CHAPTER-4

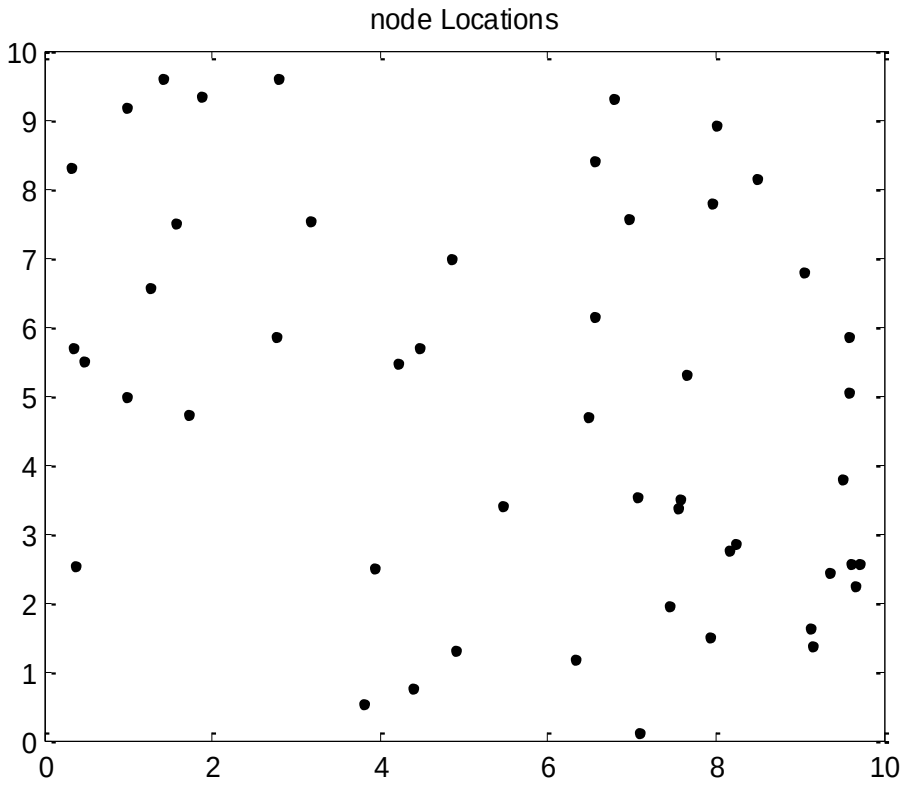
EXISTING METHODS - EXPERIMENTAL INVESTIGATIONS

The problem of node scheduling has been addressed considering a specific communication protocol and execution environment (TTP/C). The work has been largely considered for different types of heuristic algorithms for finding optimal node scheduling solution. In order to obtain optimised schedule considering the number of slots, ordering of slots and slot lengths, the execution delay is made as small as possible. The TTP bus feature, allocates a given slot in a considered TDMA round. Henceforth the process is made to be non-preemptive.

All the existing algorithms and proposed algorithm were tested with the number of nodes (ECUs) being 25 and 50. For this purpose algorithms were implemented using Matlab on an Intel Core i5 PC at 2.5GHZ with a total physical memory of 4 GB RAM.

4.1 Greedy Nearest Neighbor algorithm

The node scheduling process using Greedy Nearest Neighbor algorithm for safety critical embedded system has been investigated and simulation results are shown in figure 4.1. The distribution of node locations in an specified area is shown in fig 4.1(a). based on node locations generated the optimal scheduled Task allocation on nodes is obtained as shown in fig 4.1(b). the figure 4.1(c) illustrates the best solution (in seconds referred as units in the results) for task allocation even though the iterations increase above 50 the change in best solution is minimal. By first randomly selecting a node as reference, the task allocation is done, as the process is list based execution every node gets a task to execute provided its availability. The best tour is optimal and it reduces by 15% (approx.) from the initial allottment. The best Solution for task allocation on 50 nodes distributed randomly in the specified area is 63.36 units.



4.1 (a) Node locations

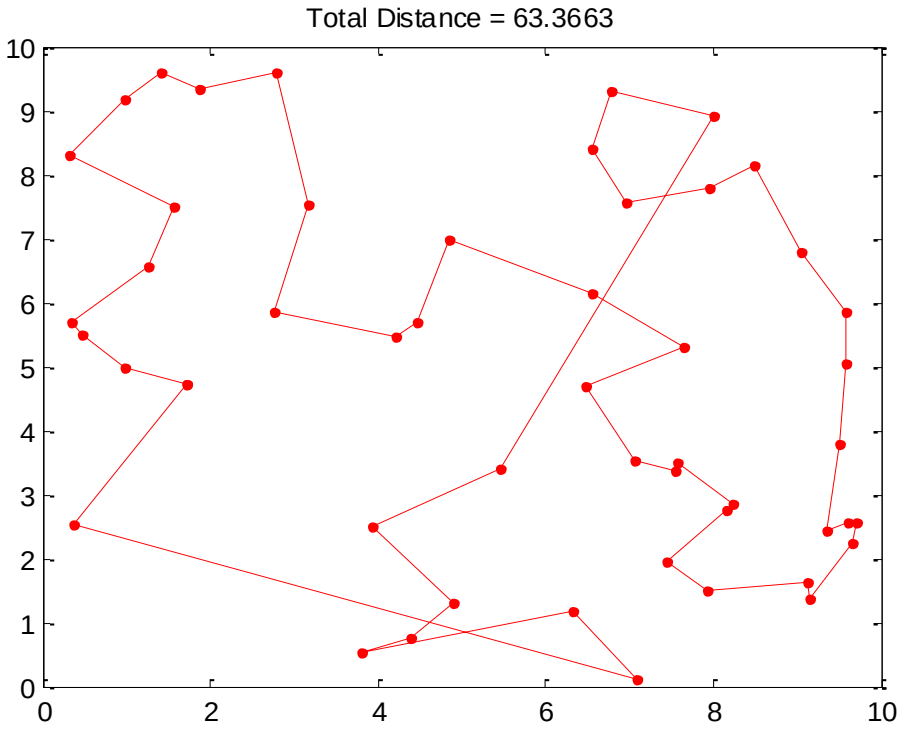


Figure 4.1 (b) Optimum scheduled Task allocation on nodes

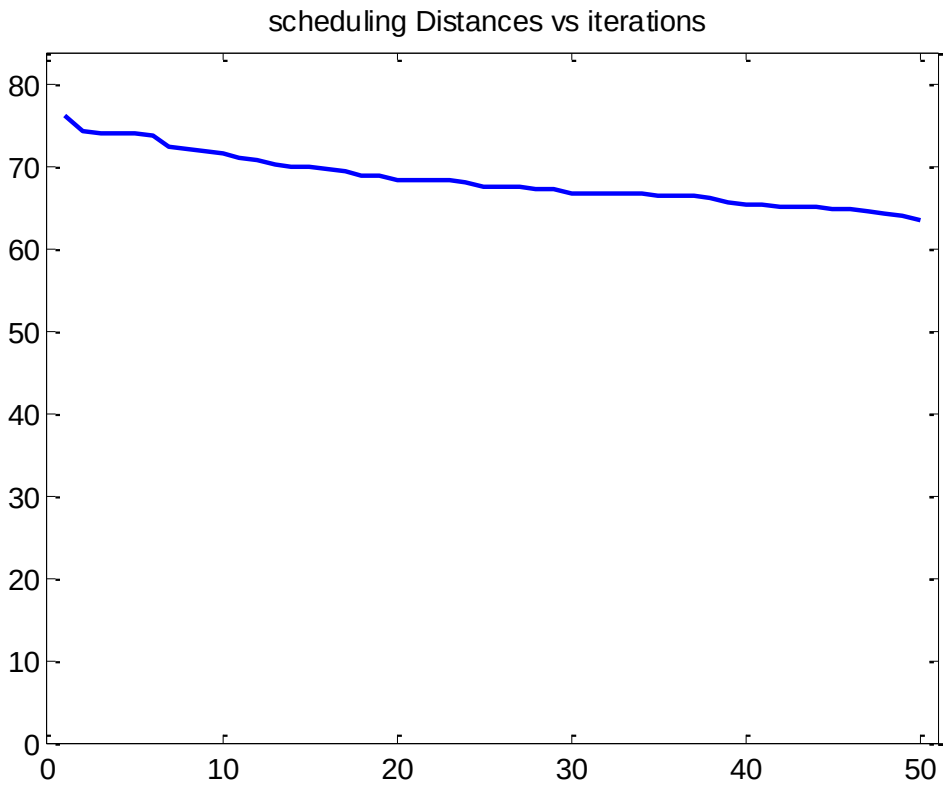


Figure 4.1. (c) Graph representing Best Solution

The experimental investigations have been also carried out for the nodes 25,30,35,40 and 45 and their node scheduling patterns are given below. In each experimentation for the respective number of nodes the process is conceded by arbitrarily selecting a node as reference, the task allocation is done, as the process is list based execution every node gets a task to execute provided its availability.

