|  |  |
| --- | --- |
|  | **MINISTRY OF EDUCATION AND TRAINING** |

|  |
| --- |
| **FPT UNIVERSITY** |
| Capstone Project Document |
| Study on Reinforcement Learning and Its Application for Delivery problem |
|  |
| |  |  | | --- | --- | | **Group 4** | | | **Group Members** | Phạm Hoàng Đức Nhã – SE61869  Phan Quốc Minh – SE62019  Hoàng Tuấn Anh – SE62103  Phạm Phú Tiến – SE61597 (Drop) | | **Supervisor** | Đoàn Nguyễn Thành Hòa | | **Ext Supervisor** |  | | **Capstone Project code** | RLFDP |   - Ho Chi Minh City, 01/2018 - |

*This page is intentionally left blank*

**

**CAPSTONE PROJECT REGISTER**

Class: Duration time: from 08/01/2018 To 30/04/2018

(\*) Profession: <Software Engineer> Specialty: <ES> <IS>

x

(\*) Kinds of person make registers: Lecturer Students

x

1. Register information for supervisor (if have)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Full name** | **Phone** | **E-Mail** | **Title** |
| Supervisor 1 | Đoàn Nguyễn Thành Hoà | 098.5050.004 | [hoadnt@fe.edu.vn](mailto:hoadnt@fe.edu.vn) | Lecturer |

2. Register information for students (if have)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Full name** | **Student code** | **Phone** | **E-mail** | **Role in Group** |
| Student 1 | Phạm Hoàng Đức Nhã | SE61869 | 01203135665 | nhaphdse61869  @fpt.edu.vn | Leader |
| Student 2 | Phan Quốc Minh | SE62019 | 01285050035 | minhpqse62019  @fpt.edu.vn | Member |
| Student 3 | Hoàng Tuấn Anh | SE62103 | 0906315471 | anhhtse62103  @fpt.edu.vn | Member |
| Student 4 | Phạm Phú Tiến | SE61597 | 0916512300 | tienppse61597  @fpt.edu.vn | Member |

3. Register content of Capstone Project

(\*) 3.1. Capstone Project name:

English: Study on Reinforcement Learning and Its Application for Delivery problem

Vietnamese: Nghiên cứu về học củng cố và ứng dụng của nó trong việc giải quyết bài toán giao hàng

Abbreviation: RLFDP

(\*) 3.2. Main proposal content (including result and product)

1. Theory and practice (document):

Reinforcement learning is a type of Machine Learning algorithm which allows software agents and machines to automatically determine the ideal behavior within a specific context, to maximize its performance. Reinforcement learning is learning what to do- how to map situations to actions- so as to maximize a numerical reward signal.

Objectives:

+ Understanding delivery problem

+ Study on Reinforcement Learning: Policy, Reward Signal, Value Function, Model of the Environment

+ Build the model for delivery problem based on Reinforcement Learning

+ Compare the proposed model with the existing models: Heuristic and approximation algorithms, Exact algorithms

+ Implement MDP, Q-learning algorithms

+ Report papers

Research: MDP- Markov Decision Process, Q-Learning, Heuristic and approximation algorithms, Exact algorithms

1. Program:

Demo application for solving the delivery problem by applying the MDP, Q-Learning algorithms

Language: python(OpenAI Gym, TensorFlow, Scikit-learn libraries),

1. Other products:

Report papers

4. Other comment (propose all relative thing if have)

|  |  |
| --- | --- |
| **Supervisor (If have)**  *(Sign and full name)* | HCM, 03/01/2018  **On behalf of Registers**  *(Sign and full name)* |

Đoàn Nguyễn Thành Hòa

**Table of Contents**

[**Definitions, Acronyms, and Abbreviations 10**](#_Toc512107472)

[**A. Introduction 11**](#_Toc512107473)

[**1. Project Information 11**](#_Toc512107474)

[**2. Introduction 11**](#_Toc512107475)

[**3. Current Situation 11**](#_Toc512107476)

[**4. Problem Definition 12**](#_Toc512107477)

[**5. Proposed Solution 12**](#_Toc512107478)

[**6. Role and Responsibility 13**](#_Toc512107479)

[**B. Software Project Management Plan 14**](#_Toc512107480)

[**1. Problem Definition 14**](#_Toc512107481)

[**1.1. Name of this Capstone Project 14**](#_Toc512107482)

[**1.2. Problem Abstract 14**](#_Toc512107483)

[**1.3. Project Overview 14**](#_Toc512107484)

[**1.3.1. Current Situation 14**](#_Toc512107485)

[**1.3.2. The Proposed System 15**](#_Toc512107486)

[**1.3.3. Boundaries of the System 15**](#_Toc512107487)

[**1.3.4. Future Plans 15**](#_Toc512107488)

[**1.3.5. Development Environment 16**](#_Toc512107489)

[***1.3.5.1.* Hardware Requirement 16**](#_Toc512107490)

[***1.3.5.2.* Software Requirement 16**](#_Toc512107491)

[**2. Project organization 16**](#_Toc512107492)

[**2.1. Software Process Model 16**](#_Toc512107493)

[**2.2. Roles and responsibilities 17**](#_Toc512107494)

[**2.3. Tools and Techniques 18**](#_Toc512107495)

[**3. Project Management Plan 19**](#_Toc512107496)

[**3.1. Software development life cycle 19**](#_Toc512107497)

[**3.2. Phase Detail 21**](#_Toc512107498)

[**3.2.1. Initialization and basic research 21**](#_Toc512107499)

[**3.2.2. Read paper and design 21**](#_Toc512107500)

[**3.2.3. Implement demo application 22**](#_Toc512107501)

[**3.2.4. Test demo application 22**](#_Toc512107502)

[**3.2.5. Write report document 23**](#_Toc512107503)

[**3.2.6. Write paper 23**](#_Toc512107504)

[**3.3. All Meeting Minutes 23**](#_Toc512107505)

[**4. Coding Convention 23**](#_Toc512107506)

[**C. Software Requirement Specification 26**](#_Toc512107507)

[**1. User Requirement Specification 26**](#_Toc512107508)

[**2. System Requirement Specification 26**](#_Toc512107509)

[**2.1. External Interface Requirement 26**](#_Toc512107510)

[**2.1.1. User Interface 26**](#_Toc512107511)

[**2.1.2. Hardware Interface 27**](#_Toc512107512)

[**2.1.3. Software Interface 27**](#_Toc512107513)

[**2.2. System Overview Use Case 28**](#_Toc512107514)

[**D. Software Design Description 29**](#_Toc512107515)

[**1. Design Overview 29**](#_Toc512107516)

[**2. Detailed Description 29**](#_Toc512107517)

[**2.1. Class Diagram 29**](#_Toc512107518)

[**3. Algorithms 30**](#_Toc512107519)

[**3.1. Delivery algorithm 30**](#_Toc512107520)

[**3.1.1. Definition 30**](#_Toc512107521)

[**3.1.2. Define Problem 31**](#_Toc512107522)

[**3.1.3. Solution 31**](#_Toc512107523)

[**3.1.4. Complexity 33**](#_Toc512107524)

[**3.1.5. Flowchart 34**](#_Toc512107525)

[**3.2. K-means++ algorithm 34**](#_Toc512107526)

[**3.2.1. Definition 34**](#_Toc512107527)

[**3.2.2. Define problem 35**](#_Toc512107528)

[**3.2.3. Solution 35**](#_Toc512107529)

[**3.2.4. Complexity 36**](#_Toc512107530)

[**3.2.5. Flowchart 36**](#_Toc512107531)

[**3.3. Other heuristic algorithms 37**](#_Toc512107532)

[**3.3.1. Ant Colony Optimization algorithm 37**](#_Toc512107533)

[**3.3.2. SA algorithm 38**](#_Toc512107534)

[**E. Task Sheet 39**](#_Toc512107535)

[**F. Appendix 43**](#_Toc512107536)

**List of Tables**

[Table 1: Definitions, Acronyms and Abbreviations 10](#_Toc512084654)

[Table 2: Role and Responsibility 13](#_Toc512084655)

[Table 3: Hardware Requirement 16](#_Toc512084656)

[Table 4: Software Requirement 16](#_Toc512084657)

[Table 5: Responsibilities 18](#_Toc512084658)

