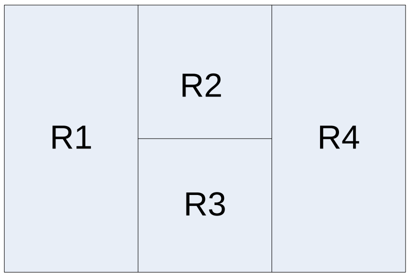
CS 455 – Module 2 Homework – Genetic Algorithms

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:

$$fitness = \sum_{region\ i} number\ of\ nonconflicting\ neighbors\ of\ region\ i$$

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

Calculate the fitness for each of the following:

i. fitness(11011011) = ____________________

ii. fitness(00010011) = ____________________

iii. fitness(11111110) = ____________________

iv. fitness(01111010) = ____________________

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: ____

Chromosome 2: ____

New Chromosome1: _____________________________________

New Chromosome2: _____________________________________

Fitness(new chromosome1) = ______________________________

Fitness(new chromosome2) = ______________________________

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

New Chromosome: _____________________________________

Fitness(new chromosome) = ______________________________

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

a. [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)?

b. [4 points] [short answer] Why is mutation important to genetic algorithms?

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

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

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 (w_i) and a value (or profit, p_i). A knapsack has a weight capacity of W. 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.

$$maximize \sum_{j=1}^{n} p_j * x_j$$

Subject to:

$$\sum_{j=1}^{n} w_j * x_j \leq W$$

$$x_j \in \{0,1\} \quad \forall j \in \{1,..,n\}$$

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 within across all K knapsacks while ensuring that no knapsack violates its weight capacity, W (same capacity for all k knapsacks).

$$maximize \sum_{i=1}^{k} \sum_{j=1}^{n} p_j * x_{i,j}$$

Subject to

$$\sum_{j=1}^{n} w_j * x_{i,j} \leq W, for\ all\ 1 \leq i \leq k$$

$$\sum_{i=1}^{k} x_{i,j} \leq 1, for\ all\ 1 \leq i \leq n$$

$$x_{i,j} \in \{0,1\} \quad \forall j \in \{1, .., k\} \text{ and } j \in \{1, .., n\}$$

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 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]

Show an example of what your chromosome would look like. Label the structure of the chromosome as appropriate. Provide a brief description of your chromosome design (1-2 paragraphs).

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]

Recall that you want to reward based on profit and penalize based upon excessive weight.

Also note, that the current roulette wheel implementation given by Dr. Stansbury on github does not work with negative fitness values. You may need to consider how this will impact your fitness function, or you may choose to modify the strategy for selecting parents based on fitness to another implementation.

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.