

**CPE341 Optimization Design and Reliablity Engineering**

**Topic :**

**`**

**Short path to travel around the airbnb hotel based in New york city using Simulated annealing**

**Created By**

**Mr. Napas Vinitnantharat**

**Present to**

**Prof Naruemon Wattana**

**Department of Computer Engineering, Faculty of Engineering**

**King Mongkut’s University of Technology Thonburi**

**Semester 2/2022**

**Chapter 1: Introduction**

**1.1 Project Description and Scope**

This project applies Simulate Annealing(SA) to solve the Traveling Saleman Problem(TSP).This Project simulate the situation of person who want to travel hotel in New York City and minimize the distance to travel all of the hotel.In this project will study applying simulated annealing to solve single objective function and compare the performance with brute force algorithm.Then we apply simulated annealing to solved the problem with multiobjective function by simulate the situation that someone want to travel 2 trip with the minimize distance and those two trip doesn’t has the same hotel to visit.

**Chapter 2: Model Formulation**

**2.2 Variable Definition**

- Path to travel all hotel

- Path to travel in the first trip

- Path to travel in the second trip

- Path to travel form hotel i to hotel j

) - Total distance to travel in path

**2.2 Objectives**

**2.2.1 Haversine Formula**

Minize the distance route to travel hotels in New york city. The total distance can be calculated by sum all of the distance between each hotel.In order to calculate the total distance between the hotel by using latitude and longitude The haversine formula have been used as shown in the following equation.

d - Distance between to hotel i and hotel j

r - Earth radius

- Destination latitude

- Starting latitude

- Destination longitude

- Starting longitude

**2.2.1 Minimize First Trip Distance**

)=

**2.2.2 Minimize Second Trip Distance**

)=

**2.2.3 Weight Sum Method**

In This study we set the weight at 0.5.So the WSM Objective is

)=+

**2.3 Decision Variable**

is set to the path to travel from hotel 1 to hotel 12.It can represent as a list of number

**2.4 Contraints**

In this problem doesn’t have any contraints

**Chapter 3: Input Data and Problem Size**

**3.1 Input Data**

In this data input we use the dataset from kaggle Airbnb Open Dataset dataset.we sample 10 of the hotel in the dataset.These are the 10 hotels host is show as the following

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| host name | borough | lat | long | price | service fee |
| Maria | Manhattan | 40.77672 | -73.95559 | 283 | 57 |
| Heather | Queens | 40.69242 | -73.79673 | 244 | 49 |
| Darkhan | Queens | 40.73455 | -73.88899 | 783 | 157 |
| Ben | Queens | 40.74422 | -73.95432 | 512 | 102 |
| Cindy | Manhattan | 40.75216 | -73.96912 | 570 | 114 |
| Aaron | Brooklyn | 40.69045 | -73.96068 | 935 | 187 |
| Kazuya | Manhattan | 40.81098 | -73.94278 | 554 | 111 |
| Haider & Galiya | Brooklyn | 40.63498 | -74.01966 | 1111 | 222 |
| Monica | Brooklyn | 40.66229 | -73.96296 | 397 | 79 |
| Roxanna | Queens | 40.72045 | -73.85782 | 108 | 22 |

**Reference Data Source**

<https://www.kaggle.com/datasets/arianazmoudeh/airbnbopendata>

**3.2 Problem Size**

for a single objective has a total of 19,958,400 solutions. The calculation is calculate by (12-1)!/2 = 19,958,400 solutions

for the multi objective has a total of The calculation is calculate by ((9-1)!/2)\*((9-1)!/2) = 19,958,400 solutions

**Chapter 4: Algorithm**

**4.1 Simulated Annealing**

In This Project the Simulated Annealing is used to find the best path solution.Simulated Annealing is c computatioal method borrowing inspiration from the field of physic introsuced by.It simulate the physical process of solid annealing. This method has been one of heuristic model to avoiding local minina.The base concept of this algorithm is accept worse candidate base o the probability dependent on the temperature and the rate of change of the fitness value or cost.

