius for the search
location = (25.0869527,121.8274974)
radius = 7000 # meter

# Query tourist attractions within the specified radius
places_result = gmaps.places_nearby(location=location, keyword='tourist', radius=radius)
results = places_result['results']

# Filter out attractions with a rating count less than 300
results_filtered = [place for place in results if place.get('user_ratings_total', 0) >= 100]
```

STEP2. 生成距離矩陣

STEP3. 產生最佳路徑

2. 研究方法

STEP1. 建立景點資料庫



- 怎麼用google api找出中心位置
- 怎麼用中心點找出附近評分高且評論不小於100的景點

STEP2. 生成距離矩陣

STEP3. 產生最佳路徑

寫入.csv檔:

```
# Write attractions to CSV
with open('Kaohsiung_Tourist_2.csv', 'w', newline='', encoding='utf-8-sig') as csvfile:
    fieldnames = ['name', 'latitude', 'longitude', 'rating', 'rating_count', 'address']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames, quotechar='"', quoting=csv.QUOTE_ALL)
    writer.writeheader()

    # Query details for each attraction and write to CSV
    for i, place in enumerate(results_filtered[:80], 1):
        place_name = place['name']

        # Use geocoding API to get the Place ID of the attraction
        geocode_result = gmaps.geocode(place_name)
        if not geocode_result:
            print(f"Could not find location: {place_name}")
            continue

        place_id = geocode_result[0]['place_id']

        # Fetch details for the attraction
        place_details = gmaps.place(place_id=place_id, fields=['name', 'rating',
        'user_ratings_total', 'geometry', 'formatted_address'])

        if 'result' in place_details:
            result = place_details['result']
            place_name = result.get('name')
            rating = result.get('rating', 'N/A')
            rating_count = result.get('user_ratings_total', 'N/A')
            location = result['geometry']['location']
            address = result.get('formatted_address', 'N/A')

            # Write data to CSV
            writer.writerow({
                'name': place_name,
                'latitude': location['lat'],
                'longitude': location['lng'],
                'rating': rating,
                'rating_count': rating_count,
                'address': address
            })

            print(f"{i}: {place_name} - Rating: {rating}, Rating Count: {rating_count}, Address: {address}")
        else:
            print(f"Could not retrieve place details for {place_name}")
```

2. 研究方法

STEP1. 建立景點資料庫



- 怎麼用google api找出中心位置
- 怎麼用中心點找出附近評分高且評論不小於100的景點

STEP2. 生成距離矩陣

STEP3. 產生最佳路徑

資料庫:

NewTaipeicity_Tourist_1.csv > data

```
1 "name", "latitude", "longitude", "rating", "rating_count"
2 "摩天瀑布", "25.06321299999999", Col 3: longitude 5, "295"
3 "Chaojing Park", "25.1434517", "121.8034149", "4.4", "16315"
4 "Hsiaojingua Outcrop", "25.1000172", "121.8452546", "4.5", "257"
5 "Sandiaoling Tunnel Bike Path Rest Point", "25.056793", "121.8430888", "4.5", "1348"
6 "陰陽海景觀台", "25.1222411", "121.8642059", "4.5", "1491"
7 "枇杷洞瀑布", "25.06339879999999", "121.8047454", "4.5", "686"
8 "Shen'ao Rail Bike (Shen'ao Station)", "25.12917939999999", "121.8144922", "4.5", "1654"
9 "合谷瀑布", "25.0614165", "121.8110348", "4.4", "160"
10 "Golden Waterfall", "25.1171278", "121.8614895", "4.4", "12038"
11 "Jinguashi Geopark", "25.1009411", "121.8545499", "4.4", "2480"
```

.CSV檔:

| | A | B | C | D | E |
|----|---|----------|-----------|--------|--------------|
| 1 | name | latitude | longitude | rating | rating_count |
| 2 | Caoshan Radar Station | 25.09478 | 121.8748 | 4.7 | 952 |
| 3 | Buyan Pavilion | 25.08924 | 121.8475 | 4.6 | 9479 |
| 4 | 摩天瀑布 | 25.06321 | 121.8054 | 4.5 | 295 |
| 5 | Hsiaojingua Outcrop | 25.10002 | 121.8453 | 4.5 | 257 |
| 6 | Sandiaoling Tunnel Bike Path Rest Point | 25.05679 | 121.8431 | 4.5 | 1348 |
| 7 | 陰陽海景觀台 | 25.12224 | 121.8642 | 4.5 | 1491 |
| 8 | 枇杷洞瀑布 | 25.0634 | 121.8047 | 4.5 | 686 |
| 9 | Shen'ao Rail Bike (Shen'ao Station) | 25.12918 | 121.8145 | 4.5 | 1653 |
| 10 | Shuqi Road | 25.10869 | 121.8435 | N/A | N/A |
| 11 | 合谷瀑布 | 25.06142 | 121.811 | 4.4 | 160 |

2. 研究方法

STEP1.建立景點資料庫

STEP2.生成距離矩陣



STEP3.產生最佳路徑

(1):

