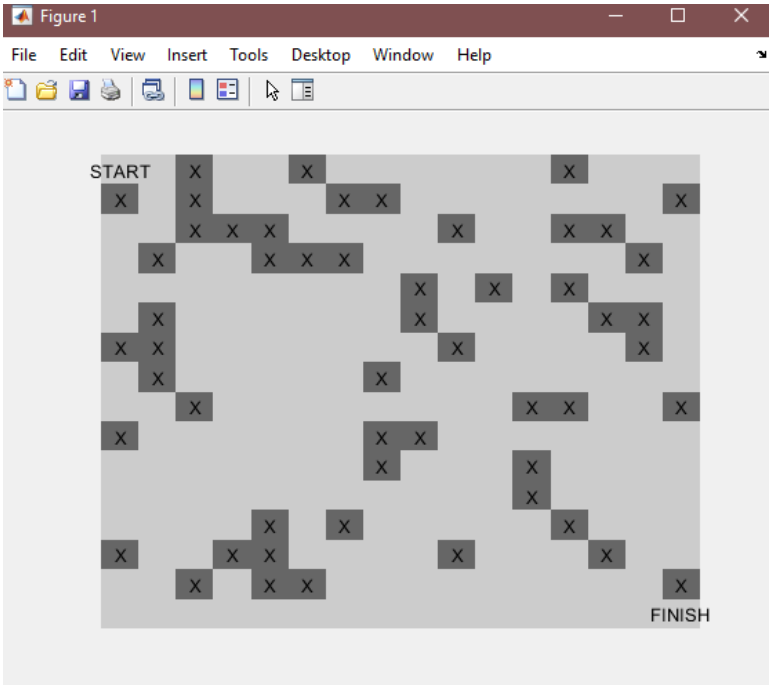
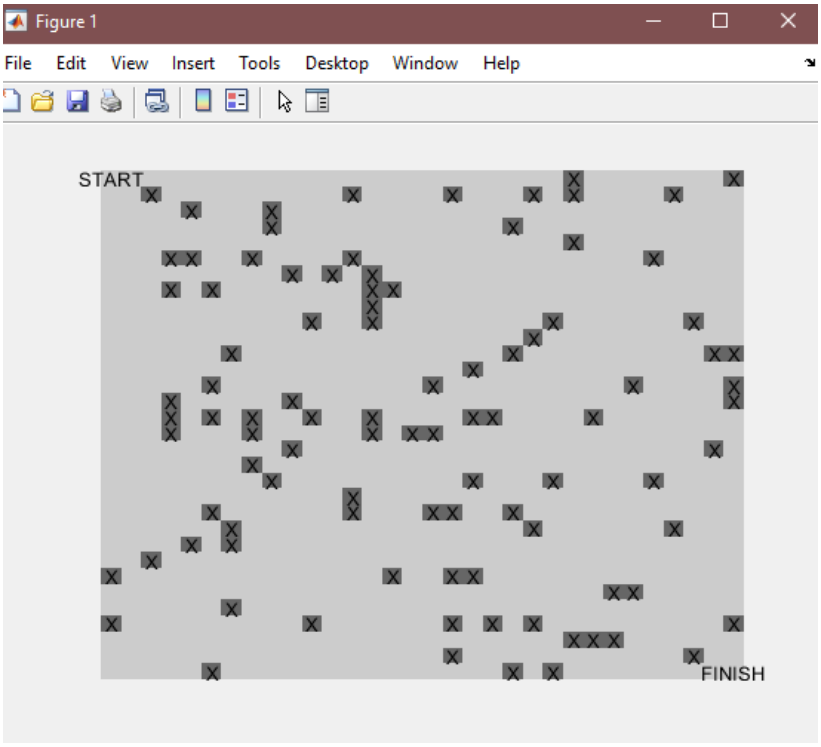
Step 1

