**­ANT COLONY OPTIMIZATION TO SOLVE**

**TRAVELLING SALESMAN PROBLEM**

A BACHELOR’S MINI PROJECT

*Submitted in partial fulfillment*

*of the requirements for the completion of the Vth semester of the*

**UNDERGRADUATE PROGRAM**

*in*

**INFORMATION TECHNOLOGY**

Submitted by

Aaquib Khwaja Mohammad Talha Khan Kunwar Krishna Ayush

IIT2011127 IIT2011073 IIT2011135

*Under the Guidance of*

**Dr. K. P. Singh**

Associate Professor

IIIT - ALLAHABAD

****

**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,**

**ALLAHABAD – 211012**

**CANDIDATES’ DECLARATION**

We hereby declare that the work presented in this project report entitled **“Ant Colony Optimization Algorithm to solve the Travelling Salesman Problem”**, being submitted as a part of Mid Semester Project Evaluation at **Indian Institute Of Information Technology, Allahabad**, is an authenticated record of our original work under the guidance and supervision of **Dr. K. P. Singh.**

**Date**: October 26th, 2013

**Place**: Allahabad

Aaquib Khwaja Mohammad Talha Khan Kunwar Krishna Ayush

IIT2011127 IIT2011073 IIT2011135

**CERTIFICATE**

This is to certify that the statement made by the candidates’ is correct to the best of my knowledge and belief. The project entitled **“Ant Colony Optimization Algorithm to solve the Travelling Salesman Problem”** is a record of candidates’ work carried out by them under my guidance and supervision.

Dr. K. P. Singh Date:

Associate Professor,

IIIT Allahabad.

**ACKNOWLEDGEMENT**

We wish to express our sincere gratitude to **Dr. K. P. Singh**, for providing us an opportunity to do our project work on **“Ant Colony Optimization Algorithm to solve the Travelling Salesman Problem”**. His keen, vital encouragement, superb guidance, and constant support are the motive force behind this project work. We are very thankful to all the technical and non-technical staffs of the college for their assistance and co-operation. Last but not the least we wish to avail ourselves of this opportunity, express a sense of gratitude and love to our friends and our beloved parents for their manual support, strength, help and for everything.

**CONTENTS**

1. Abstract 6
2. Introduction 7
3. Problem Specification And Scope 8
4. Literature Survey 10
   1. Travelling Salesman Problem 10
   2. Ant Colony Optimization 11
   3. Network Routing 13
   4. Roulette Wheel Selection 15
5. Proposed Approach 16

5.1 Travelling Salesman Problem 16

5.2 Network Routing Problem 20

1. Results 26
2. Shortcomings 31
3. Future Improvements 32
4. Work Done 33
5. Tools Used 35
6. Activity Time Chart 36
7. Appendix 37
8. References 43
9. Comments and Suggestions 44

**ABSTRACT**

In this project, the Ant Colony Optimization Technique has been applied to solve the Travelling Salesman Problem, a NP problem which has a great significance in real life.

Travelling Salesman Problem – Aim : Given a finite number of cities along with the cost of travel between each pair of them, find the cheapest way of visiting all the cities and returning to your starting point.

Then a comparative study of the time complexity has been done between the implementation of Travelling Salesman Problem using the Branch and Bound technique and the Ant Colony Optimization technique.

Further, Ant Colony Optimization algorithm has been applied to a more real life problem – Network Routing Problem.

Network Routing Problem : Aim : For different network models with different number of nodes and structure to find the shortest path with optimum throughput.

Then a time complexity analysis of the implementation of Network Routing Problem has been done.

**INTRODUCTION**

Ant algorithms are a recently developed, population-based approach which has been successfully applied to several *NP*-hard combinatorial optimization problems such as Traveling Salesman Problem (TSP), Job-shop Scheduling Problem (JSP), Vehicle Routing Problem (VRP), Quadratic Assignment Problem (QAP), etc.

