GA.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function simpleGA:
% This Function illustrate how a GA could be used to evolve a population of
% randomly generated bianry strings. The Fitness of an individual is determined
% by the number of '1'. An archive will store the fittest individual at each
% generation and the evolutionary trace plotted after the algorithm shows that
% GA is capable of improving the fitness of the evolving population
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Algorithm Definations:
% pop               An array of individuals which represent the evolving population
% pop(i)            The ith individual in the evolving population
% pop(i).chromosome The genotype value of the ith individual, which is a
%                   binary sting in this algorithm
% pop(i).fitness    The fitness of the ith individual
% fitness_trend     An array which tracks the fitness of the archive
% pop_archive       An archive which stores the fittest individual in each generation
% matingpop         An array of individuals that are being selected for crossover and mutation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function simpleGA()
close all       % Close all active windows
clc             % Clear the command window
clear           % Initialize the memory
rand('state',1) % Modify the random number generator by considering other integers
                % so that re-running the program without changes will not yield the
                % same results.
% modified randomized generator using time as a seed
% rand('state',sum(100*clock))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parameter Initialisation:
% Input your own parameters below
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

POP_SIZE=10;            % Number of individuals in the population
                        % Even POP_SIZE MUST be used for proper operation
MUTATION_RATE=0.03;     % Probability of mutation (typically <.1)
CROSSOVER_RATE=0.8;     % Probability of crossover (typically near 1)
CHRMOSOME_LENGTH=30;    % Length of the chromosome
MAX_GENERATION=50;      % Number of times the loop will run

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Algorithm Initialisation:
% The Intial Population of the algorithm will be created here. It will
% consists of POP_SIZE chrmosomes of length CHRMOSOME_LENGTH each. Each
% chrmosome will have the default fitness of 0 and each allele will be
% randomised to take the value of 0 or 1.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:1:POP_SIZE
    pop(i).chromosome = rand(1, CHRMOSOME_LENGTH);
    pop(i).fitness = 0;
    for j=1:1:CHRMOSOME_LENGTH
        if (pop(i).chromosome(j)<0.5)
            pop(i).chromosome(j)=0;
        else
            pop(i).chromosome(j)=1;
        end
    end
end
fitness_trend=zeros(1,MAX_GENERATION);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Begin of Evolution:
% The population will be evolved here. First the fitness of each chrmosome
% will be calcluated and the best chromosome will be stored in the archive.
% The mating pool will be created by applying tourament selection on the
% combined evolving population and archive. Uniform crossover and bit-flipped
% mutation will be applied subsequently. This process will be repeated for
% MAX_GENERATION generations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pop_archive=[];
for gen=1:1:MAX_GENERATION
    %Fitness Evaluation
    for n=1:1:POP_SIZE,
        pop(n).fitness = get_Fitness(pop(n).chromosome,CHRMOSOME_LENGTH);
    end
    %Archiving
    for n=1:1:POP_SIZE
        pop_archive = archive(pop(n), pop_archive);
    end
    %Tracking the fitness of the best individual
    fitness_trend(gen)=pop_archive(1).fitness;


    %Initialise the mating pool via tournament selection

    %elite is appended to the current population
    if size(pop_archive,2)~= 0
         pop = [pop pop_archive];
    end

    %elite replace the weakest individual of the current population
%     if size(pop_archive,2)~= 0
%         tmp_fitness = pop(1).fitness;
%         tmp_index=1;
%         for n=2:1:POP_SIZE,
%             if pop(n).fitness < tmp_fitness
%                 tmp_fitness = pop(n).fitness;
%                 tmp_index = n;
%             end
%         end
%         pop(tmp_index)=pop_archive;
%     end


    %Tournament selection with replacement
    for n=1:1:POP_SIZE
        matingpop(n)=tournament(pop);
    end

    %Tournament selection without replacement
%     matingpop=tournament2(pop,POP_SIZE);

    %Applying the genetic operators
    pop = crossover(matingpop, CROSSOVER_RATE,CHRMOSOME_LENGTH);
    for n=1:1:POP_SIZE,
        pop(n).chromosome = mutate(pop(n).chromosome, MUTATION_RATE);
    end
end
%Plot the fitness trace
figure
plot(fitness_trend);
title('Fitness Trace');
ylabel('Fitness')
xlabel('Generations')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of function simpleGA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function tournament:
% This Function wil randomly select two individuals from the evolving
% population. The fitter individual will be chosen and added to the
% mating pool. This will be repeated for POP_SIZE times until the mating
% pool is full
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function pair = tournament(pop)
sel = ceil(rand(1,1)*size(pop,2));
com = ceil(rand(1,1)*size(pop,2));
if pop(sel).fitness > pop(com).fitness;
    index = sel;
else
    index = com;
end
pair=pop(index);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of function tournament
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function tournament 2:
% This Function will perform tournament selection without replacement. The
% parent population will be grouped in pairs. The winner of each pair for this
% first tournament will be selected for the variation operation. This process
% will be repeated again. The winners of both tournaments will be paired together
% where the n-th winner of both groups will be together.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function popC = tournament2(pop,pop_size)

order1=randperm(size(pop,2));
order2=randperm(size(pop,2));

for i=1:2:pop_size
    if pop(order1(i)).fitness > pop(order1(i+1)).fitness;
        index1 = order1(i);
    else
        index1 = order1(i+1);
    end
    if pop(order2(i)).fitness > pop(order2(i+1)).fitness;
        index2 = order2(i);
    else
        index2 = order2(i+1);
    end
    popC(i)=pop(index1);
    popC(i+1)=pop(index2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of function tournament 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function mutate:
% This Function will mutate the different allele with a probability MUTATION_RATE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function indnew = mutate(ind, mutation_rate)
for i=1:1:size(ind,2)
    if rand(1,1)<mutation_rate
        ind(i) = 1-ind(i) ;
    end
end
indnew = ind;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of function mutate
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function crossover:
% This Function will crossover alleles between two selected chromosomes.
% Crossover will occur at a probability of CROSSOVER_RATE and if it does,
% each pairs of allele will have a 50% chance of being swapped
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function childpop = crossover(pop, crossover_rate,length);
for n=1:2:size(pop,2)
    sis = pop(n).chromosome;bro = pop(n+1).chromosome;
    if rand(1,1)<crossover_rate
        for len=1:1:length
            if rand(1,1)<0.5
                tmp = sis(len);
                sis(len)=bro(len);
                bro(len)=tmp;
            end
        end
    end
    pop(n).chromosome = sis;
    pop(n+1).chromosome = bro;
end
childpop = pop;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of function crossover
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function archive:
% This Function will select the better individuals and store it into pop_archive
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function popC = archive(ind, popB)
%Add if archive is empty
if size(popB,2)==0,
    popB(1).chromosome = ind.chromosome;
    popB(1).fitness = ind.fitness;
end

if (popB(1).fitness < ind.fitness)
    popB(1).chromosome = ind.chromosome;
    popB(1).fitness = ind.fitness;
end
popC=popB;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of function archive
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function get_fitness:
% This Function evaluate the fitness of a given chrmosome which is given by
% the number of '1'. The optimal solution is a bianry string with all
% alleles equal to one
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function fitness = get_Fitness(ind, length)
fitness = 0;
for len=1:1:length
    fitness = fitness + ind(len);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of function get_fitness
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```