16*16 with 60 obstacles



32*32 with 100 obstacles



Step 2

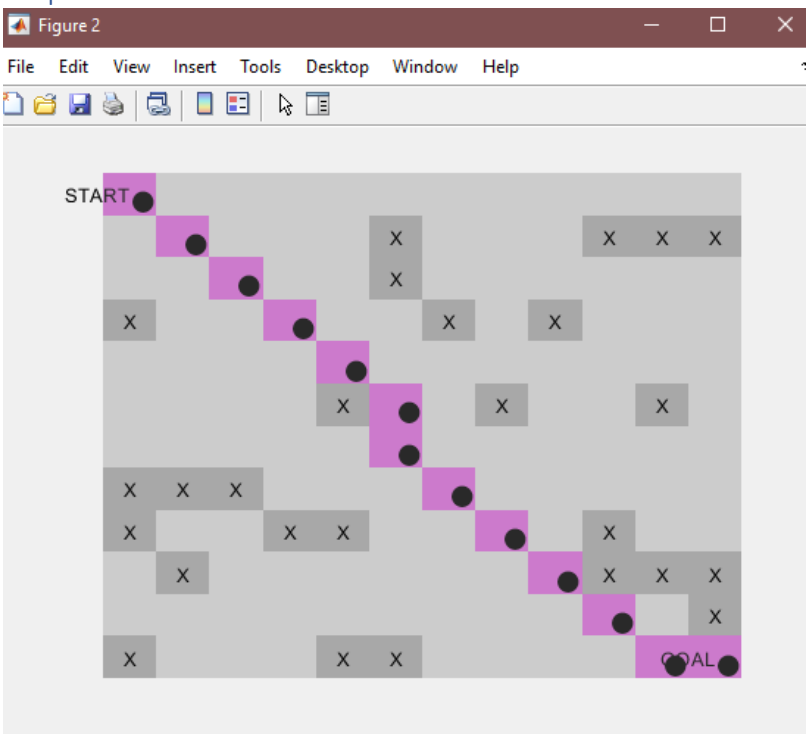


Figure 2
File Edit View Insert Tools Desktop Window Help
START
GOAL

Step 3

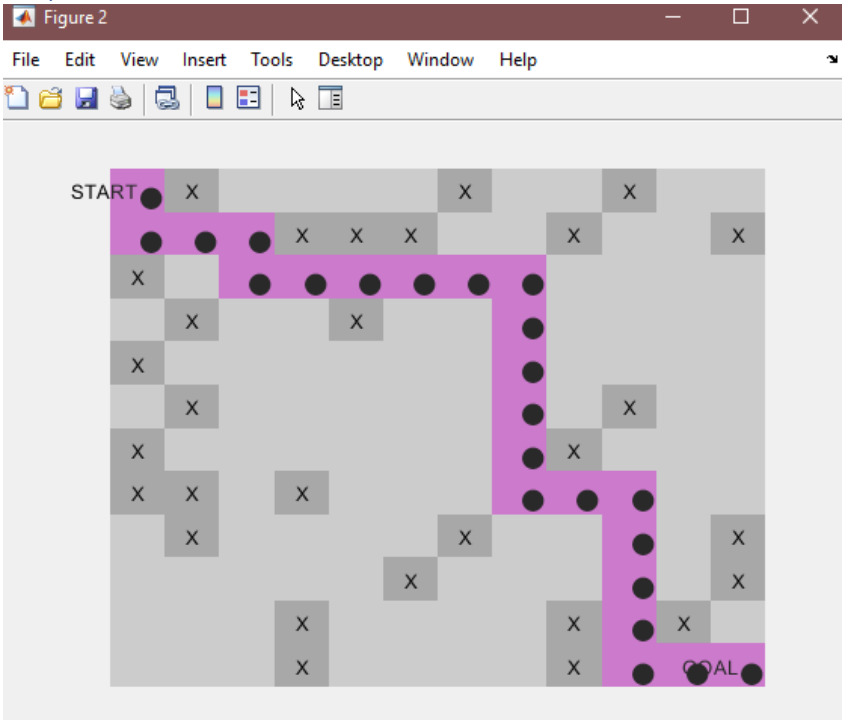


Figure 2
File Edit View Insert Tools Desktop Window Help
START
GOAL

Step 4

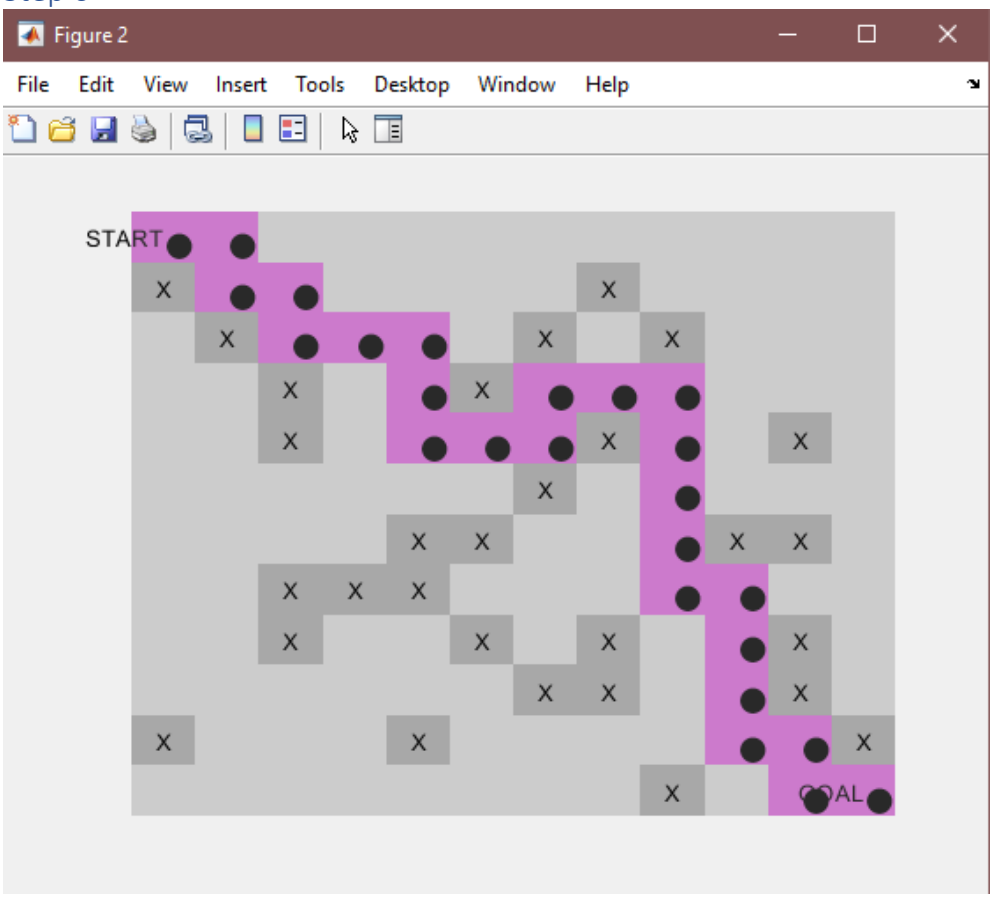
For the Q-table, it is calculated by the maximum future rewards for an action at each state, it has been updated every time an action is chosen. The value in the Q-table is updated using the equation: Q(s,a) = r + gamma*maxQ(s',a'). We got set of values like 0.0056. The reward table is where you keep track of how much you got after each phase. The