**Distributed Intelligent Systems**

**Home Work 1**

**Afroze Ibrahim Baqapuri**

**Q.1/**

**a)**

*n* corresponds to high exploitation. Greater the value of *n* means that ants show more exploitation and follow in the footsteps of the popular path being followed (probability of ants deviating from the high pheromone path is low).   
*K* corresponds to high exploration. This creates tendency of ants to deviate from the high pheromone path (more popular path) and “explore” on their own.

**b)**

Small value of *n* preferred when high exploration is desirable (instead of merely following other’s lead), for example if there are multiple path options and not all of them are explored.  
High value of *n* preferred when high exploitation and less personal exploration is desirable. For example if there are just 2 paths of unequal sizes and most ants are following one path then it is likely that is the shortest path, and it is better to just follow that path rather than explore.

**c)**

n = 5  
k = 10  
B = 23  
A = 80 -23 = 57

P(A) = ((10 + 57)^5)/((10 + 57)^5 + (10 + 23)^5)  
P(A) = 0.972

P(B) = 1 – P(A)  
P(B) = 0.028

**Q.2/**

The first ability we would like to introduce in our ants is improved memory, so the ant remembers the exact path it takes to get to the food source, and can easily back-trace that path. One obvious advantage of this would be that ants would be able to avoid loops since they would be aware that they have been to a particular location before (when they end up there again owing to the loop), so to avoid the loop they will try to go to a different path. Ants would also be induced with “patience” so that they would indefinitely continue searching for food without ever trying to “go back” from where they started in hope of trying a new path. They will only return once they find the food source. Moreover ants would be gifted with local-map-building and orientation abilities so lets say if they had to follow a zig zag path on an open space, they could compute the shortest path back to colony (given there are no obsracles), and also if they encountered loops on the way to food source they would be able to intelligently avoid loops on the way back. So basically once they have arrived to the food source they would be able to compute the minimum path back to the colony in the space they have already observed. Finally ants would have 2 types of pheromones, the normal forward pheromone which they secrete wherever they are going in search of food, and this pheromone evaporates quickly. The second pheromone is for the backward path only (from food source to ant colony) and is secreted by ants when they have found food and are returning back, it is secreted in lesser quantities but it evaporates much more slowly. We also note that the nature of ants is to show more explorative behavior in case of the first forward path pheromone, meaning that although a number of ants might take the lead of others, many would like to explore on their own. And also since the forward pheromone evaporates quickly, if not many ants are continuously following the forward pheromone trail, then it dies down quickly. But in case of the second backward pheromone ants begin to show more exploitative behavior and theres a greater probability of ants following the path with greatest amount of backward pheromone, especially if the pheromone levels are high and continue to increase with time (since they evaporate more slowly so may in fact reinforce). The amount of backward pheromone secreted by an ant on way backward from food source depends on firstly the amount of food discovered by the ant and the amount of minimum backward path computed by the ant from food source to colony (from ants previous experience in life it is assumed that it would have an idea to differentiate between large and small food sources and distances travelled to reach there).

Now we shall see how these properties of ants affect their performance on the four maps. In the first equal bridge scenario, it would not matter which bridge the ants opt to take as long as they don’t turn back on their path without locating food source. Since ants have “patience” and are prohibited to do so, hence ants would show good performance in this case.

In the map of 2 unequal paths, at first ants will show more random behavior and there will be almost equal number of ants on both (the longer and the shorter path). The ants on the longer path would reach the food source later on and then retrace their steps back to the food source using again the longer path (since they have not yet explored the shorter path so it is not their local map) and also their backward pheromone level would be of smaller quantity since they travelled a longer distance than the ants on the shorter path (although they yet do not know how much distance the other ants travelled to make their pheromone level smaller than theirs, but since the ants have been naturally fed with the function to calculate pheromone deposits according to distance so their backward pheromone level would automatically be smaller or equal to that of ants following a shorter path). Once the longer path ants reach back to the bifurcation point, many more shorter path ants would already have crossed back to the bifurcation point and so the level of pheromone deposit would already be more than the one at the longer path (and also because the ants in shorter path naturally deposit more pheromone), so when the ant is going back to the food source it would naturally be more inclined to go to the path with greater backward pheromone (the shorter path). With a couple of iterations this would converge and vast majority of ants would exploit the shorter path.

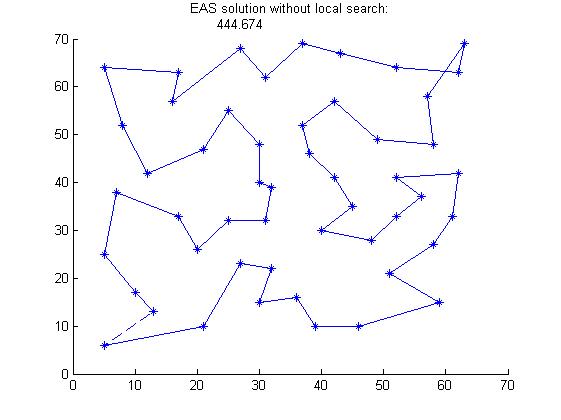
Same logic as discussed above applies to a map with multiple paths from a bifurcation point, so we will not repeat it again.

In case of open landscape map, due to the initial random behavior ants would spread out on the path and generally follow zig-zag paths along the space until they find the food. Now since the ants show a highly explorative behavior, it is highly expected that at least some ants would find the path close to the optimal path (shortest path) and these are the ants which would be able to reach back the most quickly, so newer ants would be more tempted to follow their path (since it would have backward pheromone which other paths don’t yet have). Also when the ants that took a bad path and ultimately reached the food, when they return back their path would have lower backward pheromone so they themselves would be tempted to try out the better path. In the initial phases of this scenario, the pheromone would be distributed throughout the map, with levels at maximum near shortest path. But as time goes the path would converge and since the ants would start showing increasingly exploitative behavior as the backward pheromone of one path keeps increasing.