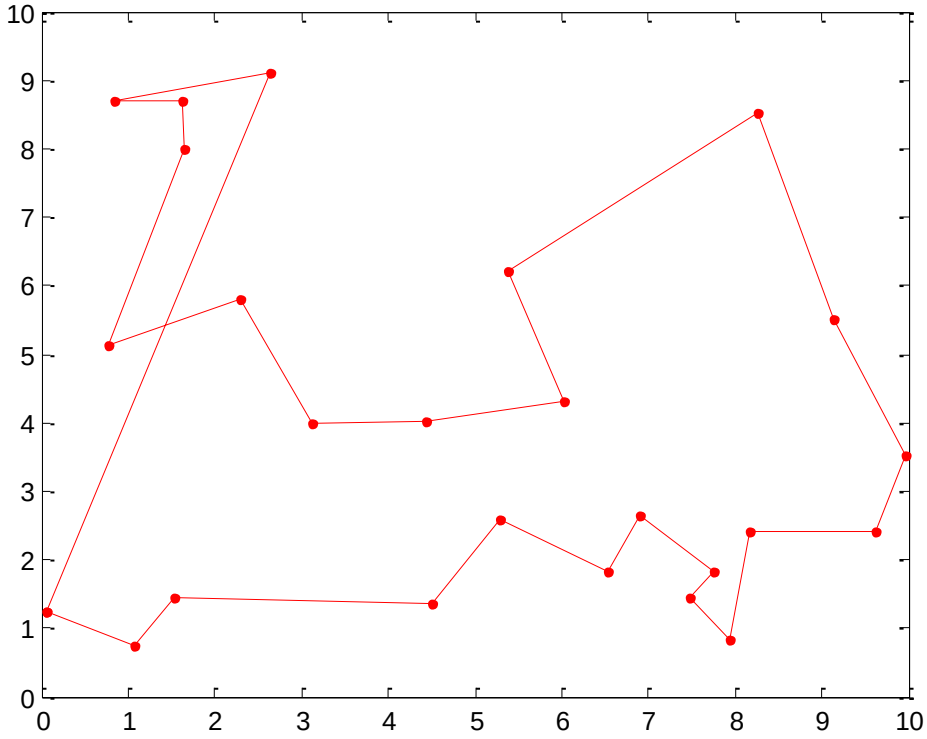


Figure 4.2. (a) Node Scheduling for 25 Nodes

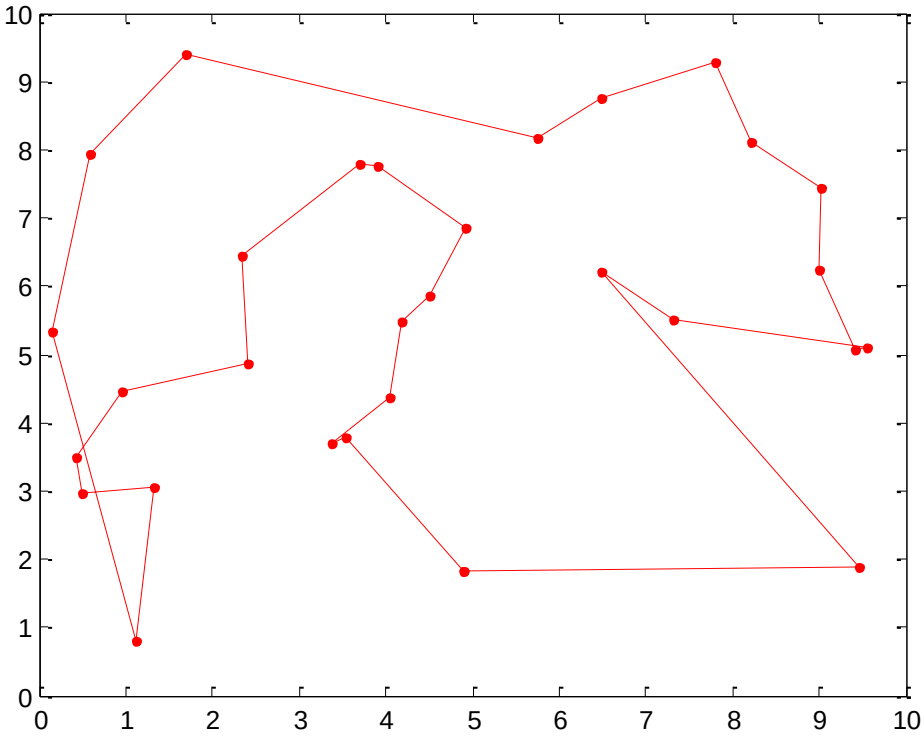


Figure 4.2. (b) Node Scheduling for 30 Nodes

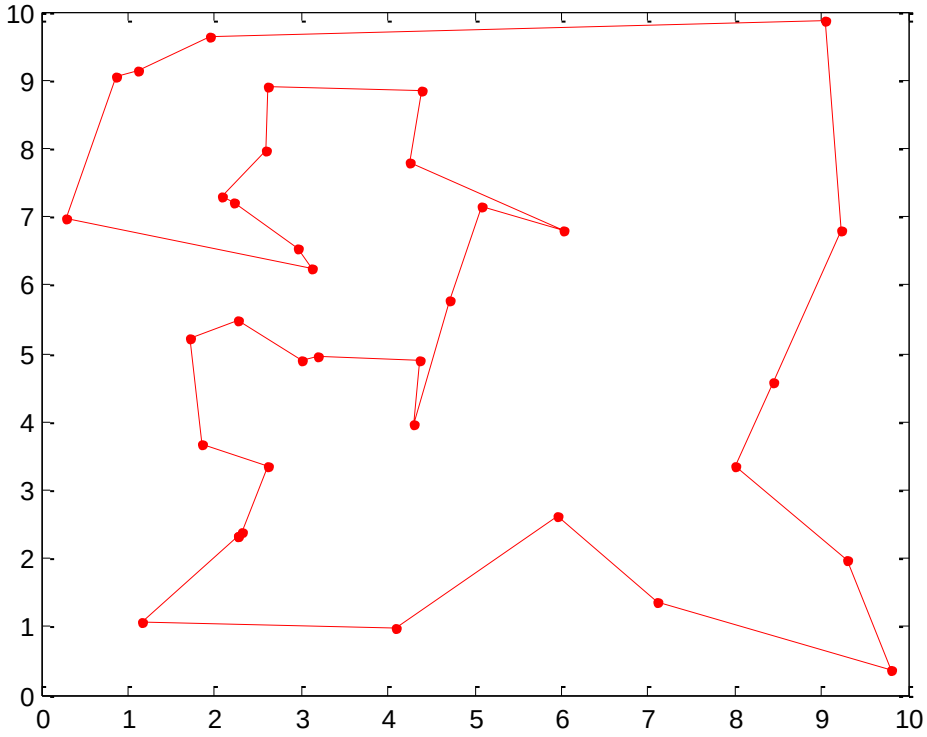


Figure 4.2. (c) Node Scheduling for 35 Nodes

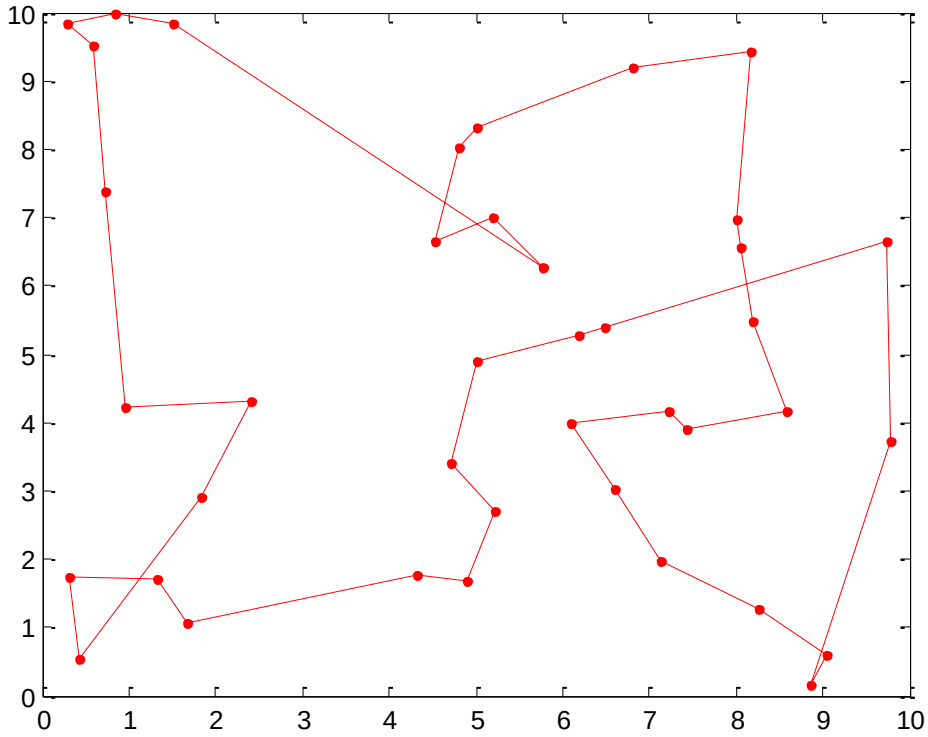


Figure 4.2. (d) Node Scheduling for 40 Nodes

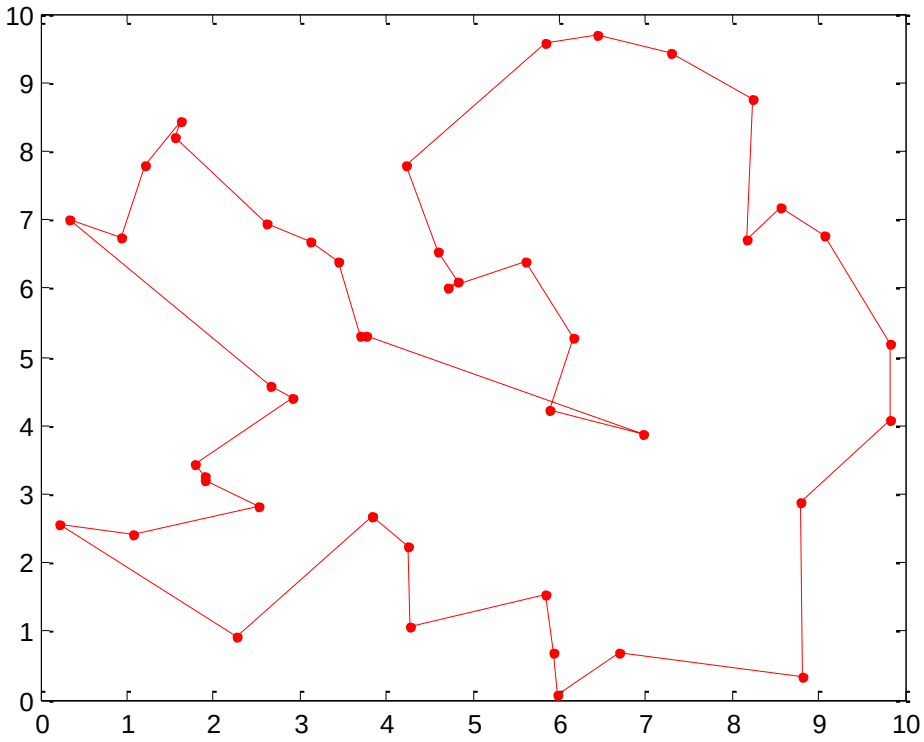


Figure 4.2. (e) Node Scheduling for 45 Nodes

4.2 Simulated Annealing

The Greedy Nearest Neighbor strategy presented earlier yields the best solution by gradually selecting the best possible node in terms of schedule length produced by its inherent property. Unlike the Greedy Nearest Neighbor approach the simulated annealing tries to get free from sticking to local optimum by randomly picking a new solution from the group of neighbors of the present solution. The obtained new solution is acceptable unless it is the better solution. However, with a certain probability, a poorer quality solution can also be accepted that is dependent upon the down turn of the cost function.

In the Simulated Annealing process the potential issues were dependent upon computation time, probability of acceptance, local optima, and cooling schedule. For better results the cooling rate should be slow, here it was kept at 0.97. Thus, the