Unit 1: The Genetic Algorithm

Notes on Implementing Base-10 Genotype Encodings in Computer Code

A simple **genetic algorithm (GA)** may be tasked with solving the optimization problem

$$\begin{aligned}&\text{maximize } f(x)\\&\text{such that } x \in [a,b]\end{aligned}$$

where $a, b \in \mathbb{R}$ with $-9 \leq a < b \leq 9$. If the function f can be evaluated but has no known symbolic expression, an algorithm can try different values of the domain and settle on one that is hopefully close to the optimal value. For example, instead of optimizing over the continuous interval $[a,b]$, we can instead do a combinatorial search over values of x that can be expressed as

$$x = x_s \times \left(x_0 + \frac{x_1}{10} + \frac{x_2}{100} + \frac{x_3}{1000} + \cdots + \frac{x_n}{10^n}\right)$$

where $x_s \in \{-1, +1\}$ represents the sign and $x_i \in \{0, 1, \ldots, 9\}$ for all $i \in \{0, 1, 2, \ldots, n\}$ encodes each of $n+1$ **base-10 digits**. To complete this combinatorial optimization problem with a GA, we represented each of these x values with a "genotype" string:

$$(x_s, x_0, x_1, x_2, x_3, x_4, x_5, \ldots, x_n)$$

where the sign and the $n+1$ digits are each different "alleles" at locations analogous to "genes".

In the implementation of the GA, it is necessary to translate the genotype string back to a real number that can be used to evaluate the objective function $f(x)$. Three example options include:

- Encode each of the $n+2$ genes as separate variables. After the crossover and mutation operators have been applied, use the formula above to calculate the corresponding x value.

- In a language like MATLAB, encode genotype as a scalar sign variable x_s and an $(n+1)$-dimensional vector $\overrightarrow{x_v}$. Then create a vector of place values $\vec{p} = \left(1, \frac{1}{10}, \frac{1}{100}, \ldots, \frac{1}{10^n}\right)^{\mathrm{T}}$ and use the inner product of these two vectors to calculate the absolute value of the answer. That is, calculate:

$$x = x_s\, \overrightarrow{x_v}^{\mathrm{T}}\, \vec{p}.$$

- Encode the genotype as a sign variable and an array with $n+1$ elements. After the crossover and mutation operators have been applied, convert the array to a single real number x by either:
    - Using a "for loop" to iterate through the elements, multiply them by their place values, and sum into a final number.
    - Using a "map" operator (to multiply by the place values) followed by a "reduce" operation (to sum into a single number).

You can find sample code snippets that may be helpful on the next page.

Sample MATLAB Code for a number of tasks that may be useful for implementing a GA:

```
% Vector of place values, long form
p = [1 0.1 0.01 0.001 0.0001]';

% More compact method of generating place values
p = 10.^[0:-1:-4]';

% Scalar storing “sign digit” as well as a vector of other digits
xs = -1;
xv = [0 1 2 3 4]';

% Product of sign digit and inner product of p and xv gives value
% (this example produces -0.1234), which can be used to evaluate
% optimization objective f( xs*xv'*p )
xs*xv'*p

% Method to mutate a single sign digit to either -1 or 1
xs = (-1)^( rand < 0.5 );

% Method to mutate a sigit within xv vector to any value from 0 to 9
xv(2) = random( 'unid', 10, 1 ) - 1;

% Method to generate two random parents
xs1 = (-1)^( rand < 0.5 );
xv1 = random( 'unid', 10, 5, 1 ) - 1;
xs2 = (-1)^( rand < 0.5 );
xv2 = random( 'unid', 10, 5, 1 ) - 1;

% To generate two offspring from a crossover of parents in xv1 and xv2
% (with crossover point between digits 2 and 3)
os1 = xs1; ov1 = [ xv1( 1:2 ); xv2(3:end) ];
os2 = xs2; ov2 = [ xv2( 1:2 ); xv1(3:end) ];

%%% MORE ADVANCED
M = 100;   % M individuals
N = 5;     % N+1 genes (1 sign gene and N digits)

% Population of M individuals with N+1 randomly drawn genes
X = [ (-1).^(rand(M,1)<=0.5) random( 'unid', 10, M, N ) ];

% Decoding vector
pv = [1 10.^(0:-1:-(N-1))]';

% Simple optimization objective (f(x)=x^2) as an anonymous function
% (note the elementwise power operation for vectorization)
f = @(x) x.^2;    % NOTE: THIS IS *NOT* THE FUNCTION USED IN ASSIGNMENT

% Calculate fitness of all M individuals simultaneously, assuming
% optimization objective is identical to fitness
F = f( X*pv );

% Normalized fitness for fitness-proportionate selection
Fn = F./sum(F);

% Draw R random parent identities according to fitness-prop. selection
R = 10;
R_parents = interp1( cumsum(Fn), 1:M, rand(R,1), 'next', 'extrap');
```