Algorithm for GA

1. Load the puzzle
2. Find the possible numbers for each cell by checking the constraints
3. generate any random sudoku puzzles of given population size
4. calculate the fitness of the generated sudoku by

* no of constraints voilation in row, column and subgrid
* if the filled solution deosnt match the possible options \* 4
* given options not fulfilled.

5. Compare the fitness of all the population to to get the global fit parent

6. Ramdomly select the few parents to produce offspring

7. Produce the offspring by copy,fresh,crossover,swap

8. Goto step 4

Algorithm for ACO

1. Fill (as maximum as possible) numbers in puzzle cells
2. randomly by set of Ants without breaking rules of game
3. Use tabu lists to reduce options of numbers)
4. Select those Ants that produced solutions having lesser empty cells and remained on puzzle
5. Allow these Ants with better solutions to update Pheromone Table
6. Destroy all Ants
7. While iterations count less than preset value:-

o Produce new Ants

o Ants generate different solutions to puzzle using Pheromone Table as a guidance tool (longer unique sequence of numbers has higher probability to select again)

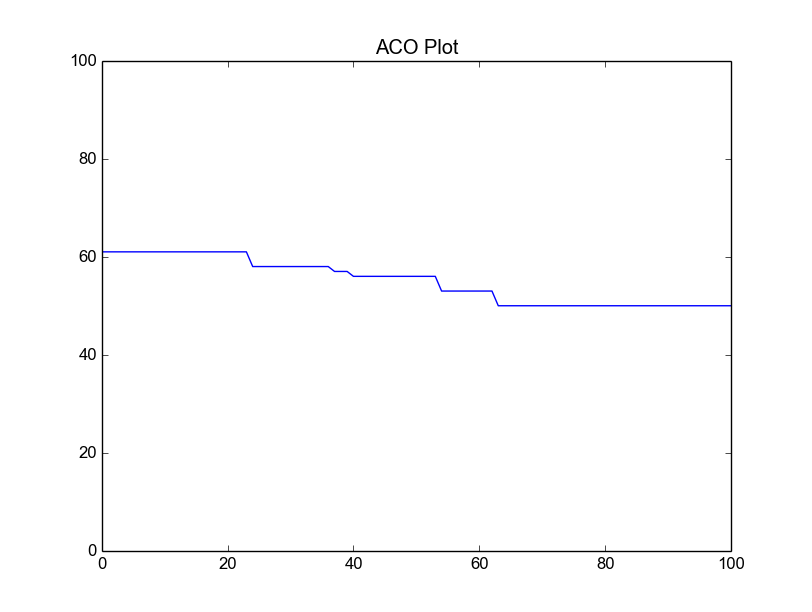
o Allow Ants to update selected positions in Pheromone Table

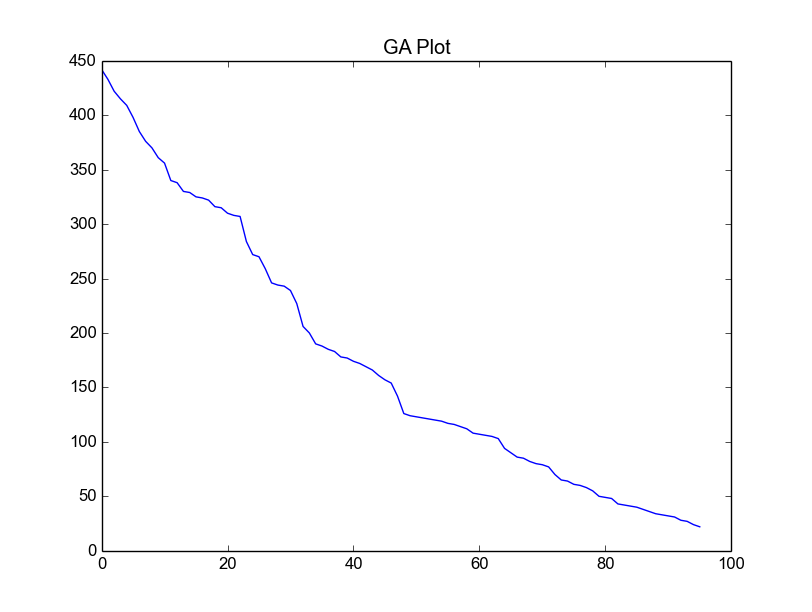
o Decay selected Pheromone Table positions

o If Pheromone value for any puzzle cell exceeds from threshold, reset it to zero

o Evaluate each Ant’s Solution

1. end while





Result:

Both the algorithm were not able to converge, but the performance of GA was better than ACO.