COMS30014 - AI

Genetic Algorithm Lab 1: The Simple GA

1. Lab Exercises

==========

Work through the following questions with a group of other students. If you run into trouble, find out how other people are making progress. Share and compare your answers. If you can explain your answer to someone else successfully, you have probably understood it. If you can complete the lab and understand what you have done, you will be able to answer questions on the exam easily and will be able to do a better piece of coursework. If you free ride on the work of others they will get the benefit of the lab and you won't. :(

For each question that requires you to run code in order to achieve an answer, define a function that executes the appropriate GA runs e.g.,

```
def q1():
    gens, best = do_the_ga()
```

Q1. Run the Standard GA Set Up once. How good was your final evolved solution, and, if it was perfect, how many generations did it take the algorithm to find it? Share your result with the other people in your group. What's the range* of performance like?

* Consider: what are good measures of performance? Average score, best score, worst score? And what are good measures of variability, variety, predictability? Max score – min score? Interquartile range? Standard deviation? Is it fair to combine the data from runs that found the solution with runs that didn't?

Q2. Explore* how the following GA parameters impact the performance** of the GA. You can divide up these evaluations between your group (e.g., you could explore the impact of varying "target length" while someone else explores the impact of varying "alphabet size") and then compare your results***. Note that the impact of one of these individual factors might interact with the impact of another one...

(Read (a) – (e) and the *, ** and *** notes below before you get started.)

a) tournament size

- vary the tournament size parameter and check impact on performance
- (you could check values between, say, 2 and 10)
- why does tournament size make a difference?
- do you expect a bigger tournament to take longer to execute? why?
- extra question: if tournament_size = pop_size, or tournament_size = 1, is the GA still a GA?

b) target length

- i.e., how do longer or shorter target strings made from the same alphabet affect the GA performance?
- is any particular target easier or more difficult for the GA to find?
- what if we set tournament size to a better value than the default
- (e.g., 3 or 4)?

c) alphabet size

- e.g., add capital letters to the alphabet...
- ...or add digits and other characters...
- how does the size of the alphabet affect the GA performance?
- (alphabet sizes you could try: 27, 53, ~100)

d) pop_size

- what's the impact of pop size?
- how does its impact interact with tournament size?
- extra question: if pop_size=1 is the GA still a GA?

e) mutation rate

- (careful to think about how to implement this one)
- by default mutation is 1 chance in 28,
- i.e., roughly 1 mutation event per offspring
- what if the chance was 2 in 28, 3 in 28, 4 in 28
- or conversely: 0.5 in 28, 0.33 in 28, 0.2 in 28, 0.1 in 28, 0.05 in 28

- the impact of mutation is not straightforward, why?
- extra questions:
- what does setting mutation=0 imply?
- what does setting mutation=1 imply?

* Consider: How do we measure "performance"? How good is the best solution that the GA finds in 1000 generations? If it finds the perfect solution, how quickly was it found? Beyond this you could also consider: how long does each generation of evolution take to simulate? Is that easy to compare across different hardware? What about how many times the fitness function was called? Or how many offspring have been generated?

** Consider: Running the GA takes time, but the GA is stochastic, so each run is different. How many runs do you need to perform in order to get a good answer? Be judicious - don't waste the lab session unnecessarily running the code more times than is necessary...

*** Consider: There are different ways to approach the task of "comparing". Which of you has a good approach? Find out how the other members of your group did it and consider using that approach so that you can make better comparisons between your group members' answers and your own answers for a), b), etc. Things to consider: How much data needs to be collected? How to summarise the data? Are there quick ways to plot the summarised data? Is there a pattern in the summarised data? Is there a statistical test that could be used to confirm the pattern? Don't get stuck on this - maybe come back to it later when you've got further with the lab questions.

Q3 Explore* how different variants of the GA perform:

a) What difference does elitism make?

Elitism is a way of improving the efficiency of a GA. It ensures that one offspring in a new generation is a perfect copy of the highest fitness individual (the elite individual) from the current generation. (The rest of the offspring are generated as normal.)

Implement elitism in the simplest way possible* and compare the performance of the GA with elitism turned on vs. turned off.

Notice that the do_the_ga() function takes a parameter called 'elitism' as an argument, by default it is set to <False>. But if you call do_the_ga(elitism=True) you can override the default. The elitism parameter is passed to the breed and mutate functions. But it is not used by these functions so far. Change these functions so that the elitism parameter affects how they work in the right way. Why does the mutate function need the elitism parameter?

Why does elitism have the effect that it has?

* Consider: one way to approach this is to add an "if elitism: .." command near the start of the breed function, and make a change to the outer loop of the mutate function

b) What difference does starting with a converged initial population make?

In the original code, we initialise the first generation of solutions to be N random strings. What if the initial population was made up of N copies of one random string? We describe this population as *converged* because all the members are the same - the population is converged on one particular solution. Dawkins' original version of this problem used a converged initial population. Why do you think he did that?

How big an effect does this change to the code make on performance?

c) What difference does sex make?

In the original code, we set crossover=0.0 as a default, which means that breeding is "asexual": each offspring is a mutated copy of one parent (its mum). Many GAs employ "sexual" crossover during breeding. Each offspring is a mutated copy of genes combined from two parents.

By setting the crossover parameter to a value between 0 and 1, you can explore what impact this has on GA performance.

There are different ways of combining two parents to form an offspring - one-point crossover is already implemented here. You can implement two-point crossover instead, or add an implementation of uniform crossover (each offspring gene is drawn from either parent with equal probability).

Why does sex help? Is one kind of crossover better than another?

Extra question: If you are interested, Google the "two-fold cost of sex"? What is it? Does it apply in genetic algorithms?

Q4 [This is an additional question that I am including in case Q1-Q3 are completed more quickly than I was expecting. :) Don't feel that you need to complete it, although having a look at it and a think about it could be useful.]

How would you measure the "convergence" in your population? Convergence is the opposite of "diversity". If every individual solution in the current population is identical we would say that the population is currently maximally converged (or minimally diverse). If every individual in the current population is different from every other individual in every way, we would say convergence is minimal and diversity is at a maximum.

Implement a measure of convergence or diversity. There are many ways to measure this aspect of a population and you are free to experiment (one option is to use the "match" between the best solution and the median solution in the current population). Track convergence over the course of one run of the standard set up GA. Describe how it behaves. Is anything surprising?

Consider your answer for Q2e. Does considering how convergence (or diversity) changes over evolutionary time help you explain what was going on with the impact of mutation rate on GA performance?