CSE496/896 AI and Heuristics in Software Engineering
Homework II

Handed out: Monda , Februar 1st, 2016
Due: Monda , Februar 15th, at 10PM
Late Assignments ill be accepted until Wednesda , Februar 17th at 10PM. A 10% late deduction ill appl until Tuesda at 10PM and 20% if submitted b Wednesda at 10PM.

There are two pieces to this assignment. You will implement an algorithm to solve a small problem and answer a set of questions related to this. You must complete and submit both parts.

Assignment: Assume ou are testing a program and ant to cover a particular branch of the program using a GA. You ould normall generate a population of solutions and then run the application and calculate a fitness based on control flo /dependence (as e have seen in the literature). For this home ork e are simplif ing the problem to make it tractable in a short time frame. You ill rite a GA for testing a program, but the fitness is going to be based on a kno n target string that ill lead to covering the specific branch ou are interested in covering. You ill use a chromosome consisting of 17 letter strings. The alleles for each gene are the same the are the ASCII characters bet een A and -- i.e. all upper and lo er case letters and the s mbols in bet een these. **The optimal solution in this case is known since we have simplified the problem**. It is the string **HeuristicSE_andAI** . Write a genetic algorithm to search for the optimal (in this case kno n) solution.

Part I: The initial parameters you should use for your genetic algorithm are as follows:

- Encode our chromosome as an 17 integer arra . Each gene can take one of 58 possible alleles. (You can either use the actual ASCII values or start at 0 or 1. This is our choice)

- Use an initial population of 32 (this population ill onl be used as a an initiali ation step). Generate the chromosomes randoml .

- Ever generation after the first ill have a population of 16.

- At each iteration, select the **best 8** (rank order) and mate using a **one-point crossover** to create t o ne individuals. Pairing should be done b alternating the individuals matched so that even and odds are matched (i.e. 1 ith 3, 2 ith 4, etc.). The others (belo the best 8) are killed off. At the end of each mating session, ou should have a population of 16 again.

- Use a mutation rate of .03

- Use t o different fitness functions as follo s (both have optimal fitness of 0):

    - The first function, tests the distance of each chromosome from the optimal

solution. It adds the square of this distance for all chromosomes. For instance in a string of length 2 ith an optimal solution of no and a current solution of os ould give ou a fitness of (1+16=17).

- The second fitness function uses a binar decision for each distance. If the allele matches it is a 0, other ise it is a one. The sum of the distances is the fitness of this individual. In the e ample above the fitness ould be 2.

- Run this algorithm 10 times for each of the fitness functions. Record the number of generations required to converge on the optimal solution for each run. Report the ma , min and standard deviation for the t o fitness functions.

- Remove the mutation step. Run the same e periments. Hint: You ma need to set a ma imum number of generations as a termination for this part!

Questions:

1) Which of the fitness functions orks better on this particular dataset? (ou can run more iterations if ou ant to get cleaner data). If the are both the same ou should comment on this as ell. When ou ans er this question ou should quantif if our measurement for better is in terms of number of generations, reliabilit of convergence, CPU time, population variation over generations, etc.

2) Comment on the effect of removing the mutation operator in the first part

3) Run some e periments to see ho a random generation ould do for this problem. E plain ho ou set up this e periment.

II. Experimenting with the Algorithm

In this section ou ill manipulate some of the parameters in this algorithm and tr to determine hat effect the have on the running of the algorithm. You should **choose at least 2** of the parameters for this algorithm from the follo ing list:

- Encoding, fitness function, selection, mutation, crossover, pairing, population si e

- For each of the t o parameters that ou have chosen, select one alternative method or representation. For instance, in selection ou ma tr tournament selection and for mutation ou ma decide to var the mutation rate over time. For crossover, ou ma choose a different crossover and for population ou ma decide to use a different si e. The goal is to pick something that ou think ill have an impact on the abilit of our search to converge (or not) to an optimal solution. You ma ant to informall tr a fe things first.

Questions to Answer:

1) For each of the parameters ou have selected, run e periments to determine hat the effect of manipulating these has on our algorithm. You can select the better of the t o fitness functions above as the basis for these e periments. Comment ith respect to the specific metrics ou have chosen. E plain ho ou have implemented the alternative parameters.

Visualization:

Create a visuali ation of the running of our algorithm for at least 2 different settings. You can do this in either 2 or 3 dimensions. (for this part ou can choose to visuali e the actual algorithm or ou ma use one of the visuali ations e have seen to sho population diversit at different stages).

Question to Ans er:

1) Comment on hat information, if an , the visuali ation provides to the developer of this algorithm.

Guidelines:

- Your program just needs to satisf the functionalit and clearl sho that it does. I am not interested in a fanc /GUI interface, **but** it should clearl print out a solution hen it is found or let me kno if one is not found.
- Include a short README file ith our program giving me directions for compiling and running. List an kno n issues (i.e. if it isn t orking properl , doesn t al a s converge, etc.)
- Your program must run on the cse server. It can be ritten in an language that e ists on cse as long as I have clear directions for compiling/running. **You may not use any GA libraries for this assignment.** The algorithm must be our o n.
- For this program ou do not need to spend time orking on run time efficienc .
- Your program should be ritten in a modular fashion so that it is eas to read/comprehend. Submit all source code.

Handing In Your Program:

1) Submit our source code and ans ers using the cse ebhandin: http://cse.unl.edu/ handin/.