ARTIFICIAL INTELLIGENCE

PROJECT REPORT

PROJECT TITLE: ***Application of Genetic Algorithm in Customer Table Assignment in Restaurants***

GROUP MEMBERS:

WAJEEHA ALI (CS – 22055)

TOOBA FATIMA (CS – 22056)

ASIYA FATIMA (CS – 22064)

BATCH: 2022

SUBMISSION DATE: **NOV 24, 2024**

SUBMITTED TO: **Ms. HAMEEZA AHMED**

INTRODUCTION

Genetic Algorithm:

The genetic algorithm (GA) is a robust metaheuristic inspired by the principles of natural selection. It belongs to the broader class of evolutionary algorithms and is widely used for solving optimization and search problems. This report explores the development and application of a genetic algorithm for the problem of assigning customers to tables in a restaurant, considering both group sizes and seating preferences.

Steps in Genetic Algorithm

Initialization:

- Create an initial population of random candidate solutions.
- Each solution is encoded as a chromosome (string of genes).

Evaluation (Fitness Function):

- Assess the quality of each chromosome using a fitness function that measures how close the solution is to the desired outcome.

Selection:

Select parent chromosomes based on their fitness scores. Common methods:

- Roulette Wheel Selection: Probability proportional to fitness.
- Tournament Selection: A subset is randomly chosen, and the fittest is selected.

Crossover (Recombination):

- Combine pairs of parents to produce offspring by exchanging genetic material.

Common methods:

- Single-Point Crossover: Split parents at a random point and exchange sections.
- Multi-Point Crossover: Split at multiple points for more variety.
- Uniform Crossover: Genes are swapped based on a probability.

Mutation:

- Randomly alter genes in a chromosome to introduce genetic diversity.
- Helps the algorithm avoid local optima by exploring new areas of the solution space.

Replacement:

Replace some or all of the population with the newly generated offspring. Strategies include:

- Generational Replacement: Replace the entire population.
- Steady-State Replacement: Replace only the weakest individuals.

Termination:

Stop the algorithm when:

- A maximum number of generations is reached.
- A solution with a satisfactory fitness is found.

- **The aim of this project is threefold:**
  - Simulate the working of a genetic algorithm using a novel example.
  - Produce a Python implementation of the genetic algorithm.
  - Apply the algorithm to a real-world problem and analyze its effectiveness.

PROBLEM STATEMENT

Restaurants often face the challenge of optimizing seating arrangements to maximize customer satisfaction while ensuring table capacity constraints are met. In this project, customers are assigned to tables based on:

- Group size: Ensuring tables are not overcrowded.
- Seating preference: Matching customers' preferences (e.g., "window," "standard," "near exit").

This problem is modeled as an optimization task where the goal is to maximize customer satisfaction while adhering to table capacity constraints.

METHDOLODY

1. Initialization

The Genetic Algorithm was initialized with a random population of solutions:

- Chromosome Representation: Each chromosome is a sequence of table IDs, where each gene corresponds to the table assigned to a specific customer group.

Example: [1, 2, 3, 4, 5] assigns Customer 1 to Table 1, Customer 2 to Table 2, and so on.

- Population Size: A total of 20 individuals were initialized in the population.

2. Evaluation (Fitness Function)

A fitness function was defined to evaluate the quality of each chromosome:

- Customer Satisfaction: A satisfaction score of 10 was awarded for matching a customer's table preference (e.g., "window"). A partial satisfaction score of 5 was assigned for a mismatch.
- Capacity Constraints: If a table's assigned customer group exceeded its capacity, a penalty of -10 was applied to the fitness score.
- Fitness Value: The total fitness of a chromosome was calculated as the sum of satisfaction scores for all customer groups while accounting for penalties.

3. Selection

To select parent chromosomes for crossover:

- Fitness-Based Selection: Higher fitness individuals were more likely to be selected, with a bias toward the top 10 solutions in the population. This ensures that better solutions have a higher chance of propagating to the next generation.

4. Crossover (Recombination)

Crossover was applied to generate new offspring by combining the genetic material of parent chromosomes:

- ❖ Method Used: Single-point crossover was employed, where a random split point was selected, and segments of two parent chromosomes were swapped to produce two offspring.
- ❖ Example:

- Parent 1: [1, 2, 3, 4, 5]
- Parent 2: [5, 4, 3, 2, 1]
- Offspring 1: [1, 2, 3, 2, 1]
- Offspring 2: [5, 4, 3, 4, 5].

5. Mutation

Mutation was introduced to maintain genetic diversity and explore new solutions:

- ❖ Mutation Rate: A mutation occurred with a probability of 20% (0.2) for each chromosome.
- ❖ Mutation Operation: One random gene in the chromosome (customer table assignment) was reassigned to a different table, ensuring that the algorithm could escape local optima.

6. Replacement

The next generation of the population was formed using a combination of:

- ❖ Elitism: The top 2 fittest individuals from the current generation were directly carried forward to the next generation.
- ❖ Offspring Replacement: Remaining population slots were filled by offspring generated through crossover and mutation.

7. Termination

The algorithm terminated after 50 generations, as predefined. The best solution in the final population was recorded as the optimal solution.

- ❖ Implementation Details
- ➢ Programming Language: Python
- ➢ Key Parameters:
- ➢ Population Size: 20
- ➢ Number of Generations: 50
- ➢ Mutation Rate: 20%

Results Summary

The Genetic Algorithm produced the following optimal solution:

Chromosome: [1, 5, 3, 4, 2]

This chromosome represents the assignment of customer groups to tables, maximizing customer satisfaction.

Fitness: 50

The highest satisfaction score achieved, with no further improvement after Generation 10, indicating early convergence.

CONCLUSION:

The Genetic Algorithm is a powerful tool for solving optimization problems like the Restaurant Table Assignment Problem. By simulating natural selection, crossover, and mutation, the algorithm effectively explores the solution space and improves over generations. The example run demonstrates how the algorithm can successfully assign customers to tables while optimizing customer satisfaction.

This report highlights the ability of genetic algorithms to solve complex problems in real-world scenarios, providing efficient and near-optimal solutions in cases where traditional methods may struggle.

EVIDENCE OF FUNCTIONALITY