Assignment 2 - Exercise 1

Vincent de Vos (0741795)
v.j.h.d.vos@student.tue.nl

Thom Hurks (0828691)
t.p.hurks@student.tue.nl

April 3, 2015

1. What does the function calcdist do?
At first the number of locations and the numbers of routes are calculated. Next the distance from John's house, which is the starting- and end-point, to the other houses is retrieved from the input matrix (the distance from John's house to others contained in row 1 for Dmatf). The loop then calculates the distance from the each first node of all routes to the second node of all routes, from the second node of all routes to the third node of all routes, etc, and adds up all those values. The result is then a vector containing the sum of all distances between subsequent nodes in a route. Lastly, it then adds up the distance from the final node back to John’s house, and then we have the vector containing the total distances of all routes.

2. Suppose that calcdist is called with the distance matrix Dmatf. Indicate which indices (i.e. which columns of Dmatf) correspond to which person in the above table

Dmatf Index	**Person in table**
1	John
2	Sami
3	Anna
4	Mary
5	Ted
6	Bill
7	Toon

3. Could the function calcdist be used to compute the total distance of the route from Sami to Anna to Mary? Motivate your answer.
No, because the function calculates a cycle, meaning it starts at a point and also ends at this same point; this start and end point being John's house. If you would still like to use this function, you could input the route Sami-Anna-Mary and then manually subtract the distance John-Sami and Mary-John from the result in order to get the correct answer.

If you would like to calculate the route Sami-Anna-Mary-Sami, however, you could use the calcdist function as-is, and adjust the input matrix such that index 1 corresponds to Sami. However, in this case you still compute a cycle back to Sami's house, which is not what was asked.

4. Write a Matlab script exhaustive.m, which generates all the possible routes for John's trip and computes the distance for each route by using the function calcdist. Hint: Use the function perms to generate all the possible routes and compute the corresponding distances by using calcdist and Dmatf. Note that calcdist assumes that the route always starts and ends in node number 1 (John in our example). What is (are) the optimal route(s) for John? What is the corresponding optimal distance?

There are two optimal routes for John: [2,5,3,7,6,4] and [4,6,7,3,5,2]. Note that these two routes are basically the same, namely the second route is the first route but then in reverse. They both have the same distance: 22.8749, which is smaller than any of the other routes.

5. Could you use the information from the distance matrix Dmatf directly as in Ant Colony Optimization (ACO)? Motivate your answer by explaining if/how you could implement the information contained in this matrix or what changes would be required.

Yes. This problem is an instance of the Traveling Salesman problem, which was specifically discussed in class as an application of ACO. For ACO we need a graph for the ants to "walk" on, we can create that graph using Dmatf as follows: for each person in the rows (or columns) of Dmatf, create a node. For all nodes, create an edge to all other nodes except the node itself (no loops). Then, look up the distance between each pair of nodes in Dmatf, and make the edge proportionally longer, or use the distance value as a weight. Example: create nodes John and Sami, create an edge between them, give the edge the weight or length 2.9961. Repeat for all nodes.

6. Describe a Genetic Algorithm implementation of this problem, clearly specifying (if applicable) fitness function, constraints, solution encoding, genetic operators (including example), termination criteria. Hint: pay special attention to the crossover and mutator operator. Note: it is not necessary to implement this solution in Matlab.

Fitness function: Take an arbitrary high number constant, that you know is higher than all possible routes (e.g. 10.000) and subtract the distance of the current route through the nodes that a chromosome represents. Shorter routes then have a higher fitness value, longer routes a lower fitness value. Alternative is to use the reciprocal of the tour distance, so 1 divided by the tour distance is the fitness value.

Constraints: Infeasible solutions are given the fitness value of zero. Infeasible solutions are solutions where a node is not visited once, but either zero times or multiple times.

Solution encoding: Since we always start and end at John's house, we do not need to include John in solution encodings. This means that we have a total of six values that we need to encode. We can simply map each person's house to a number, so Sami maps to 1, Anna maps to 2, etc. The solution representation is then a sequence of numbers.