reward begins at one and increases by one each time a moving to the right action is taken; however, the reward remains unchanged when a going down action is taken.

The dimension of the Q-table is 144*144, this is caused by the total of 12*12 = 144 cells. So, we have 144 actions, this leads to the value of the best path in Q-table a relatively large number. The dimension of the reward table is 1*23. Due to the restriction of not moving diagonally, it will take 12 + 12 – 1 = 23 steps.

Step 5

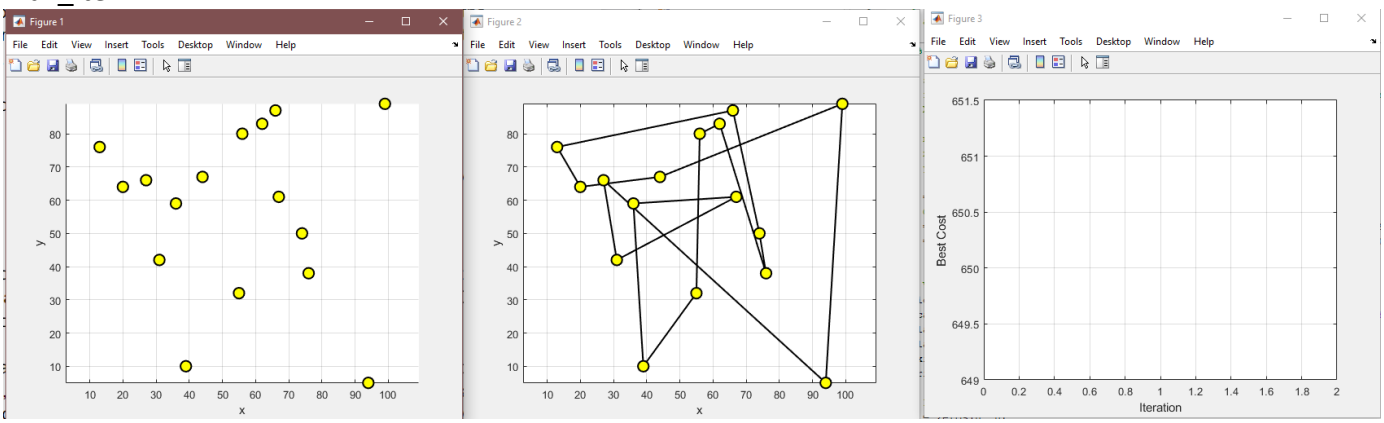
```
% Here we update the q-values as per Bellman's equation
qtable(cs,ns) = (1-alpha) * qtable(cs,ns) + alpha * (reward(cs,ns) + gamma * max_q);
```

