Module 3 Self Check

David Bishop

September 2024

1 Section 1: General, Hill Climbing, Beam Search, Simulated Annealing

General

1. If the state is 5, what does successors() return?

   If the state is 5, the successors function would return [4,6].

2. If the state is 3, what does find-best-child() return?

   If state is 3, find-best-child would return 4. 4 has the highest value out of the 2 nearest successors at 3.31.

Hill Climbing

1. Suppose we start randomly in state 7. Show and explain the exploration path of hillclimbing. Is this a global maximum?

   To answer this lets talk through the steps of this pseudocode.

   First we start at 7 which has a score of 2.72. find-best-child(7) is called which would return 8 as 8 has a higher value than 7 and 6.

   After that we are at 8 with a score of 3.09. find-best-child(8) is called which returns 9 at 3.37 higher than both 8 and 7.

   Finally we are at 9. find-best-child(9) is called. Neither 10 or 8 have higher scores thus the code ends here returning 9.

   This is a global maximum since state 9 has the highest value.

2. Suppose we start in state 2. Show and explain the exploration path of hill-climbing. Is this a global maximum?

   Once again we can step through this utilizing pseudocode.

   First we start at state 2 with a value of 2.78. We call find-best-child(2) which returns 3 as 3 is 3.14.

   After that we are at state 3 with a value of 3.14. find-best-child() is called and it returns 4 as 4 is higher than 3 at 3.31.

   We are in the final state now as find-best-child(4) is called which enters the inner if statement as neither 3 or 5 has a higher value than 4. Thus it returns 4. This is not the global maximum as there are other values which are higher, specifically 9.

3. What elaborations of hill-climbing might improve the performance of hill-climbing? Are the improvements guaranteed?

   This depends on the definition of improve. If the improvement is that it finds the highest "hill" every time we would want it to not return immediately until it has gone through the entire list. However, this could drastically slow down the program. We could attempt a random restart where instead of going only to the best spot next we randomly restart somewhere else, this is similar to what I just said but more randomized, this would cause a slow down and would not ensure a global maximum is found each time (unless you randomly went to every state). We could try a less random approach and try to find the global maximum utilizing a stochastic method allowing the algorithm to explore the less popular paths, however much like the other methods this would not ensure a global maximum is found.

Beam Search

1. Suppose we start randomly with states 2 and 7 so that k=2. Show and explain the exploration path of beam search.

   First lets restate what beam search will be doing with the given items. k=2 means that it will only handle 2 states at any given level, this works well with our functions as that is the amount of states successor returns. We will start at state 2 and 7.

   So first we would get successors for each state:

   State 2: 1 = 2.45 and 3 = 3.14
   State 7: 6 = 2.98 and 8 = 3.09

   We now have 4 states but that is too many so we choose the best of the states, 3 at 3.14 and 8 at 3.09.

   We then get the successors for these states:

   State 3: 2 = 2.78 and 4 = 3.31
   State 8: 7 = 2.72 and 9 = 3.37

   We then can choose our two successors out of the batch of 4 we have generated, 4 and 9 are the highest. These are the two "peaks" since successors would not find a higher value thus it will return 9 as that has 3.37 the highest value.

Simulated Annealing

1. Suppose we find ourselves in state 4. With reference to the pseudocode, explain how simulated annealing might permit further local search.

   Lets start by looking at what the pseudocode would do starting in state one (can't do any further states as we d not know what annealing_schedule would actually return).

   With Simulated Annealing we start with current = 4 with a value of 3.31. t = 0 as this is first step.

   We enter the do loop incrementing t to 1. temp would be whatever annealing_schedule(1) returns. Assuming it wouldnt be 0 so we pass first if statement. We then chopose a random_successor so either 3 or 5 would be chosen. We then evaluate the difference between the scores of the random successor and our current, if diff is greater than zero we move to that state, else we generate a random number and determine if e(diff/temp) is greater than it, if so we move to the random state else we stay.

   This would help permit further local search as it would allow checking random succession instead of the best succession every time, thus possibly leading to the global maximum. It is good to note that based on this pseudocode as the temperature decreases this would act much more like hill climbing.

2 Section 2: Genetic Algorithm

1. What does a random (ie, arbitrary) individual for this problem look like (ie, what does the chromosome look like? Use a List).

   Based on the informatiuon given to us I believe an arbitrary individual would look like:

   [0, 1, 1, 1, 0, 0, 1, 0, 1]

   This is because we know that we have 3 variables, meaning we need to account for 3 different items. Not only that but we know that we have to put these variables in a range of 0-7 or 8 different values. $2^3 = 8$ so we have to utilize 3 binary numbers per variable giving us a list that has 9 binary variables that can be decoded in 3's.

2. What is the phenotype for your individual from #1?

   We can take our list from part one and start converting from binary to non binary.

   [0,1,1] = 3

   [1,0,0] = 4

   [1,0,1] = 5

   Thus our phenotype is [3, 4, 5].

3. If you had the phenotype for an individual [4, 2, 5], what does its genotype (chromosome) look like?

   This would be the inverse of what we just did so:

   4 = [1, 0, 0]

   2 = [0, 1, 0]

   5 = [1, 0, 0]

   So our genotype is [1, 0, 0, 0, 1, 0, 1, 0, 0]

4. Look at the pseudocode for the genetic algorithm, do you ever need to go from phenotype to genotype (#3)?

   Overall we do not need o go from phenotype to genotype. This is because the algorithm solely works on genotype. However we would want to convert it to actually compute the fitness of the individual.

5. What would a population of 3 individuals like those from Q1 look like? Use a List of Lists.

   [[0, 1, 1, 1, 0, 0, 1, 0, 1], [1, 0, 0, 0, 1, 0, 1, 1, 1], [0, 0, 1, 1, 1, 0, 0, 1, 0]]

3 Section 3: Crossover and Mutation

a. What are the children if the gene index is 5?

   If the gene index is 5 we split right before the 5th position (technically 6th but we start index at 0). So:

   Parent 1: 01011 | 1011001000

   Parent 2: 10100 | 1011110110

   The children then get the first 5 from one parent and the rest from the second so:

   Child 1: 010111011110110

   Child 2: 101001011001000

b. What are the children if the gene index is 12?

   Parent 1: 010111011001 | 000

   Parent 2: 101001011110 | 110

   Child 1: 010111011001110

   Child 2: 101001011110000

c. What are the children if the gene index is 0?

   This implies that we don't actually split anything:

   Child 1: 101001011110110

   Child 2: 010111011001000

1. There are several varieties of mutation operators. One such mutation operator picks a random location and a random symbol.

   We also have this child with 3 possible values for each gene 01201120120221110.

a. What is the result of a mutation with location 9 and symbol 0?

A mutation at location 9 with symbol 0 means we swap the value at 9 with 0. So we initially had
012011201| 2 |0221110
Which is then turned into:
012011201| 0 |0221110
Giving us: 01201120100221110

b. What is the result of a mutation with location 2 and symbol 2?

At location 2 we have this:
01| 2 |01120120221110
As 2 is already at location 2 there is no real mutation leaving us with what we started with:
01201120120221110

2. If the mutation rate is 0.05, and rand() generates 0.0237, does mutation happen or not?

   Assuming here that mutation only happens if rand is lower than the mutation rate, then yes a mutation would happen.

3. If the crossover rate is 0.80, and rand() generates 0.8976, does crossover happen or not? What if rand() generates 0.4329?

   For 0.8976 crossover would not happen, again assuming that it only happens when rand generates below the rate. For 0.4329 crossover would happen.