ACO algorithm models the behavior of real ant colonies in establishing the shortest path between food sources and nests. Ants can communicate with one another through chemicals called pheromones in their immediate environment. The ants release pheromone on the ground while walking from their nest to food and then go back to the nest. The ants move according to the amount of pheromones, the richer the pheromone trail on a path is, the more likely it would be followed by other ants. So a shorter path has a higher amount of pheromone in probability, ants will tend to choose a shorter path. Through this mechanism, ants will eventually find the shortest path.[1]

Travelling Salesman Problem –

The traveling salesman problem (TSP), is this: given a finite number of cities along with the cost of travel between each pair of them, find the cheapest way of visiting all the cities and returning to your starting point. (Here, we consider just the symmetric TSP, where traveling from city X to city Y costs the same as traveling from Y to X.) In other words, the data consist of integer weights assigned to the edges of a finite complete graph; the objective is to find a hamiltonian cycle (that is, a cycle passing through all the vertices) of the minimum total weight.[3]

Solutions to the Travelling Salesman Problem –

1. Brute Force

Make a list of all possible Hamilton circuits

Calculate the weight of each Hamilton circuit by adding up the weights of its edges.

Choose the Hamilton circuit with the smallest total weight.

The Brute-Force Algorithm is guaranteed to find a solution, but, the algorithm is inefficient, since it has to look at all (N - 1)! Hamilton circuits, and this can take a long time.

1. Branch and Bound

The branch and bound algorithm firstly seeks a solution of the assignment problem.

Initially a minimum cost is set and at each level after updating the cost of moving to next node, if the current cost becomes greater than the minimum cost till now the whole path in the search space is pruned.

1. Our approach – Ant Colony Optimization

Implementing the Travelling Salesman Problem using ACO meta heuristics.

In this approach a colony of (artificial)ants are used to traverse the graph and find the shortest possible route to travel all the cities as described above in the introduction.

**PROBLEM SPECIFICATION**

Travelling Salesman Problem using Ant Colony Optimization.

For a given weighted graph G = (N, E) and M number of Ants, where N is the set of n cities and E is the set of edges fully connecting all cities. Each edge (i, j)ϵ Eis assigned a cost dij, which is the distance between cities i and j.

It is an NP-hard problem (A problem is NP-hard if solving it in polynomial time would make it possible to solve all problems in class NP in polynomial time).

Approaches to solve such problems:

1. Devising algorithms for finding exact solutions (they will work reasonably fast only for small problem sizes).
2. Devising "suboptimal" or heuristic algorithms, i.e., algorithms that deliver either seemingly or probably good solutions, but which could not be proved to be optimal.

Apply Ant Colony Optimization algorithm to solve the above problem.

At every node in the graph, the decision making probability is based on the amount of pheromone deposited on the outgoing edges at that node. The edge with more amount of pheromone will have more probability of being selected.  
Amount of pheromone on an edge is increased every time an ant passes through it, and keeps on evaporating as time passes.

SCOPE

Ant colony optimization algorithms have been applied to many combinatorial optimization problems, ranging from Assignment problems such as Graph coloring problem and University course Timetabling problem to Routing problems such as Travelling Salesman problem and Network routing problem.

The essence of the Traveling Salesman Problem is evident within many practical applications in real life. From a mail delivery person trying to figure out the most optimal route that will cover all of his/her daily stops, to a network architect trying to design the most efficient ring topology that will connect hundreds of computers.

The fundamental goal is to find the optimal tour. That is, to determine an order in which each location should be visited such that each location is visited only once, and the total distance traveled, or cost incurred, is minimal.

A real life example of the Ant Colony Optimization algorithm can be seen as the Network Routing Problem.

Network Routing Problem

Network routing is the process of selecting paths in a network along which to send network traffic. [13] Communication networks can be classified as either circuit-switched or packet-switched. The example of circuit switched network is the telephone network in which the physical circuit is set up at the communication start and remains the same for the communication duration. Unlike them, in packet-switched networks, also called data networks, each data packet can follow a different route and no fixed physical circuits are established. The example of data networks are LAN and the Internet. In tour project we will focus on data networks.

**LITERATURE SURVEY**

Travelling Salesman Problem –

The TSP is a NP-hard combinatorial optimization problem which has attracted a very significant amount of research by Johnson & McGeoch, 1997; Lawler et al., 1985; and Reinelt, 1994.

The TSP has played a central role in Ant Colony Optimization, because it was the application problem chosen when proposing the first ACO algorithm called Ant System (Dorigo, 1992; Dorigo, Maniezzo, & Colorni, 1991b, 1996) and it was used as a test problem for almost all ACO algorithms proposed later.[1]

The Travelling Salesman Problem(TSP) is stated as, given a complete graph, G, with a set of vertices, V, a set of edges, E, and a cost, cij, associated with each edge in E. The value cij is the cost incurred when traversing from vertex i ∈ V to vertex j ∈ V. Given this information, a solution to the TSP must return the cheapest Hamiltonian cycle of G.

