Name :p.v.ganesh
Ref.No : 20MIS7081

Assignment 7

Genetic Algorithm Problems

1. A toy manufacturing company makes two models A and B of a product on a daily basis. Company gets a profit of Rs. $(x^2 - 2)$ on sale of x units of model A and Rs. $(y^2 - 1)$ on the sale of y units of model B. Maximum demand for the models A and B is 25 units per day. How many units of models A and B should be manufactured to maximize the profit.

   - Formulate the given problem.
   - Solve the problem using Genetic Algorithm.

Formulation :

Max(Z) = (x^2 -2) + (y^2 - 1)

Subject to :
X >= 0, y >= 0;
x,y <= 25;

CODE :

```
%Name:
%Reg.No:

% Genetic Algorithm Parameters
populationSize = 6; % Population size
numberOfGenes = 2; % Number of genes in each chromosome
mutationRate = 0.01; % Probability of mutation
crossoverRate = 0.8; % Probability of crossover
generations = 6; % Number of generations

% Define the range of the variable x and y
xMin = 0;
xMax = 25;
yMin = 0;
yMax = 25;

% Initialize Population
population = randi([0, 1], populationSize, numberOfGenes);

% Main Genetic Algorithm Loop
for generation = 1:generations

% Decode chromosomes to real values in the specified range
x = decodeChromosomes(population(:, 1), xMin, xMax, numberOfGenes);
y = decodeChromosomes(population(:, 2), yMin, yMax, numberOfGenes);

% Evaluate Fitness (using the profit function)
fitness = evaluateProfit(x, y);

% Selection
selectedPopulation = selection(population, fitness);

% Crossover
offspring = crossover(selectedPopulation, crossoverRate);

% Mutation
mutatedOffspring = mutation(offspring, mutationRate);

% Replace the old population with the new population
population = mutatedOffspring;
```

```
% Display the best fitness in this generation
bestFitness = max(fitness);
fprintf('Generation %d: Best Fitness = %.4f\n', generation, bestFitness);
end

% Find and display the best solution
x = decodeChromosomes(population(:, 1), xMin, xMax, numberOfGenes);
y = decodeChromosomes(population(:, 2), yMin, yMax, numberOfGenes);
profit = evaluateProfit(x, y);

[bestProfit, bestIndex] = max(profit);
bestX = x(bestIndex);
bestY = y(bestIndex);

fprintf('Best Solution: Model A units = %.4f, Model B units = %.4f, Profit = %.4f\n', bestX, bestY,
bestProfit);

% Decode chromosomes to real values
function decodedValues = decodeChromosomes(chromosomes, minValue, maxValue, numberOfGenes)
range = maxValue - minValue;
decodedValues = minValue + (bi2de(chromosomes, 'left-msb') / (2^numberOfGenes - 1)) * range;
end

% Evaluate Profit (Objective Function)
function profit = evaluateProfit(x, y)
profit = (x.^2 - 2) + (y.^2 - 1);
end

% Selection function (Tournament Selection)
function selectedPopulation = selection(population, fitness)
tournamentSize = 5;
selectedPopulation = zeros(size(population));
for i = 1:size(population, 1)
tournamentParticipants = randperm(size(population, 1), tournamentSize);
[~, winnerIndex] = max(fitness(tournamentParticipants));
selectedPopulation(i, :) = population(tournamentParticipants(winnerIndex), :);
end
end

% Crossover function (Single-point Crossover)
function offspring = crossover(selectedPopulation, crossoverRate)
offspring = zeros(size(selectedPopulation));
for i = 1:2:size(selectedPopulation, 1)
if rand() < crossoverRate
crossoverPoint = randi([1, size(selectedPopulation, 2) - 1]);
offspring(i, :) = [selectedPopulation(i, 1:crossoverPoint), selectedPopulation(i + 1,
crossoverPoint + 1:end)];
offspring(i + 1, :) = [selectedPopulation(i + 1, 1:crossoverPoint), selectedPopulation(i,
crossoverPoint + 1:end)];
else
offspring(i, :) = selectedPopulation(i, :);
offspring(i + 1, :) = selectedPopulation(i + 1, :);
end
end
end

% Mutation function (Bit-flip Mutation)
function mutatedOffspring = mutation(offspring, mutationRate)
mutatedOffspring = offspring;
for i = 1:size(offspring, 1)
for j = 1:size(offspring, 2)
if rand() < mutationRate
mutatedOffspring(i, j) = 1 - offspring(i, j);
end
end
end
end
```

OUTPUT :

```
Command Window
>> lab9
Generation 1: Best Fitness = 135.8889
Generation 2: Best Fitness = 135.8889
Generation 3: Best Fitness = 135.8889
Generation 4: Best Fitness = 135.8889
Generation 5: Best Fitness = 135.8889
Generation 6: Best Fitness = 135.8889
Best Solution: Model A units = 8.3333, Model B units = 8.3333, Profit = 135.8889
```

2. The cost of engines plus fuel for a cargo ship (in lakhs of rupees per year for 100 tons of cargo carried) varies with speed and is given by $0.2x^2$, where x is the speed of the ship in m/s. The fixed costs of hull and crew (again in the same units) are given by $450/x$. Using the genetic algorithm, determine the operating speed of the ship for minimum total cost over the interval $[4, 16]$ (This is a single variable problem).

