

GlobeTrotter: An Optimal Travel Sequence generator.

Submitted in partial fulfilment of the requirements of the
degree of
Bachelor of Engineering

By

Saurabh Adhau
Roll No: 201414001

Taufiq Monghal
Roll No:2014140029

Shubhangi Shinde
Roll No.2014140053

Supervisor (s):
Prof. Varsha Hole



Information Technology Department
Sardar Patel Institute of Technology
2017-18

CERTIFICATE

This is to certify that the project entitled GlobeTrotter: An Optimal Travel Sequence Generator is a bonafide work of Saurabh Adhau (Roll No. 2014140001), Taufiq Monghal (Roll No. 2014140029) and Shubhangi Shinde (Roll No.2014140053) submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of Undergraduate in Bachelor of Engineering –Information Technology

(Name and sign)
Supervisor/Guide

(Name and sign)
Co-Supervisor/Guide

(Name and sign)
Department

(Name and sign) Head of
Principal

Project Report Approval for Bachelor of Engineering

Project report entitled “Globetrotter: An Optimal Travel Sequence generator” by Saurabh Adhau, Taufiq Monghal and Shubhangi Shinde is approved for the degree of Information Technology.

Examiners

1.-----

2.-----

Date:

Place:

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Saurabh Adhau (2014140001)

Taufiq Monghal (2014140029)

Shubhangi Shinde (2014140053)

Date:

Acknowledgements

We feel great pleasure in presenting the stage one report of our project titled ‘Globetrotter: An Optimal Travel Sequence generator’. We have channelized our best efforts towards a systematic approach to the project, keeping in mind the aim we need to achieve.

We are highly grateful to our project guide Mrs. Varsha Hole, Department of Information Technology, Sardar Patel Institute of Technology (SPIT) for constant encouragement, effort and guidance. She has always been involved in discussing our topic at each phase to make sure that our approach was designed and carried out in an appropriate manner and that our conclusions were appropriate, given our results.

Saurabh Adhau
Shubhangi Shinde
Taufiq Monghal

Abstract

The ready-made fixed packages provided by travel firms has always been a setback when tourists sought to find for a travel experience that are customized to their own needs. The situation then demands for a recommendation system that provides with a tailor made optimized travel sequence for the travellers. For optimizing this orienteering problem we make use of Evolutionary algorithms that consists of two popular meta-heuristics techniques; Ant Colony Optimization (ACO) and Genetic Algorithm (GA). In this work, we try to apply both techniques to create optimal travel route and compare them to determine the best one. The comparison is then accomplished to state the better one that is used in the recommendation system.

Contents

1. Introduction	8
1.1. Problem Statement	9
1.2. Literature Survey	10
1.3. Scope	10
1.4. Assumptions	11
1.5. Constraints	11
2. Analysis	12
2.1. Architecture Diagram	12
2.2. System Flow Diagram	13
2.3. Use Case Diagram	14
2.4. Sequence Diagram	14
2.5. Functional Requirements	15
2.6. Non-functional requirements	16
3. Design	17
3.1 Implementation	17
3.2 Technologies Used	34
3.3 Dataset	35
3.4 Goal Justification	36
4 References	37
4.1 Books	37
4.2 Journal Paper	37
4.3 Websites	37

1.Introduction

1.1. Problem Statement

In the current scenario, fixed travel itineraries are provided to the customers by the travel agencies with less or no consent of the user, while deciding the complete itinerary for any tour. Such itineraries directly affect the user experience since there is no dynamicity in accordance to factors like the time of the year or weather conditions. This problem persists with most of the current travel agencies who resort to traditional methods which needs to be addressed. The above problem will be eliminated by *GlobeTrotter* which provides optimal travel sequence which includes a recommendation system to recommend popular places of a specific area, that the user may want to visit. This will eventually make the customer more satisfied as compared to the previous scenario. Later, the sequence points of visit specified by the user will undergo optimization to generate optimal itineraries.

1.2. Literature Survey

1.2.1. Aris Anagnostopoulos, et al. “Tour recommendation for groups”, Received: 13 January 2016 / Accepted: 2 September 2016:

In this paper, the authors have formalized the group tour recommendation problem as a generalization of the orienteering problem. They consider several optimization objectives that provide a mathematical formulation to quantify the satisfaction of a “group of people,” which depends on the satisfaction of its members.

1.2.2. Lin Yuanyuan, Zhang Jing, “An Application of Ant Colony Optimization Algorithm in TSP”, Fifth International Conference on Intelligent Networks and Intelligent Systems 2012:

This paper has made a detailed analysis of the Ant Colony Algorithm and its parameters, and have integrated the algorithm with the TSP(Travelling Salesman problem). They have constructed the mathematical model of ant colony optimization algorithm proposed on the basis of the analysis to solve the TSP problems

1.2.3. Fabiana Lorenzi, et al. “Personal Tour: a recommender system for travel packages”, ACM International Conferences on Web Intelligence and Intelligent Agent Technology, 2011:

This paper describes the Personal Tour recommender system that helps customers to find best travel packages according to their preferences. Personal Tour is based on the paradigm of a customer recommendation request is divided into partial recommendations that are handled by different agents.

1.2.4. Gohar Vahdati, et al. “A New Approach to Solve Traveling Salesman Problem Using Genetic Algorithm Based on Heuristic Crossover and Mutation Operator” , International Conference of Soft Computing and Pattern Recognition,2009

This paper proposes a new solution for Traveling Salesman Problem (TSP), using genetic algorithm. A heuristic crossover and mutation operation have been proposed to prevent premature convergence.

1.2.5. Gao Shang, et al. "Solving Traveling Salesman Problem by Ant Colony Optimization Algorithm with Association Rule", Third International Conference on Natural Computation (ICNC 2007).

This paper proposes a new algorithm which integrates ACO and AR(Association Rule) to solve TSP problems since Association is the key in knowledge in data mining for finding the best data sequence

1.3. Scope

- 1.3.1.** Comparison between the two optimization algorithms Ant Colony Optimization (ACO) and Genetic Algorithm (GA) using explicit data.
- 1.3.2.** A recommendation system to recommend a list of popular areas.
 - 1.3.2.1.** The area that the user wants to travel may be user's location at that instant or it may a manual input location by the user.
 - 1.3.2.2.** This location will be used by the recommendation system to recommend a list of popular places to the user dynamically.
 - 1.3.2.3.** Various travel based constraints such as time and budget will be taken as input after the user selects from certain recommendations.
- 1.3.3.** Deploying the most suitable Optimisation Algorithm of the two to generate a travel sequence i.e an itinerary.

1.4. Assumptions

- 1.4.1.** Generalized data has been used for comparing the two algorithms thus results may vary with stricter data.
- 1.4.2.** The preference to user behaviour i.e popularity, distance,time and budget is equally distributed when creating the travel sequence.
- 1.4.3.** The preference to user behaviour i.e popularity, distance,time and budget is equally distributed when creating the travel sequence.

1.5. Constraints

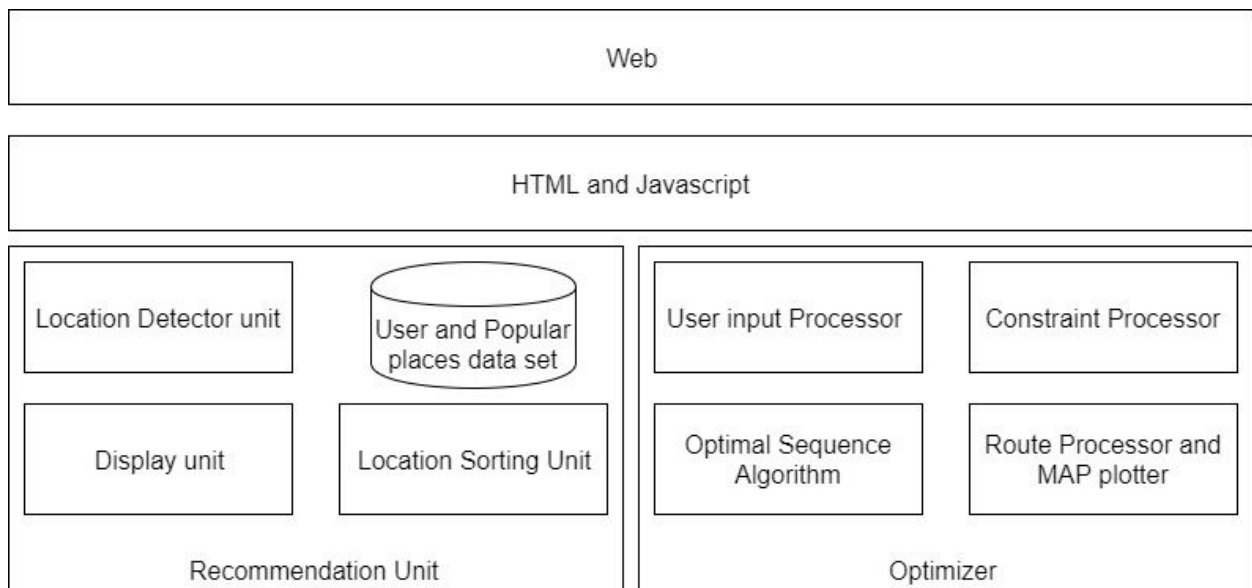
- 1.5.1.** The places available for recommendation are only restricted to certain cities in India.
- 1.5.2.** Since preferences are equalized one specific attribute will not procure heavy weightage in optimisation.
- 1.5.3.** The base used will be a web portal hence quality might deteriorate if used on mobile handsets.

2. Analysis

2.1. Architecture Diagram

The system consists of following major components namely:

1. A recommendation system to recommend a list of popular areas.
2. Optimizer to generate an optimal travel sequence.



2.1.1. Location Detector Unit: Takes explicit city input location or detects it using GPS.

2.1.2. Location Sorting Unit: Based on dataset, provides a list of recommendations, sorted based on popularity

2.1.3. Display Unit: Presents the sorted list to user.

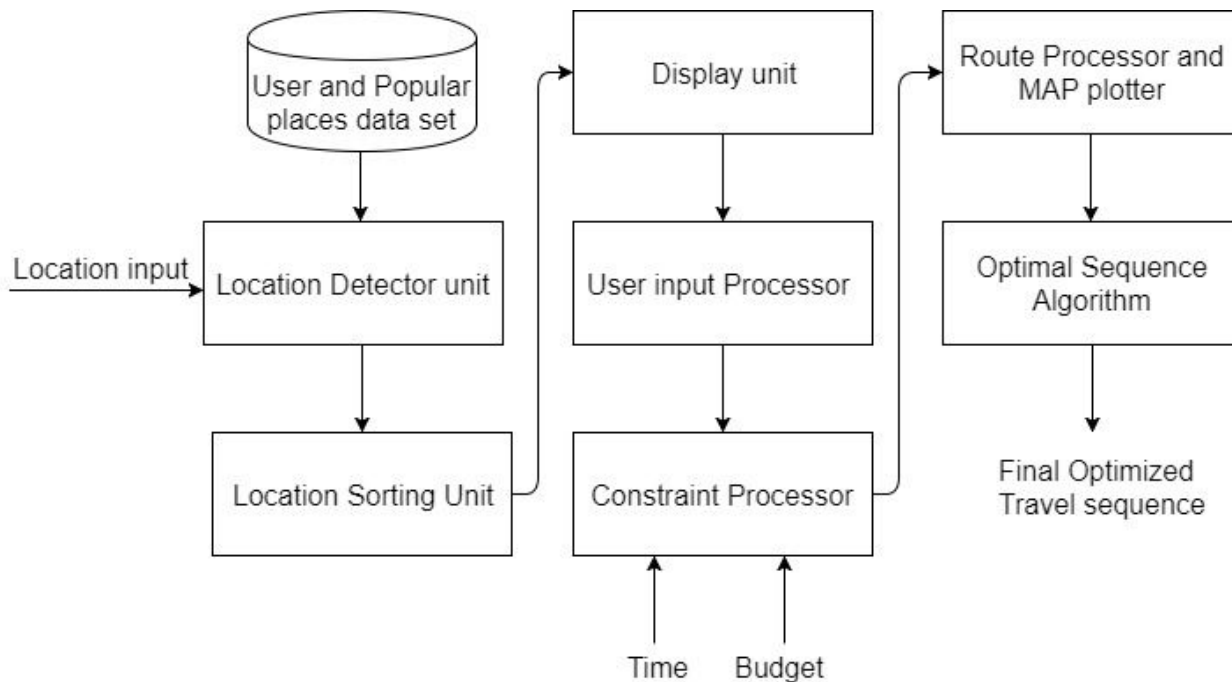
2.1.4. User input processor: User selected sub-list is processed.

2.1.5. Time and Budget constraint processor: The time and budget constraint is processed. The cost constraint shall be considered by using per km cab fare and time constraint shall be predicted using distance and real time traffic conditions. Places are dropped if needed, as per computations in this block.

