# HW1 for modern optimization methods


## B06702064 會計五 林聖硯


## 1. Optimization of a Function using Simulated Annealing

(a)

objective function:

$$f(\mathbf{x}) = 6x_1^2 - 6x_1x_2 + 2x_2^2 - x_1 - 2x_2$$

### Temperature Scheduling Parameters Settings

Assume we use exponential scheduling, with $\epsilon = 0.1$, i.e. $T_i = (1-0.1)^i \times T_0$, where $i$ is the number of iteration. Also, assume the Boltzmann constant $k_B$ is 1, i.e. the Boltzmann probability distribution is

$$P(\Delta E) = \begin{cases} e^{-\frac{\Delta E}{T}}, & \text{if } \Delta E > 0 \\ 1, & \text{otherwise} \end{cases}$$

In addition, the initial temperature will be set in Step 0.

### Mutation settings

Assume the appropriate range of mutation is $\pm 3$.

### Step 0 (initialization):

- Randomly choose **five** points for initialization. $\mathbf{x}^{(1)} = \begin{bmatrix} 6 \\ 5 \end{bmatrix}$, $\mathbf{x}^{(2)} = \begin{bmatrix} 0 \\ 7 \end{bmatrix}$, $\mathbf{x}^{(3)} = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$, $\mathbf{x}^{(4)} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$, $\mathbf{x}^{(5)} = \begin{bmatrix} 7 \\ 7 \end{bmatrix}$.

- The corresponding values of objective function are, $f(\mathbf{x}^{(1)}) = 70$, $f(\mathbf{x}^{(2)}) = 84$, $f(\mathbf{x}^{(3)}) = 170$, $f(\mathbf{x}^{(4)}) = 68$, $f(\mathbf{x}^{(5)}) = 77$

- Aussume the initial temperature is the average of the five objective value and the initial design point is the average of all five points. We get
  $T_0 = \frac{70 + 84 + 170 + 68 + 77}{5} = 93.8$, $\mathbf{x}_0 = \frac{\mathbf{x}^{(1)} + \mathbf{x}^{(2)} + \mathbf{x}^{(3)} + \mathbf{x}^{(4)} + \mathbf{x}^{(5)}}{5} = \begin{bmatrix} 5.4 \\ 6 \end{bmatrix}$

## Step 2 ($i$ = 1)

- Select two uniformly distributed random variable $u_1 = 0.52$ and $u_2 = 0.66$ for the vicinity of $5.4$ and $6$ respectively. The ranges of $\pm 3$ imply that the available ranges of $x_1$ and $x_2$ are $(2.4, 8.4)$ and $(3, 9)$
- Thus, we get $\mathbf{x}_2 = \begin{bmatrix} 2.4 + 0.52 \times (8.4 - 2.4) \\ 3 + 0.66 \times (9 - 3) \end{bmatrix} = \begin{bmatrix} 5.52 \\ 6.96 \end{bmatrix}$
- The objective value of $\mathbf{x}_2$ is $f(\mathbf{x}_2) = 29.75$, $\Delta f = f(\mathbf{x}_2) - f(\mathbf{x}_1) = -64.05$
- Since $\Delta f$ is negative, we accept $\mathbf{x}_2$.

## Step 3 ($i$ = 2)

- Select two uniformly distributed random variable $u_1 = 0.15$ and $u_2 = 0.54$ for the vicinity of $5.52$ and $6.96$ respectively. The ranges of $\pm 3$ imply that the available ranges of $x_1$ and $x_2$ are $(2.52, 8.52)$ and $(3.96, 9.96)$
- Thus, we get $\mathbf{x}_3 = \begin{bmatrix} 2.52 + 0.15 \times (8.52 - 5.52) \\ 3.96 + 0.54 \times (9.96 - 3.96) \end{bmatrix} = \begin{bmatrix} 3.42 \\ 7.2 \end{bmatrix}$
- The objective value of $\mathbf{x}_3$ is $f(\mathbf{x}_3) = 8.29$, $\Delta f = f(\mathbf{x}_3) - f(\mathbf{x}_2) = -21.46$
- Since $\Delta f$ is negative, we accept $\mathbf{x}_3$.