[Table 6: Tool and Techniques 18](#_Toc512084659)

[Table 7: Software development life cycle 20](#_Toc512084660)

[Table 8: Initialization and basic research 21](#_Toc512084661)

[Table 9: Read paper and design 22](#_Toc512084662)

[Table 10: Implement demo application 22](#_Toc512084663)

[Table 11: Test demo application 23](#_Toc512084664)

[Table 12: Write report document 23](#_Toc512084665)

[Table 13: Write paper 23](#_Toc512084666)

[Table 14: Class dictionary 30](#_Toc512084667)

[Table 15: Task sheet 42](#_Toc512084668)

**List of Figures**

[Figure 1: Software process model 17](#_Toc511854416)

[Figure 2: Use-case overview 28](file:///C:\Users\NHAPHDSE61869\Desktop\CapstoneProject\ReportVersion2\Briefcase.docx#_Toc511854417)

[Figure 3: Class diagram 29](#_Toc511854418)

[Figure 4: Ant-Q flowchart 34](#_Toc511854419)

[Figure 5: K-means++ flowchart 36](#_Toc511854420)

[Figure 6: ACO flowchart 37](#_Toc511854421)

[Figure 7: SA flowchart 38](#_Toc511854422)

# Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **SA** | Simulated Annealing |
| **ACO** | Ant Colony Optimization |

Table : Definitions, Acronyms and Abbreviations

# Introduction

## Project Information

* Project Name: Study on Reinforcement Learning and Its Application for Delivery problem
* Project Code: RLFDP
* Project Type: Desktop Application
* Start Date: 08/01/2018
* End Date: 30/04/2018

## Introduction

To determine the shortest path are one of the most addressed problems in the field of computer science. Many algorithms are currently available to solve the problems. However, since this problem is categorized as NP-hard, considering between performance and accuracy of algorithms to optimize the results in specific situations is also the dilemma which need to be solved.

Since exact algorithms are too resources-consuming, improving heuristic solutions to obtain as closest as possible the optimal result yet keeping performance to be within the acceptable threshold is currently the better approach to solve the problems. To address further, the current heuristic algorithms for shortest path problems are good only for stable environments, of which variables and states are intolerant to changes over times. Our thesis mainly focuses on how using machine learning and to be more specific, reinforcement learning would improve the accuracy of heuristic approach as well as to compare with other exact and heuristic algorithms. Furthermore, to observe how machine will “learn” to adjust itself in the inconsistent environment is also another façade of the problem we will address. The algorithm in the branch of reinforcement learning to be studied in this thesis is Ant-Q (multi-agents). To prove that the algorithm will outperform other reinforcement learning methods in terms of efficiency in time and resource usage, the collections and conclusion about the optimum of selected algorithms will be performed in this document.

## Current Situation

Choosing algorithms to solve a problem is always a difficult choice. We can not guarantee which algorithms are good or bad for research purposes. By studying previously available materials and reading some of the scientific papers on this topic, we may be able to make easier choices, but here the data is still experimental, so the error is inescapable, the collection and comparison of data from different algorithms and data sets will not normally be as expected.

Furthermore, in order to fit the situation where many delivery men would be involved in one delivery session as well as to improve the algorithm performance-wise, clustering and to be specific, K-means++ will be used to divide and conquer big delivery problems.

## Problem Definition

The current delivery problem solving algorithms always have disadvantages:

* With exact algorithms, work is reasonably fast only for small size problems.
* With heuristic algorithms, i.e., algorithms that deliver either seemingly or probably good solutions, but which could not be proved to be optimal.
* Each algorithm requires that the environment must be determined and fixed (Every changes on environment will lead to the whole algorithm to be re-run).
* Using a combination of algorithms is choosing an algorithm depending on the environment that is appropriate with that algorithm. But the method to choose the algorithm is not optimized, and there are many special cases and environments that have not optimized algorithm.

## Proposed Solution

To overcome the disadvantages of existing algorithms , we propose to use the Reinforcement Learning method, in which the Q-learning algorithm is the most efficient algorithm for solving the delivery problem.

Advantages:

* No fixed environment is required.
* Solving many special cases and environments.
* Avoid failures encountered before.
* Any changes on environments will not affect the learning process as previous learning processes are updated after each episode.

Disadvantages:

* It is necessary to have a simulated environment and a certain time to “learn” before producing optimal solution.

## Role and Responsibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Full Name | Role | Position | Contact |
| 1 | Đoàn Nguyễn Thành Hoà | Project Manager | Supervisor | [hoadnt@fe.edu.vn](mailto:hoadnt@fe.edu.vn) |
| 2 | Phạm Hoàng Đức Nhã | Developer | Leader | [nhaphdse61869@fpt.edu.vn](mailto:nhaphdse61869@fpt.edu.vn) |
| 3 | Phan Quốc Minh | Developer | Member | [minhpqse62019@fpt.edu.vn](mailto:minhpqse62019@fpt.edu.vn) |
| 4 | Hoàng Tuấn Anh | Developer | Member | [anhhtse62103@fpt.edu.vn](mailto:anhhtse62103@fpt.edu.vn) |
| 5 | Phạm Phú Tiến | Developer | Member | [tienppse61597@fpt.edu.vn](mailto:tienppse61597@fpt.edu.vn) |

Table : Role and Responsibility

# Software Project Management Plan

## Problem Definition

### Name of this Capstone Project

Official Name: Study on Reinforcement Learning and Its Application for Delivery problem

Vietnamese Name: Nghiên cứu về học củng cố và ứng dụng của nó trong việc giải quyết bài toán giao hàng

Abbreviation: RLFDP

### Problem Abstract

During researching, we applied Ant-Q algorithm which will be the main algorithm to solve the delivery problem. However, in order to fit the situation where many delivery men would be involved in one delivery session as well as to improve the algorithm performance-wise, clustering and to be specific, K-means++ will be used to divide and conquer big delivery problems.

By applying other solving delivery problem algorithms, specifically here we use ACO(Ant Colony Optimization) and SA(Simulated Annealing). The process of comparing and measuring data to give specific parameters, these parameters may be useful for selecting the right solution for different situations. The results may not be as expected but it will still be useful.

### Project Overview

#### Current Situation

* PyQt5 is mainly written in C++ language, it makes difficult to convert to python language.
* The application does not guarantee to run well if input data set is over than 50 locations.
* Comparison between asynchronous logs on data makes it difficult to make a conclusion.
* Color blurring when showing route on google maps.

#### The Proposed System

To acquire the fundamental knowledge of the basic concepts, we need to learn about Machine Learning, especially Reinforcement Learning. Nevertheless, to fully understand as well as to be able to implement the algorithm, study on Q-Learning and Ant-Q via appropriate materials is also a must.

In order to compare and study on the advantages and disadvantages of Ant-Q and other heuristic approaches, we decided to create a demo application, with a certain amount of dummy data to produce comparable results.

#### Boundaries of the System

Our program applies three selected heuristic algorithms to solve the delivery problem and show the process how algorithms work on it.

Using Ant-Q, ACO and SA algorithms to run the process is the core of the system.

Combining Ant-Q algorithm and K-mean algorithm, k-mean divides the data set into smaller pieces then apply Ant-Q algorithm on each piece of that data set.

Recording all results when the process is done, the records will be saved as logs. Log data helps user comparing the difference between logs.

Besides that, the demo application also can apply the algorithm and its parameters which belongs to selected log to process on google maps then show route results to user.

#### Future Plans

Our system is built for the only purpose that solving the delivery problem. Not only researching, testing, collecting data to make a conclusion, a comparison or a statistic. Moreover we want to apply the solution to the practical application.

About academic researching:

* We apply three algorithms which can solve the delivery problem. there are Ant-Q, ACO, SA.
* Our system allows us scale out our model, we also can apply the other algorithms which belong to other concepts of machine learning.

About practical application:

* Contributing to build solutions for Logistic.
* Optimizing existing navigation algorithms.
* Being a tool for developing Artificial intelligence (AI) which is specifically the automatic navigation robot.