Step 6

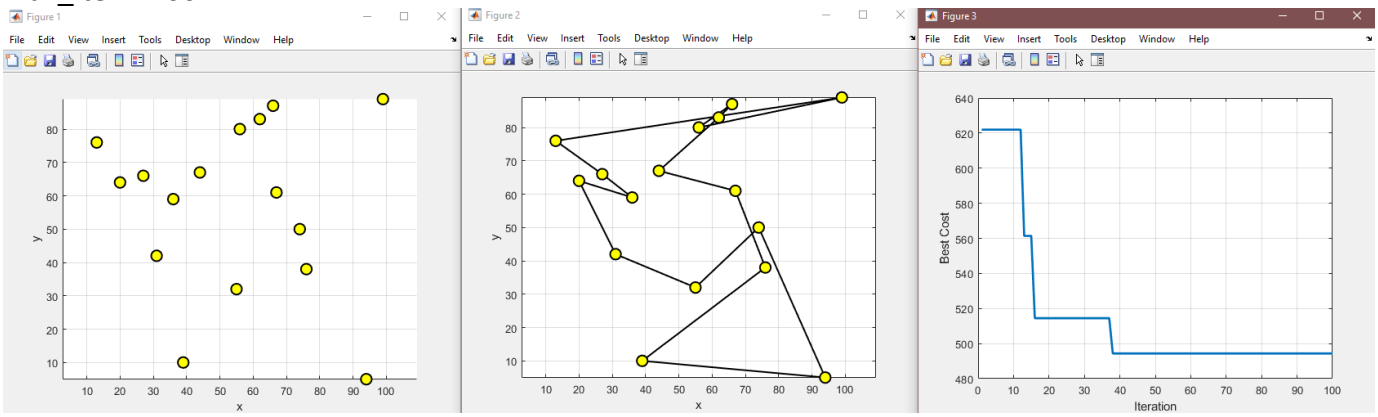


Step 7

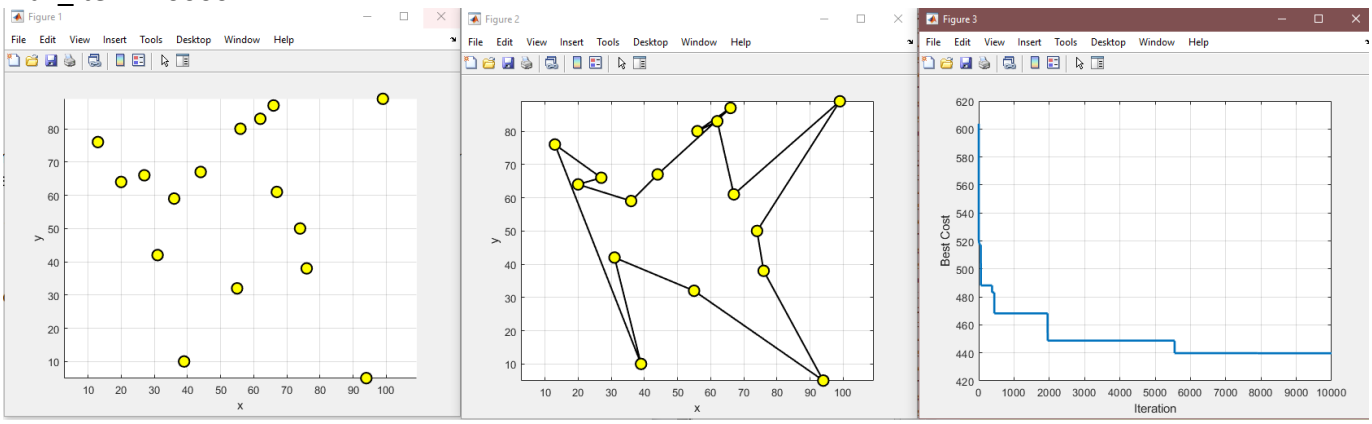
max_iter = 1



max_iter = 100



max_iter = 10000

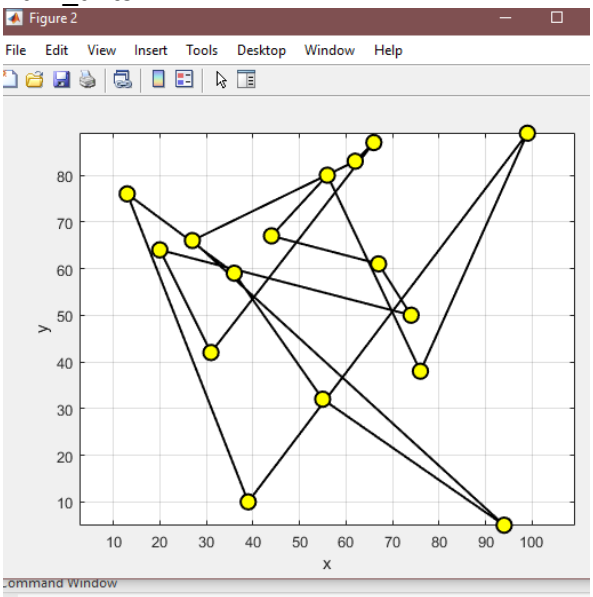


Observation: “max_iter” indicates the number of iterations, as seen in the graphs above, with more iterations, the overall result will be better. The result of the best cost value decreases as the number