(b) Solve with hill climbing algorithm

**Stoping Criteria**

- number of iteration > 100
- when the objective value do not update

Final result:

```
Iteration 37, x1 = 1.82, x2 = 3.5, best objective value = -2.67
Iteration 38, x1 = 1.82, x2 = 3.4, best objective value = -2.75
Iteration 39, x1 = 1.72, x2 = 3.3, best objective value = -2.85
Iteration 40, x1 = 1.72, x2 = 3.2, best objective value = -2.91
Iteration 41, x1 = 1.62, x2 = 3.1, best objective value = -2.99
Iteration 42, x1 = 1.62, x2 = 3.0, best objective value = -3.03
Iteration 43, x1 = 1.52, x2 = 2.9, best objective value = -3.09
Iteration 44, x1 = 1.52, x2 = 2.8, best objective value = -3.11
Iteration 45, x1 = 1.42, x2 = 2.7, best objective value = -3.15
Iteration 46, x1 = 1.42, x2 = 2.6, best objective value = -3.15
Iteration 47, x1 = 1.32, x2 = 2.5, best objective value = -3.17
```

(c) Solve with random walk algorithm

**Stoping Criteria**

- number of iteration > 100

Final result:

```
Iteration 45, x1 = 1.42, x2 = 2.7, best objective value = -3.15
Iteration 46, x1 = 1.42, x2 = 2.6, best objective value = -3.15
Iteration 47, x1 = 1.32, x2 = 2.5, best objective value = -3.17
Iteration 48, x1 = 1.32, x2 = 2.4, best objective value = -3.15
Iteration 49, x1 = 1.32, x2 = 2.5, best objective value = -3.17
```

(d) Solve with simulated annealing algorithm

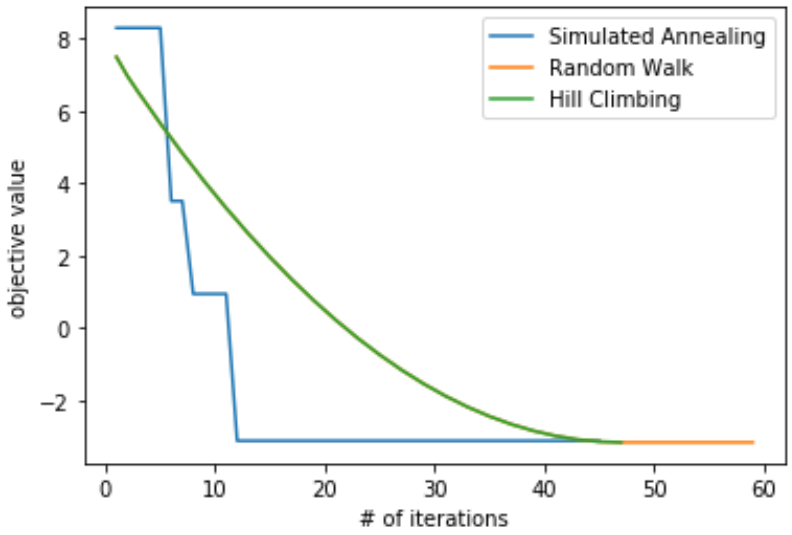
**Stoping Criteria**

- number of iteration > 1000
- when the objective value do not update either for **three times** in a row.
- when the temperature is less than **1 degree**.

