Artificial Intelligence

BS (CS) _SPRING_2025

Lab_07 Tasks

Learning Objectives:

1. Genetic Algorithm

Lab Task:

Task 1: Knapsack Problem Using Genetic Algorithm

The **Knapsack Problem** is a classic optimization problem where you have a set of items, each with a weight and a value. You are given a knapsack with a weight capacity, and the goal is to select a subset of items that maximizes the total value without exceeding the weight capacity of the knapsack.

In this task, you will use a **Genetic Algorithm (GA)** to solve a **0/1 Knapsack Problem**. Each individual in the population will represent a potential solution, where each gene in the chromosome represents whether a particular item is included (1) or excluded (0) from the knapsack.

You are given a set of n items, each with a weight w[i] and value v[i], where 1 <= i <= n.

Your task is to select a subset of the items that maximize the total value while staying within the physical volume capacity of the knapsack. The total weight and size of the selected items must be less than or equal to the weight allowed. The weight constraint is **10 Kg**.

Items	N1	N2	N3	N4	N5	N6
Value	14	23	8	9	17	15
Weight	1	3	7	4	5	6

1. **Initialization:** Generate an initial population of pop_size individuals, where each individual is a binary string of length n that represents whether each item is included or not (1 = included, 0 = not included).
2. **Fitness Function:** Evaluate the fitness of each individual by calculating the total value of the selected items and penalizing (total value – total weight) solutions that exceed the physical volume capacity of the knapsack. The fitness function should return a value that reflects the quality of the solution.
3. **Selection:** Select two individuals from the population with probability proportional to their fitness. You should use roulette wheel selection.
4. **Crossover:** Generate two offspring by applying crossover to the selected parents. You can use any crossover method you like, such as one-point crossover, two-point crossover, or uniform crossover.
5. **Mutation:** Mutate the offspring by flipping each bit in the binary string with probability **mut_prob.**
6. **Replacement:** Replace the least-fit individuals in the population with the offspring.
    - If there are chromosomes that exceed the knapsack capacity, replace the lowest-fit chromosomes (which exceeds capacity) with the offspring.
    - If no chromosome exceeds the capacity, replace the lowest-fit chromosomes with the offspring.
7. **Termination:** Repeat steps 3-6 for **max_gen** generations.