COMS30014 - AI

Genetic Algorithm Lab 2: Coevolution

1. Lab Exercises

==========

Work through the following questions in your group. If you run into trouble, find out how other people are making progress. Use the lab group chat to 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, you won't benefit from the lab. :(

Q1 Before we get started with Coevolution, consider a *non-coevolutionary* approach to the "counting-ones" problem. We evolve only *one* population, and the fitness of each member of the population is equal to the number of its bits that are set to "1". (Recall the 'Methinks it is like a Weasel' problem from last week; this is a binary version of the same toy problem).

Consider the following aspects of a fitness landscape, and state how each applies to the 'counting-ones' problem. Record your group's best answer.

a) Search Space Dimensionality

b) Search Space Epistasis

c) Number of Search Space Optima

d) Basins of Attraction

e) Deception

f) Neutrality and Neutral Networks

Each answer should be a sentence or two – succinct, accurate, definitive.

Q2 Understanding the coevolutionary genetic algorithm code:

Work in pairs. For each of the following code fragments from the code provided to you: explain the significance of one fragment to your partner, and help your partner to explain the other fragment to you. When you are happy with your explanations consider the next pair of code fragments:

From mutate():

a) `if bias==None: bias = [1.0/len(alphabet)]*len(alphabet)`

b) `i["solution"][:j]+random.choices(alphabet,bias)[0]+i["solution"][j+1:]`

From assess():

c)
```
order = list(range(len(para_pop)))
random.shuffle(order)
```

d)
```
host = host_pop[i]
para = para_pop[order[i]]
```

From apply_virulence():

e) `if max_para_fitness==0: return para_pop`

f)
```
para_pop[i]["fitness"] = (2 * normalised_score / para_lambda) –
                         ( (normalised_score**2)/(para_lambda**2) )
```

Q3 Use the coevolutionary GA code to generate the following graphs (in each case we are interested in how absolute fitness in the host population is changing over generations – but we might also benefit from seeing other measures of performance and progress, e.g., parasite fitness, host and parasite diversity/convergence, relative fitness scores, etc.):

a) Plot a graph of 600 generations of evolution against random opponents (i.e., this is *not* coevolution, just regular *evolution*) - each "host" solution is always assessed against 10 randomly generated bit strings...

What is going wrong? Why aren’t we seeing better hosts evolving?

b) Plot a graph of 600 generations of host-parasite co-evolution - i.e., each "host" solution is assessed against 10 "parasite" solutions and hosts are picked for breeding based on how many games they win, whereas parasites are picked for breeding based on how many games *they* win.

If you are successful in evolving a perfect host, how many generations did it take?

Now divide the following four tasks amongst your lab group and pool together the results that you collect – you may want to collect multiple runs for each scenario. Then explain the differences between graphs 4c, 4d, 4e, and 4f.

Note that you might need to plot additional graphs of different aspects of the evolving populations in order to explain what is going on...

c) Plot a graph of 600 generations of host-parasite coevolution with maximally virulent parasites (lambda=1.0) that have strong mutation bias advantage = 0.9

d) Plot a graph of 600 generations of host-parasite coevolution with moderately virulent parasites (lambda=0.75) that have strong mutation bias advantage = 0.9

e) Plot a graph of 600 generations of host-parasite coevolution with maximally virulent parasites (lambda=1.0) that have weak mutation bias advantage = 0.6

f) Plot a graph of 600 generations of host-parasite coevolution with moderately virulent parasites (lambda=0.75) that have weak mutation bias advantage = 0.6

Q4 If you were applying a coevolutionary genetic algorithm to a real-world problem, why might it not make sense to fix lambda at a constant value for the whole of the run? Can you think of an alternative to using fixed lambda?

Bonus question:

Q5 From evolution to coevolution:

Work in a small group. Think of a problem to do with robotics (or maybe AI software agents) that could in principle be solved by a coevolutionary GA (i.e., two co-evolving populations). Examples could include: robotic tennis players that can "return service" coevolving against smart tennis-ball-firing-machines, or AI chess players that play black coevolving against AI chess players that play white). Describe the coevolutionary set-up for your example: how would fitness be assigned, how would one population compete/co-operate with the other, how would you prevent the problems that can frustrate coevolutionary progress, how would you measure the absolute progress made by your coevolutionary search algorithm?