Final result:

```
Iteration 46, current temperature = 8.86, x1 = 0.48, x2 = 2.94, best objective value = 3.84
Iteration 47, current temperature = 8.42, x1 = 1.26, x2 = 3.66, best objective value = 0.07
Iteration 48, current temperature = 8.0, x1 = 1.26, x2 = 3.66, best objective value = 0.07
Iteration 49, current temperature = 7.6, x1 = 1.86, x2 = 3.3, best objective value = -2.75
Iteration 50, current temperature = 7.22, x1 = -0.0, x2 = 1.68, best objective value = 2.28
Iteration 51, current temperature = 6.86, x1 = -0.0, x2 = 1.68, best objective value = 2.28
Iteration 52, current temperature = 6.51, x1 = -0.0, x2 = 1.68, best objective value = 2.28
Iteration 53, current temperature = 6.19, x1 = -0.0, x2 = 1.68, best objective value = 2.28
```

(e) Observation with FE plot




From the FE plot, we can find that the SA algorithm do not update in the early stage since we may update current objective value (which is worse than current best solution) when the temperature is high. However, after the fifth iteration, we can see that the objective value decreasing dramatically and produce better answer than the other two algoriths, it may caused by the fact that the SA algorithm find a great place to optimize its value (a place that has larger gradient) from a rather unstable early stage. Thus, it ends up optimizing faster than the other two algorithms from iteration 6.

## 2. Optimization of Travel Routes for South Korea Cities

(a)

The distance matrix is below (in km), with row and column names equal to Incheon, Seoul, Busan, Daegu, Daejeon, Gwanju, Suwon-si, Ulsan, Jeonju, Cheongju-si, Changwon, Jeju-si, Chuncheon, Hongsung, Muan.

$$\begin{bmatrix} 0 & 27 & 335 & 244 & 141 & 257 & 33 & 316 & 186 & 115 & 304 & 439 & 102 & 95 & 275 \\ 27 & 0 & 330 & 237 & 144 & 268 & 31 & 307 & 195 & 113 & 301 & 453 & 75 & 111 & 290 \\ 335 & 330 & 0 & 95 & 199 & 193 & 304 & 54 & 189 & 221 & 35 & 291 & 330 & 271 & 233 \\ 244 & 237 & 95 & 0 & 117 & 171 & 212 & 75 & 130 & 130 & 72 & 324 & 236 & 191 & 215 \\ 141 & 144 & 199 & 117 & 0 & 137 & 114 & 192 & 61 & 36 & 167 & 323 & 175 & 74 & 171 \\ 257 & 268 & 193 & 171 & 137 & 0 & 238 & 222 & 77 & 173 & 161 & 186 & 311 & 162 & 44 \\ 33 & 31 & 304 & 212 & 114 & 238 & 0 & 284 & 164 & 84 & 274 & 423 & 91 & 83 & 260 \\ 316 & 307 & 54 & 75 & 192 & 222 & 284 & 0 & 198 & 205 & 67 & 341 & 296 & 266 & 265 \\ 186 & 195 & 189 & 130 & 61 & 77 & 164 & 198 & 0 & 96 & 154 & 263 & 234 & 97 & 111 \\ 115 & 113 & 221 & 130 & 36 & 173 & 84 & 205 & 96 & 0 & 190 & 359 & 139 & 74 & 205 \\ 304 & 301 & 35 & 72 & 167 & 161 & 274 & 67 & 154 & 190 & 0 & 275 & 306 & 237 & 202 \\ 439 & 453 & 291 & 324 & 323 & 186 & 423 & 341 & 263 & 359 & 275 & 0 & 498 & 344 & 165 \\ 102 & 75 & 330 & 236 & 175 & 311 & 91 & 296 & 234 & 139 & 306 & 498 & 0 & 170 & 340 \\ 95 & 111 & 271 & 191 & 74 & 162 & 83 & 266 & 97 & 74 & 237 & 344 & 170 & 0 & 180 \\ 275 & 290 & 233 & 215 & 171 & 44 & 260 & 265 & 111 & 205 & 202 & 165 & 340 & 180 & 0 \end{bmatrix}$$

(b)

It's a classic traveling salesman problem, aka **TSP**.

