Evolutionary Algorithms
Problem Set - Genetic Algorithms in MATLAB

1. Below you find the skeleton of a canonical GA implementation in MATLAB:

```
function [opt, fopt, histf] = ga(n, fitnessfct, decodefct, selectfct, stopeval)

  % GA parameters
  mu = ...
  pc = ...
  pm = ...

  % Statistics administration
  evalcount = 0;
  histf = zeros(1, stopeval);

  % Initialize population
  for i = 1:mu
    % Generate random chromosome, decode to phenotype, and evaluate
    P(:,i) = ... % random bitstring
    g(:,i) = feval(decodefct, P(:,i));
    f(i) = feval(fitnessfct, g(:,i));

    % Statistics administration
    evalcount = evalcount + 1;
    [fopt, optindex] = max(f);
    opt = P(:,optindex);
    histf(evalcount) = fopt;
  end

  % Evolution loop
  while evalcount < stopeval

    % Generate new population (recombination, mutation)
    for i = 1:mu
      p1 = feval(selectfct, P, f);
      if (rand() < pc)
        p2 = feval(selectfct, P, f);
        Pnew(:,i) = ... % crossover
      else
        Pnew(:,i) = ... % copy
      end
      Pnew(:,i) = ... % mutation
    end

    % Replace old population by new population
    P = Pnew;

    % Decode and evaluate
    for i = 1:mu
      g(:,i) = feval(decodefct, P(:,i));
      f(i) = feval(fitnessfct, g(:,i));
    end

    % Statistics administration
    [fopt, optindex] = max(f);
    opt = P(:,optindex);
    for i = 1:mu
      evalcount = evalcount + 1;
      histf(evalcount) = fopt;
    end

    % Plot statistics
    clf
    subplot(2,1,1)
    plot(histf(1:evalcount))
    subplot(2,1,2)
    bar([1:n],opt)
    xlim([1 n])
    drawnow()
  end
end
```

(a) Complete the code.

(b) Run the GA a couple of times on the ONEMAX problem (= Counting Ones problem) provided below. Use bitstrings of length 100 and an evaluation budget of 100,000 evaluations. Use the phenotype decoding function (which is a dummy function in this case as we do not need any phenotype decoding) and selection function provided below:

```
function f = ONEMAX(a)
  f = sum(a);
end

function g = no_decoding(a)
  g = a;
end

function a = select_proportional(P, f)
  cumsum_f = cumsum(f);
  r = sum(f) * rand();
  i = 1;
  while (r >= cumsum_f(i))
    i = i + 1;
  end
  a = P(:,i);
end
```

How does it perform? Do a comparison using different bitstring lengths.

(c) In the lecture, a 'fix' for proportional selection was discussed in which the fitness values are scaled. Implement a function `select_scaled_proportional` which implements this fixed proportional selection method. Does it make a difference on the Counting Ones problem?

(d) Construct a tournament selection method and compare it with the other two.

2. Implement a genotype decoding function such that the GA can be used to optimize real-valued optimization problems. Test it on a 10-dimensional version of the sphere function ($f(\vec{x}) = \sum_{i=1}^{n} x_i^2$ with $n = 10$, $x_i \in \mathbb{R}$), use search intervals $x_i \in [-10, 10]$.