    - Formulate the given problem.
    - Solve the problem using Genetic Algorithm.

Formulation :

Min(Z) = 0.2 x^2 + (450/x)
Interval [4,16]

CODE :

```
%Name:
%Reg.No:

% Genetic Algorithm Parameters
populationSize = 20; % Increase population size for better convergence
numberOfGenes = 8; % Increase the number of genes to represent a real number with more precision
mutationRate = 0.01; % Probability of mutation
crossoverRate = 0.8; % Probability of crossover
generations = 10; % Increase the number of generations for better convergence

% Define the range of the variable x (ship speed)
xMin = 4;
xMax = 16;

% Initialize Population
population = randi([0, 1], populationSize, numberOfGenes);

% Main Genetic Algorithm Loop
for generation = 1:generations
% Decode chromosomes to real values in the specified range
x = decodeChromosomes(population, xMin, xMax, numberOfGenes);

% Evaluate Fitness using the objective function (total cost)
fitness = evaluateTotalCost(x);

% Selection
selectedPopulation = selection(population, fitness);

% Crossover
offspring = crossover(selectedPopulation, crossoverRate);

% Mutation
mutatedOffspring = mutation(offspring, mutationRate);

% Replace the old population with the new population
population = mutatedOffspring;

% Display the best fitness in this generation
```

```
bestFitness = min(fitness);
fprintf('Generation %d: Best Fitness = %.4f\n', generation, bestFitness);
end

% Find and display the best solution
x = decodeChromosomes(population, xMin, xMax, numberOfGenes);
fitness = evaluateTotalCost(x);
[bestFitness, bestIndex] = min(fitness);
bestSolution = x(bestIndex);
fprintf('Best Solution: x = %.4f, Total Cost = %.4f\n', bestSolution, bestFitness);

% Decode chromosomes to real values
function x = decodeChromosomes(population, xMin, xMax, numberOfGenes)
xRange = xMax - xMin;
x = xMin + (bi2de(population, 'left-msb') / (2^numberOfGenes - 1)) * xRange;
end

% Evaluate Total Cost (Objective Function)
function totalCost = evaluateTotalCost(x)
% Objective function: Total Cost = 0.2x^2 + 450/x
totalCost = 0.2 * x.^2 + 450 ./ x;
end

% Selection function (Tournament Selection)
function selectedPopulation = selection(population, fitness)
tournamentSize = 5;
selectedPopulation = zeros(size(population));
for i = 1:size(population, 1)
tournamentParticipants = randperm(size(population, 1), tournamentSize);
[~, winnerIndex] = min(fitness(tournamentParticipants));
selectedPopulation(i, :) = population(tournamentParticipants(winnerIndex), :);
end
end

% Crossover function (Single-point Crossover)
function offspring = crossover(selectedPopulation, crossoverRate)
offspring = zeros(size(selectedPopulation));
for i = 1:2:size(selectedPopulation, 1)
if rand() < crossoverRate
crossoverPoint = randi([1, size(selectedPopulation, 2) - 1]);
offspring(i, :) = [selectedPopulation(i, 1:crossoverPoint), selectedPopulation(i + 1,
crossoverPoint + 1:end)];
offspring(i + 1, :) = [selectedPopulation(i + 1, 1:crossoverPoint), selectedPopulation(i,
crossoverPoint + 1:end)];
else
offspring(i, :) = selectedPopulation(i, :);
offspring(i + 1, :) = selectedPopulation(i + 1, :);
end
end
end

% Mutation function (Bit-flip Mutation)
function mutatedOffspring = mutation(offspring, mutationRate)
mutatedOffspring = offspring;
for i = 1:size(offspring, 1)
for j = 1:size(offspring, 2)
if rand() < mutationRate
mutatedOffspring(i, j) = 1 - offspring(i, j);
end
end
end
end
```

OUTPUT :

Command Window

```
>> lab9
Generation 1: Best Fitness = 64.9065
Generation 2: Best Fitness = 64.9012
Generation 3: Best Fitness = 64.9012
Generation 4: Best Fitness = 64.9012
Generation 5: Best Fitness = 64.9012
Generation 6: Best Fitness = 64.9012
Generation 7: Best Fitness = 64.9012
Generation 8: Best Fitness = 64.9012
Generation 9: Best Fitness = 64.9012
Generation 10: Best Fitness = 64.9012
Best Solution: x = 10.4000, Total Cost = 64.9012
```