Since the first and the last city will be **Incheon**. Assuming that you can't travel back to the city that you've visited, every time you want to travel to another city, you only have $Choices_{t-1} = (Choices_t - 1)$ choices left to choose from.
Thus, the evaluation of objective function required in exhaustive enumeration is

$$\frac{14 \times 13 \times 12 \times \cdots \times 2 \times 1}{2} = \frac{14!}{2} \text{ times}$$

(Since traveling forward and backward are of the same distances, we can divide the iteration times by 2.)

(c) Random walk algorithm

**Stoping Criteria**

- # of iteration > 100

Final result:

```
Iteration 5
Current solution is [6, 13, 11, 2, 7, 3, 10, 8, 14, 5, 4, 9, 12, 1], current length is 1675
Best solution is [6, 13, 11, 2, 7, 3, 10, 8, 14, 5, 4, 9, 12, 1], best length is 1675
==================================================
Iteration 6
Current solution is [6, 13, 8, 2, 7, 3, 10, 11, 14, 5, 4, 9, 12, 1], current length is 1501
Best solution is [6, 13, 8, 2, 7, 3, 10, 11, 14, 5, 4, 9, 12, 1], best length is 1501
==================================================
Iteration 7
Current solution is [6, 13, 8, 3, 7, 2, 10, 11, 14, 5, 4, 9, 12, 1], current length is 1405
Best solution is [6, 13, 8, 3, 7, 2, 10, 11, 14, 5, 4, 9, 12, 1], best length is 1405
==================================================
Iteration 8
Current solution is [6, 13, 4, 3, 7, 2, 10, 11, 14, 5, 8, 9, 12, 1], current length is 1369
Best solution is [6, 13, 4, 3, 7, 2, 10, 11, 14, 5, 8, 9, 12, 1], best length is 1369
==================================================
Iteration 9
Current solution is [6, 9, 4, 3, 7, 2, 10, 11, 14, 5, 8, 13, 12, 1], current length is 1364
Best solution is [6, 9, 4, 3, 7, 2, 10, 11, 14, 5, 8, 13, 12, 1], best length is 1364
==================================================
Iteration 10
Current solution is [12, 9, 4, 3, 7, 2, 10, 11, 14, 5, 8, 13, 6, 1], current length is 1357
Best solution is [12, 9, 4, 3, 7, 2, 10, 11, 14, 5, 8, 13, 6, 1], best length is 1357
==================================================
Iteration 11
Current solution is [6, 9, 4, 3, 7, 2, 10, 11, 14, 5, 8, 13, 12, 1], current length is 1364
Best solution is [12, 9, 4, 3, 7, 2, 10, 11, 14, 5, 8, 13, 6, 1], best length is 1357
==================================================
Iteration 12
Current solution is [12, 9, 4, 3, 7, 2, 10, 11, 14, 5, 8, 13, 6, 1], current length is 1357
Best solution is [12, 9, 4, 3, 7, 2, 10, 11, 14, 5, 8, 13, 6, 1], best length is 1357
==================================================
Iteration 13
Current solution is [6, 9, 4, 3, 7, 2, 10, 11, 14, 5, 8, 13, 12, 1], current length is 1364
Best solution is [12, 9, 4, 3, 7, 2, 10, 11, 14, 5, 8, 13, 6, 1], best length is 1357
==================================================
Iteration 14
Current solution is [12, 9, 4, 3, 7, 2, 10, 11, 14, 5, 8, 13, 6, 1], current length is 1357
Best solution is [12, 9, 4, 3, 7, 2, 10, 11, 14, 5, 8, 13, 6, 1], best length is 1357
```

(d) Hill climbing algorithm

### Stoping Criteria

- # of iteration > 100
- When the length of the best neighbor is greater than current solution

