

Name – Rohila Gudipati

UIN – 01058204

## Assignment 2

### 1. Optimization of Ackley's Function

#### Task 1

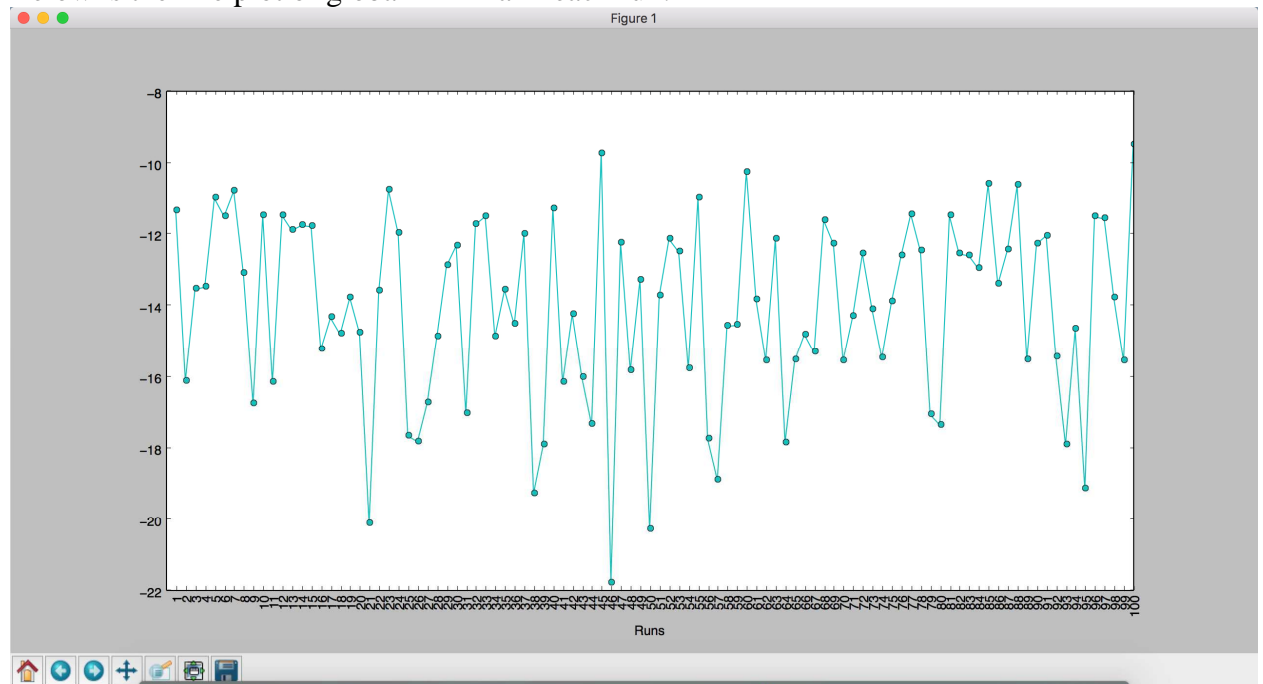
##### Compile and execution of the code:

1. The system needs to have python version 2.7 or up
2. Download all the files attached in the mail into a folder.
3. The file hill\_climb\_search.py contains hill climbing search to find global minimum of ackley's function. To run this file, type the following command in command prompt:  
python hill\_climb\_search.py
4. After a few seconds, a chart will open containing 100 runs and global minima obtained for ackley's function in each run.