P =

P - The Probability of accepting the new solution candidate

fitness(s) - In This problem we use the total distance of the path so if delta fitness < 0 mean the the new route is shorter than the previous route.

T - Tempature which is use in the control parameter

**4.1.1 Algorithm**

While Current\_Temp <= Final\_temp:

for i until i = iterationpertemp:

find neighbor

calculate the neighbor fitness value

if fitness(neighbor) < fitness(S) # new solution is better

set neighbor to be the new solution

else

random number r in range 0 to 1

if r <

set neighbor to be the new solution

else

do nothing

update the tempurature T = T\*α

**4.2. Single Objective**

We implement a simple shortest path to travel all hotel solving algorithm using Simulated Annealing

**Parameters:**

1. IntialTemp = 200

2. IteratePerTemp =200

3. FinalTemp = 0.01

4. alpha = 0.98

**Path Encoding**

The path to travel hotel is encode as the following

1. The path is represent as a list, each number in the list is represent the index of the hotel.The length of the list is equal to the number of hotel in to visit.In this case it 12 hotels.
2. The sequence of list is represent as the order that person going to visit each hotel.

**Array P Encoding**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **6** | **2** | **4** | **3** | **1** | **10** | **9** | **7** | **8** | **5** | **11** | **12** |

**Figure 4.1** Path encoding and Decoding of single objective

**4.2.1 Find Neighbor**

In order to find the Neighbor solution of the Simulate Annealing, we can apply the swap node technique as show in the following

**Step 1.** random the number in range equal to the number of hotel in this case we have 12 hotels

**Step 2.** Swap the number base on those index as show in the **Figure 4.2**

**Current Solution**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **6** | **2** | **4** | **3** | **12** | **10** | **9** | **7** | **8** | **5** | **1** | **11** |

**Neighbor Solution**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **6** | **2** | **7** | **3** | **12** | **10** | **9** | **4** | **8** | **5** | **1** | **11** |

**4.2.1 Fitness value**

**4.3. Multi Objective**

We implement a simple shortest path to travel all hotel solving algorithm using Simulated Annealing

**Parameters:**

1. IntialTemp = 200

2. IteratePerTemp =200

3. FinalTemp = 0.01

4. alpha = 0.98

**Path Encoding**

The path to travel hotel is encode as the following

1. The path is represent as a list, each number in the list is represent the index of the hotel.The length of the list is equal to the number of hotel in to visit.In this case it 10 hotels.
2. The sequence of list is represent as the order that person going to visit each hotel.

**Array P Encoding**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **4** | **3** | **8** | **10** | **9** | **7** | **8** | **5** |

**Figure :** Path encoding and Decoding of single objective

**4.3.1 Algorithm Modify**

for the multi objective function we have to modify the algorithm to support the problem

While Current\_Temp <= Final\_temp:

for i until i = iterationpertemp:

find neighbor

calculate the neighbor fitness value

if fitness(neighbor) < fitness(S) # new solution is better

set neighbor to be the new solution

else

random number r in range 0 to 1

if r <

set neighbor to be the new solution

else

do nothing

update the tempurature T = T\*α

**4.3.2 Find Neighbor**

For the nulti objective part.In order to find the Neighbor solution of the Simulate Annealing we can use swap node technique similar to the single objective.

**Current Solution**

**First Trip**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **6** | **2** | **4** | **3** | **1** | **11** | **13** | **16** | **18** |

**Second Trip**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **10** | **9** | **7** | **14** | **5** | **12** | **8** | **17** |

**Neighbor Solution**

**First Trip**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **6** | **1** | **4** | **3** | **2** | **11** | **13** | **16** | **18** |

**Second Trip**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **14** | **9** | **7** | **10** | **5** | **12** |  |  |  |

**Figure 4.5**

**4.3.3 Fitness value**

**Chapter 5: ​Results and Discussion**

**5.1. Single Objective Problem**

**5.2. Multi Objective Problem**

**Appendix**