

Homework Set 1

Due date: Monday Jan. 13.

Written Exercises

HW1.1. Do Thought Exercises 2, 3, 4, 5, and 8 in *An Introduction to Genetic Algorithms*, Chapter 1. Do the exercise and then check your solution with the solutions in the handout. Turn in answers and how you obtained them.

Below are three other written exercises (answer with a few sentences). These are exercises for thought—there is no single correct answer.

HW1.2. Chapter 1 of the book and the lectures described the appeal of biological evolution as a source of inspiration for computational search and learning algorithms. Describe **two** features of biological evolution that might make it disadvantageous as a source of inspiration for such algorithms. (Here is one example: biological populations are typically very large for a given species—on the order of millions to billions of individuals. If populations that large are needed for evolution to be successful, this might be intractable for computational methods to mimic.)

HW1.3. List **four** major differences between genetic algorithms and “steepest ascent hill climbing” (see p. 15). Here is an example: At each step, steepest ascent hill climbing performs exactly one mutation to the current string. GAs can perform more (or less) than one mutation to each string at each generation.

HW1.4. Here is a description of “elite selection” as defined in a Simple GA in C.

For all selection methods, a weight is assigned to each individual in the population as a function of its fitness. These weights are used to select pairs of parents to create the next population, with selection being done by a roulette-wheel method.

Under elite selection, the individuals are ranked by fitness, and the top NUM_ELITE individuals are assigned a weight of 1.0, meaning that they will be selected with equal probabilities. A weight of 0.0 is assigned to all other individuals, meaning that they will not be selected.

Write a similar description of the other selection methods available in the Simple GA in C: linear rank, fitness proportionate, and sigma scaling.

Computer Exercises

HW1.5. Read Chapter 1, computer exercise 1 (p. 31). You can use the “Simple GA in C” code to do this exercise. The fitness function “ $f(x) = \text{number of ones in } x$ ” is the default fitness function given with the code (this is called the “one max” fitness function). To get “fitness-proportionate selection with roulette wheel sampling”, simply make sure that in the “params” file, the SELECTION_METHOD is set to “fitness proportionate”. Use MAX_NUM_FUNCTION_EVALS equal to 5000. (For a population of 100 individuals, this runs the GA for 50 generations.)

Here, more specifically, is what you need to do.

a. Give the average generation at which the string of all 1s (fitness = 20) is first discovered. (When calculating the average, don’t include runs in which the all-1s string was not discovered.) Also give the fraction of “successful” runs, that is, the number of runs on which a string of all 1s was discovered sometime within the 50 generations.

b. **No crossover:** Give the average generation at which the string of all 1s is first discovered, as well as the fraction of successful runs for the same fitness function but with crossover turned off. That is, in the “params” file, CROSSOVER_RATE should be set to 0. Choose a typical run and plot the highest fitness at each generation versus the generation number (you can get this data from the “.short” file for that run).

c. **No mutation:** Give the average generation at which the string of all 1s is first discovered, as well as the fraction of successful runs for the same fitness function, with CROSSOVER_RATE set to 0.7 and MUTATION_RATE set to 0. Choose a typical run and plot the highest fitness at each generation versus generation (you can get this data from the “.short” file for that run).

d. Can you say anything about why the results from a, b, and c look the way they do? (If you have any ideas, put them down in a few sentences or a couple of paragraphs at most.)

HW1.6. Chapter 1, computer exercise 4. Do this only for the “one max”

fitness function. Write a function to do steepest ascent hill climbing. (You can use any parts of “Simple GA in C” code to help with this.) Start out with a random string. Run steepest ascent hill climbing for the same number of fitness function evaluations as in the GA (5000). (Increment the number of fitness evaluations each time you call `calc_fitness`.) Record how many fitness functions it takes to find the first occurrence of all 1s string. Give the average of this over 20 runs of steepest ascent hill climbing. Also give the fraction of successful runs (number of runs on which the all-1s string was discovered within the 5000 fitness function evaluations). How does this compare with the GA? Turn in your code for steepest ascent hill climbing.

HW1.7. Chapter 1, computer exercise 3. Do this only for the “one max” fitness function and for four (rather than 10) different schemas, of your choice. You can either modify the “Simple GA in C” code to record data about the schemas, or if you prefer, run the GA with the `LONG_PRINT` flag set to `TRUE` (in the “params” file). This will print out a “.long” file for each run, which contains all the individuals in the GA population. You can then write a program to gather information on the schemas using this data. For this exercise, plot the number of instances at each generation for each of your 4 schemas. For each schema in your set of 4, choose a generation t , and use the Schema Theorem to calculate the expected number of instances of the schema at generation $t + 1$, given the observed number of instances at generation t . Compare your calculations with your empirical data.