2.1.6. Route Processor and MAP plotter: Route points are processed and MAP is configured to display results.

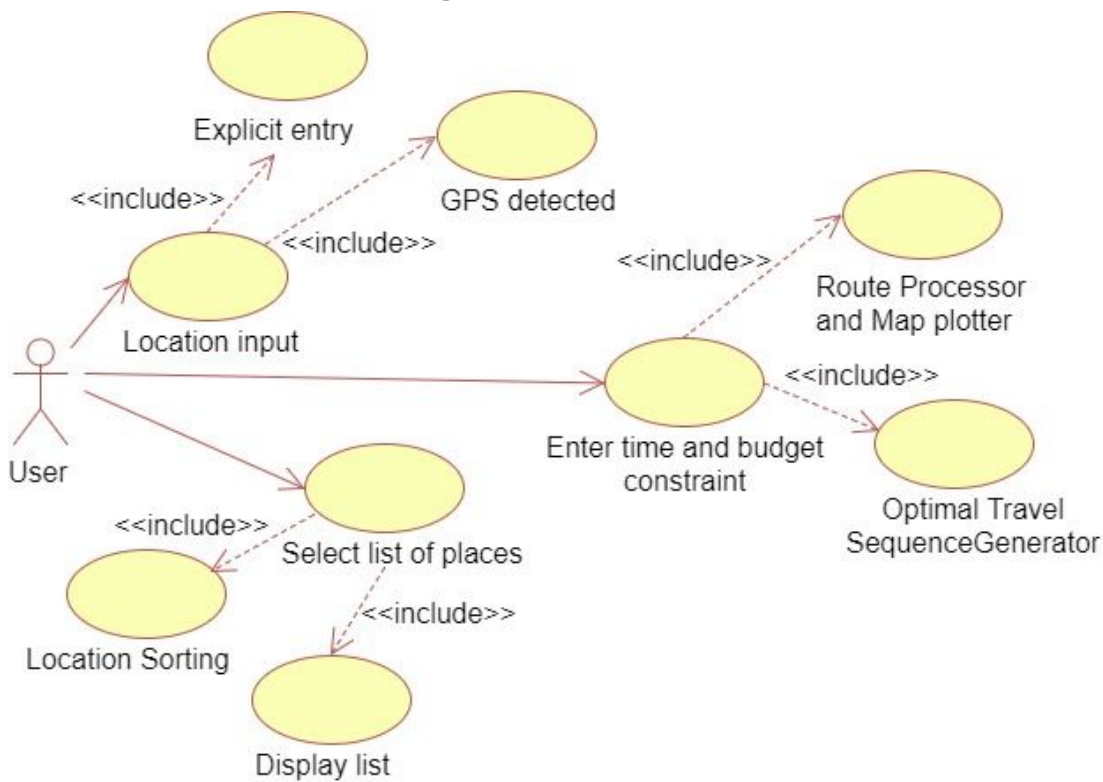
2.1.7. Optimal Sequence Algorithm: The relevant heuristic is then used to calculate optimal sequence for the user.

2.2. System Flow Diagram

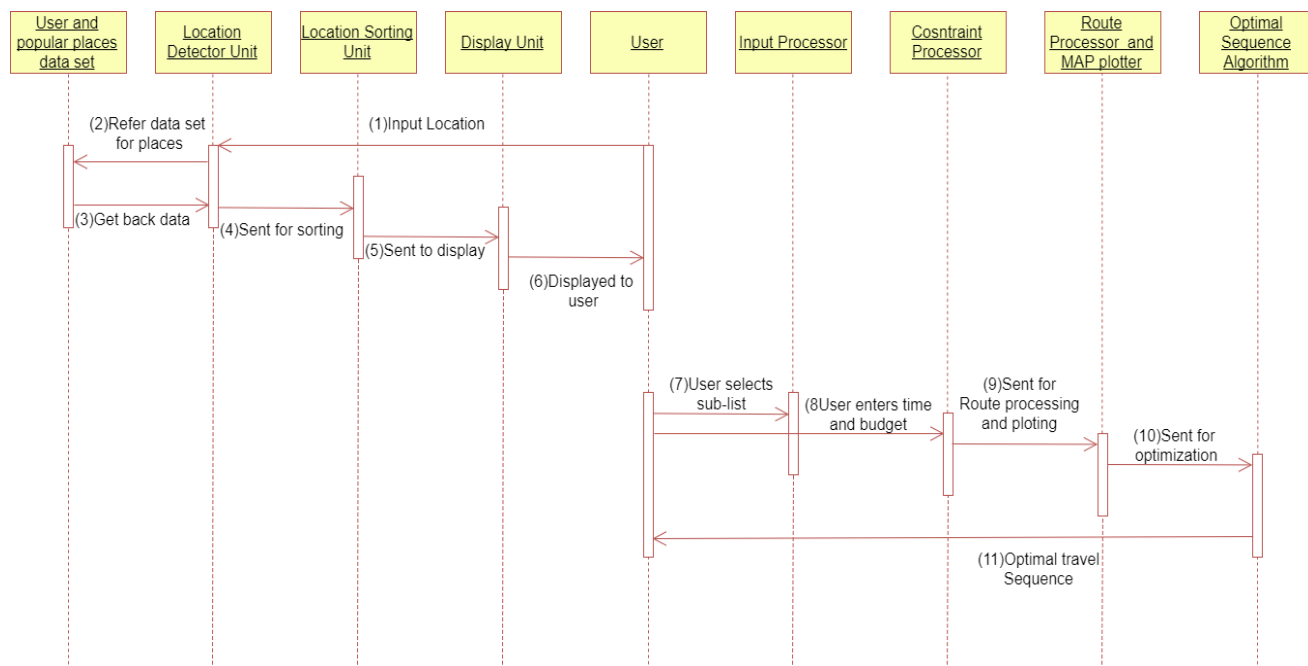


Firstly, the user enters location input. The Location Detector Unit allows the user to enter explicit city input location or detects it using GPS. Referring the user and popular places data set, the Location Sorting Unit provides a list of recommendations, based on popularity. This list is sent to the display unit for displaying it to the user. From the displayed list, the user then selects a sublist of places that he/she wants to visit. This list is sent to the user input processor. User then inputs the time and budget constraint into the Constraint Processor which is then used by the Route Processor and MAP Plotter to revise the routes and configure the map. The route points are given to the Optimal Sequence Algorithm to finally generate the optimized route.

2.3. Use Case Diagram



2.4. Sequence Diagram



2.5. Functional Requirements

2.5.1. Location Detector Unit

- **Input:**
GPS or manual input
- **Output:**
Places in the specified location
- **Description:**
User can either provide the location manually or it can be detected with the help of GPS and then it will find places in the specified location

2.5.2. Location Sorting Unit

- **Input:**
Places in the specified location
- **Output:**
Places will be sorted based on their popularity
- **Description:**
All the places in the specified location would be found and then based on the popularity all the places will be sorted in descending order.

2.5.3. Display Unit

- **Input:**
Sorted list of places
- **Output:**
Displaying it to user
- **Description:**
List of sorted places will be displayed to the user

2.5.4. Constraint Processor

- **Input:**
Time and Budget
- **Output:**
Refined Places
- **Description:**
Along with popularity, time and budget of the user would be taken into consideration and the places would be refined.

2.5.5. Route Processor

- **Input:**
Refined Places
- **Output:**
Shortest routes between all places will be found
- **Description:**
All refined places will be taken as input, it will find shortest route between all those places

2.5.6. Optimal Sequence Generator

- **Input:**
Routes found in previous stage
- **Output:**
Optimised Routes
- **Description:**
Routes in the previous stage would be taken as input and those routes would then be optimised to give the final travel sequence.

2.6. Non-functional requirements

2.6.1. Active internet connection

The system would require an active internet above 1 Mbps.

2.6.2. Loading time

The loading time of the portal should be 3-5 seconds.

2.6.3. Handling time and budget constraints efficiently

The system should be able to give a realistic result considering the time-budget constraints and users input.

2.6.4. Accuracy

The system should be able to give accurate and optimized results in the form of a travel sequence. The algorithm and the input under consideration has a huge role in deciding the accuracy of the system.

2.6.5. Usability

The system should be simple to use with a good interface

3.Design

3.1 Implementation

3.1.1 Apriori algorithm flow

Users	Locations
A	Asiatic Library, Marine Drive, Worli Sea Link
B	Gateway, Marine Drive, Juhu Beach
C	Asiatic Library, Gateway, Marine Drive, Juhu Beach
D	Gateway, Juhu Beach

Item Sets	Support
Asiatic Library	2
Gateway	3
Marine Drive	3
Worli Sea Link	1
Juhu Beach	3

Item Sets	Support
{Asiatic Library, Gateway}	1
{Asiatic Library, Marine Drive}	2
{Asiatic Library, Worli Sea Link}	1
{Asiatic Library, Juhu Beach}	1
{Gateway, Marine Drive}	2
{Gateway, Worli Sea Link}	0
{Gateway, Juhu Beach}	3
{Marine Drive, Worli Sea Link}	1
{Marine Drive, Juhu Beach}	2
{Worli Sea Link, Juhu Beach}	0

Item Sets	Support
{Asiatic Library, Marine Drive, Worli Sea Link}	1
{Gateway, Marine Drive, Juhu Beach}	2
{Asiatic Library, Marine Drive, Juhu Beach}	1

Figure shows the algorithm flow of the implemented Apriori algorithm.

3.1.2 Output of Apriori

3.1.2.1 Probability calculation

1000// No of user's behaviour.

35// No. of point of interest In Mumbai.

- 1.Gateway of India,0.518
- 2.Elephanta Caves,0.497
- 3.Colaba Causeway,0.501
- 4.Juhu Beach,0.49
- 5.Victoria Terminus,0.487
- 6.Film City,0.521
- 7.Haji Ali,0.531
- 8.Banganga Tank,0.519
- 9.Dhobi Ghat,0.494
- 10.Dharavi Slum,0.502
- 11.Marine Drive,0.507
- 12.Prince of Wales Museum,0.512
- 13.Siddhivinayak Temple,0.507
- 14.Essel World,0.513
- 15.Chor Bazaar,0.485
- 16.SGNP,0.499
- 17.Kanheri Caves,0.498
- 18.Mahalakshmi Temple,0.493
- 19.Mount Mary Church,0.487
- 20.Rajabai Clock Tower,0.476
- 21.Kamla Nehru Park,0.497

- 22.Veermata Jijabai Udyan,0.51
- 23.Dr. Bhau Daji Lad Museum,0.503
- 24.Aksa Beach,0.494
- 25.St. Thomas Cathedral,0.49
- 26.Crawford Market,0.469
- 27.Powai Lake,0.508
- 28.Jehangir Gallery,0.47
- 29.St. George Fort,0.511
- 30.Khotachiwadi Village,0.504
- 31.Worli Seaface,0.495
- 32.Global Vipassana Pagoda,0.481
- 33.Nehru Science Centre,0.497
- 34.Taraporewala Aquarium,0.518
- 35.Walk of Stars,0.503

Probability As Points of Interest Increments:

- Gateway of India, Elephanta Caves,0.262
- Gateway of India, Juhu Beach,0.272
- Gateway of India, Victoria Terminus,0.264
- Gateway of India, Film City,0.276
- Gateway of India, Haji Ali,0.271
- Gateway of India, Banganga Tank,0.265
- Gateway of India, Dhobi Ghat,0.255
- Gateway of India, Dharavi Slum,0.249
- Gateway of India, Marine Drive,0.266
- Gateway of India, Prince of Wales Museum,0.257
- and so on ...

3.1.2.2 Screenshots

Command Prompt

C:\Users\Saurabh\Desktop\GlobeTrotter1>java Apriori

Input configuration: 35 items, 1000 transactions, minsup = 35.0%

Apriori algorithm has started.