of iterations increases. This is owing to the fact that as the number of iterations rises, the likelihood of finding a lower and nearly ideal Best Cost path for those points increases.

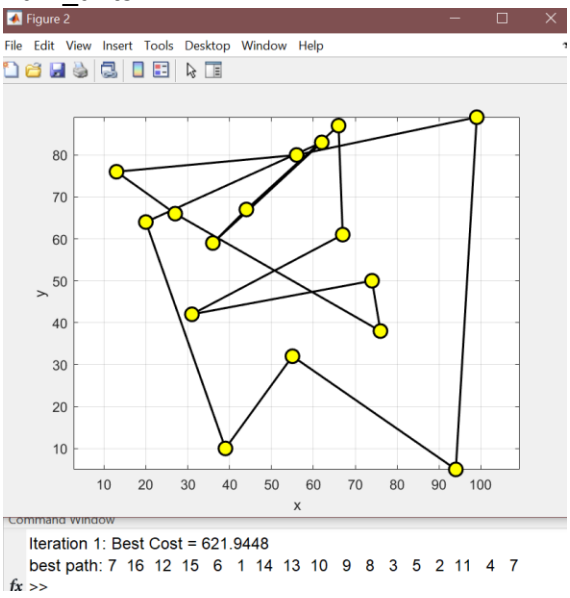
In other words, more iterations allow the algorithm to take more steps and is more likely to locate the shortest path between points, resulting in a more optimal solution.