Final result:

```
Iteration 1,  [2, 10, 3, 13, 5, 4, 9, 1, 7, 14, 11, 8, 6, 12] , best length is 2438
Iteration 2,  [2, 10, 3, 7, 5, 4, 9, 1, 13, 14, 11, 8, 6, 12] , best length is 2101
Iteration 3,  [3, 10, 2, 7, 5, 4, 9, 1, 13, 14, 11, 8, 6, 12] , best length is 1989
Iteration 4,  [3, 10, 2, 7, 8, 4, 9, 1, 13, 14, 11, 5, 6, 12] , best length is 1886
Iteration 5,  [3, 10, 2, 7, 8, 4, 9, 12, 13, 14, 11, 5, 6, 1] , best length is 1836
Iteration 6,  [8, 10, 2, 7, 3, 4, 9, 12, 13, 14, 11, 5, 6, 1] , best length is 1793
Iteration 7,  [12, 10, 2, 7, 3, 4, 9, 8, 13, 14, 11, 5, 6, 1] , best length is 1745
Iteration 8,  [12, 10, 2, 7, 3, 4, 9, 8, 5, 14, 11, 13, 6, 1] , best length is 1592
Iteration 9,  [12, 10, 2, 7, 3, 9, 4, 8, 5, 14, 11, 13, 6, 1] , best length is 1570
Iteration 10,  [12, 10, 2, 7, 3, 9, 4, 8, 5, 11, 14, 13, 6, 1] , best length is 1548
Iteration 11,  [12, 7, 2, 10, 3, 9, 4, 8, 5, 11, 14, 13, 6, 1] , best length is 1535
Iteration 12,  [12, 7, 2, 10, 3, 9, 4, 8, 5, 11, 14, 13, 6, 1] , best length is 1535
Iteration 13,  [12, 7, 2, 10, 3, 9, 4, 8, 5, 11, 14, 13, 6, 1] , best length is 1535
Iteration 14,  [12, 7, 2, 10, 3, 9, 4, 8, 5, 11, 14, 13, 6, 1] , best length is 1535
Iteration 15,  [12, 7, 2, 10, 3, 9, 4, 8, 5, 11, 14, 13, 6, 1] , best length is 1535
Iteration 16,  [12, 7, 2, 10, 3, 9, 4, 8, 5, 11, 14, 13, 6, 1] , best length is 1535
Iteration 17,  [12, 7, 2, 10, 3, 9, 4, 8, 5, 11, 14, 13, 6, 1] , best length is 1535
Iteration 18,  [12, 7, 2, 10, 3, 9, 4, 8, 5, 11, 14, 13, 6, 1] , best length is 1535
```