Frequent 1-itemsets

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]

Frequent 2-itemsets

[1 2, 1 3, 1 4, 1 5, 1 6, 1 7, 1 8, 1 9, 1 10, 1 11, 1 12, 1 13, 1 14, 1 15, 1 16, 1 17, 1 18, 1 19, 1 20, 1 21, 1 22, 1 23, 1 24, 1 25, 1 26, 1 27, 1 28, 1 29, 1 30, 1 31, 1 32, 1 33, 1 34, 1 35, 2 3, 2 4, 2 5, 2 6, 2 7, 2 8, 2 9, 2 10, 2 11, 2 12, 2 13, 2 14, 2 15, 2 16, 2 17, 2 18, 2 19, 2 20, 2 21, 2 22, 2 23, 2 24, 2 25, 2 26, 2 27, 2 28, 2 29, 2 30, 2 31, 2 32, 2 33, 2 34, 2 35, 3 4, 3 5, 3 6, 3 7, 3 8, 3 9, 3 10, 3 11, 3 12, 3 13, 3 14, 3 15, 3 16, 3 17, 3 18, 3 19, 3 20, 3 21, 3 22, 3 23, 3 24, 3 25, 3 26, 3 27, 3 28, 3 29, 3 30, 3 31, 3 32, 3 33, 3 34, 3 35, 4 5, 4 6, 4 7, 4 8, 4 9, 4 10, 4 11, 4 12, 4 13, 4 14, 4 15, 4 16, 4 17, 4 18, 4 19, 4 20, 4 21, 4 22, 4 23, 4 24, 4 25, 4 26, 4 27, 4 28, 4 29, 4 30, 4 31, 4 32, 4 33, 4 34, 4 35, 5 6, 5 7, 5 8, 5 9, 5 10, 5 11, 5 12, 5 13, 5 14, 5 15, 5 16, 5 17, 5 18, 5 19, 5 20, 5 21, 5 22, 5 23, 5 24, 5 25, 5 26, 5 27, 5 28, 5 29, 5 30, 5 31, 5 32, 5 33, 5 34, 5 35, 6 7, 6 8, 6 9, 6 10, 6 11, 6 12, 6 13, 6 14, 6 15, 6 16, 6 17, 6 18, 6 19, 6 20, 6 21, 6 22, 6 23, 6 24, 6 25, 6 26, 6 27, 6 28, 6 29, 6 30, 6 31, 6 32, 6 33, 6 34, 6 35, 7 8, 7 9, 7 10, 7 11, 7 12, 7 13, 7 14, 7 15, 7 16, 7 17, 7 18, 7 19, 7 20, 7 21, 7 22, 7 23, 7 24, 7 25, 7 26, 7 27, 7 28, 7 29, 7 30, 7 31, 7 32, 7 33, 7 34, 7 35, 8 9, 8 10, 8 11, 8 12, 8 13, 8 14, 8 15, 8 16, 8 17, 8 18, 8 19, 8 20, 8 21, 8 22, 8 23, 8 24, 8 25, 8 26, 8 27, 8 28, 8 29, 8 30, 8 31, 8 32, 8 33, 8 34, 8 35, 9 10, 9 11, 9 12, 9 13, 9 14, 9 15, 9 16, 9 17, 9 18, 9 19, 9 20, 9 21, 9 22, 9 23, 9 24, 9 25, 9 26, 9 27, 9 28, 9 29, 9 30, 9 31, 9 32, 9 33, 9 34, 9 35, 10 11, 10 12, 10 13, 10 14, 10 15, 10 16, 10 17, 10 18, 10 19, 10 20, 10 21, 10 22, 10 23, 10 24, 10 25, 10 26, 10 27, 10 28, 10 29, 10 30, 10 31, 10 32, 10 33, 10 34, 10 35, 11 12, 11 13, 11 14, 11 15, 11 16, 11 17, 11 18, 11 19, 11 20, 11 21, 11 22, 11 23, 11 24, 11 25, 11 26, 11 27, 11 28, 11 29, 11 30, 11 31, 11 32, 11 33, 11 34, 11 35, 12 13, 12 14, 12 15, 12 16, 12 17, 12 18, 12 19, 12 20, 12 21, 12 22, 12 23, 12 24, 12 25, 12 26, 12 27, 12 28, 12 29, 12 30, 12 31, 12 32, 12 33, 12 34, 12 35, 13 14, 13 15, 13 16, 13 17, 13 18, 13 19, 13 20, 13 21, 13 22, 13 23, 13 24, 13 25, 13 26, 13 27, 13 28, 13 29, 13 30, 13 31, 13 32, 13 33, 13 34, 13 35, 14 15, 14 16, 14 17, 14 18, 14 19, 14 20, 14 21, 14 22, 14 23, 14 24, 14 25, 14 26, 14 27, 14 28, 14 29, 14 30, 14 31, 14 32, 14 33, 14 34, 14 35, 15 16, 15 17, 15 18, 15 19, 15 20, 15 21, 15 22, 15 23, 15 24, 15 25, 15 26, 15 27, 15 28, 15 29, 15 30, 15 31, 15 32, 15 33, 15 34, 15 35, 16 17, 16 18, 16 19, 16 20, 16 21, 16 22, 16 23, 16 24, 16 25, 16 26, 16 27, 16 28, 16 29, 16 30, 16 31, 16 32, 16 33, 16 34, 16 35, 17 18, 17 19, 17 20, 17 21, 17 22, 17 23, 17 24, 17 25, 17 26, 17 27, 17 28, 17 29, 17 30, 17 31, 17 32, 17 33, 17 34, 17 35, 18 19, 18 20, 18 21, 18 22, 18 23, 18 24, 18 25, 18 26, 18 27, 18 28, 18 29, 18 30, 18 31, 18 32, 18 33, 18 34, 18 35, 19 20, 19 21, 19 22, 19 23, 19 24, 19 25, 19 26, 19 27, 19 28, 19 29, 19 30, 19 31, 19 32, 19 33, 19 34, 19 35, 20 21, 20 22, 20 23, 20 24, 20 25, 20 26, 20 27, 20 28, 20 29, 20 30, 20 31, 20 32, 20 33, 20 34, 20 35, 21 22, 21 23, 21 24, 21 25, 21 26, 21 27, 21 28, 21 29, 21 30, 21 31, 21 32, 21 33, 21 34, 21 35, 22 23, 22 24, 22 25, 22 26, 22 27, 22 28, 22 29, 22 30, 22 31, 22 32, 22 33, 22 34, 22 35, 23 24, 23 25, 23 26, 23 27, 23 28, 23 29, 23 30, 23 31, 23 32, 23 33, 23 34, 23 35, 24 25, 24 26, 24 27, 24 28, 24 29, 24 30, 24 31, 24 32, 24 33, 24 34, 24 35, 25 26, 25 27, 25 28, 25 29, 25 30, 25 31, 25 32, 25 33, 25 34, 25 35, 26 27, 26 28, 26 29, 26 30, 26 31, 26 32, 26 33, 26 34, 26 35, 27 28, 27 29, 27 30, 27 31, 27 32, 27 33, 27 34, 27 35, 28 29, 28 30, 28 31, 28 32, 28 33, 28 34, 28 35, 29 30, 29 31, 29 32, 29 33, 29 34, 29 35, 30 31, 30 32, 30 33, 30 34, 30 35, 31 32, 31 33, 31 34, 31 35, 32 33, 32 34, 32 35, 33 34, 33 35, 34 35]