Step 8

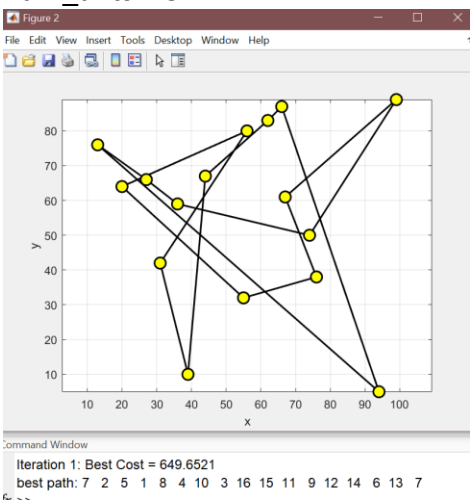
num_ants = 1



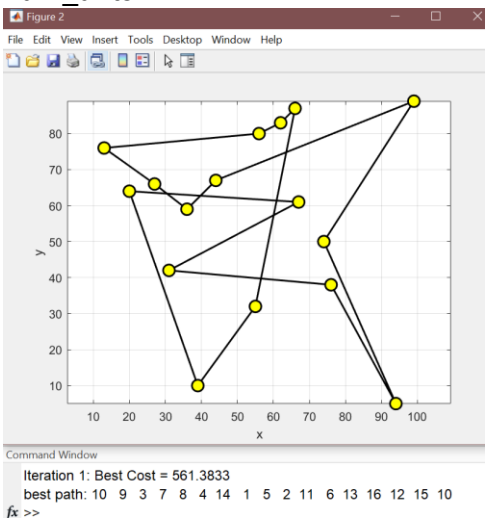
num_ants = 2



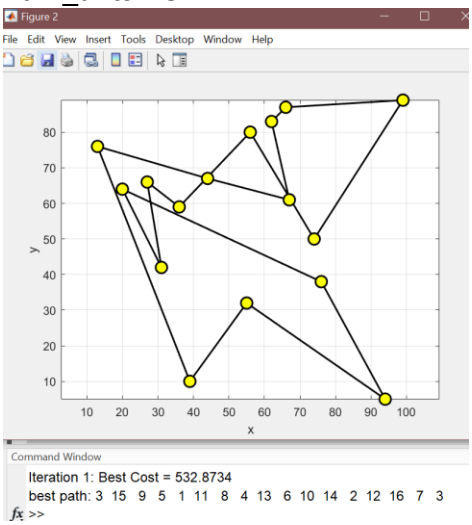
num_ants = 3



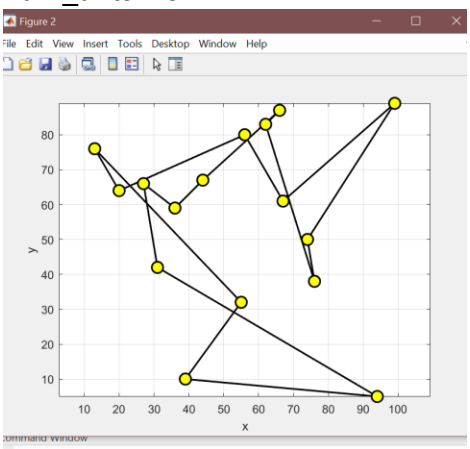
num_ants = 4



num_ants = 5

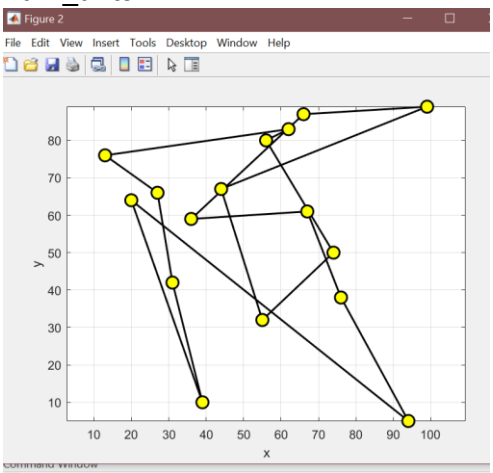


num_ants = 6



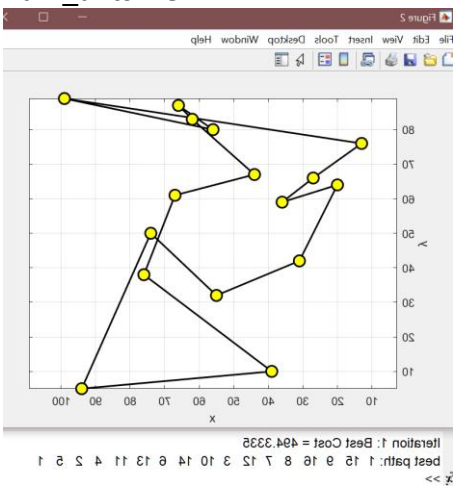

```
Iteration 1: Best Cost = 522.1354
best path: 9  5  1 14 13  6  3  8  4 10 11 15  2 16 12  7  9
>>
```