#### Development Environment

##### Hardware Requirement

|  |  |  |
| --- | --- | --- |
|  | **Minimum Requirements** | **Recommended** |
| **Operating System** | Windows 7 or later, macOS, and Linux | Windows 10 or later, macOS, and Linux |
| **Processors** | Intel® Core™ i3 processor | Intel® Core™ i5 processor 4300M at 2.60 GHz |
| **Disk Space** | 1 GB | 2 to 3 GB |
| **Memory** | 1 GB | 4 GB |

Table : Hardware Requirement

##### Software Requirement

|  |  |  |
| --- | --- | --- |
| **Software** | **Name/ Version** | **Description** |
| Environment | Python3 | Specification for python developer |
| Modeling tool | StarUML | Used to design UML diagrams |
| Source control | Github | Used for source control |
| IDE | Pycharm | Programming tool |

Table : Software Requirement

## Project organization

### Software Process Model

This project is developed using waterfall model. A customized waterfall model is applied to capable with current situations. This model includes 5 steps of researching the algorithm and developing the application:

* Initialization and basic research
* Read paper and design
* Implement demo application
* Write document
* Write paper

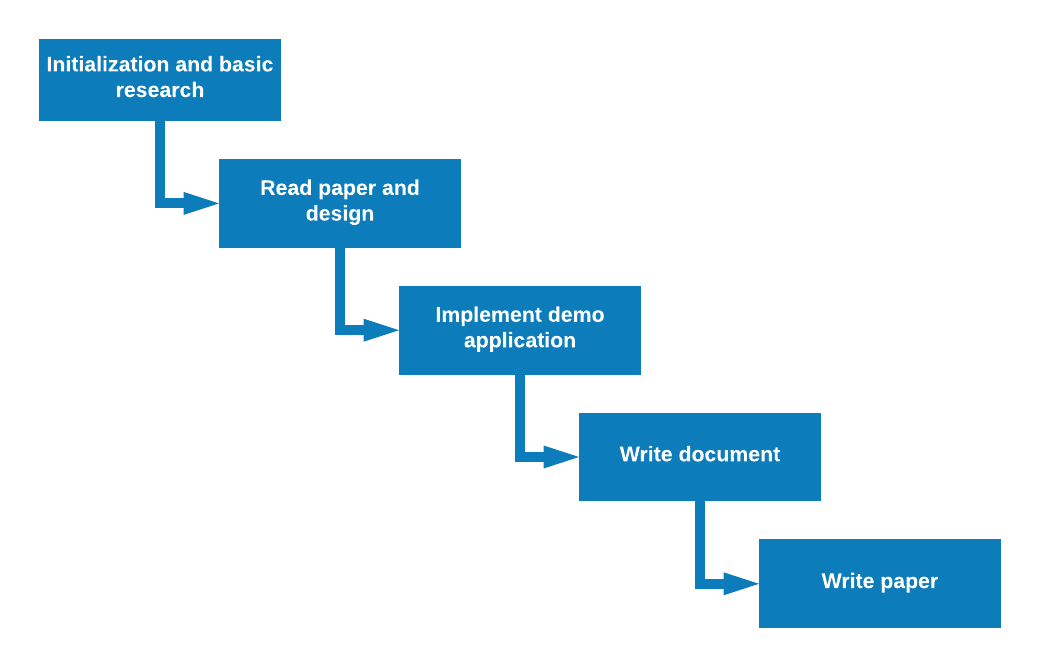


Figure : Software process model

### Roles and responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Full name** | **Role in Group** | **Responsibilities** |
| 1 | Đoàn Nguyễn Thành Hòa | Project Manager | * Specify research requirements * Control the project process * Give some advice and ensure that the research member is researching the right things. |
| 2 | Phạm Hoàng Đức Nhã | Team Leader, Developer, Tester | * Managing process * Read and understand specify paper * Prepare document and write paper * Coding * Testing |
| 3 | Phan Quốc Minh | Developer, Tester | * Read and understand specify paper * Prepare document and write paper * Coding * Testing |
| 4 | Hoàng Tuấn Anh | Developer, Tester | * Read and understand specify paper * Prepare document and write paper * Coding * Testing |
| 5 | Phạm Phú Tiến | Developer, Tester | * Read and understand specify paper * Prepare document and write paper * Coding * Testing |

Table : Responsibilities

### Tools and Techniques

|  |  |
| --- | --- |
| **Tool/ Technique** | **Name / Version** |
| Language | Python3 |
| GUI libraries | PyQt, Matplotlib |
| Google Map API libraries | googlemaps |
| Other libraries | networkx, decorator, numpy, sip |

Table : Tool and Techniques

## Project Management Plan

### Software development life cycle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phase** | **Description** | **Deliverables** | **Resource needed** | **Dependencies and Constrains** | **Risks** |
| **Initialization and basic research** | -Learn python  -Research for machine learning basic and reinforcement learning | -Basic Knowledge of python  -Understand about machine learning and reinforcement learning  -Introduction document  -Project Management Plan | 3 weeks | N/A | -Misunderstanding of machine learning  -Focus wrong things. |
| **Read paper and design** | -Reading two paper:  +Ant-Q\_ A Reinforcement Learning approach to the traveling salesman problem  +Reinforcement Learning For Solving Shortest-Path And Dynamic Scheduling Problems  -Design demo application | -Understand two papers and algorithms in papers  -Software Requirement Specification  -Software Design Description | 2 weeks | Depend on “Initialization and basic research” | -Cannot understand paper and equation in paper. |
| **Implement demo application** | -Use algorithms in two paper to implement a demo application for them. | -A demo application | 3 weeks | Depend on “Read paper” | -Implement wrong algorithms. |
| **Test demo application** | Prepare test plan and test cases  Test all functions and results | -System Implementation and Test | 2 week | Depend on “Read paper and design” and “Implement demo application” |  |
| **Complete report document** | Write “Software User’s Manual”  Review all documents | - Software User’s Manual | 2 weeks | Depend on “Read paper and design” | -Don’t have enough resource to complete documents. |
| **Write paper** | Write paper | -A paper | 2 weeks | Depend on “Implement demo application” and “Read paper” | -Don’t have enough time to complete paper.  -Don’t know and not use scientific terminology in paper. |

Table : Software development life cycle

### Phase Detail

#### Initialization and basic research

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **Learn python** | Learn python basic  Learn how to use numpy and matplotlib libraries | NhaPHD, MinhPQ, AnhHT, TienPP |
| **Study on machine learning and reinforcement learning basic** | Study on reinforcement learning basic: Policy, Reward Signal, Value Function, Model of the Environment, The Simulated Environment  Study on Supervised Learning, Unsupervised Learning, Semi-Supervised Learning and compare with Reinforcement Learning  Study on Markov Decision Process (MDP), Q-learning, Ant-Q | NhaPHD, MinhPQ, AnhHT, TienPP |
| **Write document** | Write Introduction document  Write Project Management Plan | NhaPHD, MinhPQ, AnhHT, TienPP |

Table : Initialization and basic research

#### Read paper and design

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **Read “Ant-Q\_ A Reinforcement Learning approach to the traveling salesman problem” paper** | Understand paper and Ant-Q algorithm | NhaPHD, MinhPQ |
| **Read “Reinforcement Learning For Solving Shortest-Path And Dynamic Scheduling Problems” paper** | Understand paper and Q-learning algorithm | AnhHT, TienPP |
| **Design demo application** | Design main function and GUI of demo application | NhaPHD, MinhPQ, AnhHT, TienPP |
| **Write document** | Write Software Requirement Specification  Write Software Design Description | NhaPHD, MinhPQ, AnhHT, TienPP |

Table : Read paper and design

#### Implement demo application

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **Implement GUI** | Implement GoogleMap, Log Window and Main Window | NhaPHD, MinhPQ, AnhHT, TienPP |
| **Implement Main functions** | Implement comparing function, log management and result visualization | NhaPHD, MinhPQ, AnhHT, TienPP |
| **Implement Algorithms** | Implement AntQ, K-means++, SA, ACO algorithm | NhaPHD, MinhPQ, AnhHT, TienPP |

Table : Implement demo application

#### Test demo application

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **Prepare test plan** | Prepare overview plan for testing | NhaPHD, MinhPQ, AnhHT, TienPP |
| **Prepare test case** | Provide check list for function and application | NhaPHD, MinhPQ, AnhHT, TienPP |
| **Test demo application** | Test demo application base on check list | NhaPHD, MinhPQ, AnhHT, TienPP |
| **Write document** | Write System Implementation and Test | NhaPHD, MinhPQ, AnhHT, TienPP |

