Lab 08

Genetic Algorithms

- Genetic Algorithm is a search heuristic (experience) that follows the process of natural evolution. This heuristic is used to generate useful solutions to optimization and search problems.

- Genetic algorithms are inspired by Darwin's theory about evolution. Solution to a problem solved by genetic algorithms is evolved.

- Genetic Algorithm belong to the larger class of evolutionary algorithm (EA) which generate solutions to optimization problems and using techniques inspired by natural evolution like – inheritance, mutation, selection, crossover.

- Genetic Algorithm need design space to be converted into genetic space. Genetic Algorithm works with coding variables.

- Algorithm is started with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are selected to form new solutions (offspring) are selected according to their fitness - the more suitable they are the more chances they have to reproduce.

- This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

There are three important aspects of Genetic Algorithm are

I. Definition of objective function.

II. Definition and implementation of genetic representation.

III. Definition and implementation of Genetic operators.

Advantages of Genetic Algorithm (GA) :-

1. It shows simplicity.

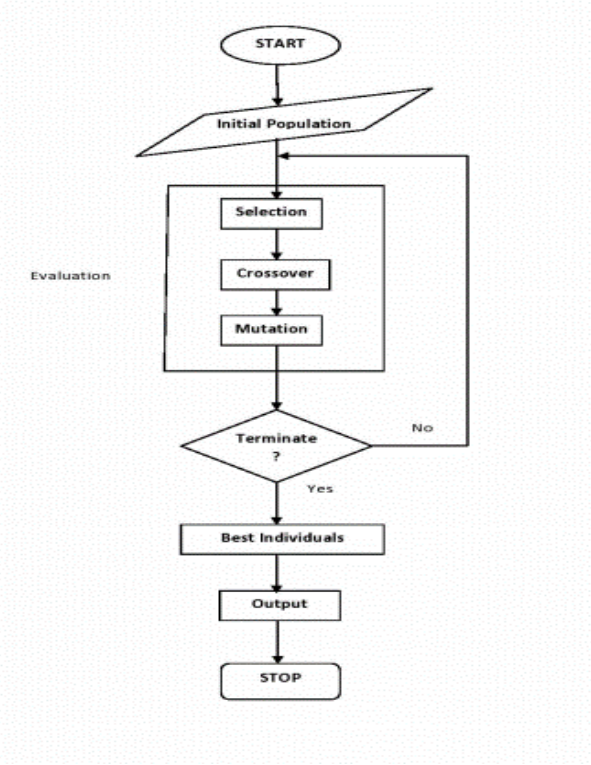
2. Ease of operation.

3. Minimal requirement.

4. Global perspective.

5. It does not guarantee to find global minimum solutions but acceptably good solutions.

Flowchart of the Algorithm



Algorithm of the Basic Genetic Algorithm:

Outline of the Basic Genetic Algorithm:

1. **[Start]** Generate random population of n chromosomes (suitable solutions for the problem)

2. **[Fitness]** Evaluate the fitness f(x) of each chromosome x in the population

3**. [New population]** Create a new population by repeating following steps until the new population is complete

- **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
- **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
- **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
- **[Accepting]** Place new offspring in a new population
- **[Replace]** Use new generated population for a further run of algorithm
- **[Test]** If the end condition is satisfied, stop, and return the best solution in current
- population
- **[Loop]** Go to step 2

Example: Match Word Finding

Here we try to guess a word from the given population of word:

Step 1: Select the word to be guessed.This value is taken through user input.

Step 2: Initialize the population. User inputs the population.

Step 3: Evaluate the population. Fitness is assigned based on number of correct letters in correct place.

Step 4: Select breeding population. Selection is done on the basis of fitness.

Step 5: Create new population. Population is created by using uniform crossover between breeding populations.

Step 6: Check for stopping condition. Here maximum fitness value in population is checked. If it is 60%.