Frequent 3-itemsets

[1 2 3, 1 2 4, 1 2 5, 1 2 6, 1 2 7, 1 2 8, 1 2 9, 1 2 10, 1 2 11, 1 2 12, 1 2 13, 1 2 14, 1 2 15, 1 2 16, 1 2 17, 1 2 18, 1 2 19, 1 2 20, 1 2 21, 1 2 22, 1 2 23, 1 2 24, 1 2 25, 1 2 26, 1 2 27, 1 2 28, 1 2 29, 1 2 30, 1 2 31, 1 2 32, 1 2 33, 1 2 34, 1 2 35, 1 3 4, 1 3 5, 1 3 6, 1 3 7, 1 3 8, 1 3 9, 1 3 10, 1 3 11, 1 3 12, 1 3 13, 1 3 14, 1 3 15, 1 3 16, 1 3 17, 1 3 18, 1 3 19, 1 3 20, 1 3 21, 1 3 22, 1 3 23, 1 3 24, 1 3 25, 1 3 26, 1 3 27, 1 3 28, 1 3 29, 1 3 30, 1 3 31, 1 3 32, 1 3 33, 1 3 34, 1 3 35, 1 4 5, 1 4 6, 1 4 7, 1 4 8, 1 4 9, 1 4 10, 1 4 11, 1 4 12, 1 4 13, 1 4 14, 1 4 15, 1 4 16, 1 4 17, 1 4 18, 1 4 19, 1 4 20, 1 4 21, 1 4 22, 1 4 23, 1 4 24, 1 4 25, 1 4 26, 1 4 27, 1 4 28, 1 4 29, 1 4 30, 1 4 31, 1 4 32, 1 4 33, 1 4 34, 1 4 35, 1 5 6, 1 5 7, 1 5 8, 1 5 9, 1 5 10, 1 5 11, 1 5 12, 1 5 13, 1 5 14, 1 5 15, 1 5 16, 1 5 17, 1 5 18, 1 5 19, 1 5 20, 1 5 21, 1 5 22, 1 5 23, 1 5 24, 1 5 25, 1 5 26, 1 5 27, 1 5 28, 1 5 29, 1 5 30, 1 5 31, 1 5 32, 1 5 33, 1 5 34, 1 5 35, 1 6 7, 1 6 8, 1 6 9, 1 6 10, 1 6 11, 1 6 12, 1 6 13, 1 6 14, 1 6 15, 1 6 16, 1 6 17, 1 6 18, 1 6 19, 1 6 20, 1 6 21, 1 6 22, 1 6 23, 1 6 24, 1 6 25, 1 6 26, 1 6 27, 1 6 28, 1 6 29, 1 6 30, 1 6 31, 1 6 32, 1 6 33, 1 6 34, 1 6 35, 1 7 8, 1 7 9, 1 7 10, 1 7 11, 1 7 12, 1 7 13, 1 7 14, 1 7 15, 1 7 16, 1 7 17, 1 7 18, 1 7 19, 1 7 20, 1 7 21, 1 7 22, 1 7 23, 1 7 24, 1 7 25, 1 7 26, 1 7 27, 1 7 28, 1 7 29, 1 7 30, 1 7 31, 1 7 32, 1 7 33, 1 7 34, 1 7 35, 1 8 9, 1 8 10, 1 8 11, 1 8 12, 1 8 13, 1 8 14, 1 8 15, 1 8 16, 1 8 17, 1 8 18, 1 8 19, 1 8 20, 1 8 21, 1 8 22, 1 8 23, 1 8 24, 1 8 25, 1 8 26, 1 8 27, 1 8 28, 1 8 29, 1 8 30, 1 8 31, 1 8 32, 1 8 33, 1 8 34, 1 8 35, 1 9 10, 1 9 11, 1 9 12, 1 9 13, 1 9 14, 1 9 15, 1 9 16, 1 9 17, 1 9 18, 1 9 19, 1 9 20, 1 9 21, 1 9 22, 1 9 23, 1 9 24, 1 9 25, 1 9 26, 1 9 27, 1 9 28, 1 9 29, 1 9 30, 1 9 31, 1 9 32, 1 9 33, 1 9 34, 1 9 35]

```
Command Prompt
14 21 24 27, 14 21 24 28, 14 21 24 29, 14 21 24 30, 14 21 24 32, 14 21 25 26, 14 21 25 29, 14 21 25 30, 14 21 25 32, 14 21 25 33, 14 21 25 34, 14 21 26 28, 14 21 26 29,
14 21 26 30, 14 21 26 32, 14 21 26 33, 14 21 27 29, 14 21 27 30, 14 21 29 30, 14 21 29 32, 14 21 30 32, 14 21 30 33, 14 21 30 34, 14 21 31 32, 14 21 31 35, 14 22 23 27
, 14 22 23 30, 14 22 25 30, 14 22 26 33, 14 22 27 29, 14 22 27 30, 14 22 27 33, 14 22 30 32, 14 22 30 34, 14 22 31 32, 14 23 24 27, 14 23 27 28, 14 23 27 30, 14 23 27 3
1, 14 23 27 35, 14 23 30 35, 14 23 31 35, 14 24 27 30, 14 25 26 32, 14 25 26 33, 14 25 26 34, 14 25 30 32, 14 25 31 32, 14 25 32 34, 14 26 30 32, 14 26 31 32, 14 26 32
33, 14 27 29 30, 14 30 31 32, 14 30 32 33, 14 30 32 34, 14 31 33 35, 15 16 17 19, 15 16 18 19, 15 16 18 21, 15 16 19 21, 15 16 19 27, 15 16 19 35, 15 16 21 25, 15 16 21
27, 15 16 21 31, 15 16 30 34, 15 17 19 34, 15 17 21 24, 15 17 21 25, 15 17 22 23, 15 17 24 33, 15 17 31 35, 15 17 33 34, 15 18 21 23, 15 18 21 24, 15 18 21 25, 15 18 2
1 32, 15 18 21 34, 15 19 21 27, 15 19 21 34, 15 19 27 34, 15 20 33 34, 15 21 22 23, 15 21 22 25, 15 21 22 30, 15 21 24 25, 15 21 24 28, 15 21 25 27, 15 21 25 28, 15 21
25 29, 15 21 25 30, 15 21 25 31, 15 21 25 32, 15 21 25 33, 15 21 25 34, 15 21 27 29, 15 21 30 34, 15 21 31 32, 15 21 31 35, 15 21 33 34, 15 22 23 27, 15 22 23 34, 15 22
24 25, 15 22 25 27, 15 22 25 30, 15 22 27 34, 15 22 30 34, 15 23 27 28, 15 23 27 34, 15 25 28 35, 15 25 29 33, 15 28 31 35, 15 30 33 34, 15 31 33 35, 16 17 19 21, 16 1
7 19 22, 16 17 19 23, 16 17 19 26, 16 17 19 27, 16 17 19 29, 16 17 19 30, 16 17 19 32, 16 17 19 35, 16 17 22 23, 16 17 23 31, 16 17 31 35, 16 18 19 23, 16
18 19 27, 16 18 19 34, 16 18 32 34, 16 19 20 34, 16 19 21 27, 16 19 21 34, 16 19 21 35, 16 19 22 30, 16 19 23 27, 16 19 24 27, 16 19 24 28, 16 19 24 30, 16 19 24 35, 16
19 25 30, 16 19 25 34, 16 19 26 32, 16 19 26 34, 16 19 27 28, 16 19 27 30, 16 19 27 32, 16 19 27 35, 16 19 28 30, 16 19 30 34, 16 19 30 35, 16 19 34 35, 16 20 23 27, 1
6 20 27 28, 16 20 27 30, 16 20 32 34, 16 21 22 30, 16 21 23 27, 16 21 23 29, 16 21 25 32, 16 21 27 35, 16 21 31 35, 16 22 23 27, 16 22 23 28, 16 22 23 29, 16 22 23 30,
16 22 23 31, 16 22 23 35, 16 22 25 30, 16 22 27 34, 16 22 27 35, 16 22 28 35, 16 22 29 35, 16 22 30 35, 16 22 31 35, 16 22 33 35, 16 22 34 35, 16 23 25 30,
16 23 27 28, 16 23 27 29, 16 23 27 30, 16 23 27 35, 16 23 28 30, 16 23 28 35, 16 23 29 34, 16 23 29 35, 16 23 30 35, 16 23 31 35, 16 24 25 28, 16 24 27 28, 16 24 28 32
, 16 24 28 35, 16 24 29 34, 16 24 34 35, 16 25 26 34, 16 25 27 32, 16 25 28 30, 16 25 28 34, 16 25 28 35, 16 25 30 32, 16 25 30 35, 16 25 32 34, 16 26 29 34, 16 26 34 3
5, 16 27 28 30, 16 27 28 32, 16 27 28 35, 16 27 29 35, 16 27 30 32, 16 27 30 35, 16 27 32 35, 16 27 33 35, 16 28 29 34, 16 28 30 32, 16 28 30 35, 16 28 34 35, 16 29 34
35, 16 30 32 34, 16 31 33 35, 17 18 19 33, 17 18 19 34, 17 18 21 31, 17 18 23 31, 17 19 20 34, 17 19 21 32, 17 19 21 34, 17 19 25 34, 17 19 26 32, 17 19 26 34, 17 19 27
32, 17 19 29 34, 17 19 31 32, 17 19 31 33, 17 19 31 34, 17 19 31 35, 17 19 32 34, 17 19 33 34, 17 19 33 35, 17 20 22 32, 17 20 26 32, 17 20 29 34, 17 20 31 32, 17 20 3
1 35, 17 20 32 34, 17 21 22 23, 17 21 22 27, 17 21 22 31, 17 21 24 25, 17 21 24 27, 17 21 24 31, 17 21 24 33, 17 21 24 34, 17 21 25 31, 17 21 25 32, 17 21 25 34, 17 21
27 31, 17 21 31 32, 17 21 31 34, 17 21 31 35, 17 22 23 24, 17 22 23 27, 17 22 23 28, 17 22 23 29, 17 22 23 30, 17 22 23 31, 17 22 23 35, 17 22 24 34, 17 22 25 34, 17 22
26 27, 17 22 26 29, 17 22 26 32, 17 22 26 34, 17 22 27 29, 17 22 27 30, 17 22 27 32, 17 22 27 34, 17 22 27 35, 17 22 31 35, 17 23 24 27, 17 23 27 28, 17 23 27 30, 17 23 27
31, 17 23 31 35, 17 24 25 33, 17 24 25 34, 17 24 26 34, 17 24 27 28, 17 24 27 29, 17 24 29 34, 17 24 31 35, 17 24 33 34, 17 25 26 32, 17 25 26 34, 17 25 27 31, 17
25 27 32, 17 25 29 34, 17 25 31 32, 17 25 31 33, 17 25 31 34, 17 25 32 34, 17 25 33 34, 17 26 27 32, 17 26 29 34, 17 26 31 32, 17 26 32 34, 17 26 33 34, 17 27 31 32, 17
27 31 35, 17 28 31 35, 17 29 31 33, 17 29 31 35, 17 29 33 34, 17 31 33 34, 17 31 33 35, 18 19 21 27, 18 19 25 34, 18 19 27 34, 18 21 22 24, 18 21 25 32, 18 21 25 33, 1
8 21 26 33, 18 21 31 32, 18 21 31 35, 18 22 25 32, 18 22 32 34, 18 24 32 34, 18 25 32 34, 18 26 32 33, 18 26 32 34, 18 27 32 34, 18 30 32 34, 18 31 33 35, 19 20 25 34,
19 20 27 34, 19 20 29 34, 19 20 30 34, 19 20 32 34, 19 21 23 27, 19 21 24 27, 19 21 25 34, 19 21 26 34, 19 21 29 34, 19 21 30 34, 19 22 24 34, 19 22 27 34, 19 22 30 34,
19 23 24 27, 19 24 25 34, 19 24 27 28, 19 24 27 29, 19 24 27 30, 19 24 27 34, 19 24 29 34, 19 25 27 34, 19 25 30 34, 19 25 32 34, 19 25 33 34, 19 26 29 34, 19 26 33 34
, 19 27 29 34, 19 29 34 35, 19 31 33 35, 19 33 34 35, 20 21 25 33, 20 22 23 27, 20 22 23 34, 20 22 24 32, 20 23 27 28, 20 23 27 30, 20 23 28 32, 20 23 31 35, 20 24 32 3
4, 20 25 32 34, 20 26 29 34, 20 27 28 29, 20 27 28 32, 20 27 28 34, 20 27 29 34, 20 29 31 35, 20 29 32 34, 20 29 33 34, 20 29 34 35, 20 30 31 32, 20 31 33 35, 21 22 23
24, 21 22 23 27, 21 22 23 29, 21 22 23 30, 21 22 24 25, 21 22 24 27, 21 22 24 29, 21 22 24 30, 21 22 24 33, 21 22 24 34, 21 22 25 33, 21 22 25 34, 21 22 27 29, 21 22 27
30, 21 22 30 34, 21 23 24 27, 21 23 24 29, 21 23 24 35, 21 23 26 29, 21 23 27 28, 21 23 27 29, 21 24 25 27, 21 24 25 28, 21 24 25 29, 21 24 25 32, 21 24 25 34, 21 24 2
7 28, 21 24 27 29, 21 24 27 30, 21 24 29 32, 21 25 26 33, 21 25 29 32, 21 25 29 33, 21 25 30 32, 21 25 30 33, 21 25 30 34, 21 25 31 32, 21 25 32 33, 21 25 32 34, 21 25
33 34, 21 29 31 32, 21 30 33 34, 21 31 33 35, 22 23 24 27, 22 23 25 27, 22 23 25 29, 22 23 27 28, 22 23 27 29, 22 23 27 30, 22 23 27 32, 22 23 27 34, 22 23 27 35, 22 23
28 30, 22 23 29 30, 22 23 29 34, 22 23 29 35, 22 23 30 35, 22 23 31 35, 22 24 25 34, 22 24 26 34, 22 24 27 34, 22 24 29 33, 22 24 29 34, 22 24 30 32, 22 24 30 34, 22 2
4 31 32, 22 24 32 34, 22 24 33 34, 22 24 33 35, 22 25 27 34, 22 25 30 32, 22 25 30 33, 22 25 30 34, 22 25 31 32, 22 25 31 33, 22 25 32 34, 22 25 33 34, 22 26 27 34, 22
26 29 34, 22 27 29 34, 22 27 30 32, 22 27 32 34, 22 27 33 35, 22 30 31 32, 22 30 32 34, 22 30 33 35, 22 31 33 35, 22 33 34 35, 23 24 27 28, 23 24 27 29, 23
24 27 35, 23 24 29 34, 23 26 29 34, 23 27 28 29, 23 27 28 30, 23 27 28 31, 23 27 28 32, 23 27 28 34, 23 27 28 35, 23 27 29 30, 23 27 29 34, 23 27 29 35, 23 27 30 35, 2
3 27 31 35, 23 29 30 35, 23 29 31 35, 23 29 34 35, 23 30 31 35, 23 31 33 35, 24 25 32 34, 24 26 29 34, 24 27 28 30, 24 27 28 32, 24 27 28 34, 24 27 29 34, 24 27 30 35,
24 27 32 34, 24 29 32 34, 24 31 32 34, 24 31 33 35, 25 26 28 35, 25 26 29 34, 25 26 30 32, 25 26 32 34, 25 26 33 34, 25 27 30 32, 25 27 32 34, 25 29 32 34, 25 30 31 32,
25 30 32 34, 25 31 32 34, 26 27 29 34, 26 28 33 34, 26 29 30 33, 26 29 33 34, 26 29 34 35, 26 32 33 34, 27 29 30 34, 27 29 32 34, 28 31 33 35, 29 31 33 35]
Execution time is: 29.263 seconds.
```

- We have considered dataset with 1000 tuples with 35 different places of Mumbai.
- Dataset is provided as input to the java file.
- Apriori algorithm starts with selecting all unique places i.e 35 places from our dataset and then assigns them a probability based on the number of people that visited that particular place.
- In the next iteration, it will consider only those places which have greater probability than minimum support as defined and then will make combinations of two places and again find probabilities.
- Similarly, this process will continue till we get set of n such places in nth iteration.
- Set of places shown below will be formed and will be recommended to the user.

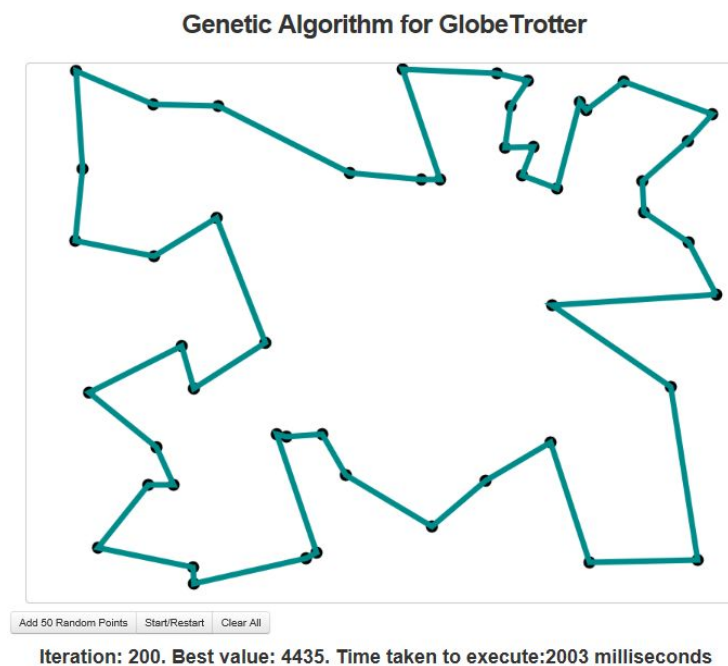
{Prince of Wales Museum, SGNP, Powai Lake, Nehru Science Centre, Walk of Stars}

{Gateway of India, Dhobi Ghat, Siddhivinayak Temple, Essel World, Mahalakshmi Temple}

3.1.4 Genetic Algorithm Simulation Results

Following are the results of Genetic Algorithm simulation for 50 cities selected at random as shown in the Figure.

Population size is said to be the number of chromosomes in one generation. Crossover probability tells us about the frequency of crossover for producing the newer generations. Mutation probability tells us about the mutation frequency on the chromosome. Elitism rate is said to be the rate of copying some part of fittest candidates to newer generations without any changes.



Parameters used are as follows:

Factor	Population size	Elitism Rate	Crossover Probability	Mutation Probability
Value	30	0.3	0.9	0.01

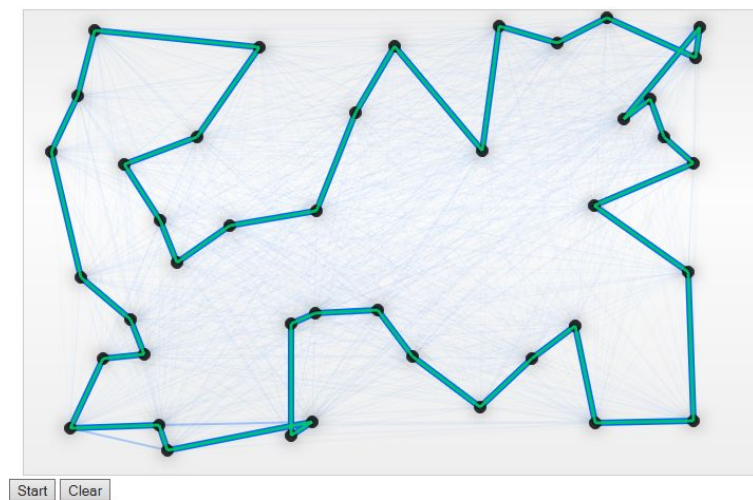
Observed Results:

Factor	Number Of Cities	Number Of Iterations	Best Distance	Execution Time(msec)
Value	50	200	4435	2003

3.1.5 Ant Colony Optimization Simulation Results

We consider 50 cities as inputs similar to the genetic algorithm. Colony size refers to the number of individuals that make up the colony, which is 30 here. Alpha factor shows how important in the path is from given paths. Beta factor shows the relative importance of the paths visibility. Rho factor gives the disappearance degree of the pheromone. The initial pheromone value refers to the ants deposit in its first iteration traversing the path till the goal. The deposit weight of this pheromone refers to the amount of chemical emitted.