Table : Test demo application

#### Write report document

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **Write document** | Write Software User’s Manual | NhaPHD, MinhPQ, AnhHT, TienPP |
| **Review all document** | Review report 1, 2, 3, 4, 5, 6 | NhaPHD, MinhPQ, AnhHT, TienPP |

Table : Write report document

#### Write paper

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **Write abstract and introduction** | N/A | NhaPHD, MinhPQ, AnhHT, TienPP |
| **Write about Ant-Q and Q-learning algorithm** | N/A | NhaPHD, MinhPQ, AnhHT, TienPP |
| **Write comparison** | N/A | NhaPHD, MinhPQ, AnhHT, TienPP |
| **Write conclusion** | N/A | NhaPHD, MinhPQ, AnhHT, TienPP |

Table : Write paper

### All Meeting Minutes

All meeting documents could be found [Meeting Minutes](https://docs.google.com/document/d/1I8FJTyIz-2_YblsUcHD7N9KDTqa7IkmbjVBbhv_R_24/edit)

## Coding Convention

Overview we use the convention of python coding, we also prefer to use PEP8. Besides that, we have some our specified coding convention:

* Back-end side: we prefer to use PEP8.
  + Variable Naming:
* Naming convention: snake case (lower\_case\_with\_underscores)
* Minimum length: 2.
* Maximum length: 50.
  + Function Naming:
* Naming convention: camel case (lowerCaseThenUpperCase)
* Minimum length: 2.
* Maximum length: 50.
* The first word: Verb (getSomething)
  + Class Naming:
    - Naming convention: Upper camel case (UpperCamelCase)
    - Minimum length: 2.
    - Maximum length: 50.
    - Meaning: The whole name must be a noun.
  + Constant Naming:
    - Naming convention: Constants should be written in uppercase characters separated by underscores. Constant names may also contain digits if appropriate, but not as the first character.
    - Minimum length: 2.
    - Maximum length: 50.
  + Comments:
    - Always comment on the previous line of the function or line of code that you want to take note.
    - The comment line must be packaged in one line.
  + Blank Lines: Two blank lines between top-level definitions, one blank line between method definitions.
  + Shebang Line: Most .py files do not need to start with a #! line. Start the main file of a program with #!/usr/bin/env python with an optional single digit 2 or 3 suffix.
  + Whitespace: Follow standard typographic rules for the use of spaces around punctuation.
* Front-end side: We prefer to use w3school coding conventions.
  + Naming Conventions:
    - Variable and function names written as camelCase
    - Global variables written in UPPERCASE (We don't, but it's quite common)
    - Constants (like PI) written in UPPERCASE
  + Hyphens in HTML and CSS:
    - HTML5 attributes can start with data- (data-quantity, data-price).
    - CSS uses hyphens in property-names (font-size).
  + Accessing HTML Elements:
    - Use the same naming convention (as JavaScript) in HTML.

# Software Requirement Specification

## User Requirement Specification

We build a demo application for researching so the demo application gives everyone permission to do the same functions like each others. We call the people who interact with the application is referred to as “USER”.

User can use all functions available in the application:

* Algorithm View:
  + Generate JSON Data File contains a set of locations data.
  + Adjust algorithm parameters
  + Apply algorithm parameters
  + Run selected algorithm.
  + Save the record as a log.
  + Run the selected log and show list route result on google maps.
  + Select cluster and show the selected cluster on google maps.
* Log View:
  + Search logs by their name and algorithm.
  + Add selected logs to the comparing table.
  + Comparing the results between logs.
  + Rename the selected log on the tree list view.
  + Remove the selected log from the tree list view.
  + View the information of the selected log.
  + Remove the selected log from the comparing table.

## System Requirement Specification

### External Interface Requirement

#### User Interface

* General requirement for graphics user interface is that the GUI should be simple, clear, intuitive and reminiscent.
* Depend on the level of importance of each component, its position and its size will be more special than the others helps user can focus on it.
* By using flat and minimalist design, the depth of each objects is also designed clearly and subtly.
* We don’t use any specific principles for a desktop application. Instead of that, we applied a general principle named “DOET” which stands for “Design of Everyday Things”.

#### Hardware Interface

* Laptop or Computer which is connected to Internet via wifi or cable.
* The device also need to meet some minimum requirements:
* CPU :    Intel, Core i3
* RAM :     4 GB
* Hard Drive :    HDD
* Graphic Card : Onboard Card/ Intel Graphic
* Connect port :  LAN : 10/100/1000 Mbps, WIFI : IEEE 802.11

#### Software Interface

* Desktop application: recommended Windows (7 or above)

### System Overview Use Case

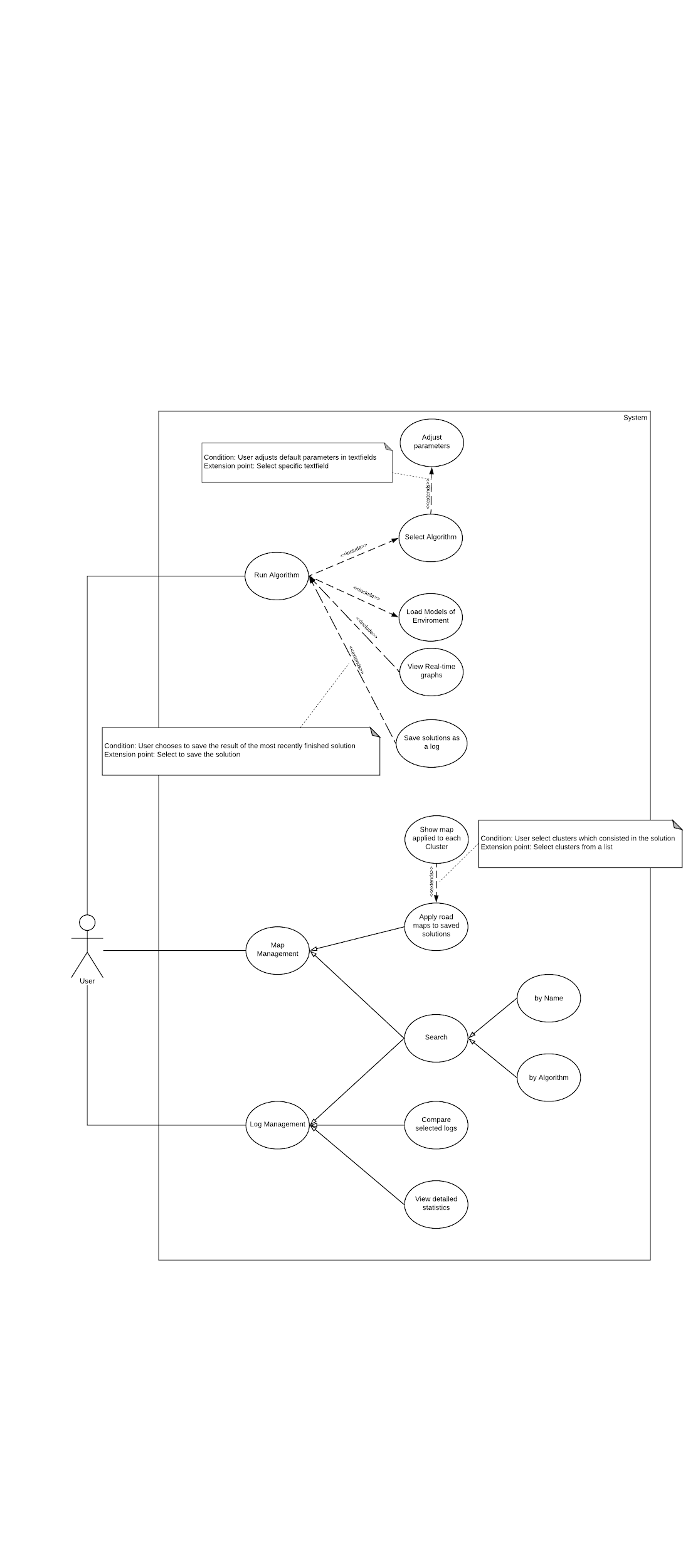


Figure : Use-case overview

# Software Design Description

## Design Overview

* This document describes the detailed descriptions and User Interface design of the demo Application System.
* The detailed descriptions of the system describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.
* The user interface describes the functionality of each component of the UI.
* Document overview:
  + Section 2: gives the detail design description which includes class diagram, class explanation, and sequence diagram to details the application functions.
  + Section 3: describe algorithms.