computation time increased progressively as the nodes increased. If the initial temperature is high the iterations have to be increased, so also the computation time increases. The probability of acceptance depends upon the temperature and its cooling schedule, initial temperature of 2000 value was considered and maximum iterations

were limited to 2000, and then found that computation time increased as the nodes were increased from 20 to 100.

If we add more constraints to the problem like disintegration of faulty nodes and erroneous data of electronic module, more local optima are created. With this problem the probability that solution may be struck in one of the local optima and chance of achieving the best results i.e, global optima is reduced. As we ran progressively the schedule pattern changed due to the above problem and it is shown in figure 4.2. The best solution of task allocation for 50 nodes obtained is 40003.141 units.

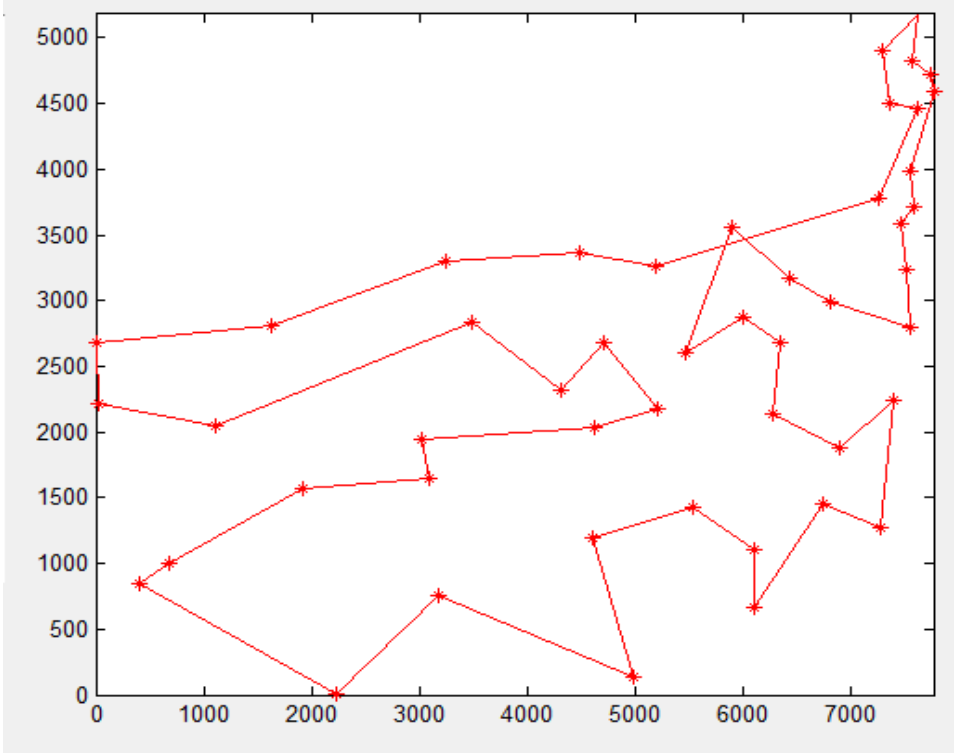


Figure 4.3 Optimum scheduled Task Allocation on Nodes

The experimental investigations have been also carried out for the nodes 25, 35 and 45 and their scheduling patterns are presented below. The parameters such as Temperature, Cooling rate and maximum iterations are maintained as same as applied for 50 nodes above.