Ant Colony Optimization for GlobeTrotter



Iteration:200. Best Distance:3167.42. Time taken to execute:64206

Parameters used are as follows:

Colony size = 30, Alpha = 1, Beta = 3, Rho = 0.1, Initial Pheromone = 1,

Pheromone Deposit Weight = 1

Observed Results:

Factor	Number Of Cities	Number Of Iterations	Best Distance	Execution Time(msec)
Value	50	200	3167.42	64206

3.1.6 Comparison of Simulation Results

Let, Best Distance = b and Execution time = c (in msec)

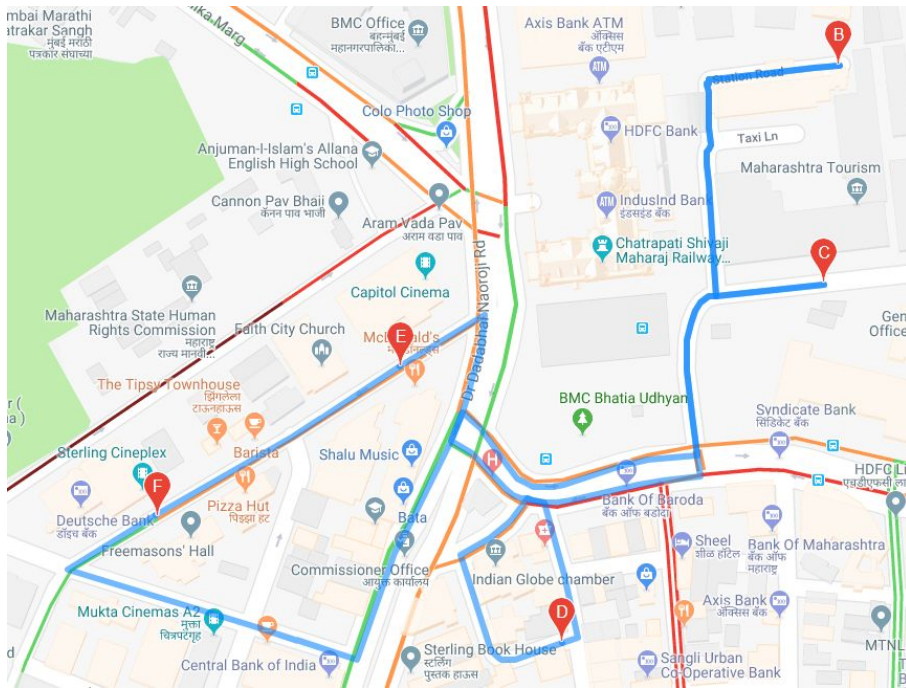
Following are the results of how both algorithms perform for different number of cities (Here, note that for a considered value of number of cities 'n', the same set of city coordinates are provided to both algorithms)

Number of cities	Number of iterations	Genetic Algorithm		Ant Colony Optimization	
		b	c	b	c
50	200	4459	1991	4475	94596
	300	4443	3010	4463	149323
	400	4438	4002	4454	194907
60	200	5083	2010	5108	94630
	300	5016	3067	5048	149563
	400	4902	4034	4989	195021
70	200	5422	2056	5720	94762
	300	5384	3134	5645	149705
	400	5323	4215	5433	195156

As observed in the comparative results, GA fares better than ACO in terms of best distance and execution time. On increasing number of iterations, best distance improves for both algorithms. However, GA gives these results for number of cities 50,60,70. But, for number of cities too large, GA becomes complicated. However since our scenario demands to work with number of cities in hundreds, GA is more appropriate to be used for optimization.

3.1.7 Map Implementation

The map visualization below shows the optimal travel sequence of a user initially at Sterling Cineplex (starting point A which eventually is the end point F as well) who wants to visit four points of interest. The visualization shows that the user is directed towards Indian Railway Catering and Tourism (point B). After point B, the next spots in the travel sequence are Welcome Stores (point C), Bora Bazaar Street Fort (point D), McDonald's (point E) and finally back to Sterling Cineplex (A is overwritten as F in the output). The algorithm used is GA as it was the better algorithm as per scenario



The Algorithm used here is Genetic Algorithm with following configurations:

Population Size = 50

Mutation Rate = 0.1

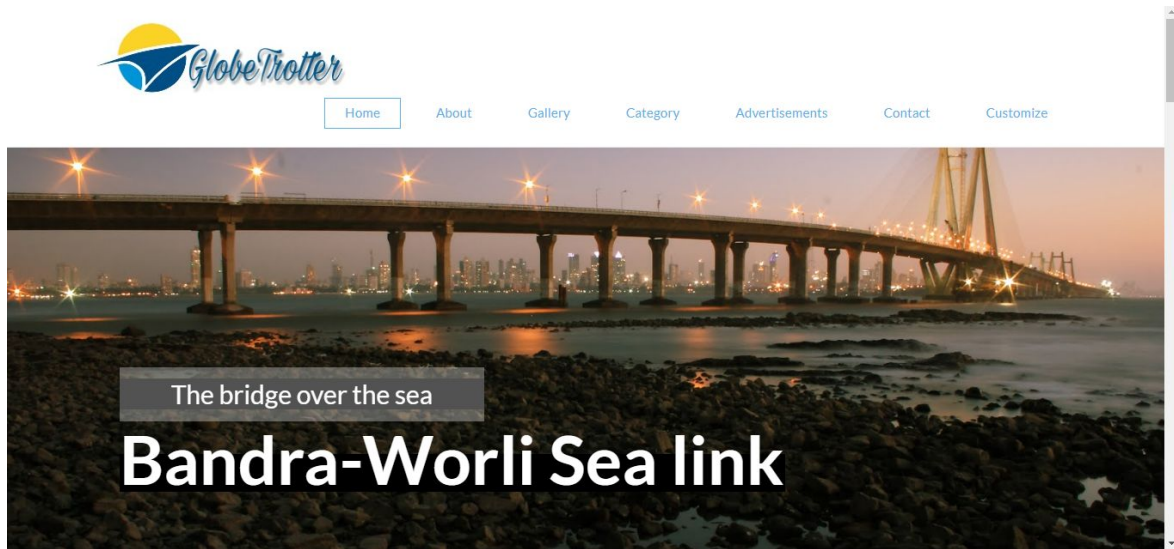
Crossover Rate = 0.5

Elitism = Enabled

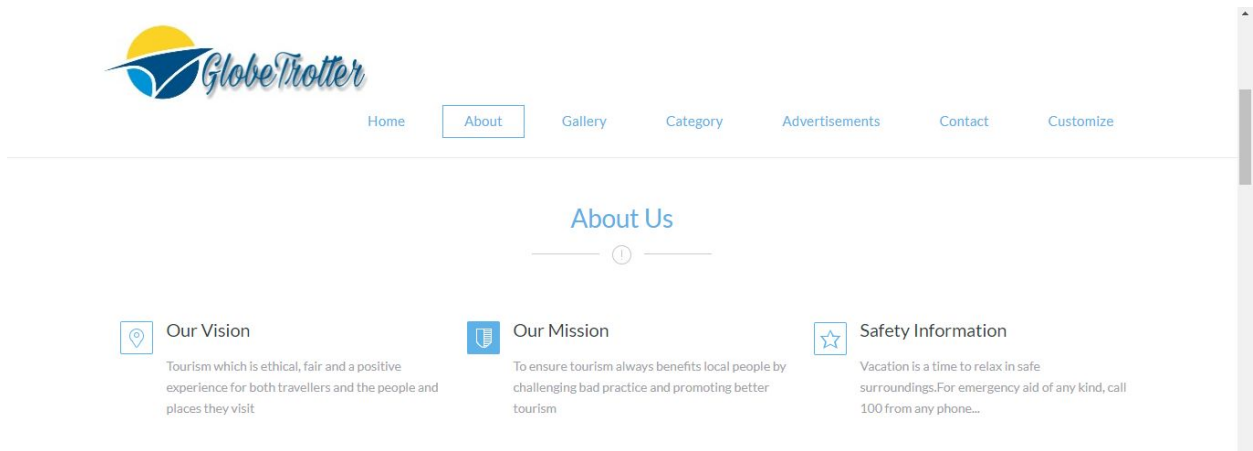
The distance during the normal hours and peak hours for the above travel sequence is the same in this case (1.9 km). The time required for travel during normal hours is 14.12 minutes (recorded at time 13:17) whereas it is 16.04 minutes during peak hours (recorded at time 18:46 on the same day).

3.1.8 Final website

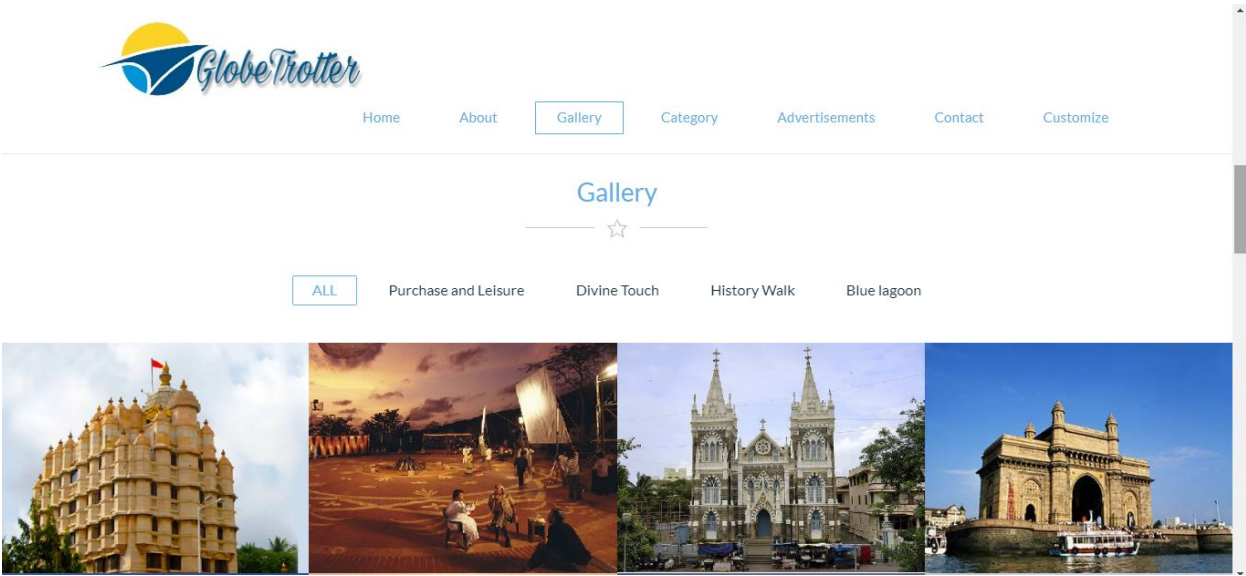
1.Home Screen



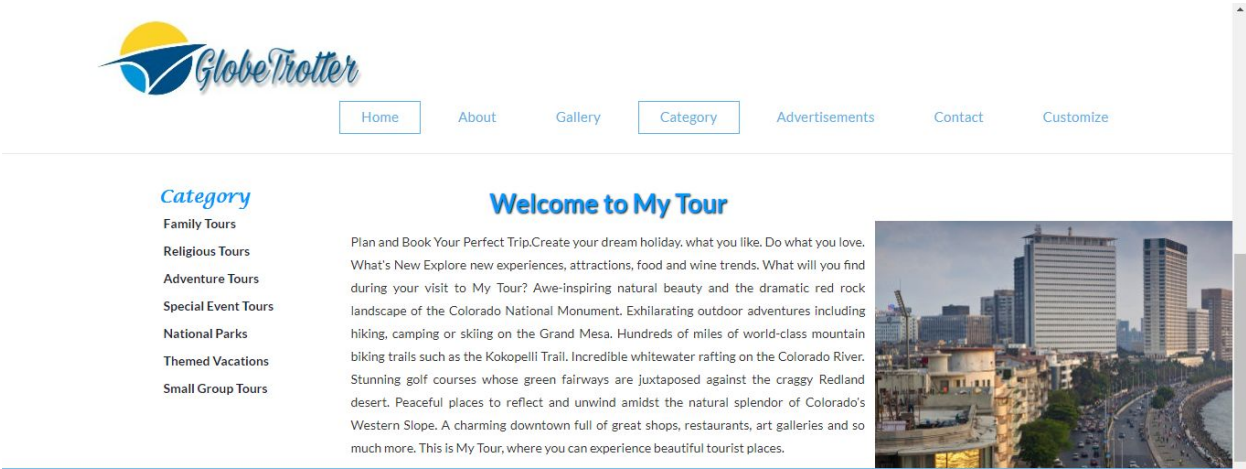
2.About Us




3.Gallery



4.Category



5.Contact



Home

About

Gallery

Category

Advertisements

Contact

Customize

Globe Trotter

Need Help ?
For fantastic suggestions you may also call our travel expert

(+91) 9779730479 - (+91) 7696303090

support[at]globetrotter.com

Marquise towers, Lower Parel, Mumbai.

