CS4300Spring 2016 Hauschild

Project 2
Due Date: 2/29/2014

Submission: Please submit your completed source files and any documentation through MyGateway submission system. **Please do not email them to me.**

[100pt] In this project you will implement a simple genetic algorithm that only uses crossover and mutation which attempts to solve the onemax problem

Start by prompting the user for two quantities, the population size and the size of the string. Using this information, create a population of randomly generated binary strings of the specified size.

Suppose the population given is N=100. Each generation you start by using tournament selection(with $k = 2$) twice to select two parents and then perform crossover to generate two children which you will add to a child population. You continue to do this until your child population has N-1 members (99 in this case). This means you will have done 99*2 total tournament selections and 99 uniform crossovers (notes on crossover later), with each crossover generating two children from two parents. Note that parents can be selected multiple times, and in fact a crossover could be between two identical (even the same individual) parents. Once you are done generating the child population, add the highest ranked fitness individual from the parent population to the child population and have this new population of children replace the entire parent population. Repeat this process for each generation.

Your fitness function is the onemax function, which is the sum of 1s in the string. The global optimum is simply the size of the string, with higher fitness being desired.

Your crossover operator that you should use is uniform crossover. Uniform crossover generates a child by randomly picking from which parent to get each bit (chance of a bit from either of the parents is 50%). Uniform crossover should generate two children, that are opposite of each other. This crossover operator itself (not the individual chance to take a bit from each parent when applying the operator) should only be applied with probability of $p_c = 0.6$. What this means is, 40% of the time, your uniform crossover operator should simply copy the parents as the children, without modification. 60% of the time, it should perform the crossover that generates new children by randomly taking bits from each.

You should also use a mutation operator that, after you generate a child using crossover or copy, you mutate an individual bit with probability of 1/n where n is the size of the string. If a bit is selected for mutation, simply flip its value to 0 if it was 1 and 1 if it was 0.

Each generation display the average, best and worst fitness found in the population. Have your algorithm terminate if the global optimum is discovered or average fitness did not improve from the previous generation. Your average, best and worst fitness should allow non-integer values.

NOTE: Due to the algorithm being simplified (Using a simplified replacement strategy and only using elitism with one individual), it is possible that the results can be poor depending on the initial population and the settings. Given reasonable values, it should get very close to maximum fitness. However, do not assume it will find the global optimum every time.