A Hamiltonian cycle is a cycle that visits each node in a graph exactly once.

NP –Hard Problem -

NPis the set of decision problems with the following property: If the answer is YES, then there is a proofof this fact that can be checked in polynomial time. Intuitively, NP is the set of decision problems where we can verify a YES answer quickly if we have the solution in front of us.

A problem π is NP-hardif a polynomial-time algorithm for π would imply a polynomial-time algorithm for every problemin NP. In other words:

|  |
| --- |
| **π is NP-hard ⬄ If π can be solved in polynomial time, then P = NP** |

Combinatorial Optimization –

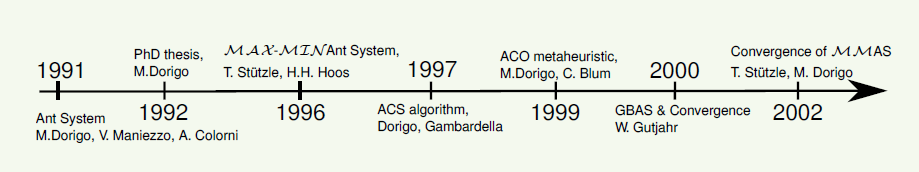
Combinatorial optimization is a topic that consists of finding an optimal object from a finite set of objects. In many such problems, exhaustive search is not feasible. It operates on the domain of those optimization problems, in which the set of feasible solutions is discrete, and in which the goal is to find the best solution. Some common problems involving combinatorial optimization are the traveling salesman problem ("TSP") and the minimum spanning tree problem ("MST").

Swarm Intelligence –

Swarm intelligence is a computational intelligence technique to solve complex real-world problems. It involves the study of collective behavior of individuals in a population who interact locally with one another and with their environment in a decentralized control system.  
Example – Ant Colony Optimization

Ant Colony Optimization –

Was developed by Marco Dorigo (Italy) in his PhD thesis in 1992.



Technique for solving problems which can be expressed as finding good paths through graphs.

Each ant tries to find a route between its nest and a food source.

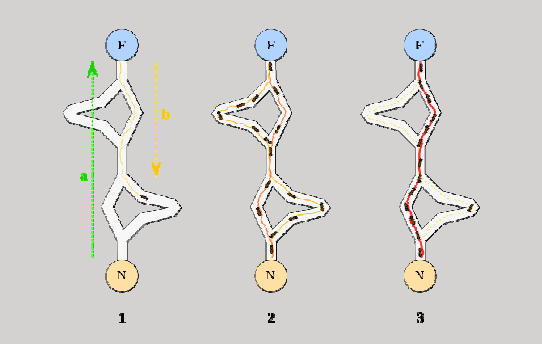
The behavior of each ant in nature –

1. Wander randomly at first, laying down a pheromone trail.
2. If food is found, return to the nest laying down a pheromone trail.
3. If pheromone is found, with some increased probability follow the pheromone trail.
4. Once back at the nest, go out again in search of food.

However, pheromones evaporate over time, such that unless they are reinforced by more ants, the pheromones will disappear.[2]

Description –

1. The first ant wanders randomly until it finds the food source (F), then it returns to the nest (N), laying a pheromone trail.
2. Other ants follow one of the paths at random, also laying pheromone trails. Since the ants on the shortest path lay pheromone trails faster, this path gets reinforced with more pheromone, making it more appealing to future ants.



1. The ants become increasingly likely to follow the shortest path since it is constantly reinforced with a larger amount of pheromones. The pheromone trails of the longer paths evaporate.

Pheromones –

A pheromone is a chemical an animal produces which changes the behavior of another animal of the same species (animals include insects). Some describe pheromones as behavior-altering agents. Many people do not know that pheromones trigger other behaviors in the animal of the same species.

Meta-heuristic –

Heuristic – Pertains to the process of gaining knowledge or some desired result by intelligent guesswork rather than by following some pre established formula. The term seems to have two usages:

1. Describing an approach to learning by trying without necessarily having an organized hypothesis or way of proving that the results proved or disproved the hypothesis. That is, "trial-by-error" learning.
2. Pertaining to the use of the general knowledge gained by experience, sometimes expressed as "using a rule-of-thumb."