```
df = pd.read_csv('NewTaipeiicity_Tourist_1.csv')
distances = []

for i in range(10):
    row = []
    for j in range(10):
        if i == j:
            row.append(0)
        else:
            time.sleep(0.05)
            origin = df.loc[i, 'name']
            destination = df.loc[j, 'name']
            result = gmaps.distance_matrix(origin, destination)
            if 'distance' in result['rows'][0]['elements'][0]:
                distance = result['rows'][0]['elements'][0]['distance']['text']
                row.append(distance)
            else:
                row.append(None)
    distances.append(row)
```

2. 研究方法

STEP1.建立景點資料庫

STEP2.生成距離矩陣



(2):

```
for i in range(10):
    for j in range(i+1, 10): # Start from i+1 to avoid overwriting already calculated distances
        if distances[i][j] is not None:
            distances[j][i] = distances[i][j]
        elif distances[j][i] is not None:
            distances[i][j] = distances[j][i]

for i in range (10):
    print(distances[i], end='\n')
for i in range (10):
    places = []
    for i in range(10):
        places.append(df.loc[i, 'name'])
print(places)
```

STEP3.產生最佳路徑

2. 研究方法

STEP1.建立景點資料庫

STEP2.生成距離矩陣



距離矩陣:

```
#NewTaipeicity_Tourist_1
distances = [
    [0, 18.3, 21.9, 25.1, 19.6, 1.6, 15.8, 13, 20.5, 19.5],
    [18.3, 0, 13.7, 20.0, 8.7, 17.5, 2.4, 18.4, 9.6, 11.3],
    [21.9, 13.7, 0, 9.7, 8.5, 21.0, 11.2, 21.9, 7.8, 1.6],
    [25.1, 20.0, 9.7, 0, 21.2, 24.2, 17.4, 25.1, 16.5, 10.3],
    [19.6, 8.7, 8.5, 21.2, 0, 18.7, 6.2, 19.6, 1.0, 6.3],
    [1.6, 17.5, 21.0, 24.2, 18.7, 0, 14.9, 1.6, 19.6, 18.6],
    [15.8, 2.4, 11.2, 17.4, 6.2, 14.9, 0, 16.0, 7.1, 8.9],
    [13, 18.4, 21.9, 25.1, 19.6, 1.6, 16.0, 0, 20.5, 19.5],
    [20.5, 9.6, 7.8, 16.5, 1.0, 19.6, 7.1, 20.5, 0, 5.4],
    [19.5, 11.3, 1.6, 10.3, 6.3, 18.6, 8.9, 19.5, 5.4, 0]
]
```

STEP3.產生最佳路徑

2. 研究方法

STEP1.建立景點資料庫

STEP2.生成距離矩陣

STEP3.產生最佳路徑



- 放入距離矩陣以及資料庫

```
distances = [
    [0, 10.8, 13.8, 11.9, 8.4, 10.5, 5.7, 10.9, 8.6, 4.2],
    [10.8, 0, 8.3, 6.5, 18.3, 20.4, 15.6, 5.5, 18.4, 14.1],
    [13.8, 8.3, 0, 1.8, 21.3, 23.4, 18.5, 2.8, 21.4, 17.1],
    [11.9, 6.5, 1.8, 0, 19.5, 21.6, 16.7, 1.0, 19.6, 15.3],
    [8.4, 18.3, 21.3, 19.5, 0, 6.7, 2.0, 17.7, 0.1, 9.6],
    [10.5, 20.4, 23.4, 21.6, 6.7, 0, 4.9, 20.7, 8.9, 12.5],
    [5.7, 15.6, 18.5, 16.7, 2.0, 4.9, 0, 15.7, 3.9, 7.6],
    [10.9, 5.5, 2.8, 1.0, 17.7, 20.7, 15.7, 0, 18.6, 14.2],
    [8.6, 18.4, 21.4, 19.6, 0.1, 8.9, 3.9, 18.6, 0, 9.7],
    [4.2, 14.1, 17.1, 15.3, 9.6, 12.5, 7.6, 14.2, 9.7, 0]
]

ratings = []