#### Output:

```
assignment2 — python hill_climb_search.py — 80x6
[Rohilas-MacBook-Pro:assignment2 rohila$ python hill_climb_search.py
]
```

Below is the line plot of global minima in each run:



## Task 2

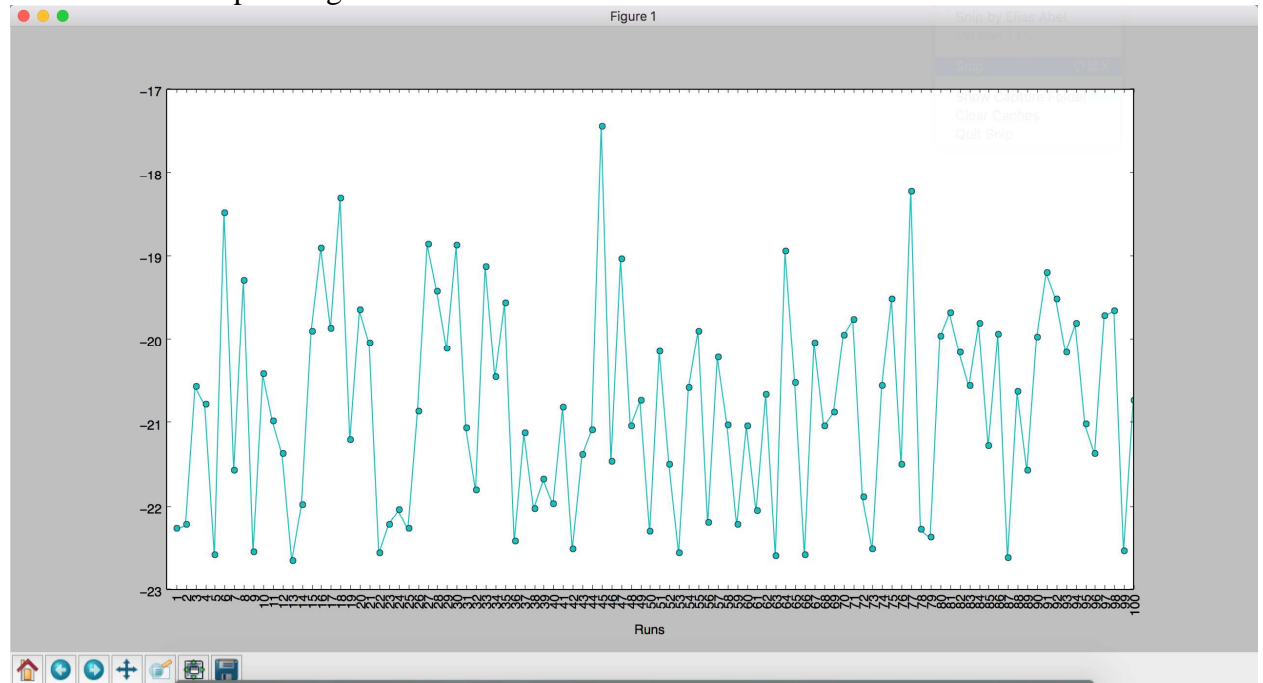
### Compile and execution of the code:

1. The system needs to have python version 2.7 or up
2. Download all the files attached in the mail into a folder.
3. The file differential\_evolution.py contains differential evolution algorithm to find global minimum of ackley's function. To run this file, type the following command in command prompt:  
python differential\_evolution.py
4. After a few seconds, a chart will open containing 100 runs and global minima obtained for ackley's function in each run.

### Output:

```
assignment2 — python differential_evolution.py — 80×6
[Rohilas-MacBook-Pro:assignment2 rohila$ python differential_evolution.py
]
```

Below is the line plot of global minima in each run:



## Task 3

Comparison of Hill climbing search and differential evolution algorithm to find global minima of Ackley's function

The problem with hill climbing search is it stops optimization at local minima.

The global minima for ackley's function is around -22 at the point(0,0).  
Differential evolution algorithm is more efficient than hill climbing search as the global minima in each run is nearly -22 in the case of differential evolution algorithm.  
In case of hill climbing search, minima reaches -22 only once in a 100 runs.

## 2. Travelling Salesman Problem

### Task 1

#### Algorithm:

This program is implemented using genetic algorithm where the following steps take place.

1. The first step is selection of a population
2. Second step is crossover of 2 parents to generate children
3. Third step is mutation of the each element in the population

In implementing this algorithm, I have used cyclic cross over method as discussed in the class. For mutation, I have randomly selected 2 cities in the each path and based on mutation probability I have swapped the cities.

