**CS 455 – Module 2 Homework – Genetic Algorithms**

This assignment can be completed individually or with one partner. It is strongly encouraged that CS 455 students partner-up with students in the CS 455 section, and likewise, the CS 595A students partner with CS 595A students.

Submission Criteria:

* Submit your work as a zip file containing
  + Solutions to written exercises
  + Source code for implementation exercise
    - Jupyter notebook with Python code (recommended)
    - Or, Python source code only
* If you are working with a partner, you must indicate clearly in your assignment with whom you are collaborating.

**Problem #1: Map coloring problem – Given the following map [24 points]**

|  |  |
| --- | --- |
| Each region can have one of four colors:   * Red – 00 * White – 01 * Blue – 10 * Green – 11   Chromosome a string of color assignments in the order of the regions:  e.g. for chromosome, 01101011  01|10|10|00 {r1=white,r2= blue, r3=blue,  r4 = red} |  |

Fitness is calculated using:

e.g. fitness(01101011) = 2+2+2+2 = 8

Calculate the fitness for each of the following:

1. fitness(11011011) = \_\_\_\_\_\_\_\_\_8\_\_\_\_\_\_\_\_\_
2. fitness(00010011) = \_\_\_\_\_\_\_\_\_6\_\_\_\_\_\_\_\_\_
3. fitness(11111110) = \_\_\_\_\_\_\_\_\_4\_\_\_\_\_\_\_\_\_
4. fitness(01111010) = \_\_\_\_\_\_\_\_\_6\_\_\_\_\_\_\_\_\_

Given the calculated fitness above, perform crossover on the two fittest chromosomes **after** the 4th bit (counting left to right from 0 to 7).

Chromosome 1: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11011011\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Chromosome 2: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_00010011\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

New Chromosome1: \_\_\_\_\_\_\_\_\_\_\_\_\_11011**011**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

New Chromosome2: \_\_\_\_\_\_\_\_\_\_\_\_\_00010**011**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fitness(new chromosome1) = \_\_\_\_\_\_\_\_8\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fitness(new chromosome2) = \_\_\_\_\_\_\_\_\_6\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mutate the chromosome’s 01100011 5th bit and calculate the new fitness

New Chromosome: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_01100111\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fitness(new chromosome) = \_\_\_\_\_\_\_\_\_\_6\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Problem #2: Local Search and Evolutionary Computing [10 points]**

1. [4 points] [short answer] Identify and describe two methods of helping a local search become unstuck from a local maxima (if maximizing) or minima (if minimizing)?
2. Mutation causes the search to randomly explore new regions
3. Change where in the chromosome that crossover occurs to help change the system to grow in a different way
4. [4 points] [short answer] Why is mutation important to genetic algorithms?

It helps prevent the GA from getting stuck in local maxima or optima and it helps the GA explore new areas/solution paths.

1. [1 point] [True or False] Each pair of chromosomes selected as parents must crossover.

**False**

1. [1 point] [True or False] Mutation probability should be kept significantly lower than crossover probability

**True**

**Problem #3: Multi-knapsack Problem GA [66 points total]**

The knapsack problem is a classic constraints and optimization problem. You will be asked to use a GA to implement a variety of the knapsack problem called the multi-knapsack problem.

You are free to build off of the GA code provided by Dr. Stansbury on github, but you are not required to.

***What is the Knapsack Problem?***

A set of items (x) have both a weight () and a value (or profit, ). A knapsack has a weight capacity of . The goal is to maximize the value of items stored within the knapsack such that the total weight of its contents does not exceed W.

Subject to:

}

The first constraint says that the weight capacity cannot be exceeded. The second constraint says that each item in x has a value of 0 – not in knapsack or 1 – in the knapsack.

**What is the Multi-Knapsack Problem?**

A set of k knapsacks can carry the items. We wish to maximize the total value of items stored across all K knapsacks while ensuring that no knapsack violates its weight capacity, W (same capacity for all k knapsacks).

Subject to

} and }

The first constraint says that no knapsack can exceed its capacity. The second constraint states that each item can only exist in one knapsack. The final constraint states that each item in x can only have a value of 0 – not in the bag ith bag or 1 – in the ith bag.

**Task 1: Show graphically how you would represent the state of the multi-knapsack problem as a chromosome. Briefly justify your design. [15 points]**

A screenshot of a cell phone

Description automatically generated

Each major section of the chromosome represents an item. The number stored in each section is the bag number that that item is stored in. So a chromosome with item 0 stored in bag 3 and item 1 stored in no bag would look like: [011][000] where in this example there would be a maximum of 7 bags. The first set of 3 digits represents the first item, and the second set of 3 digits represents the second item.

**Task 2: Specify the equation you will use to calculate the fitness of a chromosome for the multi-knapsack problem? Briefly justify your design. [15 points]**

This function will reward the chromosome directly based upon profit. It multiplies the profit from each bag by how off weight it is. By taking the absolute value of the available weight, if the bag is either overweight or underweight, it will then decrease the value for that bag, and therefore the value of that chromosome. Then, it finds the overall sum of these profit to weight ratios for all bags.

**Task 3: Implement the multi-knapsack problem. [36 points]**

For your implementation, your program should be able to receive four problem inputs:

* p – a list of n values. Each value is the profit of the ith available item.
* w – a list of n values. Each value is the weight of the ith available item.
* W – a scalar value representing the weight capacity
* k – a scalar value representing the number of knapsacks

Your solution should list the following:

* Indices of the items that will be put into the knapsack.
* Final weight of the knapsack.
* Final profit from the knapsack.
* Final fitness of the knapsack state.

**Task 4: Submit.**

Submit the answers to task 1 and task 2 with the rest of your homework answers for problems 1 and 2.

Submit your source code in the .zip file. You can submit as a .py python script or a .ipyn Jupyter notebook.