num_ants = 7

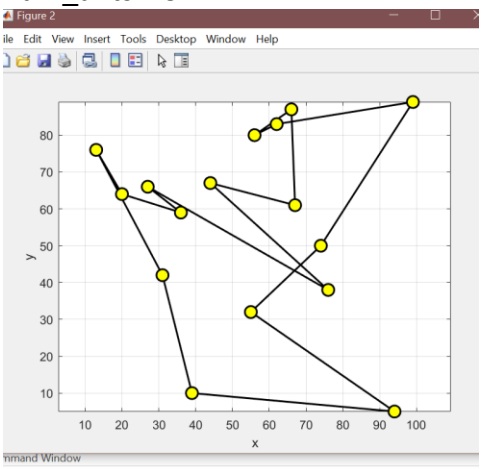



```
Iteration 1: Best Cost = 607.1453
best path: 13  4 14 16  8 11  6  2  5  9 12 15  7  3 10  1 13
>>
```

num_ants = 8

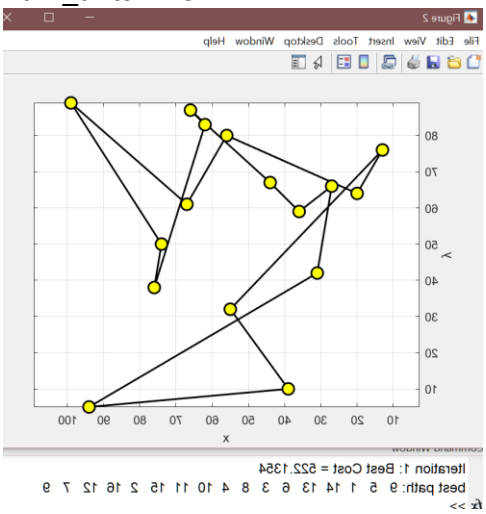


num_ants = 9



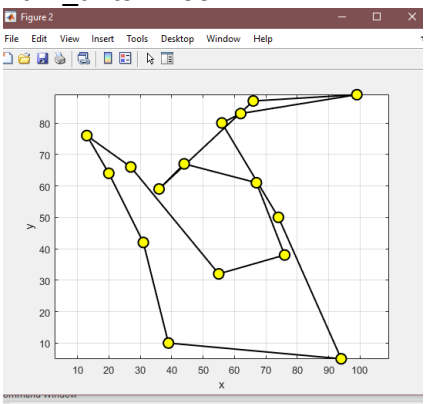

```
Iteration 1: Best Cost = 494.4798
best path: 6  11  13  10  14   3   5   1  15   2   9  12   7  16   8   4   6
>>
```

num_ants = 10



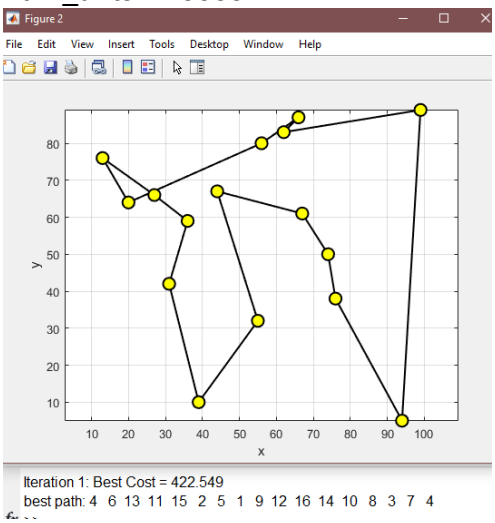
```
Iteration 1: Best Cost = 522.1364
best path: 9  5  1  14  13  6  3  8  4  10  11  15  2  16  12  7  9
>>
```

num_ants = 100




```
Iteration 1: Best Cost = 472.0915
best path: 10  14   1  13   4   6  11   8   7  12   9  15   2   5  16   3  10
>>
```

