Ha Noi University of Science and Technology

School of Information & Communication Technology

Software Requirement Specification – SRS

Version 1.0

Evolutionary Algorithm Visualization

Subject: Software Engineering

Group No. 5

Lương Minh Triết

Phạm Vũ Huy

Nguyễn Xuân Phong

*Hà Nội, ngày 10 tháng 5 năm* *2021*

Table of Content

Table of Content ii

1 Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 Glossary 1

1.4 Reference Document 2

2 General Discription 3

2.1 Actor 3

2.2 General Use Case Diagram 3

2.3 Decompose Use Case Diagram 3

2.3.1 Decompose Use Case “Input Solution” 3

2.4 Professional Process 4

2.4.1 Software Usage Process 4

2.4.2 Input Process 5

3 Functional Requirement 6

3.1 Use Case UC001 “Input Solution” 6

3.2 Use Case UC002 “Visualization” 7

1. **Introduction**
   1. ***Purpose***

This document demonstrates evolutionary algorithm which contains Genetic, Algorithm, Particle Swarm Optimization and Hill Climbing algorithm. All the algorithms are visually demonstrated using JavaFX. This document describes the purposes and the functionalities of the application, the interface and how the application reacts to interactions.

* 1. ***Scope***

Evolutionary algorithms are a heuristic-based approach to solving problems that cannot be easily solved in polynomial time and anything else that would take far too long to exhaustively process. When used on their own, they are typically applied to combinatorial problems.

An EA contains four overall steps: **initialization, selection, genetic operators, and termination**. These steps each correspond, roughly, to a particular facet of natural selection, and provide easy ways to modularize implementations of this algorithm category.

In this application, the users are able to provide and create an initial population of solutions. Then, they will be able to choose one of the three algorithms to solve the problem and the chosen approach will be visually demonstrated to the users.

* 1. ***Glossary***

- **Genetic algorithms: Genetic algorithms simulate the process of natural selection to generate high-quality solutions for optimization problems and search problems.**

* Particle swarm optimization: The general idea of PSO is inspired by a flying swarm of birds searching for food. A PSO is used to track the local mode of the similarity measure and to seek a good local minimum, and then the conjugate gradient is utilized to find the local minimum accurately.
* Hill climbing: Hill-climbing solves the problems where we need to maximize or minimize a given real function by choosing values from the given inputs. This solution may not be the global optimal maximum.
  1. ***Reference documents***

- <https://www.geeksforgeeks.org/genetic-algorithms/>

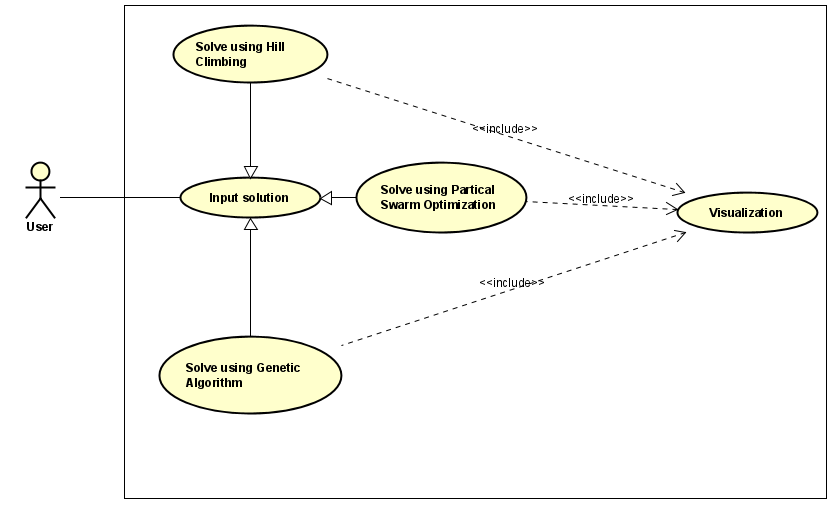
- <https://www.sciencedirect.com/topics/engineering/particle-swarm-optimization>

