**Project Report**

**Artificial Intelligence**

Compiled by:   
Pooja Jhunjhunwala  
UIN: 224002725

**Analysis of Exploitation versus Exploration for ACO Dynamic Shortest Path Problem**

**Shortest Path Problem**

Given a weighted, directed graph with n nodes and m edges, and a source and target, find the shortest path between the source and the target vertices in the graph

**Ant Colony Optimization**

In nature, ants wander randomly until they find a food source. The wandering ants deposit a chemical substance known as pheromone on the path that they tread on. Subsequent ants tend to prefer a path with higher concentration of pheromone over one with lesser or no pheromone. This tends to further deposition of pheromone on the same path, and a reinforcement of that path. However, the pheromone evaporates with time, reducing the quantity of pheromone available. A longer path will take more time to be traversed and therefore, be subject to evaporation longer than a shorter path. This leads to the shorter path being reinforced more than the longer path, and eventually most ants converge to the shortest path

**Exploration versus Exploitation**

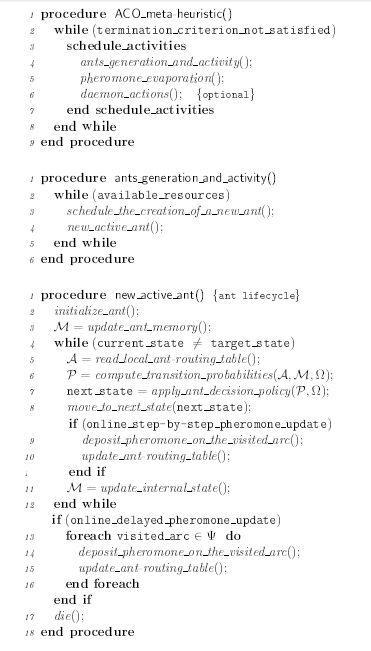
Exploration refers to risk taking behavior in a system. It involves discovering new possibilities, which might not always be optimal, but which are unknown, in the hope that there might exist a better solution in the undiscovered areas. Exploration often involves random behavior.  
Exploitation, on the other hand, makes use of all the information that has already been gathered by the system to make an informed decision. Perfect exploitation does not make use of any randomness in making decisions.

**Objective**

The objective of this project is to analyze the performance of an ACO shortest path algorithm for varying degrees of exploration v/s exploitation. The performance of varying degree of exploration is to be tested against dynamic graphs. It is expected that as the graph becomes more dynamic, a higher degree of exploration will increase the performance of the algorithm. The analysis of dynamic graphs is the future scope of the project. Currently, the project analyzes the performance of the algorithm with increasing levels of exploration.

**Algorithm**

The algorithm is inspired from the paper Ant Colony Optimization Meta Heuristic by Marco Dorigo and Gianni Di Caro. The algorithm, as stated in the paper is:



**Implementation details**

The program was implemented from scratch to allow the maximum level of flexibility with the parameters involved. The solution was coded in Python making use of various libraries like random, matplotlib, and numpy. The implementation for the segments of the program are as follows:

The ants run as parallel threads on a global graph. A pheromone array keeps track of the pheromone on each edge of the graph. Each edge is initialized with some minimum amount of pheromone (maximum edge weight/ 10).

**Pheromone Update:**

Each ant performs 2 types of updates on the pheromones:

1. Online step update:  
   Every time an ant makes a transition from one node to another, it deposits pheromone equal to the reciprocal of the edge weight on the edge connecting the two nodes. This helps in strengthening the shorter edges more than the longer edges.
2. Online delayed update:  
   Every time an ant reaches the target node, it updates all the edges in its path with the pheromone of value (1/path cost). This helps in reinforcing the shorter paths faster than the longer paths, and to ensure that the ants do not perform only local optimization

In each case, the pheromone update happens by the following rule:  
pheromone[i][j] = (1 – d) \* pheromone[i][j] + d\*newPheromone

Where d is the decay constant, that is, it determines how much of the old pheromone should be decayed before adding the new pheromone. For d = 1, it completely evaporates the old pheromone

**Picking the next node:**

In order to pick the next node, the ant has an option of performing exploitation or exploration. This choice is determined by the exploitation constant, alpha. The probability of selecting a particular node depends on the following equation:

weight(i) = (1 – alpha) \* 1 + alpha \* pheromone(i)  
Where pheromone(i) is the amount of pheromone on edge i

Therefore, for all outgoing edges from a node, the probability of picking edge i is given by:  
weight(i)/sum(weight(j)) over all outgoing edges from the node

An alpha value of 1 will ensure complete exploitation, whereas an alpha value of 0 will imply complete exploration. As alpha goes from 1 to 0, the level of exploration will increase.

