Faculty of Computing and Information Technology

University of the Punjab,
Lahore

Artificial Intelligence Lab 7

Instructor: Qamar U Zaman

Genetic Algorithm Lab Manual

Introduction to Genetic Algorithms

Genetic algorithms (GAs) are optimization and search techniques inspired by the principles of genetics and natural selection. GAs are used to solve complex problems by evolving solutions over time. They are particularly useful for problems where the search space is large and traditional optimization techniques are not effective.

Key Concepts

- **Population:** A set of potential solutions to the problem.
- **Chromosomes:** Representations of the solutions in the population.

- **Genes:** Parts of a chromosome, representing specific traits or variables of the solution.
- **Fitness Function:** A function that evaluates how good a solution is. •

Selection: The process of choosing the best solutions to reproduce. •

Crossover: Combining parts of two solutions to create new offspring. •

Mutation: Randomly altering parts of a solution to introduce diversity.

Problem 1: Traveling Salesman Problem (TSP)

Problem Description

The Traveling Salesman Problem (TSP) is a classic optimization problem where a salesman must visit a set of cities exactly once and return to the starting city. The objective is to find the shortest possible route that visits each city and returns to the origin.

Genetic Algorithm Approach

1. **Representation:**
   - Each chromosome represents a possible tour of the cities.
   - A gene represents a city, and the sequence of genes represents the order in which the cities are visited.
2. **Fitness Function:**
   - The fitness of a tour is the inverse of its total distance. A shorter tour has a higher fitness value.
3. **Selection:**
   - Use methods like roulette wheel selection or tournament selection to choose parent solutions based on their fitness.
4. **Crossover:**
   - Apply crossover techniques like ordered crossover (OX) or partially mapped crossover (PMX) to generate new offspring.
5. **Mutation:**
   - Use mutation operators such as swap mutation, where two cities in the tour are randomly selected and their positions are swapped.
6. **Algorithm Steps:**
   - Initialize a population of random tours.
   - Evaluate the fitness of each tour.
   - Select parents based on fitness.

- Apply crossover and mutation to create a new population.
- Repeat until a stopping criterion is met (e.g., a fixed number of generations or a satisfactory fitness level).

Necessary Details

- **Distance Matrix:** A matrix where the entry at row i and column j represents the distance between city i and city j.
- **Stopping Criteria:** Define a maximum number of generations or a convergence threshold.

Problem 2: Knapsack Problem

Problem Description

The Knapsack Problem is a combinatorial optimization problem where you are given a set of items, each with a weight and a value. The objective is to determine the number of each item to include in a knapsack so that the total weight does not exceed a given limit and the total value is maximized.

Genetic Algorithm Approach

1. **Representation:**
   - Each chromosome represents a potential solution.
   - A gene is a binary value (0 or 1) indicating whether an item is included in the knapsack.
2. **Fitness Function:**
   - The fitness is the total value of the items in the knapsack if the total weight is within the limit; otherwise, the fitness is zero.
3. **Selection:**
   - Use methods like roulette wheel selection or tournament selection to choose parent solutions based on their fitness.
4. **Crossover:**
   - Apply crossover techniques like single-point crossover or uniform crossover to generate new offspring.
5. **Mutation:**
   - Use mutation operators such as bit-flip mutation, where a gene is randomly selected and its value is flipped (0 becomes 1 and 1 becomes 0).
6. **Algorithm Steps:**
   - Initialize a population of random solutions.

- Evaluate the fitness of each solution.
- Select parents based on fitness.
- Apply crossover and mutation to create a new population.
- Repeat until a stopping criterion is met (e.g., a fixed number of generations or a satisfactory fitness level).

Necessary Details

- **Item List:** A list of items, each with a specified weight and value.
- **Weight Limit:** The maximum weight the knapsack can carry.
- **Stopping Criteria:** Define a maximum number of generations or a convergence threshold.

Practical Implementation

Problem 1: Traveling Salesman Problem (TSP)

1. **Create a Distance Matrix:** Define the distance matrix representing the distances between cities.
2. **Initialize Population:** Generate an initial population of random tours.
3. **Evaluate Fitness:** Calculate the fitness of each tour based on the total distance.
4. **Selection, Crossover, and Mutation:** Implement the selection, crossover, and mutation operators to evolve the population.

Problem 2: Knapsack Problem

1. **Define Item List:** List the items with their weights and values.
2. **Initialize Population:** Generate an initial population of random solutions.
3. **Evaluate Fitness:** Calculate the fitness of each solution based on the total value and weight.
4. **Selection, Crossover, and Mutation:** Implement the selection, crossover, and mutation operators to evolve the population.