**Q.3/**

Performance obtained = 444.674

Attaching resulting image for reference. Code and .fig file provided in the supplementary folder.

****

**Q.4/**

**Differences:**  
**1)** In TSP it is required to visit every node in the graph, in navigation problem it just required to reach one node (the final goal node), irrespective of nodes are followed in between.  
**2)** In TSP you have to come back to the original starting node, while in navigation node you don’t have to return back (might return back as another navigation trip but not in the current one). The navigation problem is more like a routing problem .

**Heuristics:**  
**1)** **Shortest route:** Simply the weight of the edges connected with the current node, giving preference to a lower weight (where weight represents length of path).  
**2) Fastest route:** Try to minimize the product of weight of available edges and current traffic on that edge.  
**3. Fuel efficient:** Number of traffic lights, signals, speed breakers and crossings in the next edge we choose to take. Fuel efficiency is increased if we travel at a constant speed (no significant acceleration).

**Q.5/**

First we arrive at the general equation:

m[1] = [150(1 – (s1/4000))] + [200(1 –( s2/4000))] + [-30(1 – (s3/4000))] + [25(1-(s4/4000))]  
m[1] = 295 + (-150\*s1 - 200\*s2 + 30\*s3 + 25\*s4)/4000

Similarly,  
m[2] = 295 + (25\*s1 + 30\*s2 – 200\*s3 – 150\*s4)/4000

Now, we get the respective value of sensors by mapping the graph with the distance of obstacles from sensor (since the figure was not drawn to scale we take approximate values and only consider if object is in line of sight of the sensor):

Thus, we calculate, d(s4) = 3.5 cm, d(s3) = 4cm, d(s2)=d(s1)=0 cm   
And so s4 = 3250, s3 = 3000, s2=s1=0

Hence:   
m[1] = 337.8  
m[2] = 23.1

So the motor moves towards

**Q.6/**

Using the same general equation calculated above:

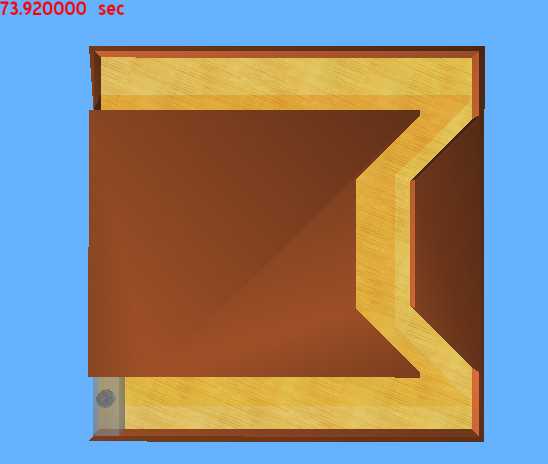
Here we note that d(s1) = d(s4) and d(s2) = d(s3), so in fact we get:

m[1] = m[2] = 337.8

We observe that due to the symmetry of placement of robot sensors and obstacles, the effect is that both motors get signal of same level of speed change and overall effect is that motor does not change direction at all. Thus the robot will keep going forward and hit that obstacle and get stuck over there.

**Q.7/**

Code provided in supplementary folder.

****

**Q.8/**

**5 particles:** mean = 0.00044, standard deviation = 0.0022  
**6 particles:** mean = 0.000269, standard deviation = 0.000889  
**10 particles:** mean = 0.004, standard deviation = 0.0048  
**15 particles:** mean = 0.031, standard deviation = 0.027

Best is with 6 particles (250 iterations) because mean and standard deviation are the least (we want these to be as close to zero as possible, which is the actual solution of sphere function). This is because increasing number of particles, we decrease the total iterations which have an overall bad effect If iteration count goes too low and deteriorates the performace.

**Q.9/**

**5 particles:** mean = 27.9, standard deviation =45.6  
**6 particles:** mean =25.9, standard deviation =48.23  
**10 particles:** mean = 25.9, standard deviation = 40.6  
**15 particles:** mean = 31.68, standard deviation = 33.71

Best is with 10 particles (150 iterations) because although standard deviation of 12 particle simulation is smaller, the overall mean of 10 particle simulation is smaller and that wins because the standard deviations are comparable to each other(we want these to be as close to zero as possible, which is the actual solution of rosenbrock function).

**Q.10/**

Results for rosenbrock function are significantly much worse than those for sphere function. This is because rosenbrock function is more complex and it requires both more particles and more iterations to solve it, but when we increase particles we decrease iterations which is not good.   
The function has a valley which is easy to locate by the particle swarms, but inside the valley locating the global minimum is not trivial and requires more particles and iterations.   
In sphere function, all slopes lead towards the global minimum (there is only one minimum so no issue of getting stuck in local minimum), so it is comparatively easy to locate. In rosenbrock, although the valley is easy to locate, but once inside the valley only one direction points towards the global minimum and all other directions point away from it so it is not easy to locate that. And hence we need to do more exploration of the available space, in other words more particles and more iterations.

**Q.11/**

Code provided in the supplementary folder.

Selected 5 waypoints to begin with, but algorithm “learned” itself to follow just the first waypoint and ignore the rest (since this led to non-wastage of time if the other waypoints did not lie on the best path to the goal). The best waypoint selected was **(1.10 , 0.39)** and total distance travelled was **2.01** units.