(e) Tabu search algorithm

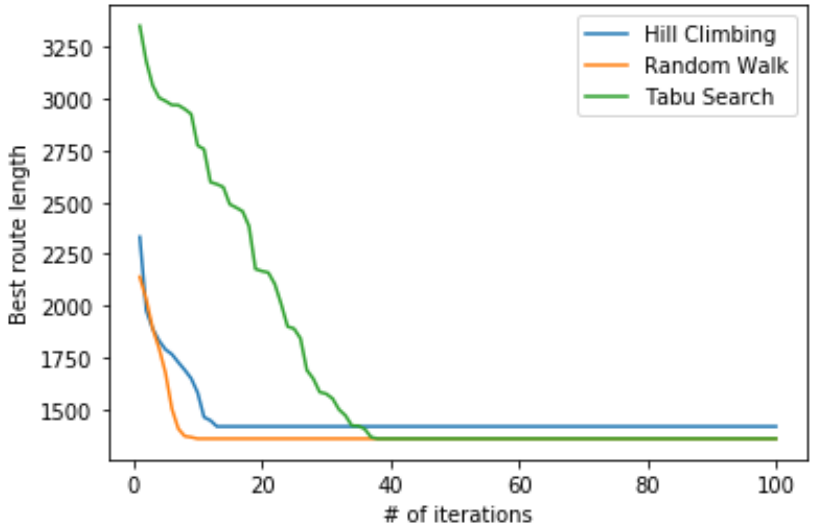
### Stoping Criteria

- # of iteration > 100

Final result:

```
Iteration 18,  [8, 7, 11, 6, 4, 3, 2, 10, 5, 14, 13, 9, 12, 1] , best length is 2209
Iteration 19,  [8, 7, 11, 6, 13, 3, 2, 10, 5, 14, 4, 9, 12, 1] , best length is 2205
Iteration 20,  [8, 7, 11, 9, 13, 3, 2, 10, 5, 14, 4, 6, 12, 1] , best length is 2162
Iteration 21,  [8, 7, 11, 9, 4, 3, 2, 10, 5, 14, 13, 6, 12, 1] , best length is 2028
Iteration 22,  [8, 7, 11, 14, 4, 3, 2, 10, 5, 9, 13, 6, 12, 1] , best length is 1992
Iteration 23,  [8, 7, 11, 14, 4, 3, 2, 10, 5, 13, 9, 6, 12, 1] , best length is 1982
Iteration 24,  [8, 7, 11, 14, 5, 3, 2, 10, 4, 13, 9, 6, 12, 1] , best length is 1827
Iteration 25,  [8, 7, 11, 14, 5, 3, 2, 10, 4, 9, 13, 6, 12, 1] , best length is 1788
Iteration 26,  [8, 7, 11, 14, 5, 10, 2, 3, 4, 9, 13, 6, 12, 1] , best length is 1728
Iteration 27,  [8, 7, 5, 14, 11, 10, 2, 3, 4, 9, 13, 6, 12, 1] , best length is 1723
Iteration 28,  [8, 7, 5, 14, 11, 2, 10, 3, 4, 9, 13, 6, 12, 1] , best length is 1716
Iteration 29,  [8, 4, 5, 14, 11, 2, 10, 3, 7, 9, 13, 6, 12, 1] , best length is 1621
Iteration 30,  [8, 4, 5, 14, 11, 2, 10, 7, 3, 9, 13, 6, 12, 1] , best length is 1541
Iteration 31,  [8, 4, 5, 14, 11, 10, 2, 7, 3, 9, 13, 6, 12, 1] , best length is 1512
Iteration 32,  [9, 4, 5, 14, 11, 10, 2, 7, 3, 8, 13, 6, 12, 1] , best length is 1439
Iteration 33,  [9, 8, 5, 14, 11, 10, 2, 7, 3, 4, 13, 6, 12, 1] , best length is 1403
Iteration 34,  [13, 8, 5, 14, 11, 10, 2, 7, 3, 4, 9, 6, 12, 1] , best length is 1347
Iteration 35,  [13, 8, 5, 14, 11, 10, 2, 7, 3, 4, 9, 6, 12, 1] , best length is 1347
Iteration 36,  [13, 8, 5, 14, 11, 10, 2, 7, 3, 4, 9, 6, 12, 1] , best length is 1347
Iteration 37,  [13, 8, 5, 14, 11, 10, 2, 7, 3, 4, 9, 6, 12, 1] , best length is 1347
Iteration 38,  [13, 8, 5, 14, 11, 10, 2, 7, 3, 4, 9, 6, 12, 1] , best length is 1347
```

(f) FE plot




(g) Observation

From the FE plot, we can find that the three algorithms converge nearly to the same answers. In the early stage of tabu algorithm, the best objective value is rather unstable since we move to the best neighbor solution anyways. However, the TS algo. leads to better solution in the end since it may effectively escape from the local minima.
From the result of random walk algorithm, we can see that it jumps back and forth from two local minima and couldn't escape this loop.
From the result of hill climbing algorithm, we can see that it stucks in a local minimum and have no way to escape from it.