## Detailed Description

### Class Diagram

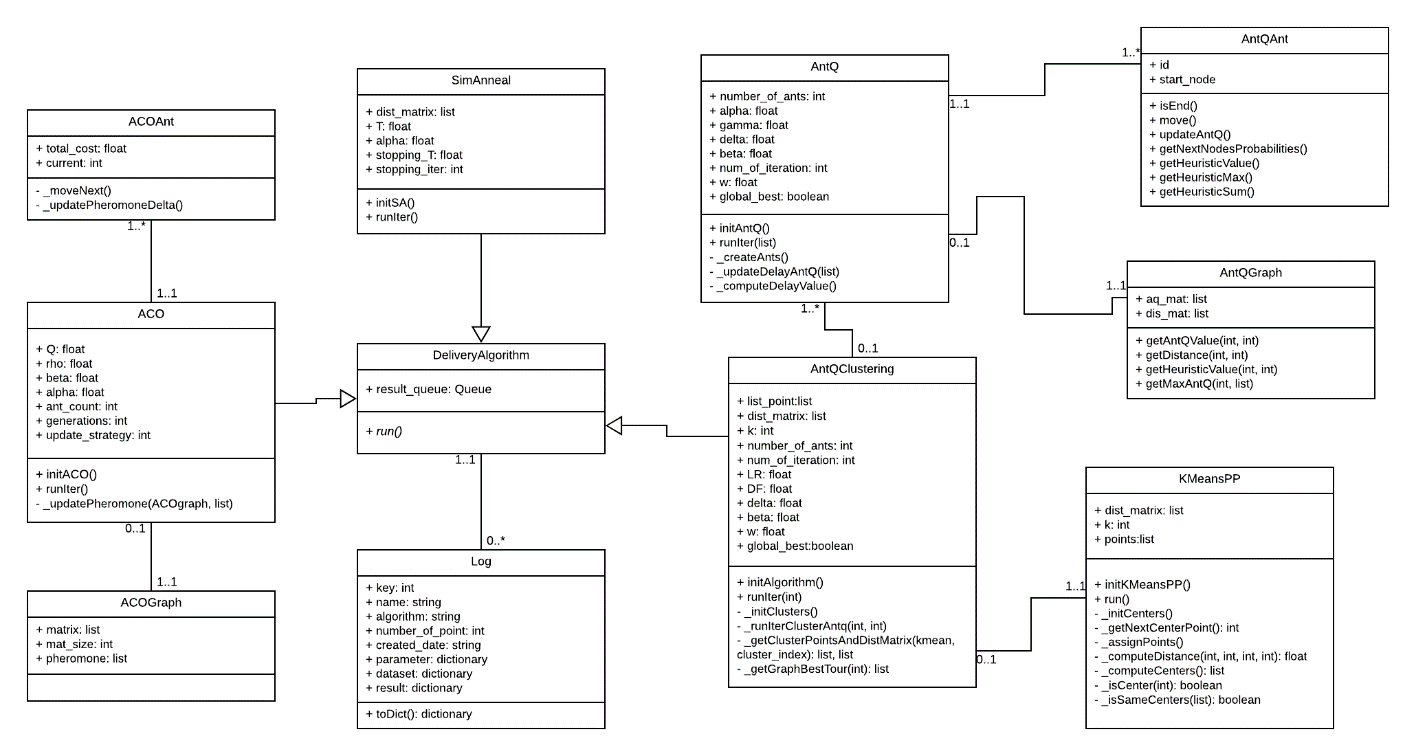


Figure : Class diagram

|  |  |
| --- | --- |
| Class dictionary: describe Class | |
| **Class Name** | **Description** |
| ACOAnt | Contain the Ant Colony Optimization Algorithm’s Ant informations and methods. |
| ACO | Contain parameter and implementation of Ant Colony Optimization Algorithm |
| ACOGraph | Contain informations and methods of Ant Colony Optimization Algorithm’s Graph |
| SimAnneal | Contain parameter and implementation of SA Algorithm |
| Ant | Contain the AntQ Algorithm’s Ant informations and methods. |
| AntQ | Contain parameter and implementation of AntQ Algorithm |
| AntQGraph | Contain informations and methods of AntQ Algorithm’s Graph |
| KmeansPP | Contain parameter and implementation of KMean Algorithm |
| AntQClustering | Contain parameter and implement the combination of AntQ and Kmean |
| Log | Contain the information of log |
| DeliveryAlgorithm | An parent class contains the methods and variables which every algorithm have to implement |

Table : Class dictionary

## Algorithms

### Delivery algorithm

#### Definition

The travelling salesman problem (TSP) asks the following question: "Given a list of locations and the distances between each pair of locations, what is the shortest possible route that visits each location and returns to the origin location?" It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

#### Define Problem

We have a list of locations what can perform on google map. Each location has two attributes that are latitude and longitude. Google also provides some services to solve this problem but we can’t change any parameters between two locations on it and Google only gives us a fixed result with the same list locations. We need more than what Google can do that finding out the best route that visits each location in unstable environment. The exterior conditions can change time, distance which is parameter we travel between each pair of locations such as the weather, traffic situation, etc.

#### Solution

To solve this problem, we use Ant-Q algorithm. By applying Ant-Q, we can create multi agents to find the best route. Given a set of n locations, and for each pair of locations a distance drs (distance from R to S), the TSP is stated as the problem of finding a minimal length closed tour that **visits each location once**. An instance of the TSP is given by a graph (N,E), where:

* + - N,|N| =n, is the set of locations
    - E is the set of edges between locations (a fully connected graph in the Euclidean TSP).

In case **drs≠dsr** we have the more general asymmetric traveling salesman problem (ATSP). Ant-Q algorithms apply indifferently to both problems.

Before going into the formula, we briefly go through the parameters

* **AQ(r,s):** is a positive real value associated to arc (r,s) and is the AntQ algorithm counterpart of Q-learning Q-values. AQ(r,s)'s are changed at run time and are intended to indicate how useful it is to make move s (i.e., to go to location s) when in state r
* **HE(r,s):** is a heuristic function which evaluates the goodness of move s when in location r. For example, in the ATSP HE(r,s) is the inverse of the distance between locations r and s.
* **δ and β:** weigh the relative importance of the learned AQ-values and the heuristic values
* **q:** is a value chosen randomly with uniform probability in [0, 1], and q0 (0≤q0≤1) is a parameter: the smaller q0 the higher the probability to make a random choice
* **S:** is a random variable selected according to the distribution which gives the probability with which an ant in location r chooses the location s to move to.

An agent k situated in location r moves to location s using the following rule, called action choice rule (or state transition rule):

* Pseudo-Random

(1)

* Random-Proportional

(2)

* Pseudo-Random-Proportional is a compromise between the pseudo-random state choice rule typically used in Q-learning and the random-proportional action choice rule typically used in Ant System.

We apply the third way Pseudo-Random-Proportional.

The next step is determining AQ-value, AQ-value will be updated by the following rule:

(3)

We tested two types of delayed reinforcement, called global-best and iteration-best. The reinforcement ∆AQ is always zero except after each agent has completed its tour. ∆AQ(r,s) is computed by the following formula:

* Global-best

(4)

where W is a parameter which in all experiments was set to W=10 (this value was found to be a good one with AS, and was not optimized for Ant-Q), is the agent who made the globally best tour from the beginning of the trial, and is its tour length.

* Iteration-best

(5)

where is the agent who made the best tour in the current iteration of the trial, and is its tour length.

We set a parameter called Iteration limitation. When number of iterations meets the condition, the result will be given by all the steps above.