Name


Contact No

Email

Message

6.Customise

i.Select Month of Travel



Home

About

Gallery

Category

Advertisements

Contact

Customize

Customize your travel plan

☆

Select Month of Travel:

April

ii.Select your Interest

Nature

Historical

Entertainment

Beaches

Religious

Globe Trotter: An optimal travel sequence generator

Page | 27

Saurabh Adhau | Taufiq Monghal | Shubhangi Shinde

iii.Select Budget of your Choice

Select Bugdet of your choice:

0-500₹

500-1000₹

1000-2000₹

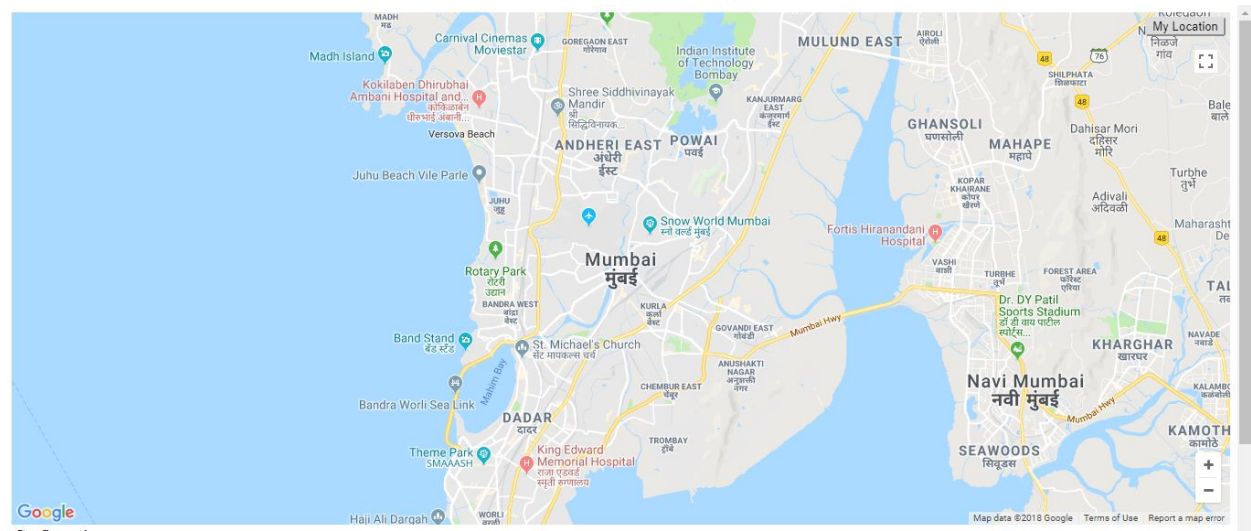
2000-5000₹

Submit

GoToMap

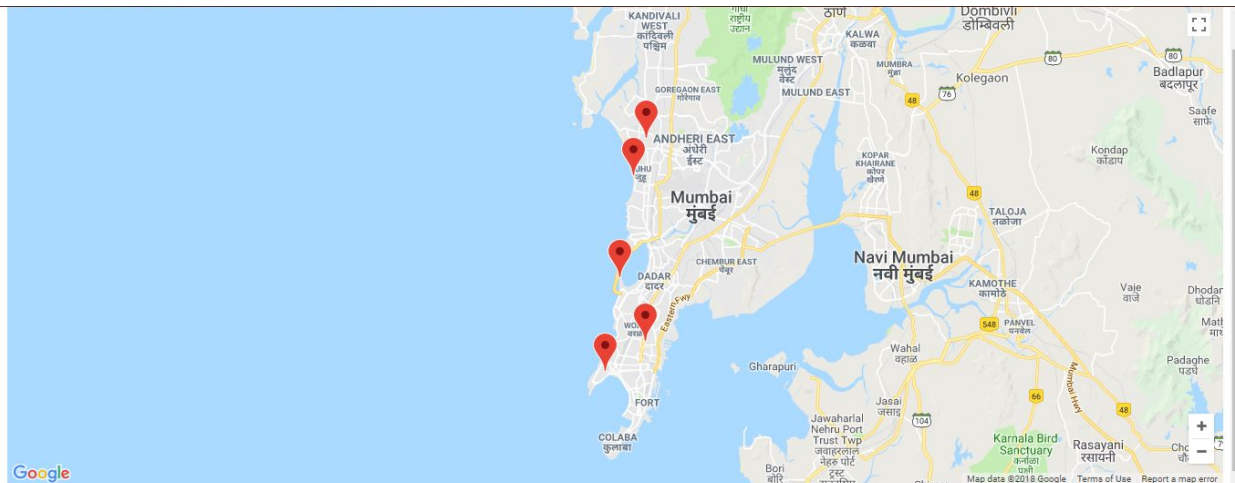
iv.Click submit

Then click on GoToMap

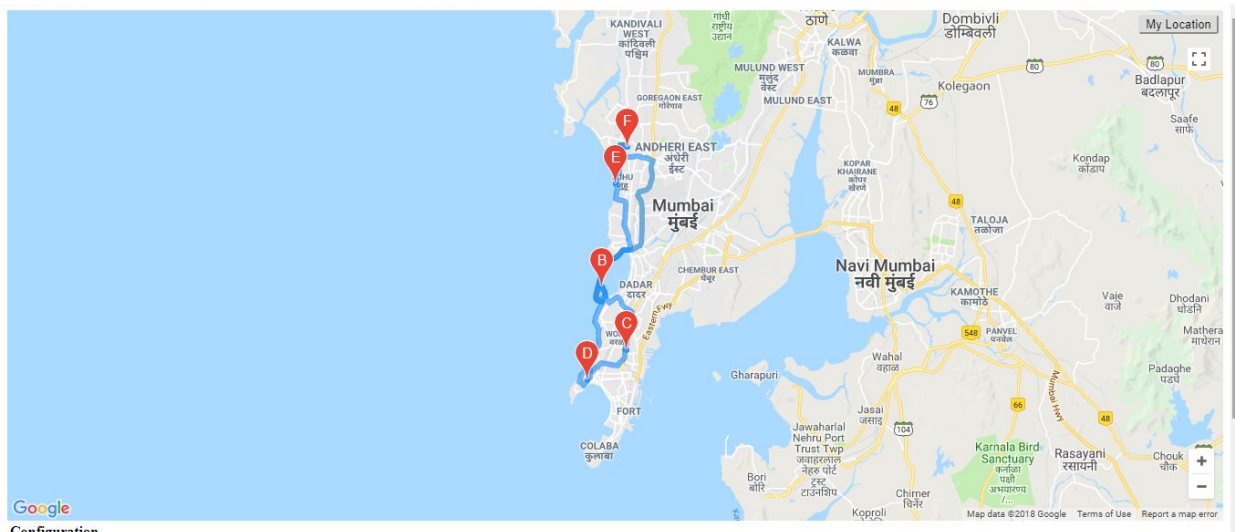


v.Click on My Location tab.

Your location will be marked. Then click on the plot button



vi.Now click Start button to get the optimized route from your location to other places as per interest and budget and and back



3.1.9 Test Cases:

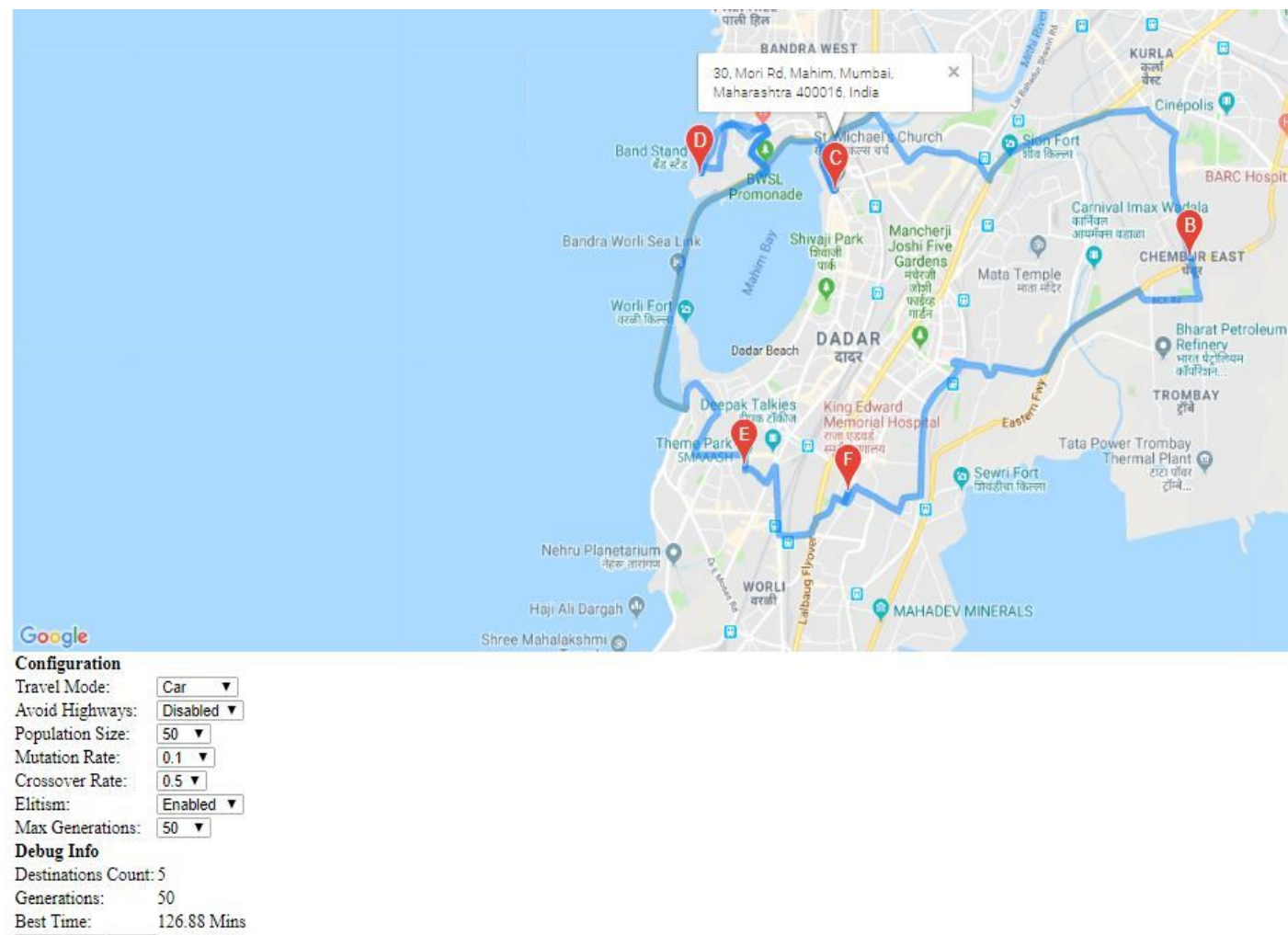
Globe Trotter: An optimal travel sequence generator

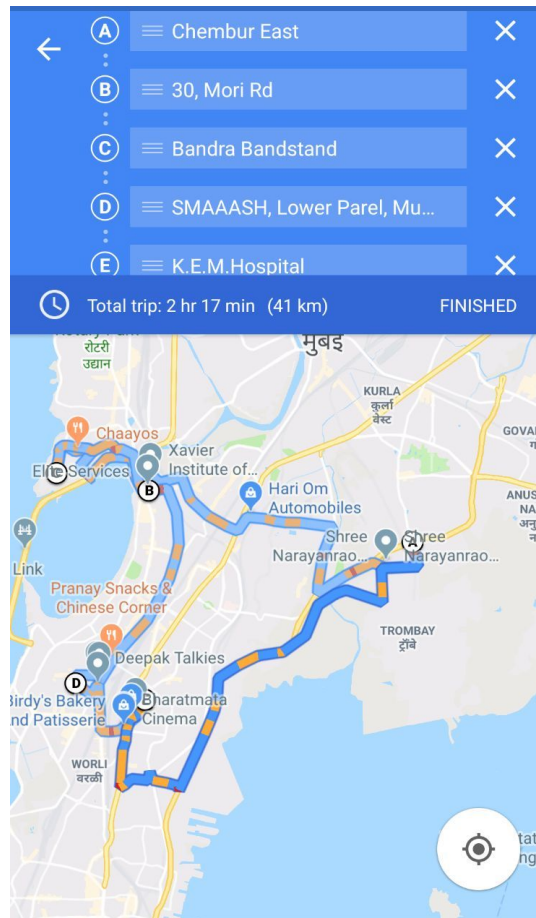
Page | 29

Saurabh Adhau | Taufiq Monghal | Shubhangi Shinde

We have taken 5 test cases for validating the correctness of the route optimisation given by our genetic algorithm by comparing the time taken to traverse the location. Following is the screenshot of the result and the time taken.