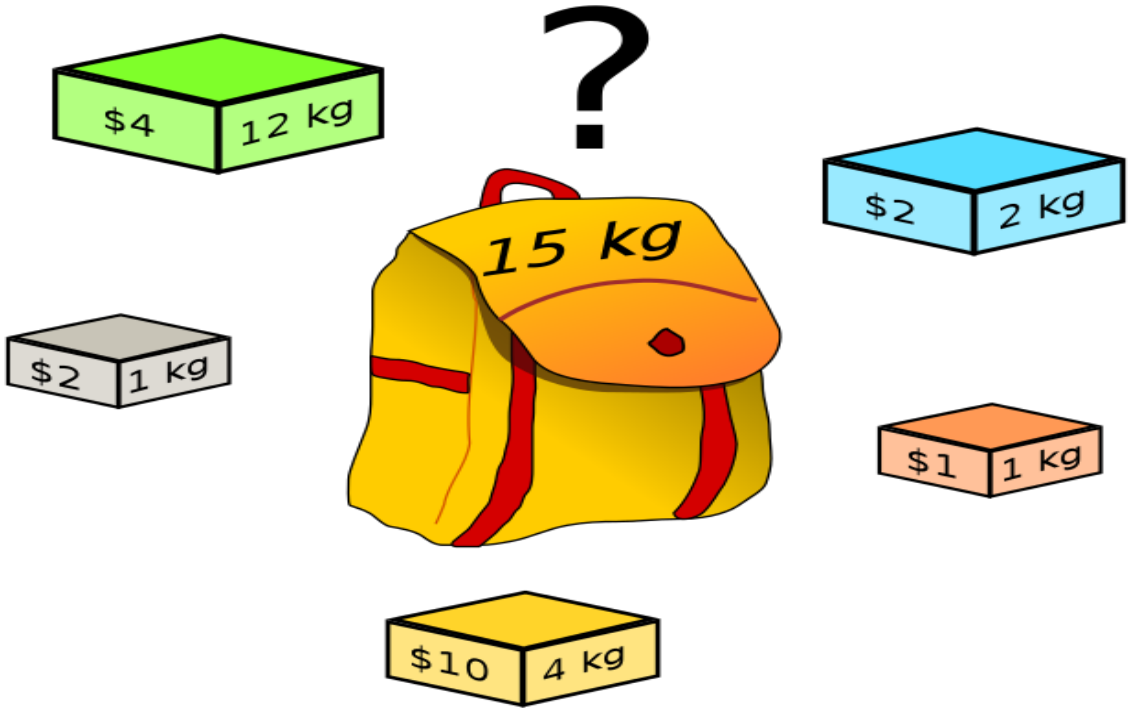
Step 7: If stopping condition is not true, go to Step 3; else return the offspring with highest fitness value

Lab Task

In this Lab we will solve the 0-1 Knapsack Problem using Genetic Algorithm.

Knapsack Problem

The knapsack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.



Code Provided:

Mainly 4 functions are already provided to you for your ease

1) **Generate Population**: Generate Population , given the size and backpack_Capacity

2) **parent_selection(population) :**

Select a parent from Population

Find 2 Fittest Individual to select parent

Check Total sum value of fittest individuals

3) **def apply_crossover:** Apply Crossover and Mutation on population, Given crossover probability and mutation probability

4) def apply_mutation(chromosome, backpack_capacity, mutation_probability):

Apply Mutation on chromosomes , given Mutation probability

------- Tasks To perform -----

1) def find_two_fittest_individuals(population):

Find Top 2 Fittest Individual from Population

2) def calculate_fitness(population, items, max_weight):

Calculate Fitness of population, given Items (weight,value) and max weight in action

Run The code Given at the end to check fitness values of your algorithm after every 100th generations.

Submission Guidelines:

- Lab must be submitted in the google classroom.
- Submission other than google classroom won't not be accepted.
- You are required to submit a python (version 3 compatible) file named after Your RollNo.