CS 224 Introduction to Python — Spring 2019

Closed Lab 3

University of Wisconsin – La Crosse — *March 6*

Functions

Description: Genetic algorithms (GAs) find approximate solutions to difficult problems by employing the principles of Darwinian evolution. The algorithm maintains a population of candidate solutions that are iteratively improved through a number of generations using reproduction, mutation, and survival of the fittest. Reproduction takes the form of *crossover*. In crossover, two individuals (parents) are *recombined* to create some number of new individuals (children). Each child contains some of the genome of each parent. For this assignment, you will implement several crossover functions.

The genomes for this assignment are binary strings with fixed length 64. (In general, genomes can be of any size and contain bits, integers, real numbers, etc.) That is, each individual has a list of 64 bits that encodes its solution to the problem.

Details: Details of the crossover operations appear below. Each will be implemented as a Python function that returns two children. You should also write a main program that calls the crossover functions to test them.

1. Uniform crossover: Much like the problem from Quiz 3, uniform crossover exchanges bits. This function will create a copy of each parent. `child1` is a copy of `parent1` and `child2` is a copy of `parent2`. Each bit in `child1` is replaced with the corresponding bit from `parent2` with probability `indpb`. `child2` is similarly altered by `parent1`.

   parameters:

   - `parent1`: list of bits
   - `parent2`: list of bits
   - `indpb`: probability in `[0.0, 1.0]`, default value = `0.2`

   returns:

   - `child1`: list of bits, defined above
   - `child2`: list of bits, defined above

2. One-point crossover: In one-point crossover, a **split point** is selected in the genome. Assume that position `k` is chosen (at random) as the split point. Then `child1` would inherit genes `0..k` from `parent1` and genes `k+1..n-1` from `parent2`. `child2` is created from the symmetric case.

   parameters:

   - `parent1`: list of bits
   - `parent2`: list of bits

   returns:

   - `child1`: list of bits, defined above
   - `child2`: list of bits, defined above

3. Two-point crossover: In two-point crossover, two **split points** are selected in the genome. Assume that positions `j` and `k` are chosen (at random) as the split points. Then `child1` would inherit genes `0..k` and `j+1..n-1` from `parent1` and genes `k+1..j` from `parent2`. In other words, the *middle section* of `child1` comes from `parent2`. `child2` is formed by the symmetric case.

   parameters:

   - `parent1`: list of bits
   - `parent2`: list of bits

   returns:

   - `child1`: list of bits, defined above
   - `child2`: list of bits, defined above