Nick Rebhun

Artificial Intelligence

Assignment #2

1. Implement a genetic algorithm to find a solution to the Traveling Salesman Problem for the following distance matrix:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-														
2	1	-													
3	10	12	-												
4	2	9	4	-											
5	17	4	18	5	-										
6	13	2	20	3	11	-									
7	19	7	1	11	5	18	-								
8	11	19	4	14	8	6	20	-							
9	6	12	16	19	2	4	9	12	-						
10	5	4	13	1	6	3	10	7	11	-					
11	17	14	20	11	15	15	5	20	8	14	-				
12	1	3	2	7	12	7	13	2	13	14	14	-			
13	12	17	6	16	8	10	3	6	16	10	15	1	-		
14	7	18	15	5	8	16	18	3	15	20	8	13	9	-	
15	10	16	18	2	17	9	19	9	3	19	1	4	17	1	-

Instructions:

- There are 15 cities indicated by 1-15. The distance from city 1 to city 1 is nothing. The distance from city 2 to city 1 is 1. The distance from city 3 to city 1 is 10, and so forth.
- Write the algorithm in pseudo code or in any notation you want that finds the shortest route to take between the cities.
- You do not need to implement the algorithm in a programming language.

- You are just defining and describing a solution to the Traveling Salesman Problem using the information in the chart.

1) Generate 2 random, non-repeating paths (“A” and “B”), with the same start and endpoints, which may be of varying length
2) with an index “N” starting at 1, use a fitness test to evaluate the weight of the “N”th edge for each path.
3) Create a third, working path “C”, and assign its “N”th term to be the node with the more desirable weight, between “A” and “B” (calculated in the previous step)
4) If endpoint has not been reached by either path:
    1) increment the index, use a fitness test to evaluate the weights of the “N”th edge of both “A” and “B” relative to the “N”th node of “C”
    2) assign path “C”s “N”th term to be the node with the more desirable weight (calculated in the previous step)
    3) repeat until an endpoint is reached
5) else if endpoint is reached by one path but not the other:
    1) use a fitness test to evaluate the weights of the “N”th edge of both “A” and “B” relative to the “N”th node of “C”. For whichever path has not reached the endpoint, add up all remaining edges, and use that to represent the weight of the unfinished path
    2) assign path “C”s “N”th term to be the node with the more desirable weight (calculated in the previous step)
6) Evaluate the longest two connecting edges in path C, and mutate the vertex-node to any other random node not already present in path C
7) Repeat steps 1-7, replacing path A with path C, and using a new randomly generated path for path B