A meta-heuristic is a heuristic method for solving a very general class of computational problems in the hope of obtaining a more efficient or more robust procedure. Meta-heuristics are generally applied to problems for which there is no satisfactory problem-specific algorithm or heuristic; or when it is not practical to implement such a method. Most commonly used meta-heuristics are targeted to combinatorial optimization problems.[7]

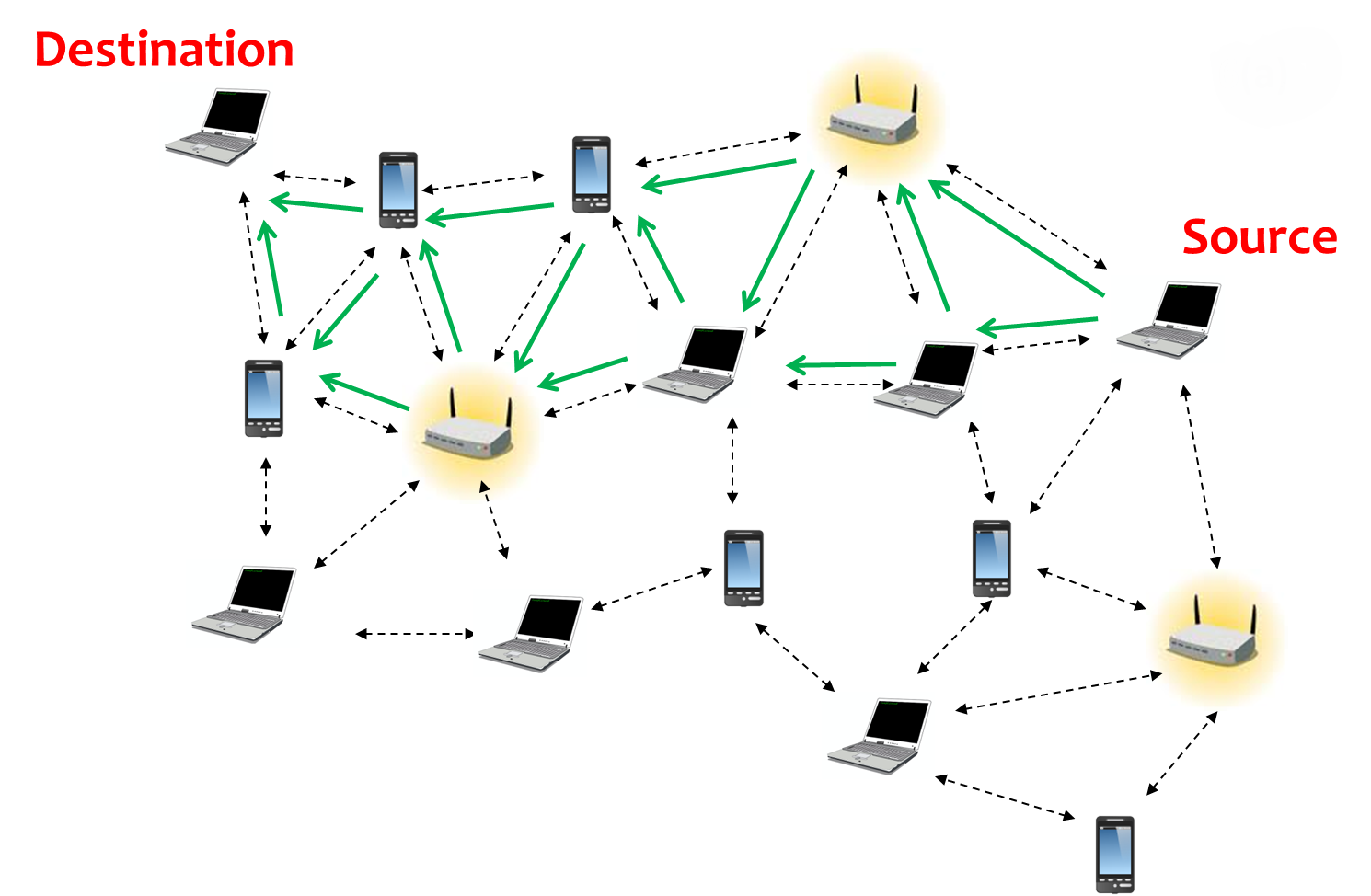
Network Routing Problem -

Network routing is the process of selecting paths in a network along which to send network traffic. [13] Communication networks can be classified as either circuit-switched or packet-switched. The example of circuit switched network is the telephone network in which the physical circuit is set up at the communication start and remains the same for the communication duration. Unlike them, in packet-switched networks, also called data networks, each data packet can follow a different route and no fixed physical circuits are established. The example of data networks are LAN and the Internet. In tour project we will focus on data networks.