with open('Taoyuan_Tourist_2.csv', newline='', encoding='utf-8') as csvfile:
    reader = csv.DictReader(csvfile)
    for i, row in enumerate(reader):
        if i >= 0 and i <= 9:  # 第四列的2到11行
            rating = float(row['rating'])
            ratings.append(rating)
# print(ratings)
```

2. 研究方法

STEP1. 建立景點資料庫

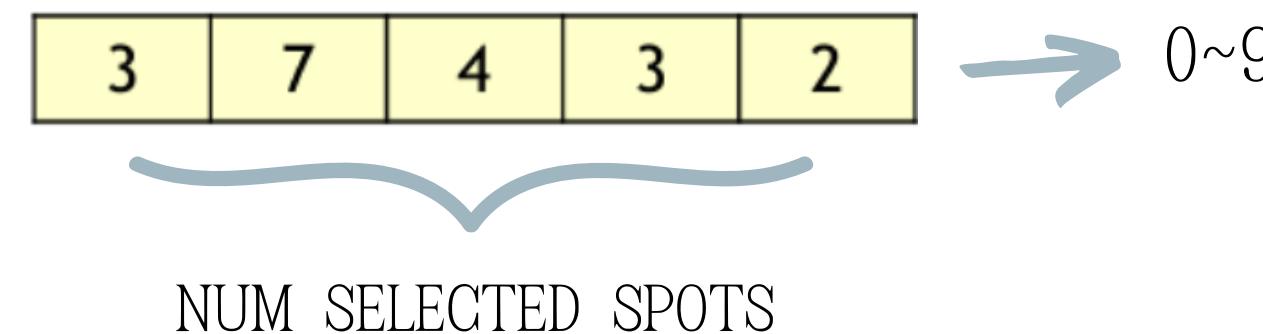
STEP2. 生成距離矩陣

STEP3. 產生最佳路徑



- 放入距離矩陣以及資料庫
- 染色體 & 適應度函數
(考量：距離、評分)

- 染色體：discrete decision variables



- 適應度函數：

x_i : 各景點之評分， $i = 0,1,2,3,4,5,6,7,8,9$

y_{jk} : 由第 j 個景點到第 k 個景點的距離(km) = $\begin{cases} R^+, & j \neq k \\ 0, & j = k \end{cases}$

$j = 0,1,2,3,4,5,6,7,8,9$

$k = 0,1,2,3,4,5,6,7,8,9$

距離矩陣：
$$\begin{bmatrix} y_{00} & \cdots & y_{0k} \\ \vdots & \ddots & \vdots \\ y_{j0} & \cdots & y_{99} \end{bmatrix}$$

\Rightarrow 適應度函數 $f(x, y) = \sum x_i - \sum y_{jk}$

2. 研究方法

STEP1.建立景點資料庫

STEP2.生成距離矩陣

STEP3.產生最佳路徑



- 放入距離矩陣以及資料庫
- 染色體 & 適應度函數
(考量：距離、評分)

```
# 參數設置
POPULATION_SIZE = 100
MAX_GENERATIONS = 1000
MUTATION_RATE = 0.1
NUM_SELECTED_SPOTS = 3

# 適應度函數，最大化評分總和並最小化距離總和
def fitness_function(chromosome):
    total_distance = 0
    total_rating = 0

    # 計算選擇的景點的評分總和
    for spot in chromosome:
        total_rating += 1 * ratings[spot]

    # 計算選擇的景點的距離總和
    start = chromosome[0]
    prev = start
    for current in chromosome[1:] + [start]:
        total_distance += distances[prev][current]
        prev = current

    # 適應度函數為評分總和減去距離總和
    return total_rating - total_distance
```

2. 研究方法

STEP1. 建立景點資料庫

STEP2. 生成距離矩陣

STEP3. 產生最佳路徑



- 放入距離矩陣以及資料庫
- 染色體 & 適應度函數
(考量：距離、評分)

結果：