Genetic operator, mutation: We use multiple uniform mutation, where instead of giving each selected element a random new value, we shuffle the values of all selected elements. If we just assign a random value per element, we could generate infeasible solutions where nodes appear multiple times or zero times in the solution. If elements are shuffled, then each node will always appear exactly once in the solution. This is under the assumption that the input is already a feasible solution. Given feasible input solutions, this mutation operator will definitely not generate unfeasible solutions.

Example: For input ABCDEFG we can randomly select indices 3 and 7 and then shuffle those to create the mutated solution ABGDEFC.

Genetic operator, crossover: a typical crossover operator for a TSP problem would be to take a subset of the solution of one parent, and add the cities of the other parent to "fill in the gaps", while leaving the relative order of the cities of the other parent intact.

Example: ABCDEFG <crossover> BECGAFD. Take subset of solution 1, e.g. CDE. Offspring solution then becomes xxCDExx, where the x's still need to be filled in by cities of parent solution 2. If we eliminate CDE from parent 2, since those are already used, we get BGAF. Fill this in from left-to-right in the offspring and we obtain BGCDEAF as the offspring.

Termination criteria: We can terminate if the population as a whole is converging, so if the fitness values of the best individual and the worst individual are within a certain user-determined percentage apart.

7. Explain two forms to include the constraint that Anna has to be picked up right after Toon in the implementation you described above.

One way to do this is to adjust the genetic operators. The mutation operator would have to be adjusted to always make sure Anna and Toon stay together, so they are either both not shuffled, or they are shuffled as a pair and stay together. The crossover operator could be adjusted to always select the crossover points such that they do not split up Toon and Anna; so Toon and Anna are either both together in the static part or they are both in the part that is crossed over.

An alternative is to redefine the input matrix and merge together Toon and Anna to one “Toon and Anna” node. We then have to recalculate the distance values in the following way: the distance values from everyone to “Toon and Anna” would be the distance values from everyone to Toon. The distance values from “Toon and Anna” to everyone else would be the distance value from Toon to Anna plus the distance values from Anna to everyone else. With this adjusted input matrix, we can just run the regular genetic algorithm without specialized changes to the operators or special constraints.

8. Describe a solution encoding that allows you to calculate the lower bound of the number of expected offspring, at a given iteration, of a solution containing Anna as the second person to be picked up and Toon as the last one. Note: it is not necessary to implement this solution in Matlab.

The schema encoding would be, assuming Anna’s value in the encoding scheme is 2 and Toon’s value is 6, “*2***6” where the *’s represent “don’t cares”. The variables from the Schema theorem then become Ld = 4, L = 6, o = 4. We can then use the Schema theorem to predict the presence of a particular schema in future generations. We can calculate the expected number of offspring of the chromosome in the next generation using the mh(i+1) formula from the slides. This gives a lower bound on the number of instances of our schema at the next generation.

9. Describe a Particle Swarm Optimization of this problem (without the constraint that Anna has to be picked up right after Toon). Note: it is not necessary to implement this solution in Matlab.

Fitness function: the same as with the genetic algorithm.
Search Space: the set of all valid tours (visiting every house exactly once)
Position: This is in essence the solution encoding, so it can be the same as with the generic algorithm.
Velocity function: Given a position, produce a new position. This can be done by doing a permutation of the input position. By doing the permutation the same way each time, the particles can “fly in the same direction”, one can calculate a particle’s previous position, etc.

Objective function: Since our search space is a connected graph (all houses are connected to one another) our function has a finite number of values with the global minimum yielding the best solution.

10. John is trying to decide the amount of food (x1) and drinks (x2) to buy for the party. Since the group is quite heterogeneous, Sami has devised a mathematical function to calculate the cost of the food, taking the form

$$R(x) = 20 + x_1^2 + x_2^2 - 10(cos2\pi x_1 + cos2\pi x_2)\,.$$

• Find the minimum of food cost using the genetic algorithm toolbox. Use crossover in a singlepoint and uniform mutation.
• Use as starting point the best solution found using genetic algorithm, use constrained minimization, fmincon to try to find a better solution.
• Increase the number of iterations for the genetic algorithm and check if you obtain a lower cost.
• Plot the obtained optimization results. Include all intermediary iterations and comment on your findings. Highlight the final solution of each algorithm (with a different color and adequate legend).