- <https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/>

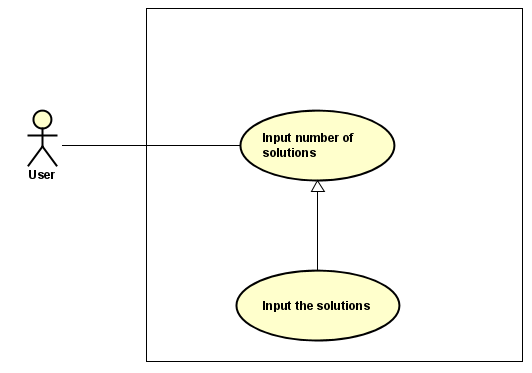
1. **General description**
   1. ***Actors***

The application includes 1 actor: Users

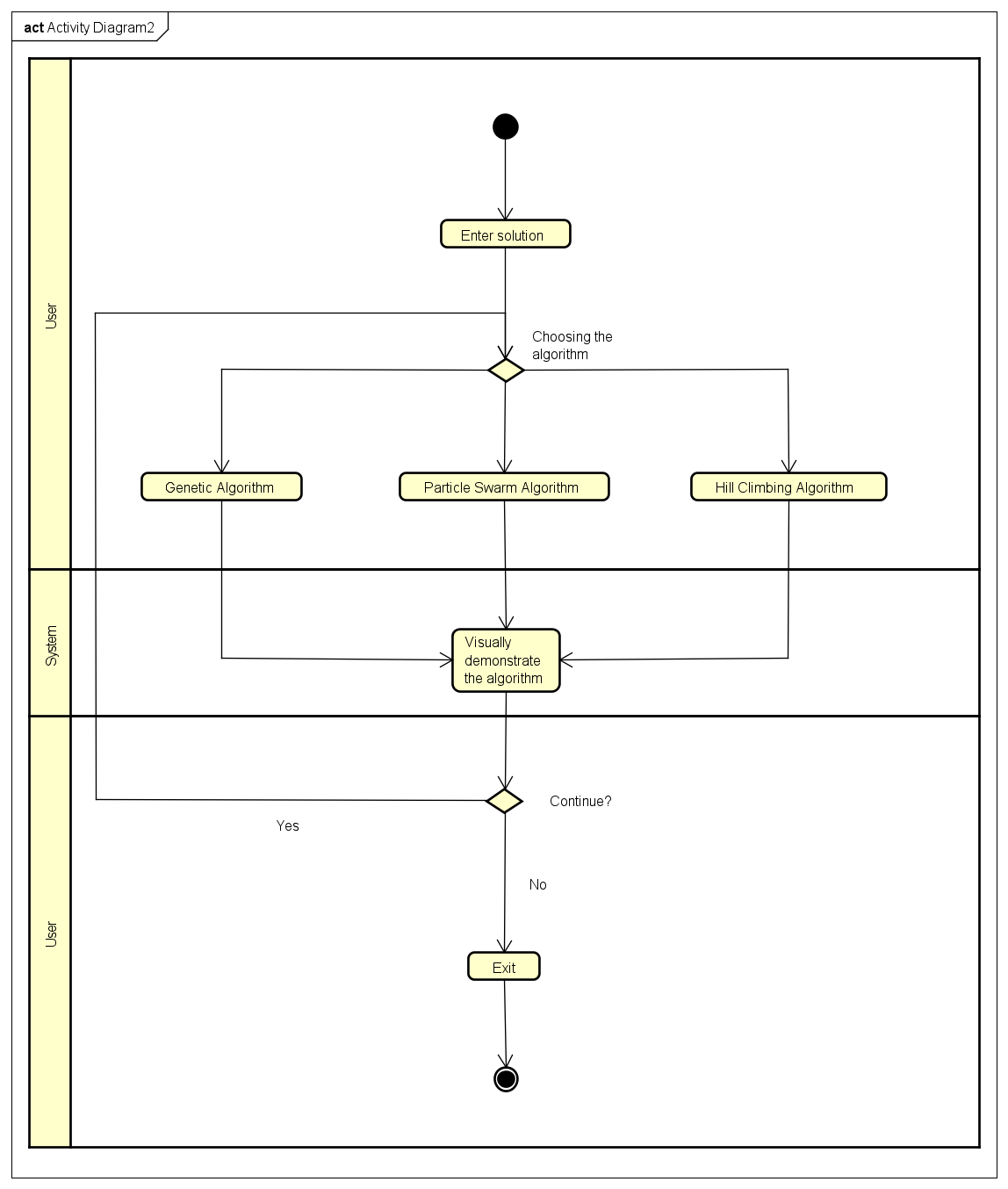
* 1. ***General Use Case Diagram***



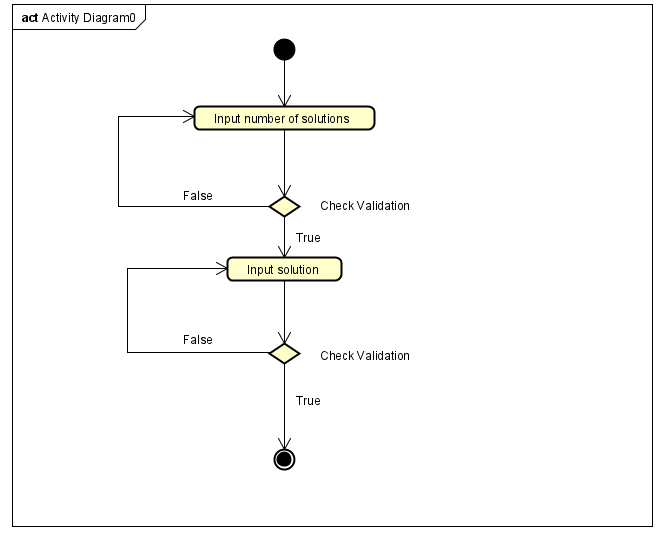
* 1. ***Decompose Use Case Diagram***
     1. ***Decompose Use Case “Input Solution”***



* 1. ***Professional Process***
     1. ***Software Usage Process***



* + 1. ***Input Process***



1. **Functional Requirements**
   1. ***Use Case UC001 “Input Solution”***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Code** | UC001 | **Use Case Name** | Input Solution |
| **Trigger** | User | | |
| **Precondition** | None | | |
| **Basic Path** | |  |  |  | | --- | --- | --- | | **No.** | **Performed by** | **Action** | | 1. | System | Display the interface for user to input data | | 2. | User | Input the population of solutions randomly | | 3. | System | Check if the number sastisifies the condition | | 4. | User | Input solutions | | 5. | System | Check if the solutions sastisifies  the condition | | 6. | System | Call Use Case “Giving visual representation” | | | |
| **Alternative Path** | |  |  |  | | --- | --- | --- | | **No.** | **Performed by** | **Action** | | 3a. | System | Inform user about invalid input and let them input again | | 5a. | System | Inform user about invalid input and let them input again | | | |
| **Postcondition** | None | | |

* 1. ***Use Case UC002 “Visualization”***

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Code** | UC002 | **Use Case name** | Visualization |
| **Trigger** | System | | |
| **Precondition** | Graph inputted successfully | | |
| **Basic Path** | |  |  |  | | --- | --- | --- | | **No.** | **Performed by** | **Action** | | 1. | System | Display algorithm menu | | 2. | User | Choose algorithm to visualize | | 3. | System | Run chosen algorithm | | 4. | System | Display graph visualization when graph is completed | | 5. | User | Choose to exit or try other algorithms | | | |
| **Alternative Path** | |  |  |  | | --- | --- | --- | | **No.** | **Performed by** | **Action** | | 3a. | System | If graph is not completed then update graph | | 5a1. | System | If user choose to exit then end use case | | 5a2. | System | If user choose to try other algorithms then call Use Case “Giving visual representation” again | | | |
| **Postcondition** | None | | |