num_ants = 10000



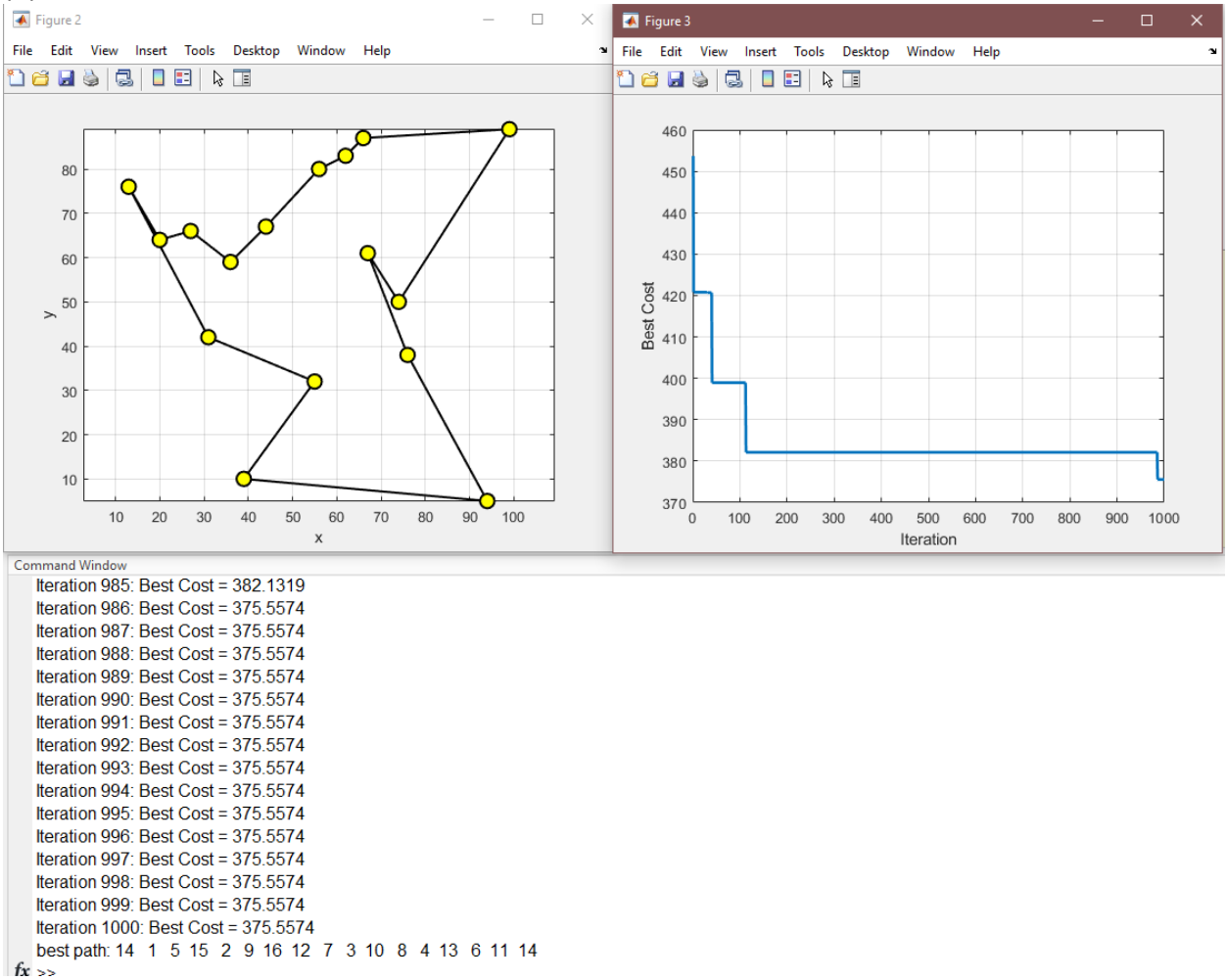
Observation: “num_ants” has similar effect as the “max_iter”. From the graph, when “num_ants” = 1, 100, 10000, their best cost values are 709, 472, 422 respectively. Higher “num_ants” value will lead to a lower best cost value which means a better performance. This makes sense because logically if we have more ants participate to search the path, then, we will have more possible paths, which leads to require less time to find the optimal solution.

Step 9

(a)

```
% Parameters
max_iter = 1000;  % Maximum number of iteration
num_ants = 1000;    % Number of ants
Q        = 1;     % q constant
alpha    = 1;     % Phromone coefficient
beta     = 1;     % Preference for shortest path
rho      = 0.1;   % Evaporation rate
tau0     = 10*Q/(n*mean(L(:))); % Initial phromone
```

(b) total cost: 375.5574



(c) This combination works well since, in the studies above, increasing both the number of iterations and the number of ants results in more optimal solutions. Furthermore, I do not believe this is the global minimum. As the number of iterations and ants grows, so does the complexity of the problem. Even so, there will be some lower-cost value. With more of both “max_iter” and “num_ants” value, there will be a better result.