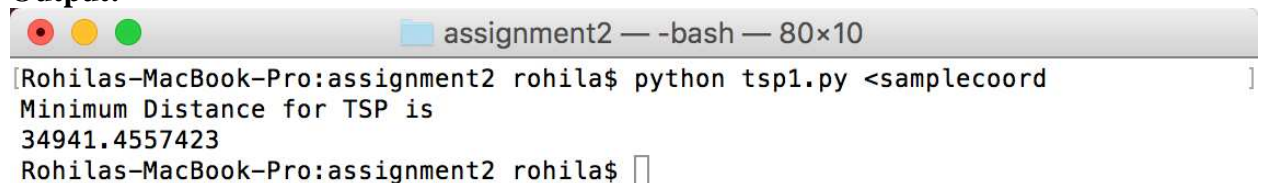
#### Compile and execution of the code:

1. The system needs to have python version 2.7 or up
2. Download all the files attached in the mail into a folder.
3. The file tsp1.py contains a genetic algorithm to shortest distance between given set of cities and back to the city from where we started. To run this file, type the following command in command prompt one after other:

```
python tsp1.py <samplecoord  
python tsp1.py <qatar  
python tsp1.py <uruguay.txt
```

4. After a few seconds, The minimum distance is displayed.

#### Output:



```
assignment2 — -bash — 80x10  
[Rohilas-MacBook-Pro:assignment2 rohila$ python tsp1.py <samplecoord  
Minimum Distance for TSP is  
34941.4557423  
Rohilas-MacBook-Pro:assignment2 rohila$ ]
```

```
assignment2 — -bash — 80x14
[Rohilas-MacBook-Pro:assignment2 rohila$ python tsp1.py <qatar
Minimum Distance for TSP is
41556.7792277
Rohilas-MacBook-Pro:assignment2 rohila$ ]
```

## Task 2

### Algorithm:

This program is implemented using genetic algorithm with different crossover function than described in Task 1.

In implementing this algorithm, I have used PMX cross over method as discussed in the class. For mutation, I have randomly selected 2 cities in the each path and based on mutation probability I have swapped the cities.

### Compile and execution of the code:

1. The system needs to have python version 2.7 or up
2. Download all the files attached in the mail into a folder.
3. The file tsp2.py contains a genetic algorithm to shortest distance between given set of cities and back to the city from where we started. To run this file, type the following command in command prompt one after other:

```
python tsp2.py <samplecoord
python tsp2.py <qatar
python tsp2.py <uruguay.txt
```

4. After a few seconds, The minimum distance is displayed.

### Output:

```
assignment2 — -bash — 80x8
[Rohilas-MacBook-Pro:assignment2 rohila$ python tsp2.py <samplecoord
Minimum distance for TSP is
28559.17697
Rohilas-MacBook-Pro:assignment2 rohila$ ]
```

```
assignment2 — -bash — 80×12
[Rohilas-MacBook-Pro:assignment2 rohila$ python tsp2.py <qatar
Minimum distance for TSP is
35452.420004
Rohilas-MacBook-Pro:assignment2 rohila$ ]
```

### Task 3

Analysis and comparison of 2 genetic algorithms to solve TSP

The output of genetic algorithm depends on the process involved in crossover and mutation.

In the first implementation, I have used cyclic cross over method which gives fairly good results.

In second implementation I have used PMX (Partially matched crossover), this algorithm gives better results than cyclic cross over method.

In both the cases, selection is done using roulette wheel method and mutation method is same which is, swapping of 2 cities in each element of population, depending upon mutation probability.

Time complexity of PMX is more than cyclic crossover but it is more efficient in terms of results. PMX crossover has higher time complexity than cyclic crossover method.

The algorithm directly depends on population size, cross over probability, mutation probability. With increase in these 3 values, the accuracy increases.

The above two genetic algorithms to solve TSP are able to give results for the sample 29 cities and Qatar map (194 cities), but are taking huge amount of time for the other map (uruguay). The map of Uruguay consists 734 cities and due to this time taken by the program is increased exponentially.