Software Requirements Specifications

Introduction through Requirements and Specification Lens

Purpose

The genetic algorithm project is designed to represent a software program that implements a genetic algorithm to represent genetic evolution of chromosomes. The genetic algorithm utilizes a population of chromosomes represented typically by 10 x 10 grids of on/off states. Each chromosome represents a potential representation of a genetic variation of a chromosome. The algorithm evolves the population over multiple generations with operators like selection, crossover, and mutation to meet a fitness criterion for the chromosomes.

The software provides a flexible and configurable framework to apply a genetic algorithm towards evolution representative problems. Allowing users to define factors like population size, elitism, generations elapsed, mutation rate, and fitness models, the program enables the exploration of different genetic algorithm configurations. The visualization of evolutionary progress, along with the display of a fitness graph, assists users in analyzing and understanding an algorithm's effectiveness.

This SRS document outlines the requirements, features, and functionalities of the software program. It serves as a comprehensive guide for the development team, stakeholders, and users to understand the scope and behavior of the system.

Scope of Project and Functional Requirements Specification

The scope of the genetic algorithm project encompasses the development of a software program that implements a genetic algorithm to represent chromosome evolution and solve optimization problems. The software provides a versatile framework that simulates the principles of natural selection and genetics to iteratively improve a population of "chromosomes".

Each chromosome holds information to represent a number of genes and which state they are in (in this case, on and off, similar to a boolean). The population size, determined by the user, determines the number of chromosomes present in each generation. The fitness of each chromosome is evaluated using the chosen fitness model, which can include the "simple" fitness model or other user-defined models.

Throughout the execution of the genetic algorithm, the software program provides a visual interface that displays the evolutionary progress. Users can observe the best-performing chromosome from each generation, accompanied by its fitness value. Additionally, the software generates graphical representations of the fitness trends for the best, average, and worst chromosomes over time.

The software program emphasizes flexibility and configurability, enabling users to explore a wide range of genetic domains, experiment with various parameter settings, and assess the performance of the genetic algorithm in different contexts. The user-friendly interface and interactive features allow users to gain insights into these genetic optimization processes.

Requirements Specification

Interface Requirements

1. The user interface should graphically display evolutionary progress, including a grid representation of chromosomes and information about each generation.

2. The user interface should provide input fields for configuring parameters such as population size, elitism, generations, fitness, and mutation rate.

3. The software reasonably and automatically adjusts the speed at which information is displayed, allowing users to interpret the progress easily.

4. The user interface should include a graph component that visually represents the fitness trends of the best, average, and worst chromosomes over time.

5. The software should support multiple UI options, such as JSwing or JavaFX, to provide flexibility and accommodate user preferences.

6. For the JavaFX version, a Chromosome Viewer window should be accessible via a connecting button. The Chromosome Viewer allows users to create their own fitness "end result" and observe the chromosomes' evolution towards that goal.

Nonfunctional Requirements

1. Performance:

    - The software should execute the genetic algorithm efficiently, with reasonable response times even for large population sizes and lengthy generations.
    - The graphical display and updates should be smooth and responsive, providing real-time visual feedback to users.
2. Usability:
    - The user interface should be intuitive, with clear labels and input fields for configuring parameters such as population size, elitism, generations, and mutation rate.
    - The software should provide appropriate error handling and validation for user inputs to ensure accurate parameter configuration.
    - The timer for displaying information should be adjustable, allowing users to control the speed of information presentation.
    - The user interface should be visually appealing and well-organized, facilitating easy interpretation and understanding of the displayed data.
3. Portability:
    - The software should be platform-independent, capable of running on various operating systems without significant modifications.
    - The application should be compatible with different UI options, such as JSwing or JavaFX, to accommodate user preferences and provide flexibility in the choice of user interface technology.
4. Extensibility:
    - The software should be designed to allow for easy integration of additional fitness models, enabling users to define and incorporate their own fitness criteria.
    - The system should be extensible to support future enhancements, such as the addition of new genetic operators or optimization techniques.

Acceptance Criteria

1. The graphical display of chromosomes and relevant information should be rendered accurately and consistently for each generation.

2. Users should be able to input and adjust parameters such as population size, elitism, generations, and mutation rate, and observe the corresponding effects on the genetic algorithm's behavior.

3. The timer control should allow users to slow down or speed up the display of information, ensuring it is interpretable and comprehensible to human users.

4. The graph component should accurately represent the fitness trends of the best, average, and worst chromosomes over generational time, providing visual insights into the performance of the genetic algorithm.

5. The software should offer different UI options (JSwing or JavaFX) without any loss of functionality or data consistency.

6. In the JSwing version, the Chromosome Viewer window should open correctly upon clicking the connecting button. Users should be able to define their own fitness "end result" and observe the chromosomes evolving towards that goal within the Chromosome Viewer.