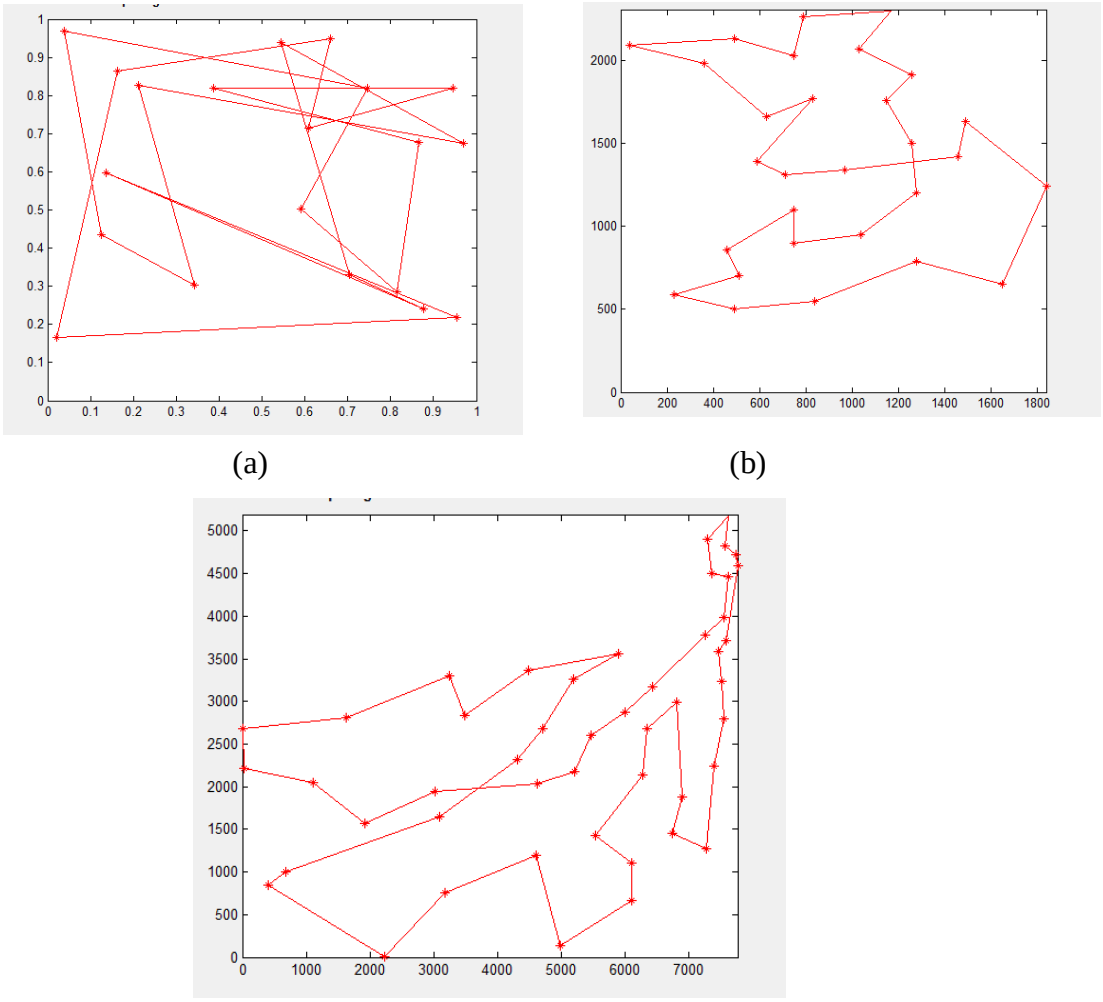


Figure 4.4. Node Scheduling for (a) 25 Nodes (b)35 Nodes (c) 45 Nodes

4.3 Ant Colony Algorithm

For the distributed computer embedded system having large number of nodes, scheduling on the system creates numerous local optima in simulated annealing approach. This becomes worse as best solution for cooling process is gradual and focusing on current solution. Since this requires large number of iterations, the scheduling process becomes slow to obtain best solution. Due to above explained constraints it is evident that the performance of SA is limited for Task allocation problems. Hence experimental investigations based on ACO have been carried in this research work with TSP.

Here the parameters are set to the following values: α=1(effect of ants' sight), β= 4 (trace's effect), e =.15 (evaporation coefficient). The maximum iteration is set

100 and m (number of ants) is 10. The task allocation in ACO is based on the concept of how the simulated ants are expected to select the next node depending on the amount of the pheromone in a trail and the distance to the next node. The figure 4.5 presents the results obtained for this method and the best solution of task allocation is 170.3367 units.