The main function of data networks is to assure the efficient distribution of information among its users. This is done through the exploitation of the n etwork control system. One of the most important components of control system is routing. Routing refers to the distributed activity of building and using routing tables.

Routing table is a common component of all routing algorithms. It is used to hold information to make the local forwarding decisions. One routing table is maintained by each node in the network. It tells the nodes incoming data packets which link to use next in order to continue its travel to the destination node of that data packet.

However, one of the main aspects of the network routing problem is that it is non-stationary. Meaning that, one of the routing characteristics is that the traffic over the network changes all the time. Additionally, the nodes and links of the network can suddenly go out of the service, and new nodes and links can be added at any moment. All these characteristics have to be considered in order to create an optimal solution to this problem, shortest path routing.



Shortest path routing –

Shortest path routing is implementing shortest path algorithm on solving the network routing problem. Its objective is to determine the shortest path (minimum cost) between two nodes, where the sum of the costs of its constituent edges is minimized.

Till today, many routing algorithms used for solving shortest path have been accepted. One of them is Dijkstra's algorithm. Dijkstra's algorithm, conceived by Dutch computer scientist Edgar Dijkstra in 1959, is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge costs, producing a shortest path tree. [14]

For a given source node in the graph, the algorithm finds the path with lowest cost (i.e. the shortest path) between that node and every other node. It can also be used for finding costs of shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been reached. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. As a result, the shortest path first is widely used in network routing protocols, most notably OSPF (Open Short Path First).

As I have mentioned, network routing is of a variable nature and nodes and links of the network can suddenly break, as well as the new one can be created. Therefore we have OSPF, which is a dynamic routing protocol and it keeps track of the complete network topology and all the nodes and links within that network.  
In addition to this, if for some time the best route converges within the network, and suddenly the link failure occurs, the OSPF will detect it very quickly in the topology and will converge on the new loop-free routing structure. So to conclude with this section, we now have convergence as one of the potential problems to solve in the following sections as well as in the implementation we prepared.

Congestion Control –

Congestion is an important issue that can arise in packet switched network. Congestion is a situation in Communication Networks in which too many packets are present in a part of the subnet, performance degrades. Congestion in a network may occur when the load on the network *(i.e.* the number of packets sent to the network) is greater than the capacity of the network *(i.e.* the number of packets a network can handle).