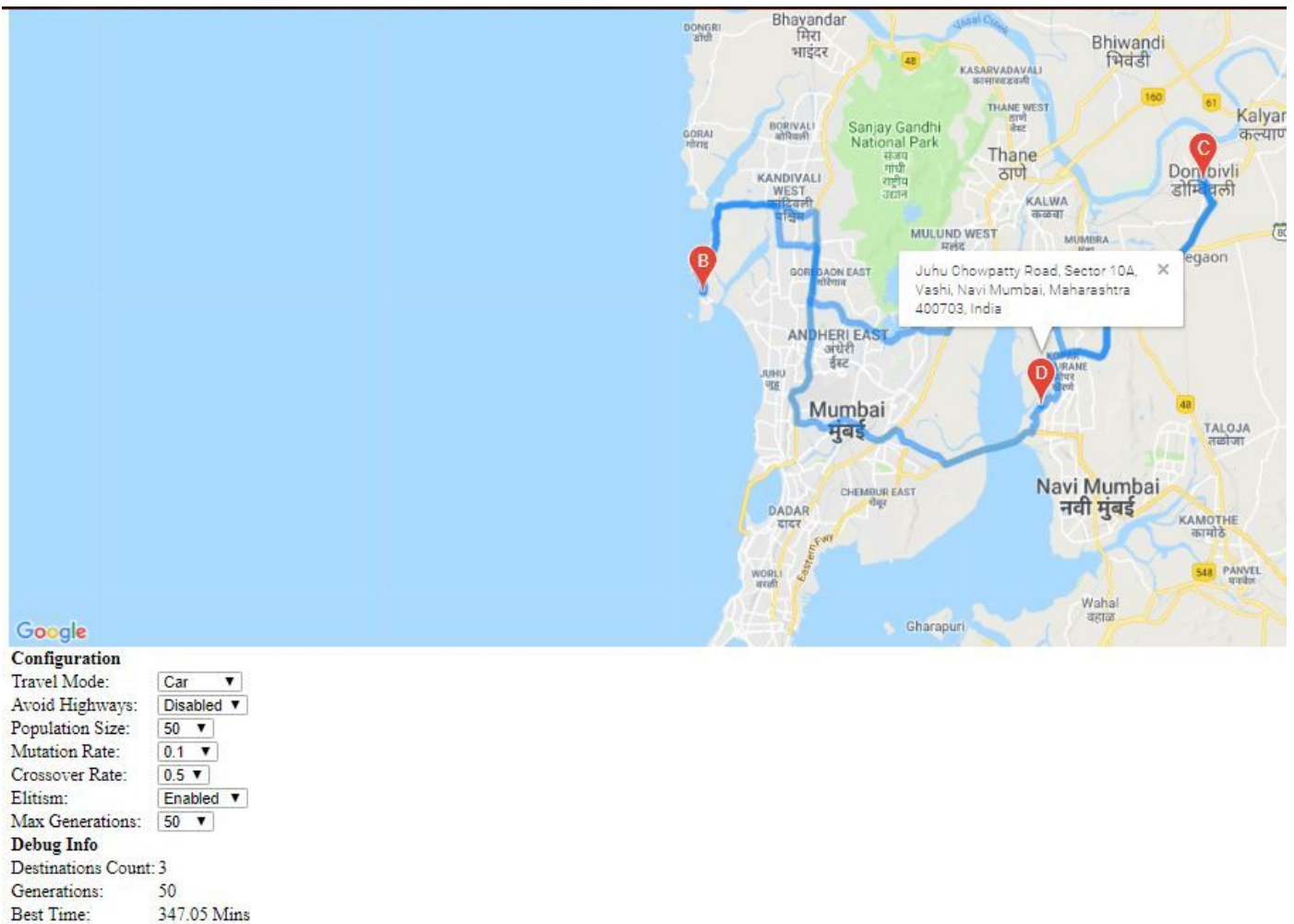
- 1. Time taken:
 - a. Genetic Algorithm: 126.88 mins
 - b. Google map: 2hr 17min= 137 mins

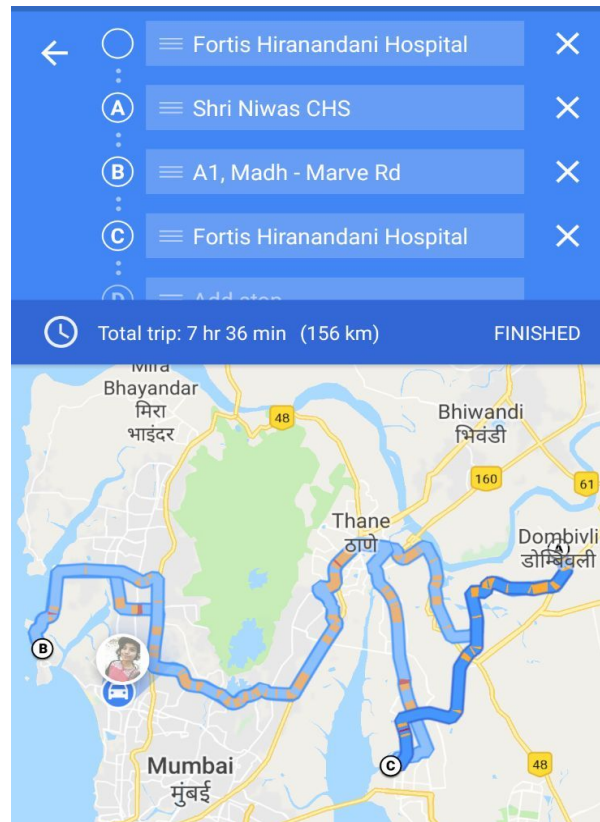




2. Time taken:

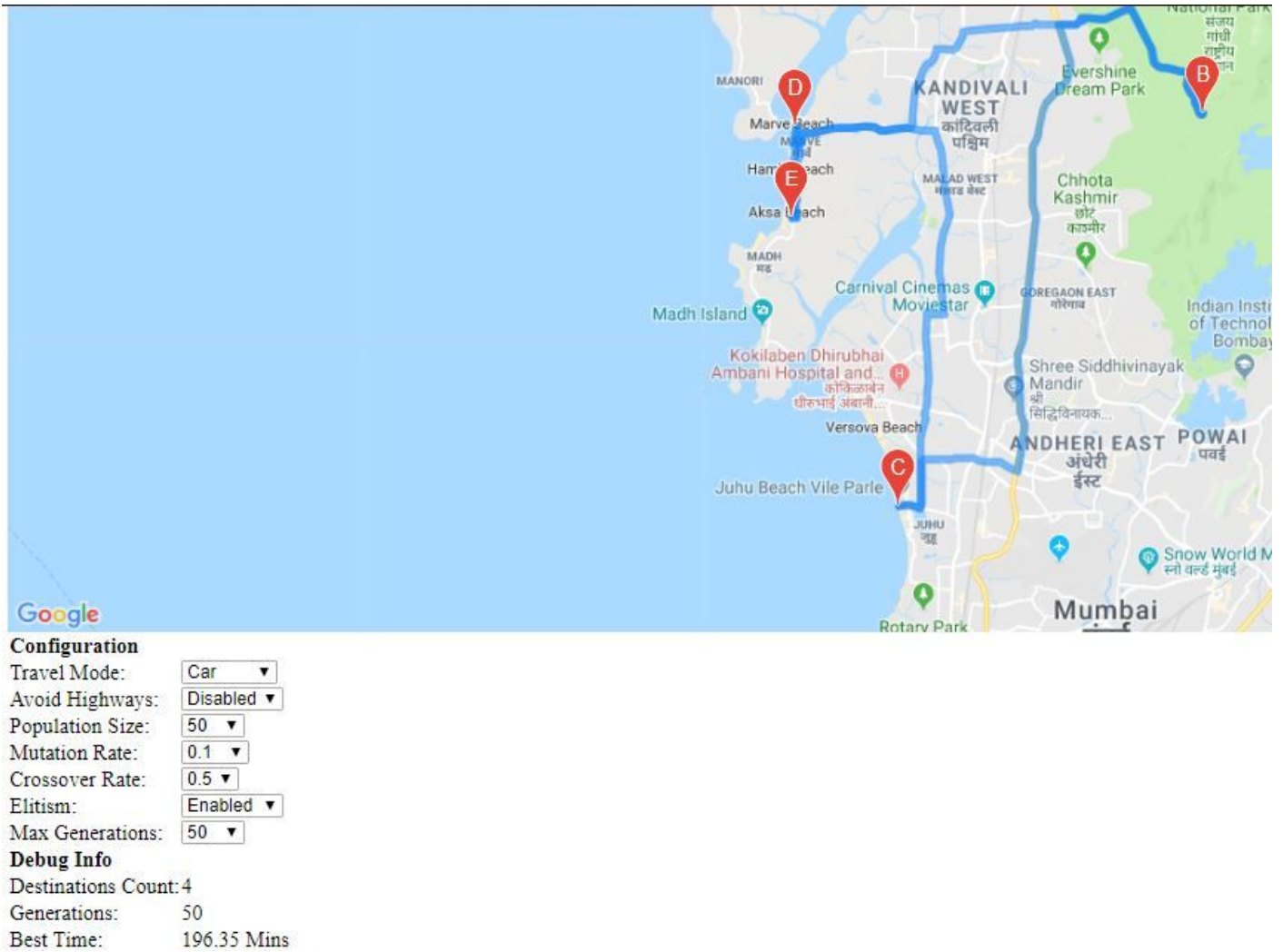
- a. Genetic Algorithm: 347.05 mins
- b. Google map: = 7hr 36 mins=456 mins

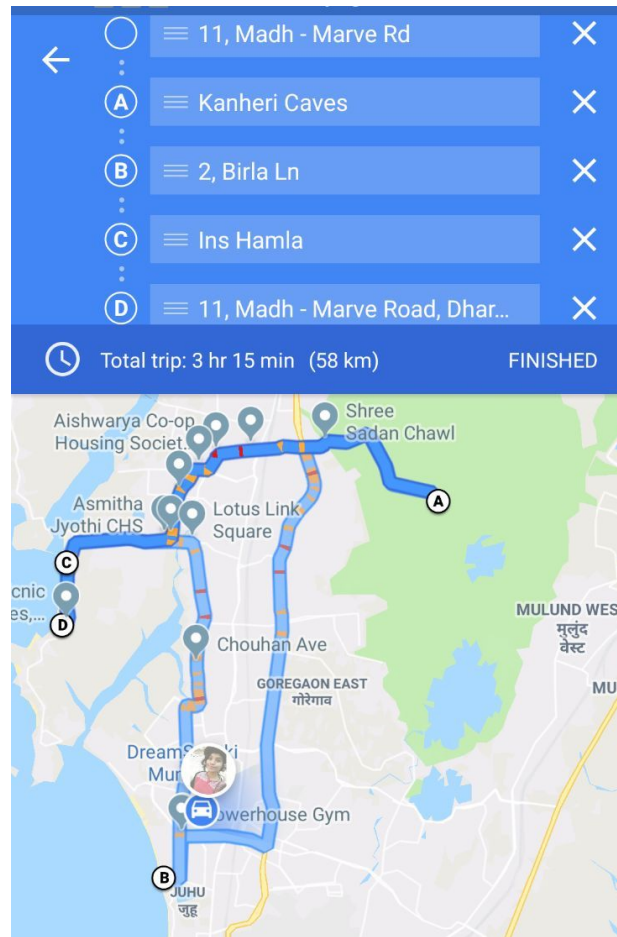




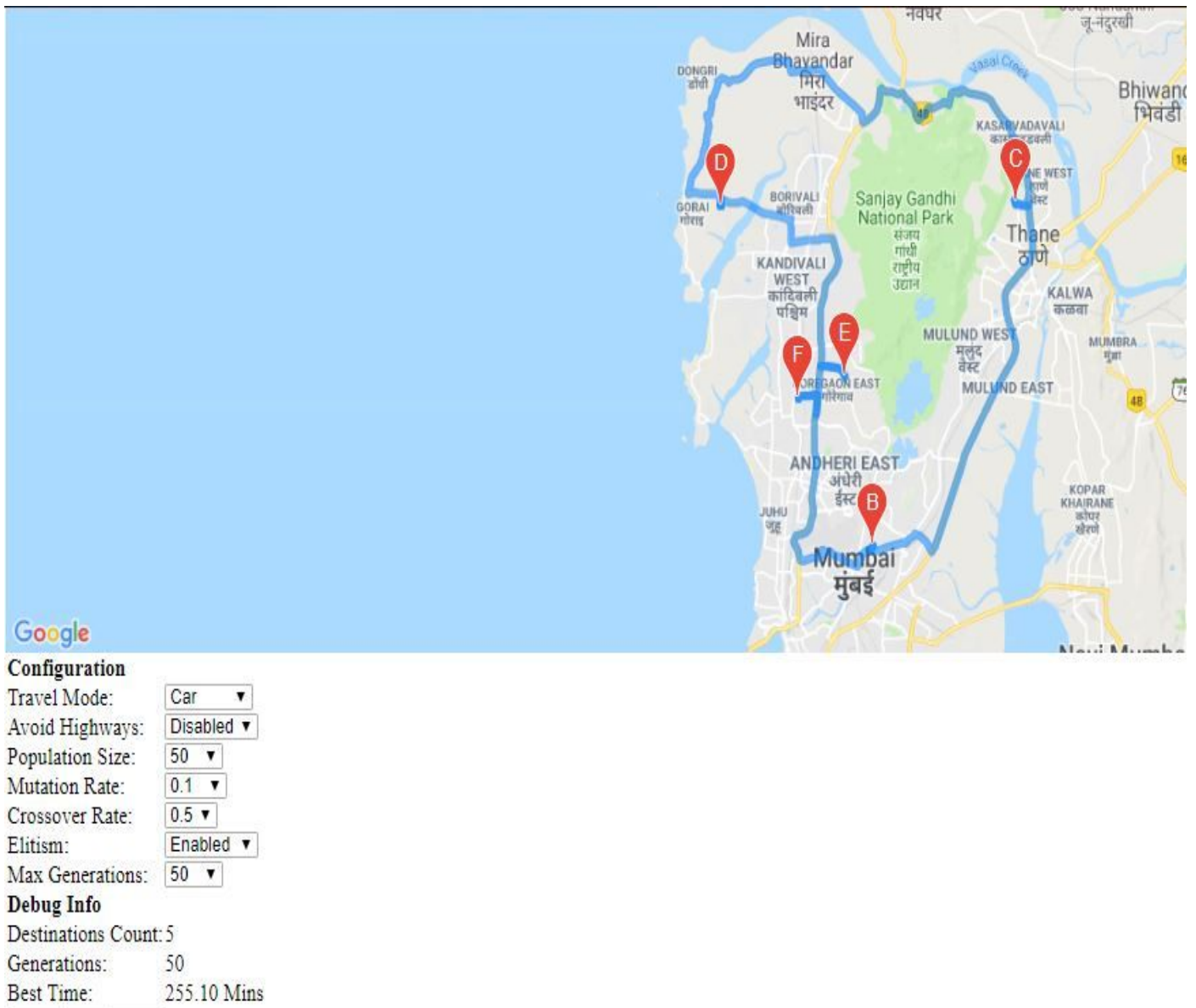
3. Time taken:

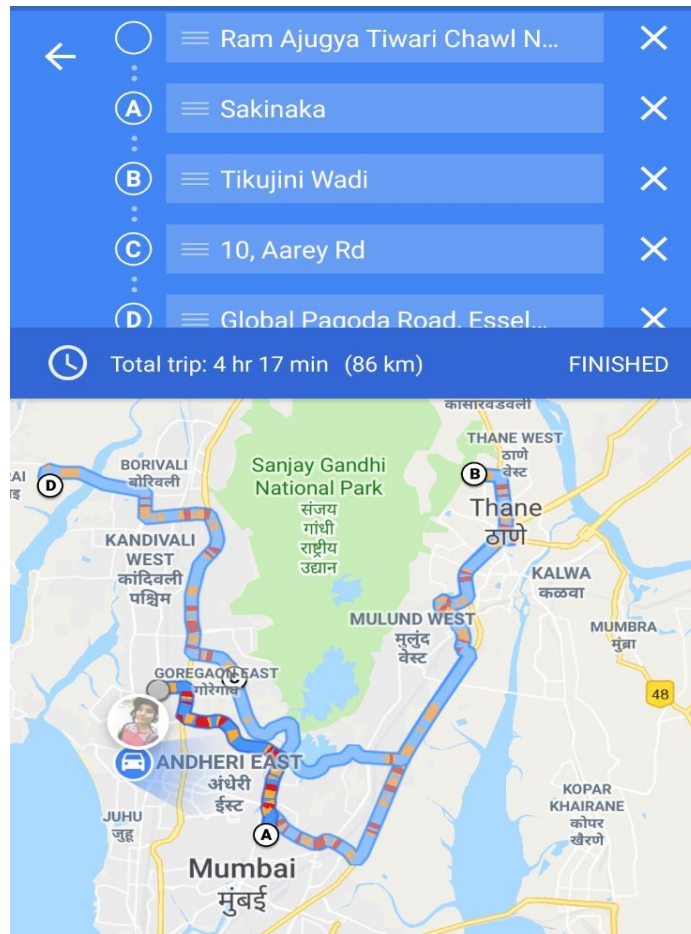
- a. Genetic Algorithm: 196.35 mins
- b. Google map: hr min = 3hr 15 mins = 195 mins



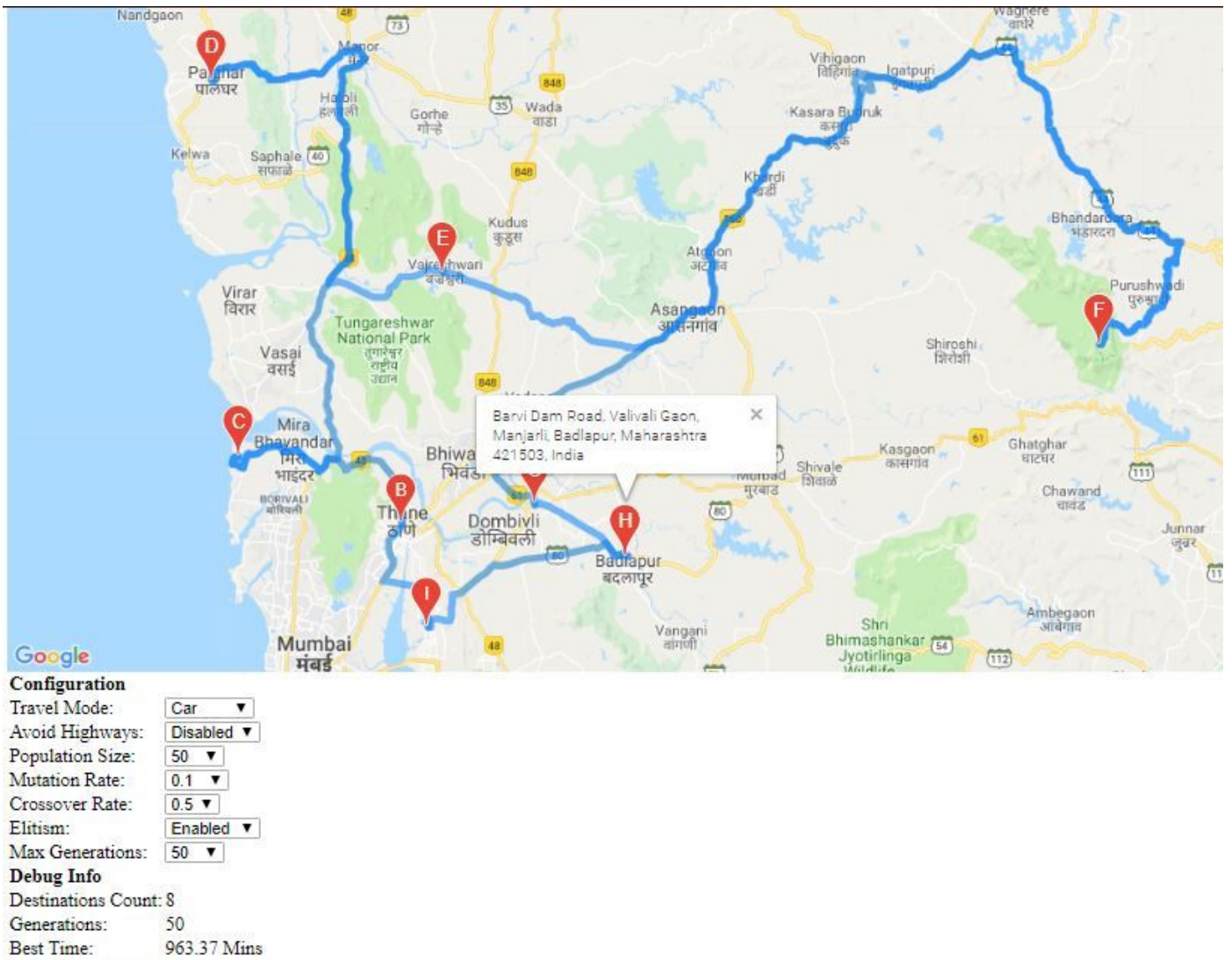


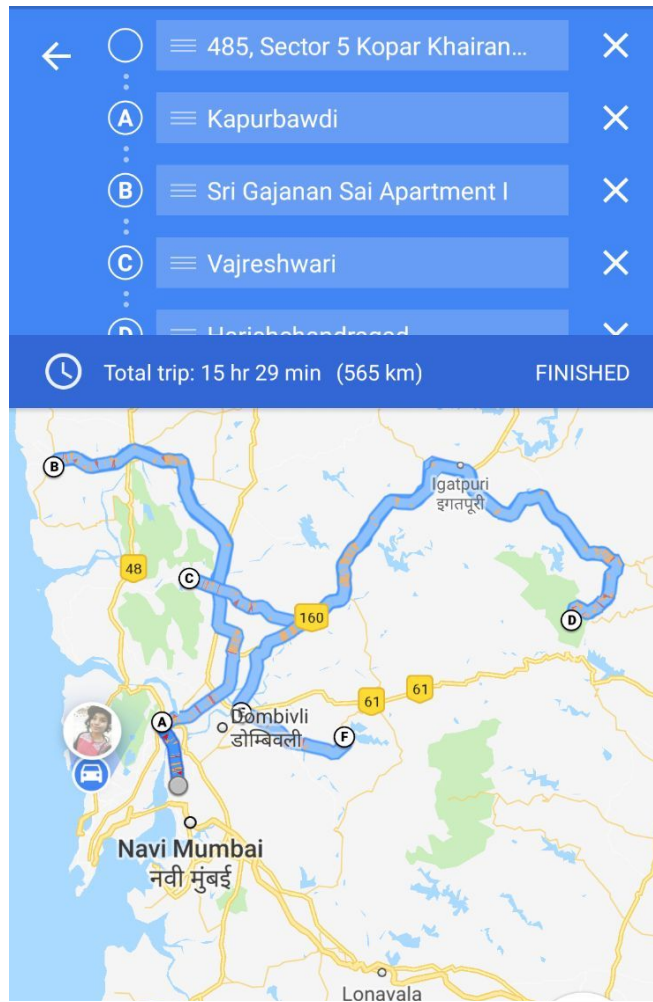
- a. Genetic Algorithm: 255.10 mins
- b. Google map: hr min= 4 hr 17 mins= 247 mins





- Genetic Algorithm: 963.37 mins
- Google map: 15 hr 29 mins = 929 mins





3.2 Technologies Used

3.2.1 Java

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture.

Version Used: Java 1.8.0_131

3.2.2 Eclipse

Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications. **Version Used:** Neon.

3.3 Dataset:

3.3.1 Dataset (1000 tuples) with 35 places

- Execution time was 590.262 seconds
- Frequent 5 itemsets i.e 5 places in each set based on the popularity were generated
- Such 137 sets were obtained

User Alias	Places in Mumbai					
	Gateway of India	Elephanta Caves	Colaba Causeway	Juhu Beach	Victoria Terminus	Film City
A	0	1	1	1	0	1
B	1	0	0	1	0	0
C	1	1	1	0	0	1
D	0	1	1	1	0	1
E	0	0	0	1	1	0
F	1	1	1	0	1	0
G	0	1	0	1	1	0
H	1	0	0	0	0	0
I	1	0	0	1	0	1
J	1	1	0	0	1	1

The 1 and 0's denote whether the user visited the popular place or did not visit.

Therefore,

1=Visited

0=Not Visited

The dataset is self-created by researching the user behaviour from the comments section of the tripadvisor page of each 35 places.

3.4 Goal Justification

1. The goal of this module is to find set of places considering the popularity of those places within the specified location provided manually or through GPS.
2. This goal would be achieved using Apriori algorithm wherein the location would be taken as input and dataset of that location would be fetched and then Apriori algorithm would be applied where the places would be assigned a probability which is calculated by $(\text{No of users visited place } X) / (\text{Total number of users})$. In the second iteration combination of two such places would be made and assigned respective probabilities and it would be continued until we get set of places above our support count and finally those sets would be recommended to the user.

4 References

4.1 Books

- 4.1.1 Data Mining Concepts and Techniques Han, Kamber.
- 4.1.2 Optimization: Algorithms and Applications by Rajesh Kumar Arora.

4.2 Journal Paper

- 4.2.1 Aris Anagnostopoulos, "Tour recommendation for groups", Sapienza University of Rome, Rome, Italy, 16 September, 2016.
- 4.2.2 Wei Luo, " An Improved Ant Colony Optimization and Its Application on TSP Problem", Coll. of Phys. & Inf. Eng., Fuzhou Univ., Fuzhou, China, 04 May 2017.
- 4.2.3 Gao Shang, " Solving Traveling Salesman Problem by Ant Colony Optimization Algorithm with Association Rule", Jiangsu University of Science and Technology, China, 05 November 2007.
- 4.2.4 Lin Yuanyuan, " An Application of Ant Colony Optimization Algorithm in TSP", Training Center of Comput. & Language, Tianjin Univ. of Technol. & Educ., Tianjin, China, 11 December 2012.
- 4.2.5 Jianxin Zhang, " Solving TSP with Novel Local Search Heuristic Genetic Algorithms", Inf. Eng. Sch., Beijing Univ. of Sci. & Technol., Beijing, 07 November 2008.
- 4.2.6 Fabiana Lorenzi, " Personal Tour: a recommender system for travel package", Invenio Software Inteligente, Canoas, Brazil, 10 October 2011.

4.3 Websites

- 4.3.1 <http://blog.hackerearth.com/beginners-tutorial-apriori-algorithm-data-mining-r-implementation> [Accessed: 07- Aug- 2017].
- 4.3.2 <http://www.sciencedirect.com/science/article/pii/S1571064505000333> [Accessed: 05- Sep- 2017].
- 4.3.3 <https://www.slideshare.net/karthiksankar/genetic-algorithms-3626322> [Accessed: 09- Sep- 2017].