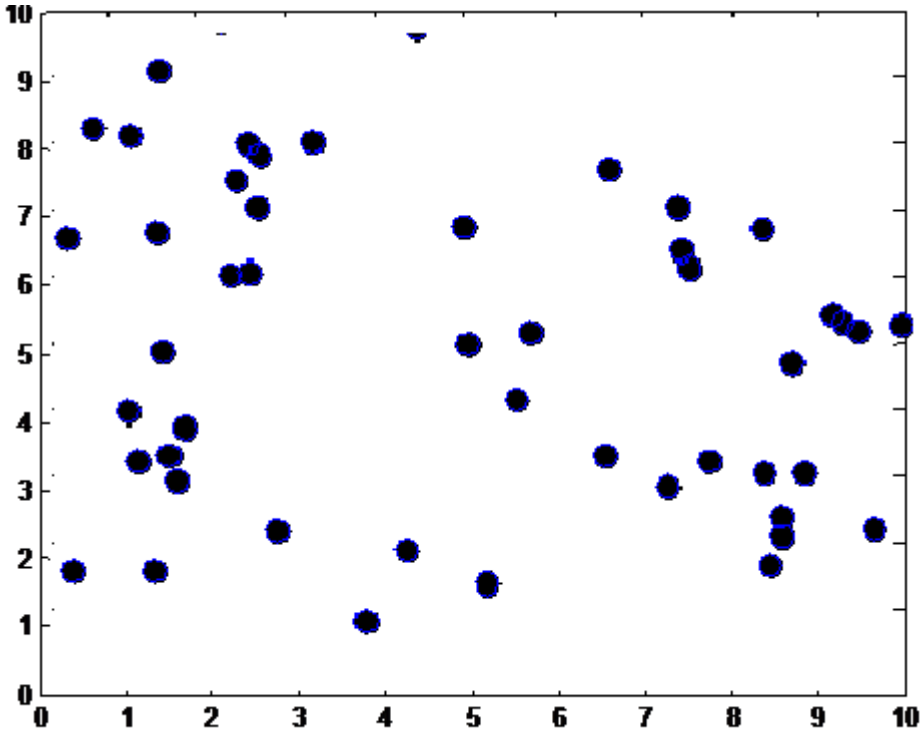


Figure 4.5 (a) Node locations

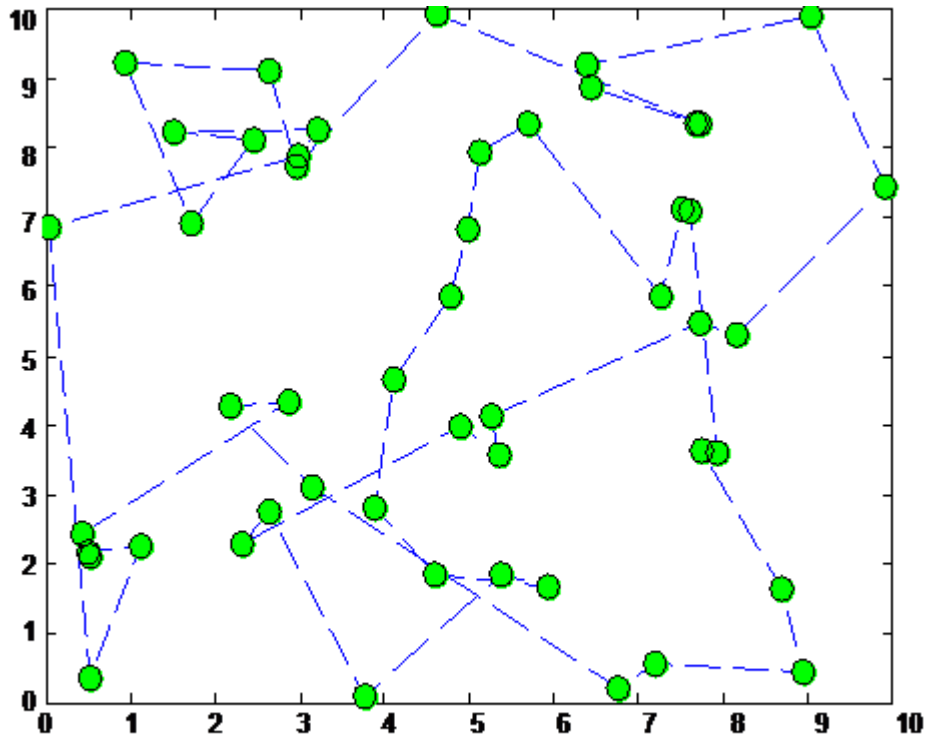


Figure 4.5 (b) optimum scheduled Task Allocation on Nodes

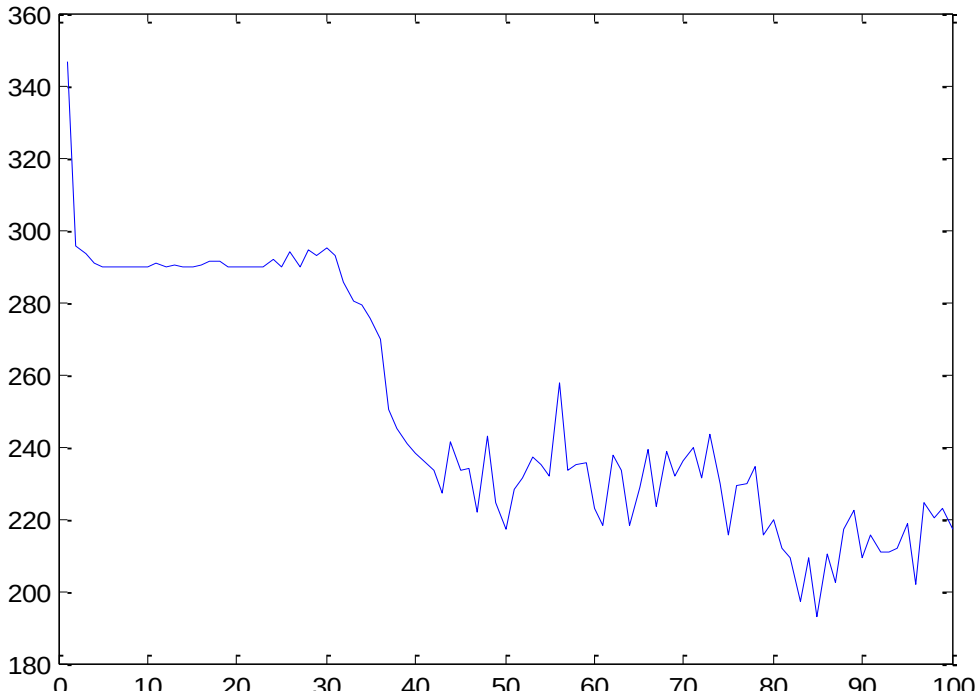


Figure 4.5 (c) Graph representing Best Solution

The experimental investigations have been also carried out for the nodes 25,30,35,40 and 45 and their corresponding scheduling patterns are depicted in below figures. The task allocation on nodes in ACO is based on the idea of how the simulated ants are anticipated to select the next node depending on the amount of the pheromone in a trail and the distance to the next node.