**Pheromone Evaporation:**

The pheromone from all edges is evaporated at regular intervals so as to ensure that the ants do not start reinforcing a local suboptimal solution. This process runs as a separate thread, independent of the ants running and operates on the same global pheromone matrix.

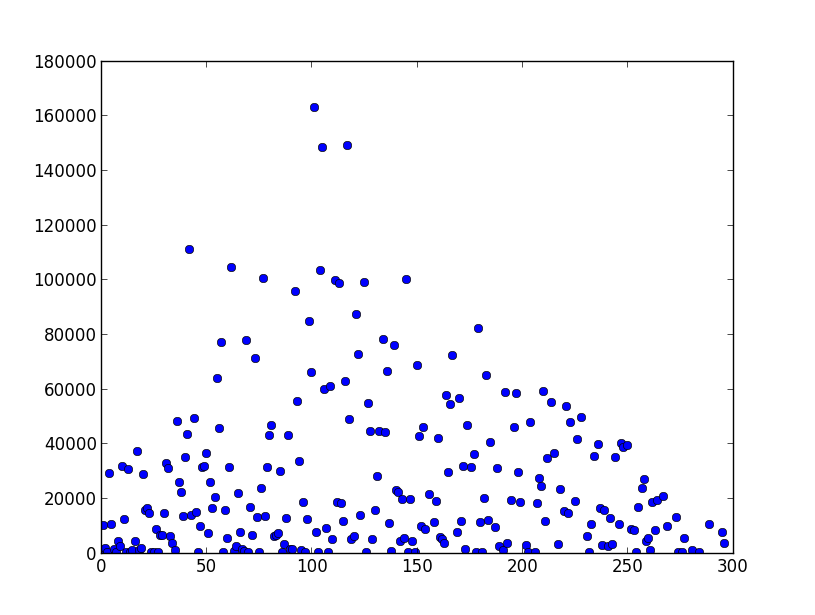
**Daemon Activities:**

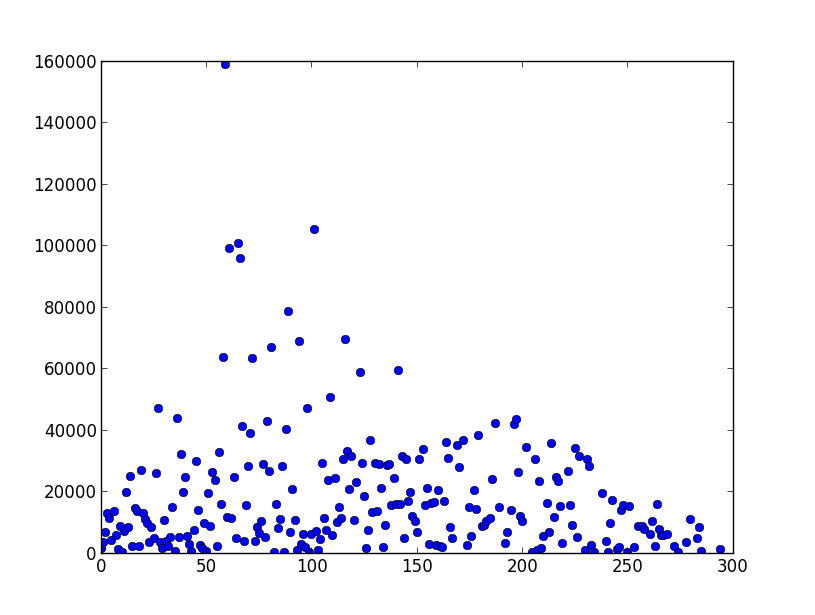
The daemon activity is a parallel process that runs at regular intervals in the background and does a global update of the pheromone in the system based on the shortest path that has been achieved that far. It keeps a track of the shortest path that has been found that far, and updates the pheromone on that path to further reinforce that path.

