**Solutions:**

Some code laying out example solutions to the problems from the lab (but no graphs) is here:  [ga-code-coev-answers.py](https://www.ole.bris.ac.uk/bbcswebdav/pid-8277598-dt-content-rid-46021109_2/xid-46021109_2)  
   
Some code that also generates some nice graphs is here:  [ga-coev-code-xkcd.py](https://www.ole.bris.ac.uk/bbcswebdav/pid-8277598-dt-content-rid-47255415_2/xid-47255415_2)  
   
(It imports functions from the graph plotting code - so include this slightly updated version in the same directory or your Python environment:  [ga\_plot\_xkcd.py](https://www.ole.bris.ac.uk/bbcswebdav/pid-8277598-dt-content-rid-47255416_2/xid-47255416_2))

# Lab Answers:  
#  
# -- 1  
#  
# a) Search Space Dimensionality  
#    the dimensionality of the space is the same as the length of the bitstring, L  
# b) Search Space Epistasis  
#    none - the contribution of each bit is independent of the others  
# c) Number of Search Space Optima  
#    one global optimum - no local optima - this is always the case with zero epistasis  
# d) Basins of Attraction  
#    one basic on attraction because there is one optimum  
# e) Deception  
#    no deception because only one basin of attraction  
# f) Neutrality and Neutral Networks  
#    no neutrality in one-bit neighbourhood of each solution  
#    there is neutrality between a solution and some of its 2-bit

#    neighbours as two bit-flips could introduce a '1' and remove

#    a '1' at the same time leaving fitness unchanged.

# -- 2  
#  
# a) if there is no mutation bias supplied to the mutation function, then we want to have an equal chance of each  
#    alphabet symbol being chosen when a gene is mutated, so if there are A symbols in the alphabet we want to set  
#    bias to be a list of A values with each value = 1/A - with a two-bit alphabet, bias = [0.5, 0.5]  
# b) if we need to mutate location j in a solution, we need to add three parts together to make the mutated solution:  
#    part 1 is the original bits before location j  
#    part 2 is a random bit selected from the alphabet using the specified bias probabilities  
#    part 3 is all the original bits after location j  
# c) for each round of competition we want to randomly pair each host with a different parasite  
#    so we start by setting order to be the list [0 ... N-1]  
#    then we shuffle this list so that each element i in the list points to a unique random number between 0 and N-1  
# d) we pit host number i against parasite number order[i] where order[i] is now a unique random number between 0 and N-1  
#    (a more naive approach would be to 1. repeatedly pick a random host and random parasite and make them fight  
#    or to 2. loop through the hosts and make each host fight a randomly selected parasite  
#    - what's the problem with this? some parasites will play more games than others, which will bias  
#    our evaluation of parasites and introduce additional noise into evolution...)  
# e) if the best fitness in the para population is zero then all parasites have fitness zero and we don't need to  
#    map them through the parasite virulence function as they will still all be the same value, so leave them alone  
#    this avoids a division by zero error  
# f) this line implements the parasite virulence function from Cartlidge and Bullock (2004).  
#    a parasite with normalised fitness, x, is given a new fitness value = 2x/lambda - (x^2/lambda^2)  
#    zero fitness always maps to zero, irrespective of the value of lambda or of the other fitness values in the population  
#    if lambda = 1, fitness is maximum for x = 1 and fitness ranks are therefore left unchanged  
#    if lambda = 0.75, fitness is maximum for x = 0.75 and lowest for x=0, the fitness for x=1 is 0.8889  
#    if lambda = 0.5, fitness is maximum for x = 0.5 and lowest for x=0 and x=1

# -- 3

# (a) Evolution is not under any pressure to improve the host solution once hosts are somewhat better than random bitstrings.  
# (b) Coevolution took around 200 generations to find a nearly perfect host. # (c) With strong bias in favour of the parasites, hosts get left behind and the populations disengage.

#     Parasites drift to 90% 1s (their mutation bias), and hosts drift to their average genotype = ~50% 1s

(d) Despite the strong bias in favour of the parasites, hosts no longer get left behind because of moderate virulence.

#     Parasites climb to around 97% 1s and so do hosts.

#     Why don't hosts and paras reach 100% 1s? Moderate virulence prevents parasites from really applying the final push.

# (e)  Now with weaker bias in favour of the parasites, hosts no longer get left behind even with maximum virulence parasites.

#      Hosts are able to keep pace with parasites, and both climb to around 98% or 99% 1s.

#      Hosts and paras get closer to 100% 1s. Maximum virulence parasites are happy to punish hosts for missing even 1 or 2 '1's.

#      What if we reduced host mutation rate might that slow them down and allow disengagement to occur?

# (f) Staying with weaker bias in favour of the parasites, but employing moderate virulence parasites...

#     Hosts are able to keep pace with parasites, and both climb to around 93% or 95% 1s

#     No problems with disengagement, but moderate virulence parasites are not interested in punishing hosts for missing the last few '1's.

# -- 4  
#  
# The lambda parameter governs how hard the algorithm is trying to resist disengagement.  
# If the parasite population has a strong advantage over the hosts (if their task is much easier), then lambda needs to  
# be low, but if the hosts and parasites have an equally challenging task lambda needs to be higher. Since which  
# population is enjoying an advantage might change over evolutionary time, it might make sense to have the lambda  
# parameter change adaptively to reflect what the balance of power might be at any point in coevolutionary progress.  
# See: Cartlidge, J. and Ait-Boudaoud, D. (2011). Autonomous virulence adaptation improves coevolutionary optimization.  
# IEEE Transactions on Evolutionary Computation, 15(2), 215-229.

# -- 5  
#  
# Many possible answers and approaches here. Q5 is primarily to occupy students who have run out of things do to. :)