#### Complexity

Where n is number of locations and m is number of agents. m should belong to the range [0.6·n,  n]. So the complexity of the algorithm is:

(6)

#### Flowchart

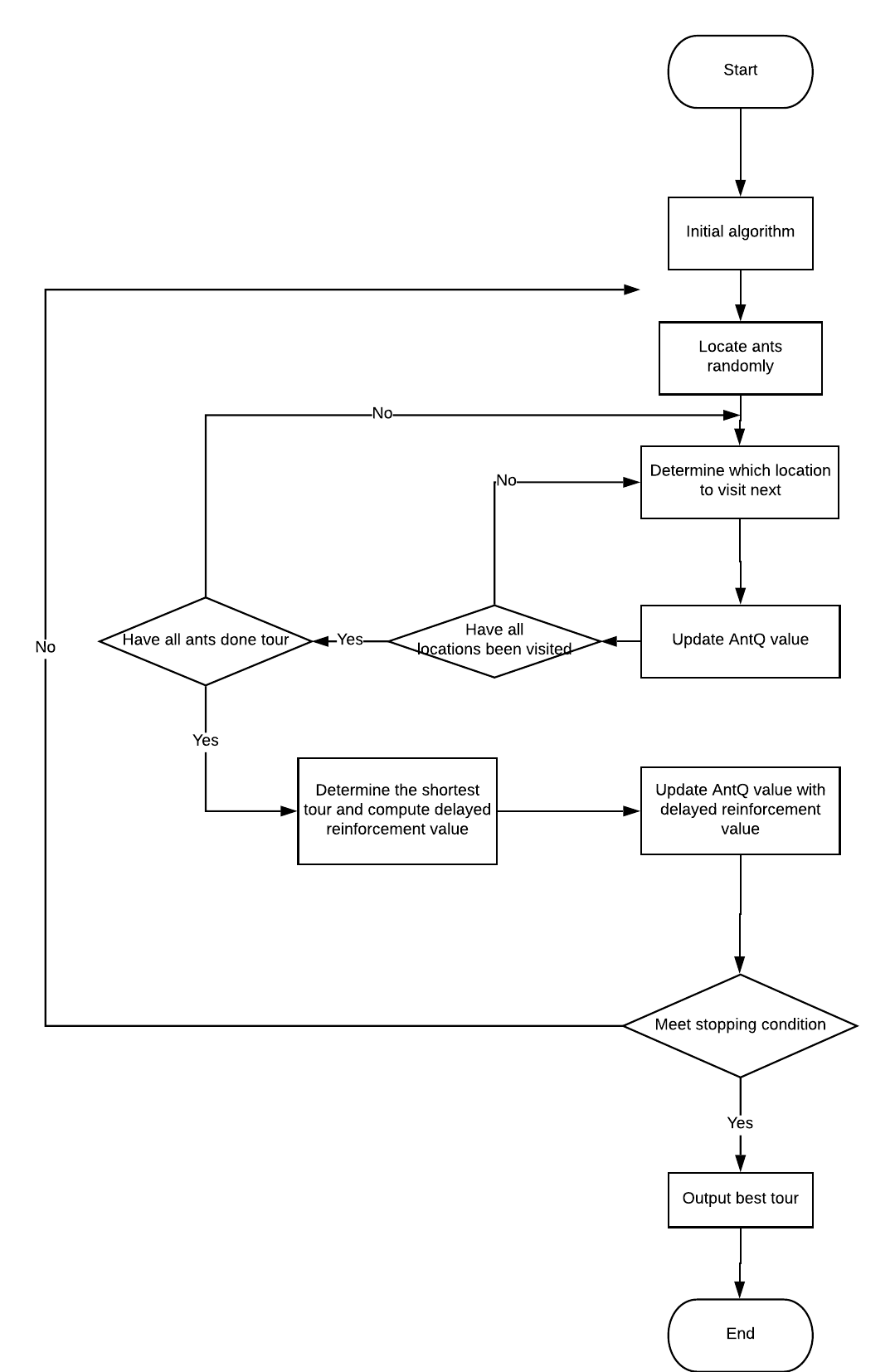
****

Figure : Ant-Q flowchart

### K-means++ algorithm

#### Definition

K-mean is the basic and most common clustering algorithm of Unsupervised Learning – a type of Machine Learning . The standard algorithm was first proposed by Stuart Lloyd in 1957. In 1975, it was introduced in detail by Hartigan. With clustering problem, we have N points in M dimensions, k-mean is used to divide this N points into k cluster (group) with k has given.

#### Define problem

As described in the delivery problem, we have a list of locations that we have to deliver to. To improve the performance of delivery, we divide them into several groups. We assume that each location is a point in 2-dimensions space then separating those points into several clustering group.

#### Solution

To solve this problem, we use k-mean algorithm. Given n points in m dimensions and the number of cluster k. We find which cluster each point belongs to. In the standard k-mean algorithm, we will choose initial k centers randomly. According to experiment, if we have the good initial centers, the result will be better. In 2007, David Arthur and Sergei Vassilvitskii proposed a improved k-means algorithm, called **k-means++ algorithm**.

K-means algorithm performs the following steps:

* Step 1a: Choose one center uniformly at random from among the data points.
* Step 1b: For each point x, computing the distance D(x) between point x and the nearest center of it that has already been chosen. Then, choose a new point at random as a new center with probability
* Step 1c: Repeat step 1b until k centers have been chosen.
* Step 2: Compute the distance between each point and centers.
* Step 3: Assign each point to the cluster whose center has the minimum distance to that point.
* Step 4: Update center point for each cluster by computing mean of points in this cluster which is determined in assignment step.
* Step 5: Repeat step 2 until there is no changes of points in clusters.

#### Complexity

Complexity of K-Means algorithm O(n\*k\*d\*i), where

* n: number of points.
* k: number of clusters.
* d: number of dimensional
* i: number of iterations.

#### Flowchart

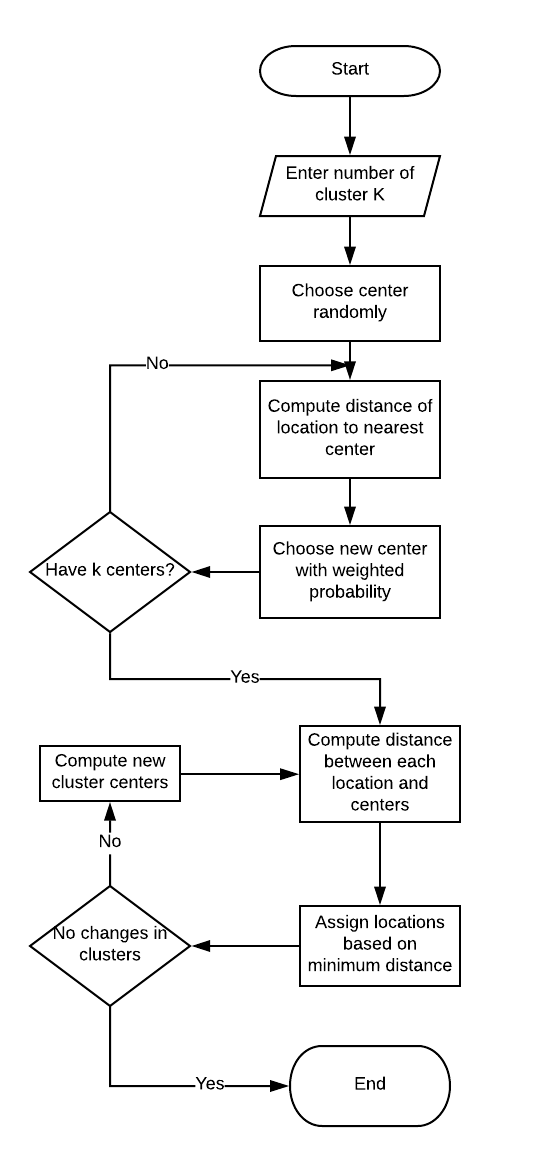
****

Figure : K-means++ flowchart

### Other heuristic algorithms

#### Ant Colony Optimization algorithm

Ant colony optimization algorithm (ACO) is a probabilistic technique for solving delivery problem. Initially proposed by Marco Dorigo in 1992 in his PhD thesis.

The base of ACO is simulating the real behaviour of ants in nature. Ants deposit pheromone on the way while walking, the others can follow the previous pheromone by probability while passing by. The higher pheromone, the higher probability with which the ants will follow. Because it takes less time for ants to find food and carry them back to the nests through the shortest path, after a long enough period of time, more ants will choose this shortest path and deposit pheromone on it. In this way, all ants eventually choose the shortest path.