# 3. More on Optimization of Travel Routes for South Korea Cities.

(a) Solving TSP with simulated annealing

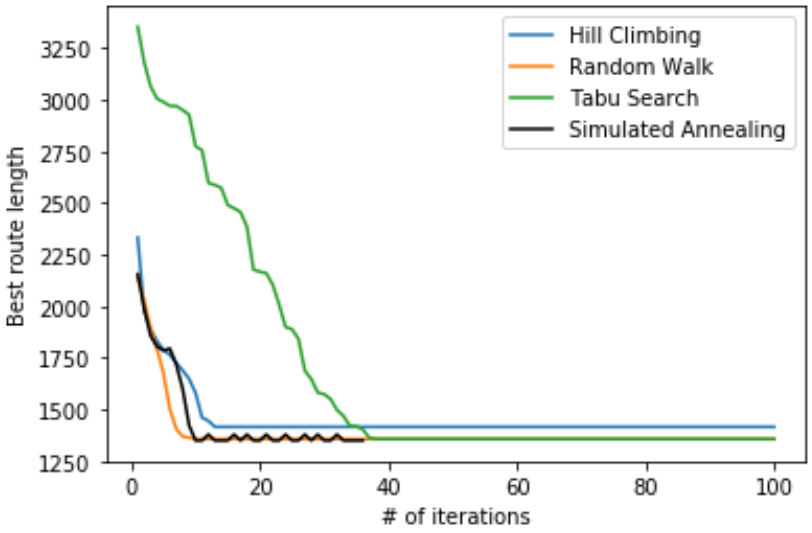
**Stoping Criteria**

- number of iteration > 100
- when the objective value do not update either for **three times** in a row.
- when the temperature is less than **1 degree**.

Final result:

```
Iteration 4 [6, 9, 3, 7, 2, 10, 11, 14, 12, 1, 4, 8, 5, 13] , best length is 1805
Iteration 5 [6, 9, 3, 7, 2, 10, 11, 14, 1, 12, 4, 8, 5, 13] , best length is 1786
Iteration 6 [1, 9, 3, 7, 2, 10, 11, 14, 6, 12, 4, 8, 5, 13] , best length is 1795
Iteration 7 [1, 12, 3, 7, 2, 10, 11, 14, 6, 9, 4, 8, 5, 13] , best length is 1717
Iteration 8 [1, 12, 3, 7, 2, 10, 11, 14, 5, 9, 4, 8, 6, 13] , best length is 1598
Iteration 9 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 6, 13] , best length is 1422
Iteration 10 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 11 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 12 [1, 12, 3, 7, 10, 2, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1379
Iteration 13 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 14 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 15 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 16 [1, 12, 3, 7, 10, 2, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1379
Iteration 17 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 18 [1, 12, 3, 7, 10, 2, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1379
Iteration 19 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 20 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 21 [1, 12, 3, 7, 10, 2, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1379
Iteration 22 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 23 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 24 [1, 12, 3, 7, 10, 2, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1379
Iteration 25 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 26 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 27 [1, 12, 3, 7, 10, 2, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1379
Iteration 28 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 29 [1, 12, 3, 7, 10, 2, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1379
Iteration 30 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 31 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 32 [1, 12, 3, 7, 10, 2, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1379
Iteration 33 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 34 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 35 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
Iteration 36 [1, 12, 3, 7, 2, 10, 11, 14, 5, 8, 4, 9, 13, 6] , best length is 1350
```

(b) FE plot - Hill climbing, random walk, simulated annealing, tabu search




(c) Observation

From the result of simulated annealing and FE plot, we can see that the SA algorithm effectively find the optimal solution in other three algorithms. I think that it may be due to the fact the the boltzmann mechanism may let it escape from local minimum in the early stage.

In coonclusion, the random walk, tabu search and simulated annealing algorithms can help us finding the optimal solution in **this** traveling salesman problem since they all have some sort of **mechanism** to help them escaping from the local minima.

I have to admit that some of the result is a little bit ***cherry-picking*** since the solution generated by each algorithm is largely affected by a randomly initialized solution in the beginning. Sometimes, when the solution is generated in a *bad* place, the algorithm won't converge to the optimal (or even sub-optimal) solution.