Evan Edelstein

EN.605.645.82.SP26

module 4 – self check

General:

1) States 4 and 6 ([4,6])
2) The neighbors of state 3 are state 2 and state 4. State 4 would be returned by find-best-child(), since its evaluation function value (3.31) is greater than state 2's (2.78)

Hill-climbing:

3) The successor states from state 7 are state 6 and 8. Since state 8 has a higher evolution function value (3.09) compared to state 6 (2.98) it is chosen as the candidate successor state. Since it has a higher value than state 7 (2.72), state 8 is chosen as the successor state. Next, the neighbors of state 8, state 9 and state 7, are compared, and state 9 is chosen (3.37 > 2.72) as a candidate. Since state 9 has a higher value than state 8, state 9 is transitioned to. Next, the neighbors of state 9, state 10 and state 8, are compared, and state 10 is chosen (3.26> 3.09) as a candidate. Since state 9 has a higher value than state 10, state 9 is kept as the current state, and the hill climbing terminates. Since state 9 has the highest evaluation function value of all the states, it is a global maximum. The table below shows each iteration of the hill-climbing:

Current	Children	Best child	Successor
7	6,8	8	8
8	7,9	9	9
9	8,10	10	9

4) Following the same logic as 3, at each iteration we find the neighbors of the current state, pick the neighbor with the highest evaluation function value and then choose a successor by comparing the evaluation function value of the current state and the best neighbor. The table below shows each iteration of the hill-climbing:

Current	Children	Best child	Successor
2	1,3	3	3
3	2, 4	4	4
4	3,5	5	4

This is not a global maximum since the algorithm did not reach state 9, which has the highest evaluation function value.

5) There are three ways of improving the hill-climbing local search algorithm, each of which introduce randomness into the algorithm to improve the chances of reaching the global optimum by avoiding getting stack in a local-optimum. Option 1, Stochastic Hill-Climbing, generates a set of random possible successor states instead of just looking at the neighboring states. The second improvement, Random Restart, runs the hill climbing algorithm multiple times, over a set of randomly chosen initial states and picks the best solution (using the evaluation function) from the result of each hill-climb. Simulated Annealing, the third improvement, uses a 'temperature' value to introduce a random chance that a successor state with a smaller evaluation function value than the current state is chosen as the successor. Since all three improvements rely on randomness to avoid local optimums, it is not guaranteed that they will find a better solution than standard hill-climbing.

Beam Search:

6) Starting with states 2 and 7, and k =2, we would generate each states successor, [1,3] and [6,8], respectively. Next, the starting states and neighbors are evaluated using the evaluation function:

    1: 2.45 6: 2.98

    2: 2.78 7: 2.72

    3: 3.14 8: 3.09

    The top k = 2 states by the evaluation function are [3, 8]. Next the neighbors of these states are evaluated to get: [[2,4], [7,9]]. They are evaluated as:

    2: 2.78 7: 2.72

    3: 3.14 8: 3.09

    4: 3.31 9: 3.37

    The top k = 2 states are as [9, 4]. Next the neighbors of these states are evaluated to get: [[8, 10], [3, 5]] Evaluating these neighbors we get:

    3: 3.14 8: 3.09

    4: 3.31 9: 3.37

    5: 3.23 10: 3.26

The top k =2 of which are [9,4] again, so the algorithm terminates and state 9 and 4 are compared, and state 9, with the higher evaluation function value is returned.

Visually we can represent this search by highlighting the states compared in each step:

Step	Current	Searched states
1:	[2, 7]	[1 2 3 4 5 6 7 8 9 10]
2:	[3,8]	[1 2 3 4 5 6 7 8 9 10]
3:	[4,9]	[1 2 3 4 5 6 7 8 9 10]

7) Since the neighbors of state 4 have lower evaluation functions value than it, standard hill climbing would terminate when it reached state 4. This is true if the initial state was between 1 and 6, since these states converge to state 4. Simulating annealing provides two mechanisms to leave the local optimum of state 4. Firstly, the successor state is chosen at random [line 8], which allows the hill-climbing to reach states 7-10, even if it started in states 1-6. This is because the difference in value between state 9 and any state [line 9] would result in state 9 being the successor state [line 10]. Secondly, the temperature value allows for a random move to a successor state with an evaluation function lower than state 4's [line 11]. Opening the possibility to move away from state 4's local optimum.

Genetic Algorithm

8) An individual in this population has a phenotype of 3 variables between 0 and 7. We can represent each of these numbers using a 3-bit binary number, and the gene can be the concatenation of each variables 3-bit encoding. An example individual could be [001011101] (color used to split up each number)
9) The phenotype for this individual would be [1, 3, 5]
10) The genotype for the individual would be [100010101]
11) The pseudocode suggest that we do not need to encode the phenotype into a genotype. The only place where this would be applicable would be to provide the starting population to the algorithm, but in the pseudocode the starting population is generated randomly.
12) A population of 3 individuals could look like [[001011101], [011011001], [111001000]]
13) The children would be 010101011110110 and 101011011001000
14) The children would be 010111011000110 and 101001011111000
15) The children would be 101001011110110 and 010111011001000 (switch parents)

16) The resulting child would be 0120112002022111O
17) The resulting child would be 02201120120221110
18) If rand < Pm the mutation occurs. Since 0.0237 is smaller than 0.05 the mutation occurs.
19) Since 0.8976 is greater than 0.80 the mutation does not occur. 0.4329 is less than 0.80 so the mutation occurs.