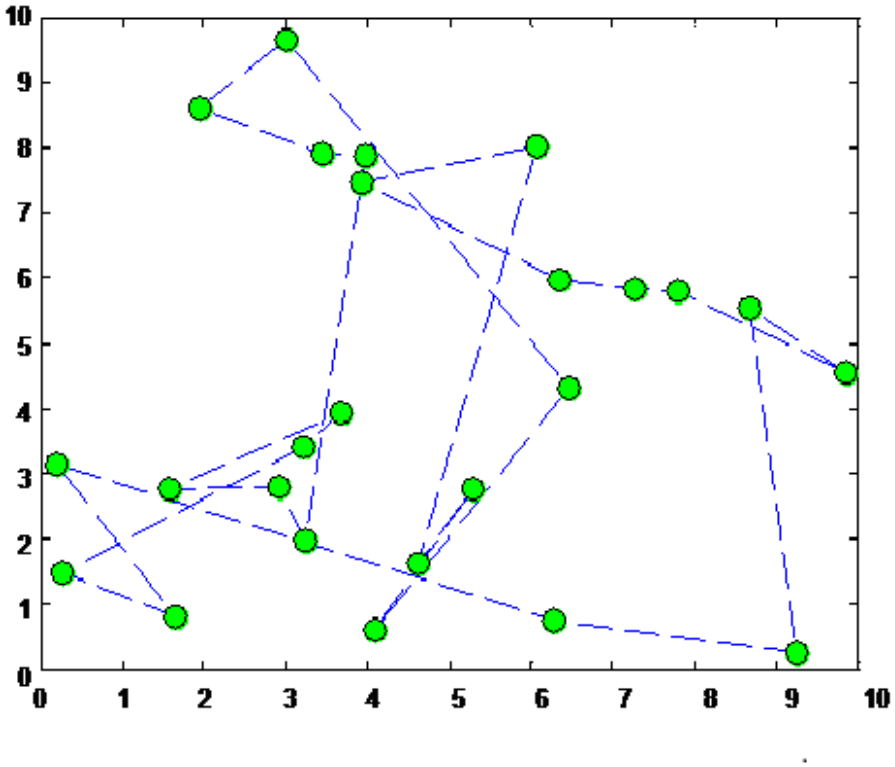


Figure 4.6. (a) Node Scheduling for 25 Nodes

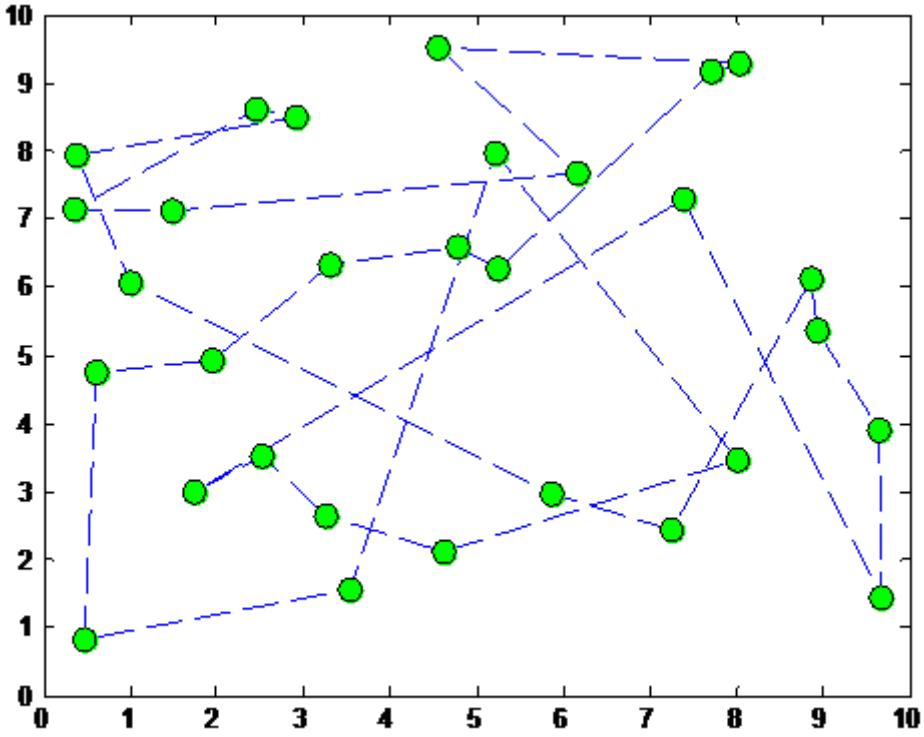


Figure 4.6. (b) Node Scheduling for 30 Nodes

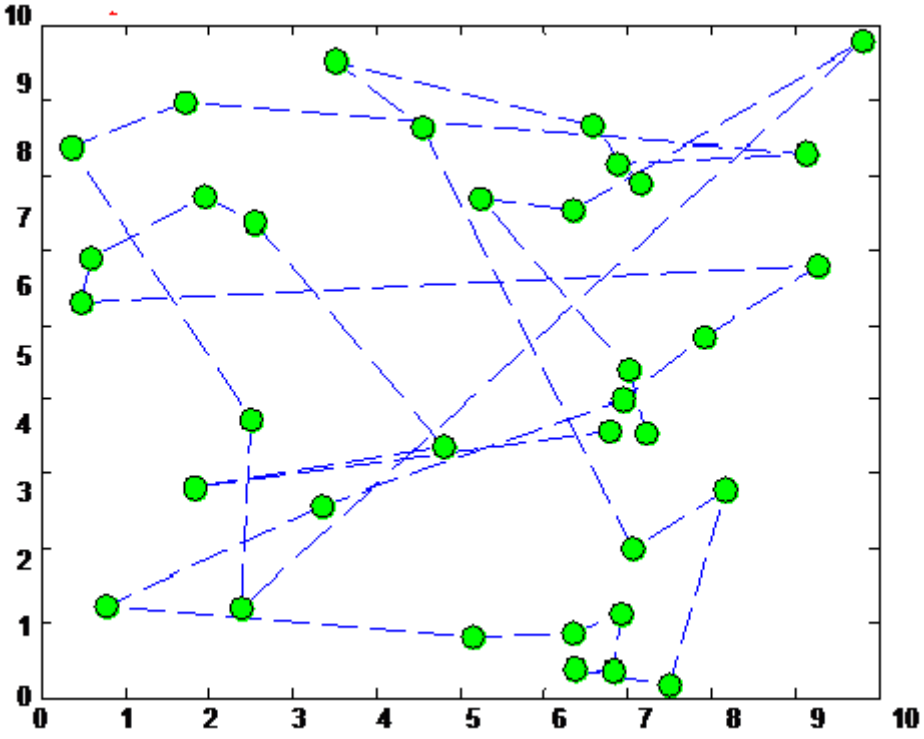


Figure 4.6. (c) Node Scheduling for 35 Nodes

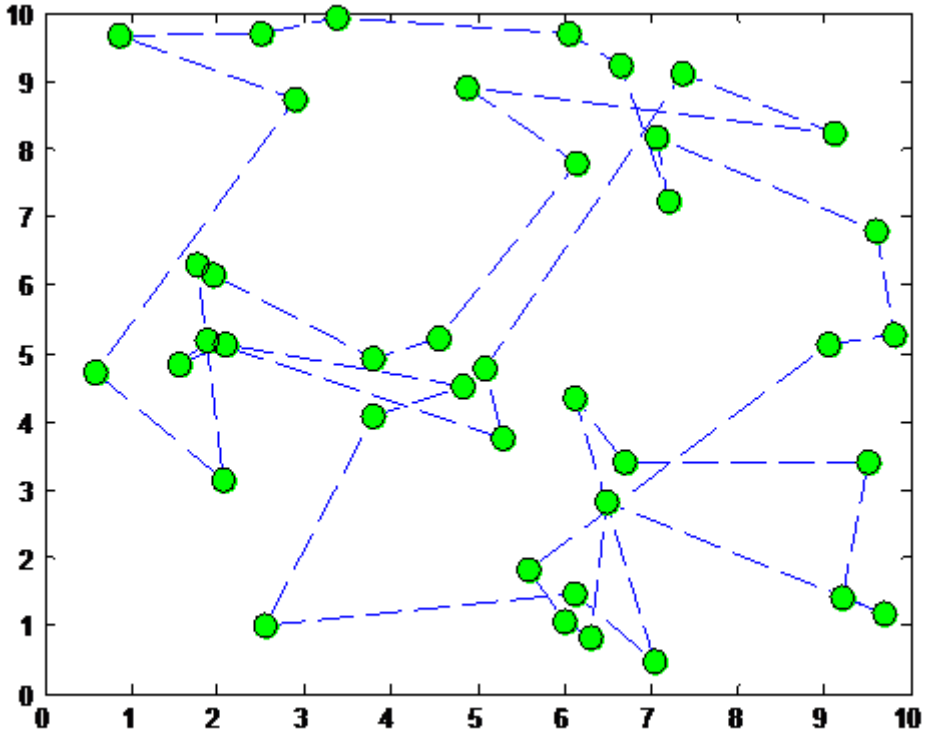


Figure 4.6. (d) Node Scheduling for 40 Nodes

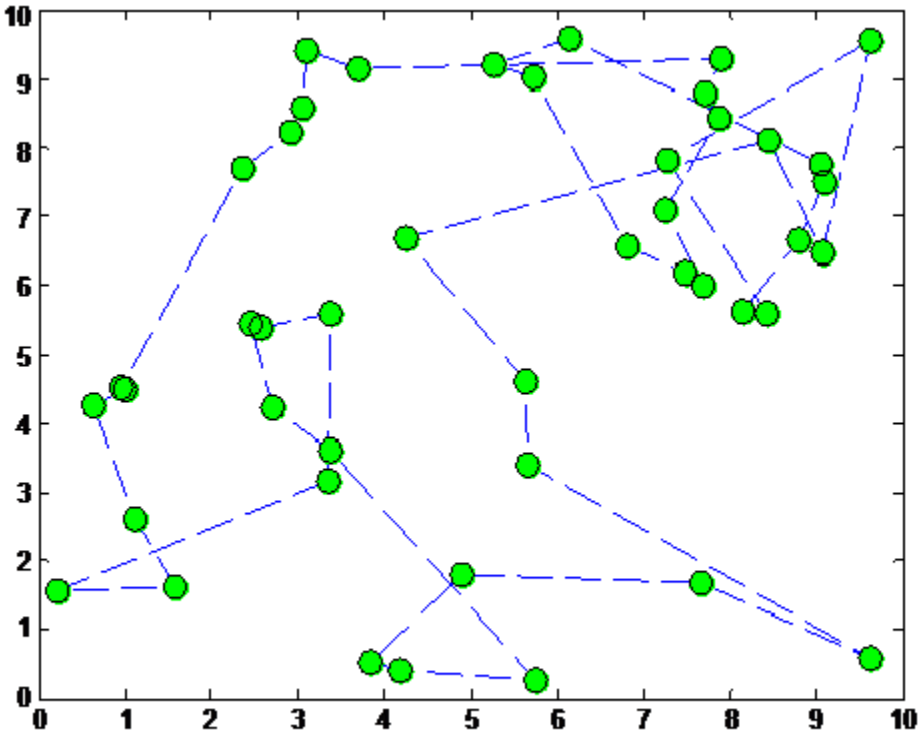


Figure 4.6. (e) Node Scheduling for 45 Nodes

4.4 Genetic Algorithm

As the number of ants in ACO are increased subsequently the running time will increase, further results may not be good. So the number of ants has to fix to a lower level. On the other hand, lower the number of ants will not yield a good solution even though better running times are achieved. Unlike ACO, Genetic Algorithm produces a new generation of solution better than the previous solution and this process continues until a stopping condition is met. For real time embedded applications, it is required to recognize for limited resources utility the algorithm has to run several times to yield optimal results, for which ACO may not be better choice. In this work in order to solve the coding the inputs of main concern are population size and matrix of point to point distances/costs. The results obtained show that approximately 15 - 60% scheduling time is reduced when compared to previous algorithms. When large population size is considered the Genetic Algorithm run slows down, whereas, a small size leads to traveling around a little search space. This is due to the crossover and mutation operators inherent with Genetic Algorithm.

