MINI PROJECT - II

**Group Travel Plan**

**Using Genetic Algorithms**

S. R. S. Report



**INSTITUTE OF ENGINEERING AND APPLICATIONS**

5th April 2019

Yash Punetha (161500639)

**Table of Contents**

**REVISION HISTORY II**

**DOCUMENT APPROVAL II**

**1. INTRODUCTION** 3

1.1 Purpose 3

1.2 Scope 3

1.3 Definitions 4

1.4 References 4

1.4.1 Dataset 4

1.4.2 Implementation 4

**2. GENERAL DESCRIPTION 4**

2.1 Project Perspective 4

2.2 Tools Used 5

2.3 General Constraints 5

2.4 Assumptions and Dependencies 5

**3. SPECIFIC REQUIREMENTS** 6

*3.1 Hardware and Software Interfaces* 6

3.2 Use Cases 7

3.3 Classes / Objects 8

**4. ANALYSIS MODELS** 10

4.1 Sequence Diagram 10

**A. APPENDICES** 11

A.1 Appendix 1 11

# 1. Introduction

## Anyone who has planned a trip for a group of people, or even for an individual, realizes that there are a lot of different inputs requires, such as what's everyone flight or train schedule should be, how many cars, or automobiles to be rented, which station or airport is nearest. Many outputs must be considered, such as total cost, time spent waiting at airports, and time taken off work. As the inputs can't be mapped with a simple formula, the problem of finding the best solution lends itself to optimization.

## 1.1 Purpose

Optimization finds the best solution to a problem by trying many different solutions  
and scoring them to determine their quality. Optimization is typically used in cases  
where there are too many possible solutions to try them all. The simplest but least  
effective method of searching for solutions is just trying a few thousand random  
guesses and seeing which one is best.

## 1.2 Scope

There are many ways this problem can be expanded. You might combine it with a weather search to optimize for combinations of prices and warm temperatures at potential destinations, or with a hotel search to find destinations with a reasonable combination of flight and hotel prices.

## 1.3 Definition

**Cost Function:** The cost function is the key to solving any problem using optimization, and it’s usually the most difficult thing to determine. The goal of any optimization algorithm is  
to find a set of inputs—flights, in this case—that minimizes the cost function, so the  
cost function has to return a value that represents how bad a solution is.

**Random Searching:** *Random searching* isn’t a very good optimization method, but it makes it easy to understand exactly what all the algorithms are trying to do, and it also serves as a  
baseline so you can see if the other algorithms are doing a good job.

**Hill Climbing:** Randomly trying different solutions is very inefficient because it does not take advantage of the good solutions that have already been discovered. An alternate method of random searching is called hill climbing. Hill climbing starts with a random solution and looks at the set of neighboring solutions for those that are better (have a lower cost function). This is analogous to going down a hill

**Simulated Annealing:** *Simulated annealing* is an optimization method inspired by physics. Annealing is the process of heating up an alloy and then cooling it down slowly. Because the atoms are first made to jump around a lot and then gradually settle into a low energy state,  
the atoms can find a low energy configuration.

**Genetic Algorithm:** Another set of techniques for optimization, also inspired by nature, is called geneticalgorithms. These work by initially creating a set of random solutions known as the population. At each step of the optimization, the cost function for the entire population is calculated to get a ranked list of solutions.

## 1.4 References

**1.4.1 Dataset**

[**http://shop.oreilly.com/product/9780596529321.do**](http://shop.oreilly.com/product/9780596529321.do)

[**http://www.kayak.co.in**](http://www.kayak.co.in)

[**http://www.kayak.com/labs/api/search**](http://www.kayak.com/labs/api/search)

**1.4.2 Implementation**

Wikipedia **–** [**www.wikipedia.com**](http://www.wikipedia.com/)

Software Engineering, Seventh Edition, Ian Sommerville.

Analyticsvidhya **-** [**https://www.analyticsvidhya.com**](https://www.analyticsvidhya.com)

Medium- [**http://www.medium.com**](http://www.medium.com)

StackOverflow –[**http://stackoverflow.com**](http://stackoverflow.com)

Github – [**www.github.com**](http://www.github.com/)

# 2. General Description

The following section describes the various requirements and modules of the Project.

## 2.1 Project Perspective

**Drawbacks:**

Requires a large amount of data.