**Performance**

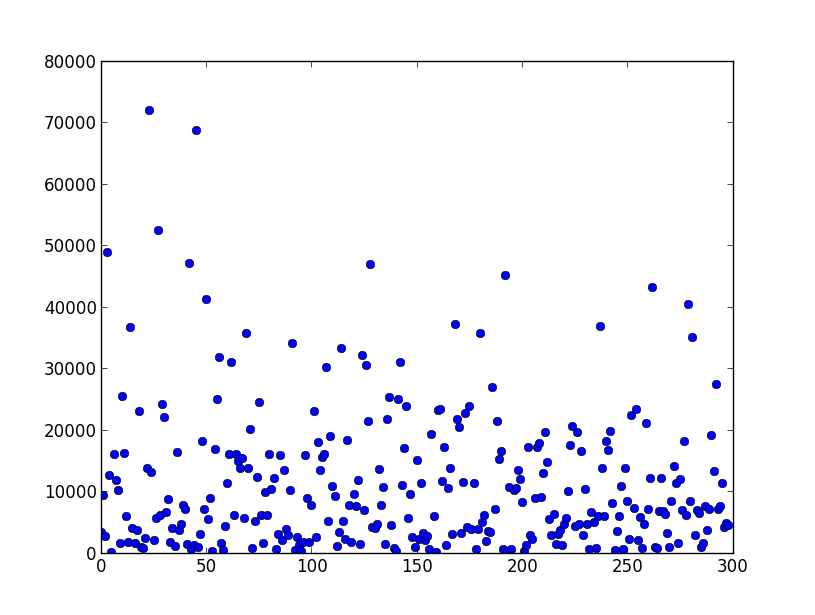
The performance of the program for the values of alpha as 1, 0.5 and 0 are shown below. The y axis represents the error, that is, the value of shortest path determined by that generation of ants minus the correct shortest path in the graph as determined by Dijkstra’s algorithm. The x axis represents the generations of ants.

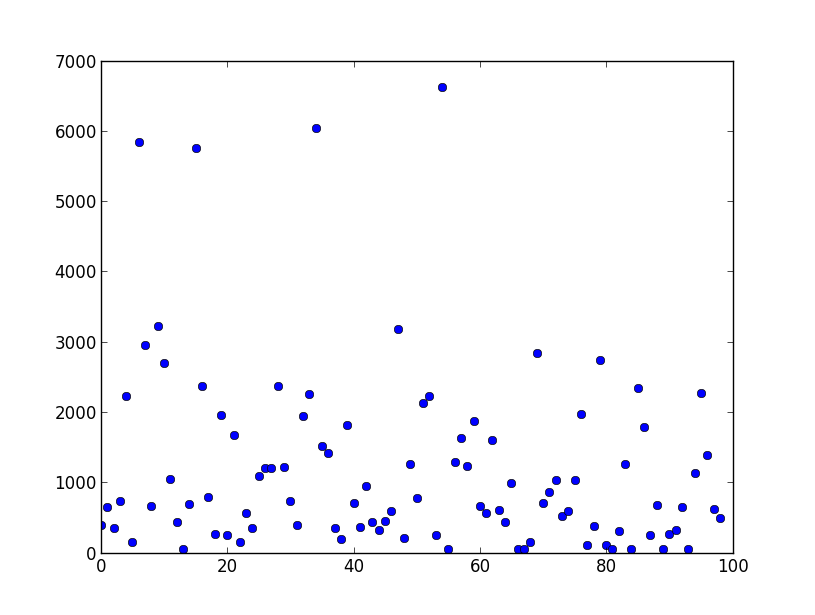
Alpha = 1:



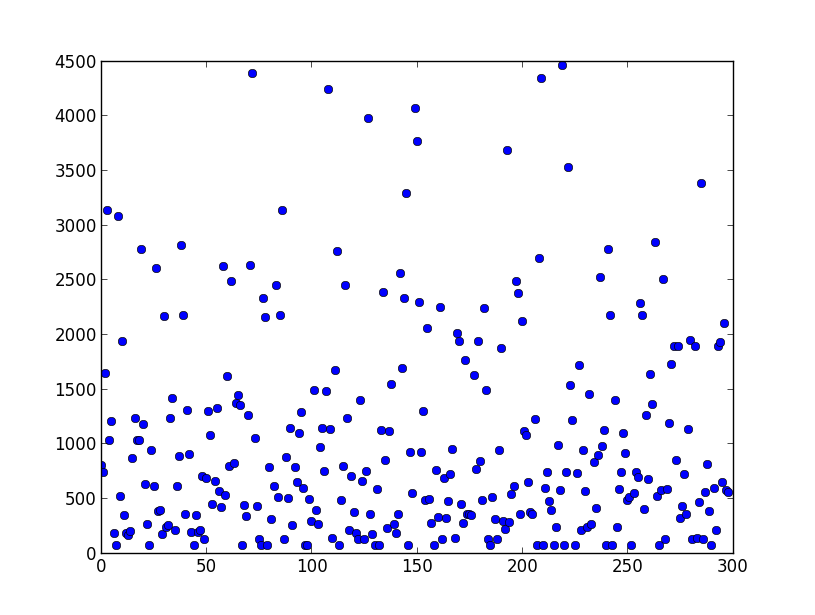


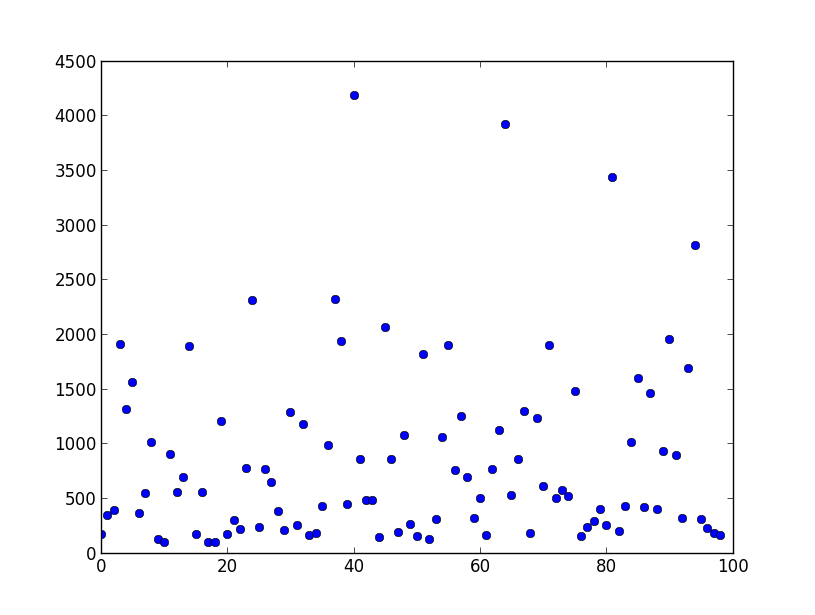
Alpha = 0.5





Alpha = 0





**Observation**

As we can see from the above results, for a static graph, as we move from exploitation towards exploration, the results worsen. This is owing to the fact that in exploration we disregard the information which we have collected so far. However, in many cases, exploration is required to move out of a local suboptimal solution.

Also, in the case where alpha = 1, we almost always tend to reach an error of zero. However, as alpha decreases, the chances of reaching a zero error also decrease. In the case of alpha = 0, getting a zero error is purely coincidental, as the ants do not follow any strategy in moving from one state to the other.

**Conclusion**

Exploration does not help in finding the shortest path in a static graph. However, it might be useful in dynamic graphs. It is the future scope of this project to analyze the performance of the algorithms for dynamic graphs.

**References**

<http://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms>  
<http://www.indigosim.com/tutorials/exploration/t1s1.htm>  
<http://natcomp.liacs.nl/SWI/papers/ant.colony.optimization/The.ant.colony.optimization.meta.heuristics.pdf>  
<http://www.scholarpedia.org/article/Ant_colony_optimization>  
<http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot>

**Aggie Code of Honor**

**“On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work.”**