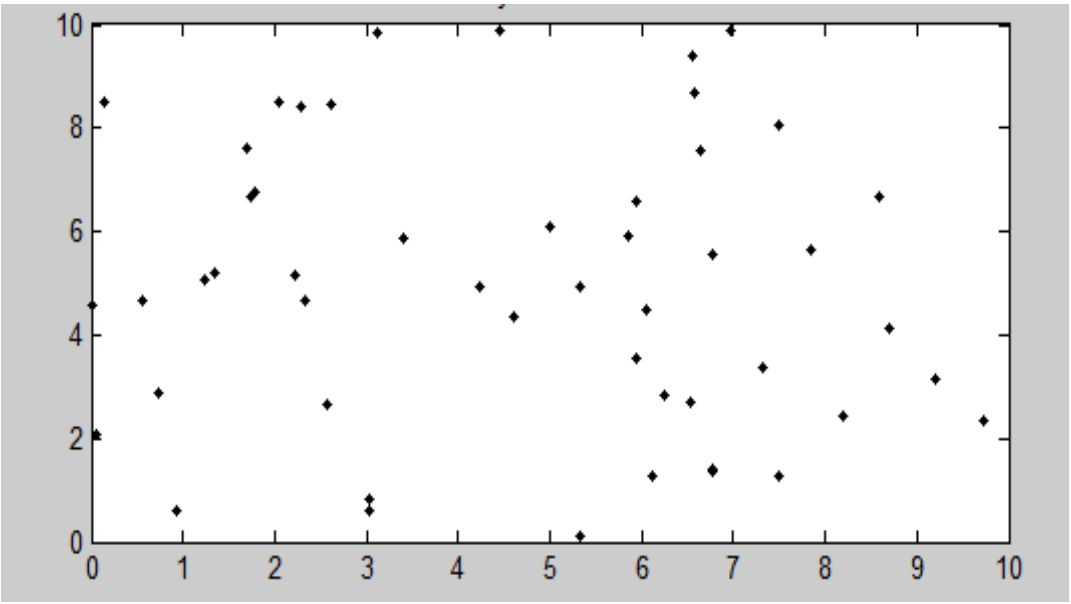


Figure 4.7 (a) Node locations

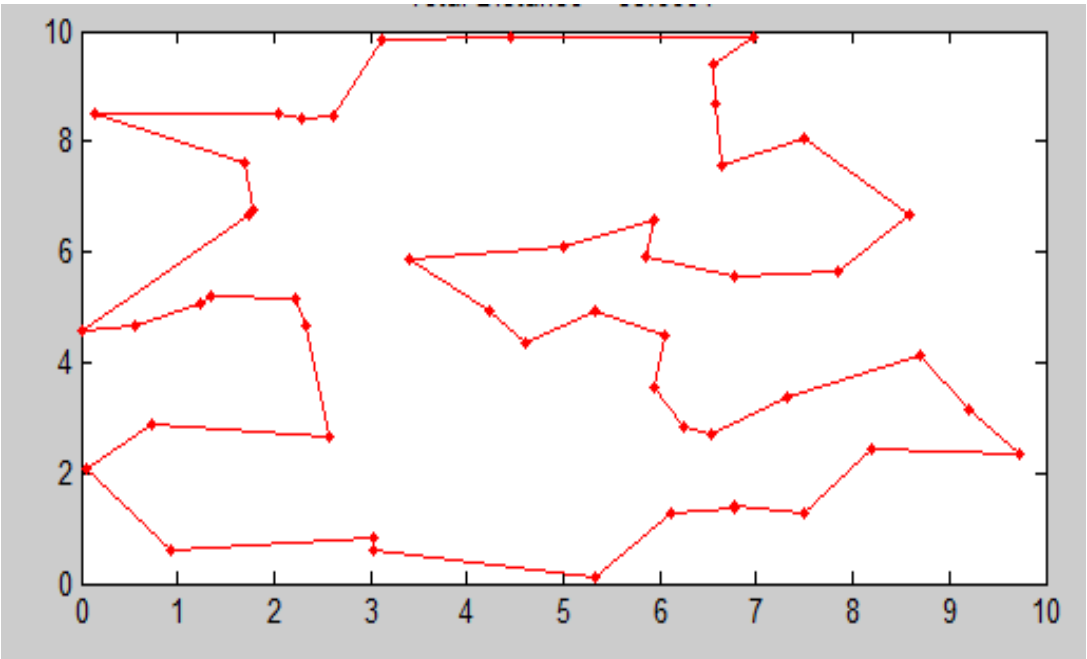


Figure 4.7 (b) Optimum scheduled Task Allocation on Nodes

The simulation results are shown in figure 4.4 and the best solution of task allocation for 50 nodes is 55.66 units.

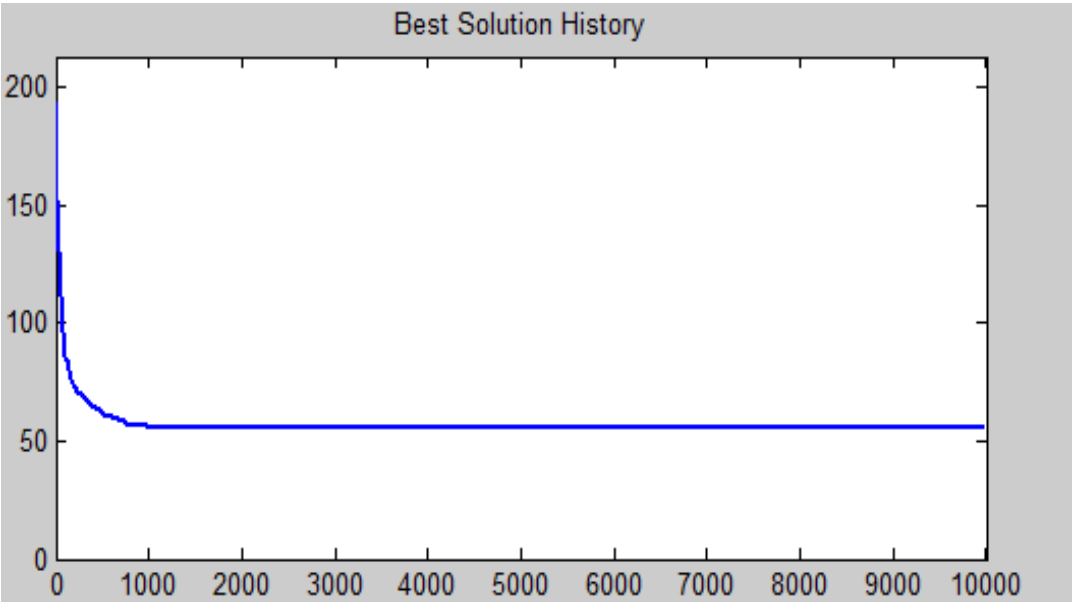


Figure 4.7 (c) Graph representing Best Solution

The experimental investigations have been also carried out for the nodes 25,30,35,40 and 45. While implementing GA, in order to unravel the coding the attributes of foremost concern in this research were population size and matrix of point to point distances/costs.