Flowchart:

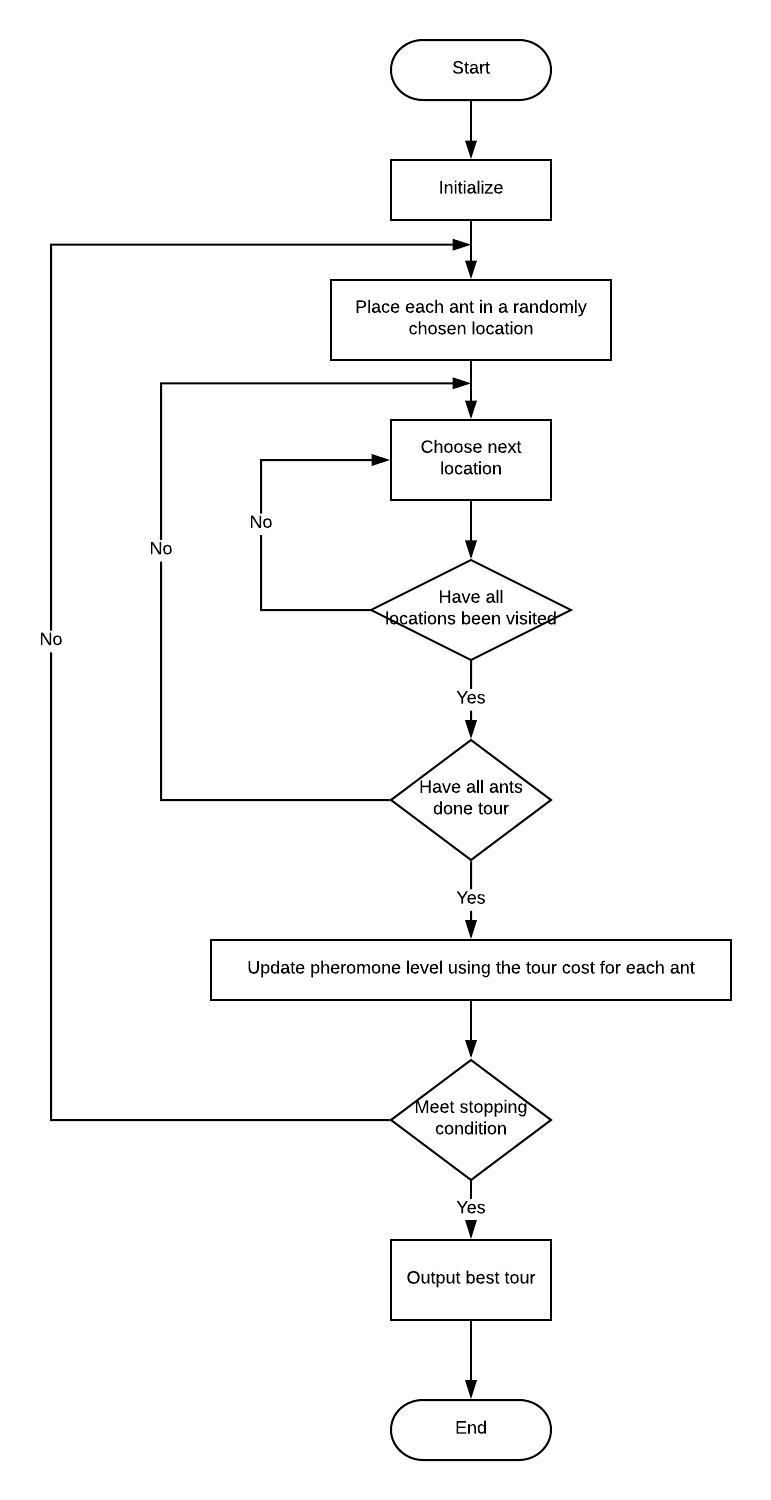


Figure : ACO flowchart

#### SA algorithm

SA algorithm was a method introduced in 1983 by Scott Kirkpatrick, Daniel Gelatt and Mario Vecchi. It is a popular intelligent optimization algorithm which has been successfully applied in many fields, include solving delivery problem.

With SA, a random tour is chosen and its permutation is slightly modified by switching the order of at least two points to obtain a new tour. If this new tour’s distance is shorter than the original tour, it will become the new tour of the interest. Else if this new tour has a greater distance than the original tour, there will exist some probability that this new tour is adopted anyways. At each step along the way, this probability decreases to 0 until a final solution is settled upon.

Flowchart:

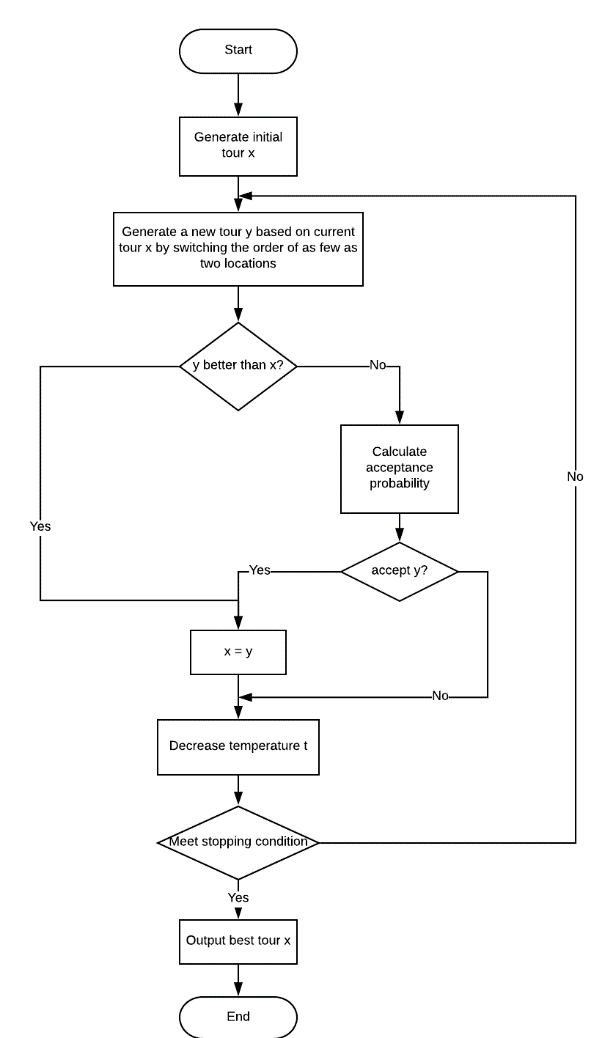


Figure : SA flowchart

# Task Sheet

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Product Deliverables** | **Task** | **NhaPHD** | **MinhPQ** | **AnhHT** | **Unit** | **Size** |
| 1 | **Report1 - Introduction** | Project Information |  |  | **O** |  | 1 |
| Introduction |  |  | **O** |  | 1 |
| Current Situation | **O** |  |  |  | 1 |
| Problem Definition |  |  | **O** |  | 1 |
| Proposed Solution | **O** |  |  |  | 1 |
| Role and Responsibility |  |  | **O** |  | 1 |
| 2 | **Report2- Software Project Management Plan** | **Problem Definition** |  |  |  |  |  |
| Name of this Capstone Project |  |  | **O** |  | 1 |
| Problem Abstract |  | **O** |  |  | 1 |
| Project Overview |  | **O** |  |  | 1 |
| **Project organization** |  |  |  |  |  |
| Roles and responsibilities |  |  | **O** |  | 1 |
| Tools and Techniques | **O** |  |  |  | 1 |
| **Project Management Plan** |  |  |  |  |  |
| Software development life cycle | **O** |  |  |  | 1 |
| **Phase Detail** |  |  |  |  |  |
| Initialization and basic  research | **O** |  |  |  | 1 |
| Read paper | **O** |  |  |  | 1 |
| Implement demo  application | **O** |  |  |  | 1 |
| Write report document | **O** |  |  |  | 1 |
| Write paper | **O** |  |  |  | 1 |
| All Meeting Minutes |  | **O** |  |  | 1 |
| Coding Convention |  |  | **O** |  | 1 |
| 3 | **Report 3- Software Requirement Specification** | User Requirement Specification |  |  | **O** |  | 1 |
| **System Requirement Specification** |  |  |  |  |  |
| **External Interface**  **Requirement** |  |  |  |  |  |
| User Interface |  | **O** |  |  | 1 |
| Hardware Interface |  | **O** |  |  | 1 |
| Software Interface |  | **O** |  |  | 1 |
| System Overview Use Case |  |  | **O** |  | 1 |
| **List of Use Case** |  |  |  |  |  |
| **Run Algorithm** |  |  |  |  |  |
| Adjust  parameters |  |  | **O** |  | 1 |
| Select an  algorithm |  |  | **O** |  | 1 |
| Load model of  environment | **O** |  |  |  | 1 |
| Run the algorithm |  |  | **O** |  | 1 |
| View real-time  graph |  |  | **O** |  | 1 |
| Save solution as  log |  | **O** |  |  | 1 |
| **Map management** |  |  |  |  |  |
| Search solutions |  | **O** |  |  | 1 |
| Apply road map  to saved solutions |  | **O** |  |  | 1 |
| Apply road map  to each cluster |  | **O** |  |  | 1 |
| **Log Management** |  |  |  |  |  |
| Compare selected  logs | **O** |  |  |  | 1 |
| View detailed  statistics | **O** |  |  |  | 1 |
| Software System Attribute | **O** |  |  |  | 1 |
| 4 | **Report 4- Software Design Description** | Design Overview |  | **O** |  |  | 1 |
| **Detailed Description** |  |  |  |  |  |
| Class Diagram | **O** |  |  |  | 4 |
| Class Diagram Explanation | **O** |  |  |  | 1 |
| **Interaction Diagram** |  |  |  |  |  |
| Run AntQ and K-Means  algorithm |  |  | **O** |  | 1 |
| Run ACO Algorithm |  |  | **O** |  | 1 |
| Run SA Algorithm |  | **O** |  |  | 1 |
| Get List Log |  | **O** |  |  | 1 |
| Compare logs |  | **O** |  |  | 1 |
| **User Interface Design** |  |  |  |  |  |
| **Main User Interface** |  |  |  |  |  |
| Graph UI |  | **O** |  |  | 1 |
| Google Maps UI |  | **O** |  |  | 1 |
| Log User Interface |  | **O** |  |  | 1 |
| **Algorithms** |  |  |  |  |  |
| AntQ algorithm |  |  | **O** |  | 5 |
| K-means++ algorithm | **O** |  |  |  | 4 |
| **Other heuristic algorithms** |  |  |  |  |  |
| Ant Colony Optimization  algorithm |  | **O** |  |  | 3 |
| Simulated Annealing  algorithm |  | **O** |  |  | 3 |
| 5 | **Report 5 - Software Implementation and Test Document** | **Introduction** |  |  |  |  |  |
| Overview |  |  | **O** |  | 1 |
| Test Approach |  |  | **O** |  | 1 |
| Performance Measures |  |  | **O** |  | 1 |
| Test Plan |  | **O** |  |  | 1 |
| **Application Implementation** |  |  |  |  |  |
| **GUI** |  |  |  |  |  |
| LogWindow | **O** |  |  |  | 1 |
| **MainWindow** |  |  |  |  |  |
| GraphUI |  |  | **O** |  | 1 |
| GoogleMap UI |  | **O** |  |  | 1 |
| **Algorithm** |  |  |  |  |  |
| K-mean | **O** |  |  |  | 4 |
| AntQ |  |  | **O** |  | 4 |
| Combination of Kmean  and AntQ | **O** |  |  |  | 3 |
| Simulated Annealing |  | **O** |  |  | 4 |
| Ant Colony Optimization |  | **O** |  |  | 4 |
| **Function** |  |  |  |  |  |
| Visualize result | **O** |  |  |  | 2 |
| Show log on GoogleMap |  | **O** |  |  | 1 |
| Compare log |  |  | **O** |  | 2 |
| Manage logs | **O** |  |  |  | 1 |
| **System Testing Test Case** |  |  |  |  |  |
| GUI | **O** |  |  |  | 1 |
| **Main Window Function** |  |  |  |  |  |
| Graph UI Function |  |  | **O** |  | 1 |
| Googlemap UI Function |  |  | **O** |  | 1 |
| Log Window Function |  |  | **O** |  | 1 |
| 6 | **Report 6 - Software User's Manual** | **Installation** |  |  |  |  |  |
| Prepare dependencies | **O** |  |  |  | 1 |
| Install application | **O** |  |  |  | 1 |
| **User guide** |  |  |  |  |  |
| Generate data | **O** |  |  |  | 1 |
| Apply new parameter |  | **O** |  |  | 1 |
| Run algorithm |  | **O** |  |  | 1 |
| Run on Google Map |  | **O** |  |  | 1 |
| View a cluster on googlemap |  |  | **O** |  | 1 |
| Search logs |  |  | **O** |  | 1 |
| Add log to table |  |  | **O** |  | 1 |
| Compare logs |  |  | **O** |  | 1 |

Table : Task sheet

|  |  |  |
| --- | --- | --- |
| **Notes** | |  |
| **No.** | **Function Types** | **Function Point Size** |
| 1 | Insert, Delete, Update, Simple Search, Login, Register, simple function or simple query on one table | 1 |
| 2 | Computed functions or joined query | 2 |
| 3 | Complex functions or multiple subquery | 3 |
| 4 | Constraint process and data integrity | 4 |
| 5 | Functions with implementing algorithms | 5 |

# Appendix

1. Luca M. Gambardella , Marco Dorigo. Ant-Q: A Reinforcement Learning approach to the traveling salesman problem
2. Péter Stefán , László Monostori , Ferenc Erdélyi. REINFORCEMENT LEARNING FOR SOLVING SHORTEST-PATH AND DYNAMIC SCHEDULING PROBLEMS
3. Gilbert L. Peterson, Gary B. Lamont, Maj Christopher B. Mayer. Scaling Ant Colony Optimization with Hierarchical Reinforcement Learning Partitioning
4. Bob Story . Reinforcement Learning with Python
5. Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, andbAngela Y. Wu. An Efficient k-Means Clustering Algorithm: Analysis and Implementation
6. David Arthur, Sergei Vassilvitskii. k-means++: The Advantages of Careful Seeding
7. Machine learning – Wikipedia: https://en.wikipedia.org/wiki/Machine\_learning
8. Reinforcement learning – Wikipedia: https://en.wikipedia.org/wiki/Reinforcement\_learning
9. Supervised learning – Wikipedia: https://en.wikipedia.org/wiki/Supervised\_learning
10. Cluster analysis – Wikipedia: https://en.wikipedia.org/wiki/Cluster\_analysis
11. Markov property – Wikipedia: https://en.wikipedia.org/wiki/Markov\_property
12. Markov chain – Wikipedia: https://en.wikipedia.org/wiki/Markov\_chain
13. Markov decision process – Wikipedia: https://en.wikipedia.org/wiki/Markov\_decision\_process
14. Q-learning – Wikipedia: https://en.wikipedia.org/wiki/Q-learning
15. k-means++ - Wikipedia: https://en.wikipedia.org/wiki/K-means%2B%2B
16. k-means clustering - Wikipedia: https://en.wikipedia.org/wiki/K-means\_clustering
17. Ant colony optimization algorithms - Wikipedia: https://en.wikipedia.org/wiki/Ant\_colony\_optimization\_algorithms
18. Simulated Annealing – Wikipedia: https://en.wikipedia.org/wiki/Simulated\_annealing
19. Travelling salesman problem - Wikipedia:https://en.wikipedia.org/wiki/Travelling\_salesman\_problem
20. NP-hardness - Wikipedia: https://en.wikipedia.org/wiki/NP-hardness
21. K-means Clustering – Machine Learning Cơ Bản: https://machinelearningcoban.com/2017/01/01/kmeans/
22. Giới thiệu về Machine Learning - Machine Learning Cơ Bản: https://machinelearningcoban.com/2016/12/26/introduce/
23. Phân nhóm các thuật toán Machine Learning - Machine Learning Cơ Bản: https://machinelearningcoban.com/2016/12/27/categories/
24. Simulated Annealing applied to the Traveling Salesman Problem: https://pdfs.semanticscholar.org/e34f/94561872a20130addfaec69de096582516de.pdf
25. Donald A. Norman. Design of Everyday Things
26. Federico Greco. Travelling Salesman Problem