**Our Plan:**

To minimize the cost and time, and find the best route.

**2.2 Tool Used:**

**Anaconda :** Anaconda is a free and open source distribution of the Python and R programming languages for data science and machine learning related applications, that aims to simplify package management and deployment.

**Python :** Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

**Spyder :** Itis a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It offers a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

**2.3 General Constraints:**

This system is dataset dependent.

**2.4 Assumption and Dependencies**

Scrapped data is from limited sources.

Operating system should be Windows or Linux.

**3. Specific Requirements**

**3.1 Hardware and Software Interfaces**

**Minimum**

|  |  |
| --- | --- |
| **HARDWARE** | **SOFTWARE** |
| 1.2 GHz processor | Windows/Linux |
| 2Gb RAM | Python 3, Git, Anaconda |
| 10 GB HDD | Panda, Numpy |

**Recommended**

|  |  |
| --- | --- |
| **HARDWARE** | **SOFTWARE** |
| **2 GHz Processor and above** | Windows/Linux |
| AMD RADEON GPU(R5 M430 and above) | Anaconda |
| 8Gb RAM and above | Python 3 |
| 50 GB HDD and above | Panda, Numpy |

## 

**3.2 USE CASE Diagram**

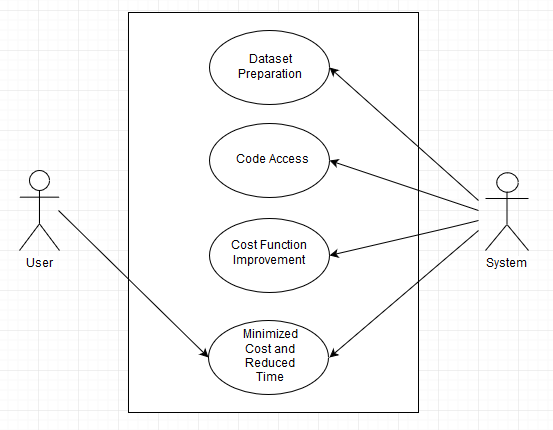


Fig1: Use case diagram

|  |  |
| --- | --- |
| **USE CASE** | **DESCRIPTION** |
| Update dataset | Admin can update dataset. |
| Code access | Admin can update/modify the code as per the requirements. |
| Cost Function Improvement | Admin can increase the efficiency by updating the models |
| Minimized Cost And Reduced Time | User gets the overall cost and waiting time. |

## 3.4 Classes / Objects

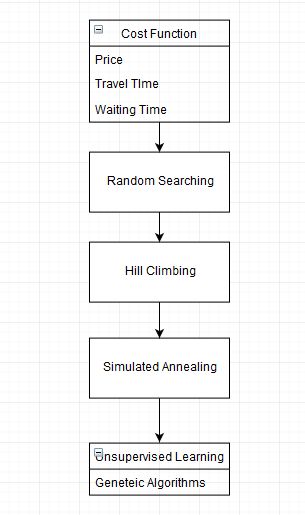


Fig 3: Class diagram

|  |  |
| --- | --- |
| **Class** | **Description** |
| Cost Function | The cost function has to return a value that represents how bad a solution is. |
| Random Searching | It makes it easy to understand exactly what all the algorithms are trying to do |
| Hill Climbing | Hill climbing starts with a random solution and looks at the set of neighboring solutions for those that are better |
| Simulated Annealing | It uses a variable representing the temperature, which starts very high and gradually gets lower |
| Genetic Algorithm | At each step of the optimization, the cost function for the entire population is calculated to get a ranked list of solutions. |

# 4. Analysis Models

## 4.1 Sequence Diagram

## C:\Users\Yash\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2.png

Fig3: Sequence diagram

## A. Appendices

**A.1 Appendices**

**A**

*API*

*Anaconda*

**C**

*Class Diagram*

*Constraints*

*Cost Function*

**D**

*Dataset*

**G**

*Genetic Algorithm*

**H**

*Hill Climbing*

**I**

*Implementation*

**L**

*Linux*

**M**

*Machine learning*

**N**

*Numpy*

**O**

*OS*

**P**

*Python*

*Pandas*

**R**

*Random Searching*

**S**

*Sequence Diagram*

*Spyder*

*Simulated Annealing*

**U**

*Use Case*

**W**

*Windows*

*Warnings*