```
Generation 983: Best fitness = 9.10, Best chromosome = [0, 3, 9]
Generation 984: Best fitness = 9.10, Best chromosome = [0, 3, 9]
Generation 985: Best fitness = 9.10, Best chromosome = [3, 9, 0]
Generation 986: Best fitness = 9.10, Best chromosome = [0, 3, 9]
Generation 987: Best fitness = 9.10, Best chromosome = [0, 3, 9]
Generation 988: Best fitness = 9.10, Best chromosome = [3, 9, 0]
Generation 989: Best fitness = 9.10, Best chromosome = [3, 9, 0]
Generation 990: Best fitness = 8.30, Best chromosome = [9, 3, 0]
Generation 991: Best fitness = 9.10, Best chromosome = [0, 3, 9]
Generation 992: Best fitness = 9.10, Best chromosome = [0, 3, 9]
Generation 993: Best fitness = 6.90, Best chromosome = [2, 0, 3]
Generation 994: Best fitness = 9.10, Best chromosome = [9, 0, 3]
Generation 995: Best fitness = 9.10, Best chromosome = [0, 3, 9]
Generation 996: Best fitness = 9.10, Best chromosome = [9, 0, 3]
Generation 997: Best fitness = 8.30, Best chromosome = [0, 9, 3]
Generation 998: Best fitness = 8.30, Best chromosome = [0, 9, 3]
Generation 999: Best fitness = 8.30, Best chromosome = [0, 9, 3]
```

Figures now render in the Plots pane by default. To make them also appear inline in the Console, uncheck "Mute Inline Plotting" under the Plots pane options menu.

Best solution: [3, 9, 0]

3. 研究結果

起點陽明交大 ->

(苗栗) 貓裡山公園 -> 功維敘隧道 -> 玉清公園 -> 後龍溪河濱公園 -> 黃金小鎮休閒農 ->
(台中) 麗寶落羽松大道 -> 花梁鋼橋 -> 大坑八號登山步道 -> 紙箱王創意園區 -> 湖心亭 -> 綠空鐵道1908 -> 興大園道阿勃勒黃金步道 -> 康橋水岸公園 ->
(南投) 省府日常散策 -> 南投燈會水舞廣場 -> 中興新村拱門 -> 中興新村親情公園 -> 中興新村兒童公園 ->
(雲林) 斗六籽公園 -> 近水樓台湖畔森林咖啡 -> 太平老街 ->
(嘉義) 朴子溪自行車道 -> 嘉藝點水道頭文創聚落 -> 荷苞嶼生態園區 ->
(台南) 虱目魚小子 -> 蜀葵花文化節 -> 老塘湖藝術村 -> 赤崁文化園區 -> 臺南市立體育公園 -> 巴克禮紀念公園 ->
(高雄) 台灣滷味博物館 -> 岡山公園 -> 橋仔頭糖廠 -> 愛河灣水樂園 -> 光榮碼頭 -> 駁二藝術特區 ->
(屏東) 雙流瀑布步道 -> 星砂灣 -> 臺灣最南點 -> 龍磐公園 -> 牡丹灣 ->
(台東) 南迴海天一線 -> 太麻里平交道 -> 鐵道藝術村 -> 郡界遊憩區 -> 水往上流遊憩區 -> 金樽陸連島 ->
(花蓮) 富里花海景觀區 -> 南安瀑布 -> 八通關古道東段登山口 -> 遠雄海洋公園 -> 太平洋公園 -> 松園別館 ->
(宜蘭) 斑比山丘 -> 水岸-森林物語 -> 陳定南紀念園區 -> 羅東林業文化園區 -> 玉兔鉛筆學校觀光工廠 ->
(新北) 陰陽海景觀台 -> 金瓜石地質公園 -> 黃金瀑布 -> 文山草堂 -> 石碇鱸魚島 ->
(台北) 榴錦時光生活園區 -> 國立中正紀念堂 -> 二二八紀念公園 -> 國立故宮博物院 -> 芝山岩文化史蹟公園 ->
(桃園) 天空繩橋 -> 宇內溪戲水區 -> 大溪中正公園 -> 大溪橋 -> 大溪河濱公園 ->
(新竹) 李克承博士故居 -> 天公壇公園 -> 城隍廟廣場 -> 新竹公園 ->

回陽明交大

3. 研究結果 - 成果示範

- MyMap 分次疊圖的成果



3. 研究結果 - 將網頁寫入本地網頁



3. 研究結果 - 方法比較



網路部落格

| | 我們的方法 | 網路部落格 (同為逆時針) |
|--------|---|---|
| 總騎乘公里數 | 1360.9km | 1218km |
| 路線 | 北：台2線、台7線 中：台3線 南：台1線、台61線、台17線 東：台9線、台26線、台11線、 台23線 | 北：台2線、台1線 中：台13線、台3線 南：台63線、台74線 東：台9線、台11線、台2線 |
| 騎乘難易度 | 123.718 km/perday  | 文中建議：150-200 km/perday |
| 天數 | 11天 | 9天 |
| 景點數 | 74  | 42 |

3. 研究結果 - 分析可行性

- 騎乘容易度：每天行駛距離適當
- 景點喜好：挑選的景點評論佳、評分高
- 個人需求：想睡就睡、想吃就吃



4. 結論與未來展望

- 演算法
 - 基因演算法參數
 - 將所有縣市一同放入距離計算最佳解
 - 考量縣市交界起始點設定問題
- 網頁呈現
 - 繪圖方法：有更多景點間的路線規劃
 - 架設網站提供服務

謝謝大家聆聽