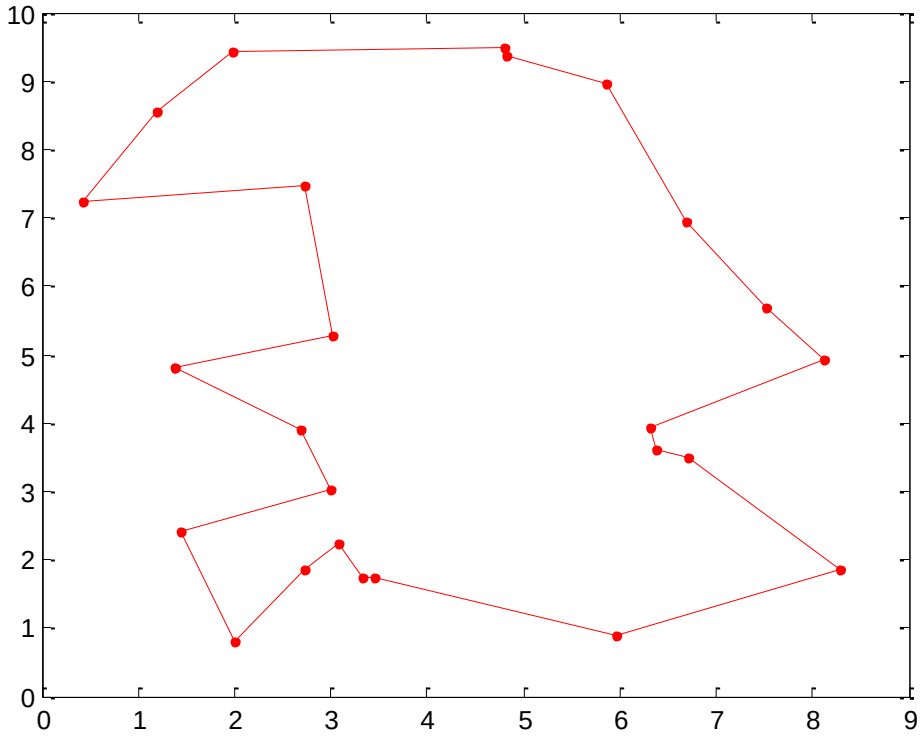


Figure 4.8. (a) Node Scheduling for 25 Nodes

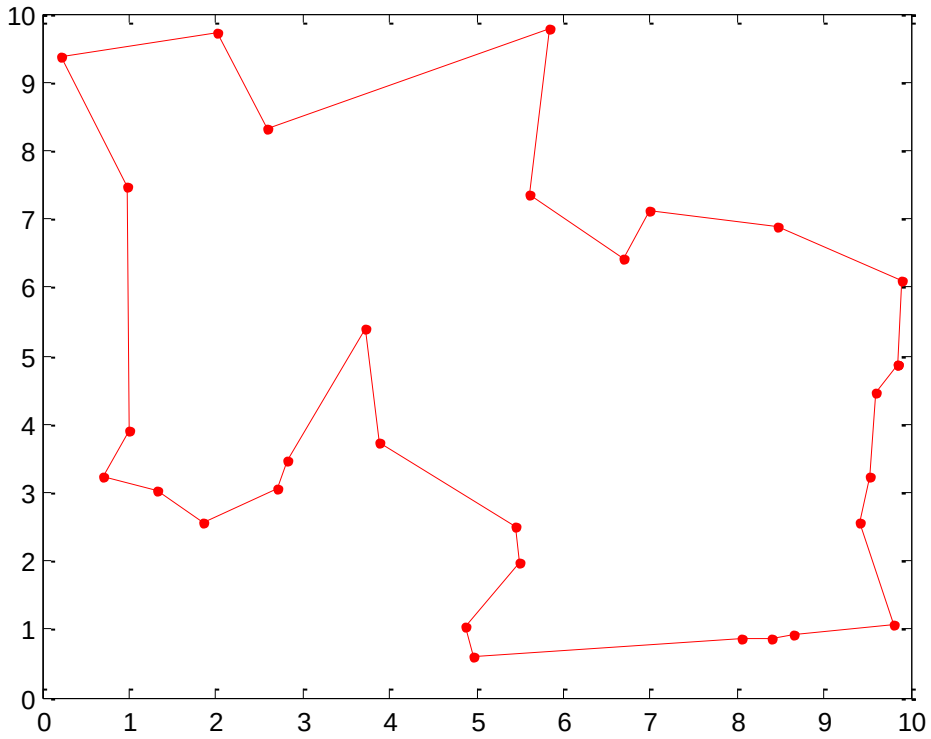


Figure 4.8. (b) Node Scheduling for 30 Nodes

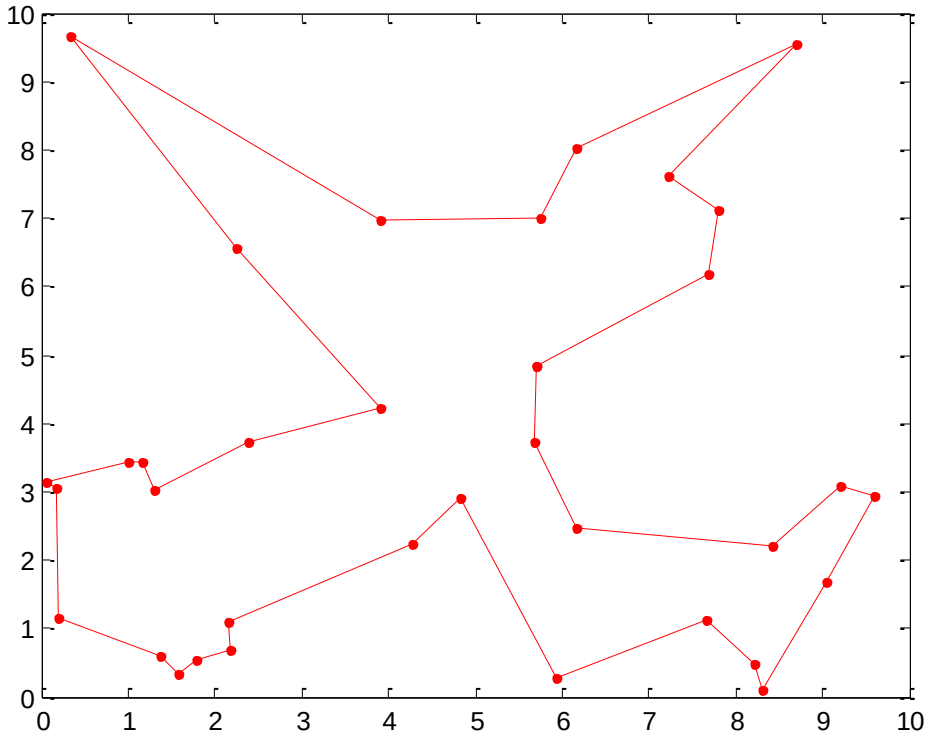


Figure 4.8. (c) Node Scheduling for 35 Nodes

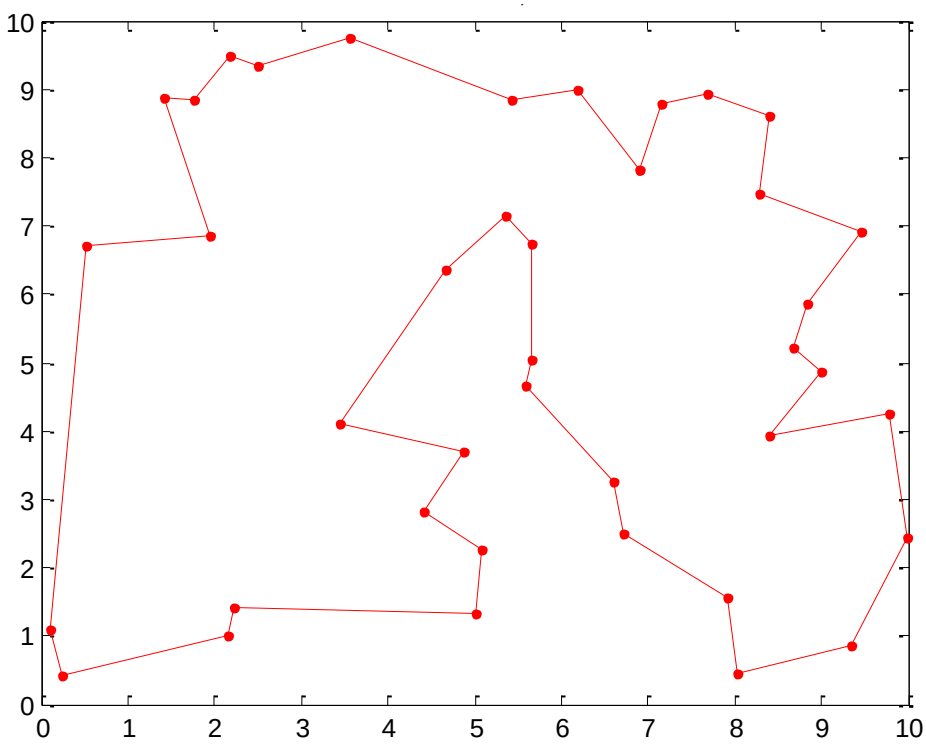


Figure 4.8. (d) Node Scheduling for 40 Nodes

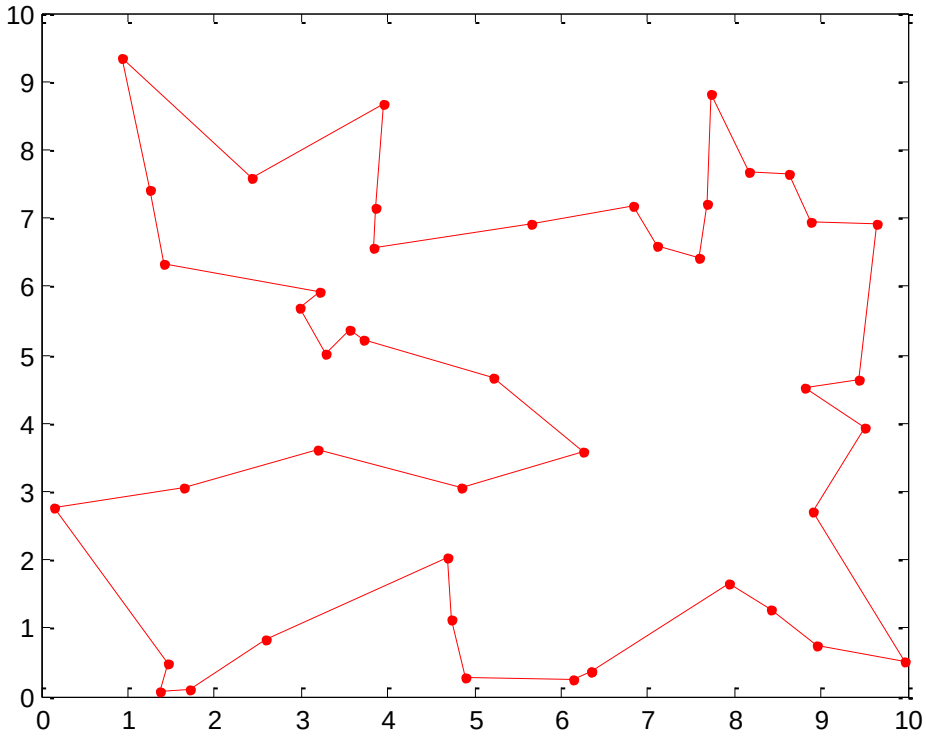


Figure 4.8. (e) Node Scheduling for 45 Nodes