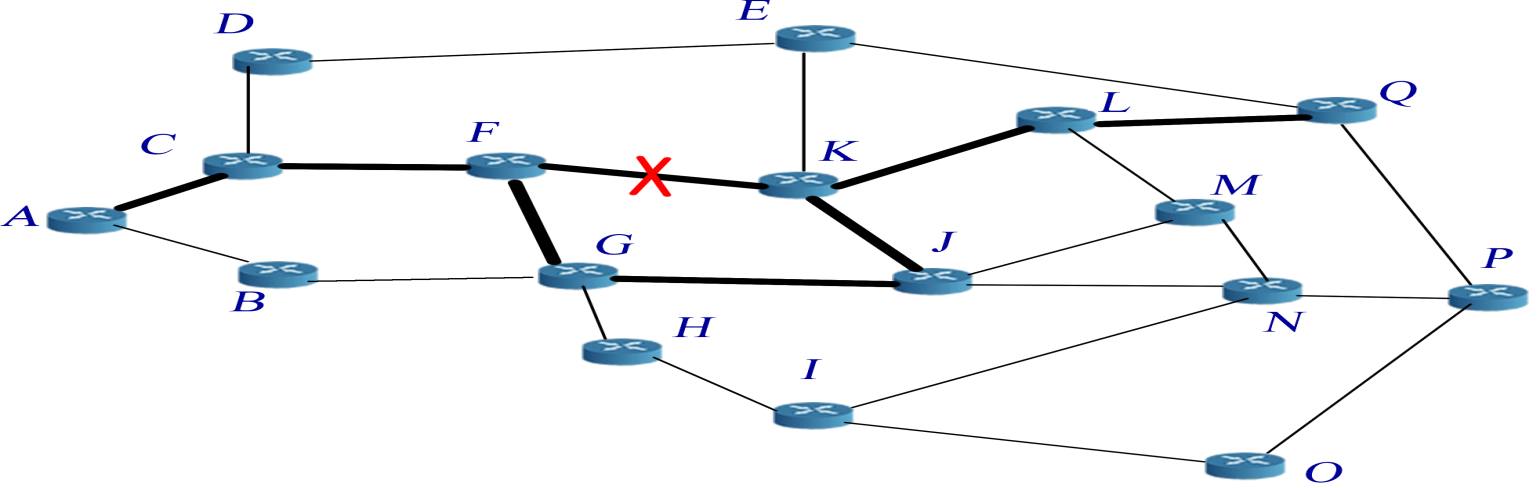
Cause of congestion –

When more packets were sent than could be handled by intermediate routers, the intermediate routers discarded many packets, expecting the end points of the network to retransmit the information. When this packet loss occurred, the end points sent extra packets that repeated the information lost, doubling the data rate sent, exactly the opposite of what should be done during congestion. This pushed the entire network into a 'congestion collapse' where most packets were lost and the resultant throughput was negligible.

Link Failure –

Link failure in a network is an issue which is very commonly observed in packet-switched networks.

It is the most common network failure that occurs due to fiber cut. A link failure results in a change in the network topology.



Roulette Wheel Selection –

The idea behind the roulette wheel selection technique is that each individual is given a chance to become a parent in proportion to its fitness. It is called roulette wheel selection as the chances of selecting a parent can be seen as spinning a roulette wheel with the size of the slot for each parent being proportional to its fitness. Obviously those with the largest fitness (slot sizes) have more chance of being chosen. Thus, it is possible for one member to dominate all the others and get selected a high proportion of the time.

**PROPOSED APPROACH**

**Travelling Salesman Problem** -

According to Marco Dorigo and Thomas Stutzle,

Ant Colony optimization is a metaheuristic in which a colony of artificial ants cooperate in finding good solutions to difficult discrete optimization problems, TSP being one of them.

Although the ACO metaheuristic can be applied to any interesting combinatorial optimization problems, the real issue is how to map the considered problem to a representation that can be used by the artificial ants to build solutions.

Informally, an ACO algorithm can be imagined as the interplay of two procedures:

1. ConstructAntsSolutions
2. UpdatePheromones

ConstructAntsSolutions –

Manages a colony of ants that concurrently and asynchronously visit adjacent states of the considered problem by moving through neighbour nodes of the problems construction graph. They move by applying a local decision policy that makes use of pheromone trails and heuristic information. In this way, ants incrementally build solutions to the optimization problem. Once an ant has built a solution, or while the solution is being built, the ant evaluates the (partial) solution that will be used by the UpdatePheromones procedure to decide how much pheromone to deposit.

UpdatePheromones -

Is the process by which the pheromone trails are modified. The trails value can either increase, as ants deposit pheromone on the components or connections they use, or decrease, due to pheromone evaporation .From a practical point of view, the deposit of new pheromone increases the probability that those components that were either used by many ants or that were used by at least one ant and which produced a very good solution will be used again by future ants. Differently, pheromone evaporation implements a useful form of forgetting: it avoids a too rapid convergence of the algorithm toward a suboptimal region, therefore favoring the exploration of new areas of the search space.

OUTLINE OF ACO METAHEURISTIC -

procedure ACO Metaheuristic

ScheduleActivities

ConstructAntsSolutions

UpdatePheromones

end-ScheduleActivities

end-procedure

SOLUTION TO THE ABOVE PROBLEM –

Transformation of parameters –

|  |  |
| --- | --- |
| **NATURE** | **PROGRAM** |
|  |  |
| Natural habitat | Graph (nodes and edges) |
| Nest and food | Nodes in the graph: start and destination |
| Ants | Agents, our artificial ants |
| Visibility | The reciprocal of distance, η |
| Pheromones | Artificial pheromones, τ |
| Foraging behavior | Random walk through graph (guided by pheromones) |

Parameters -

Construction graph:

The construction graph is identical to the problem graph. The set of nodes correspond to the cities and the connections correspond to the set of edges, and each connection has a weight which corresponds to the distance dij between nodes i and j. The states of the problem are the set of all possible partial tours.

Constraints:

The only constraint in the TSP is that all cities have to be visited and that each city is visited at most once. This constraint is enforced if an ant at each construction step chooses the next city only among those it has not visited yet.

Pheromone trails and heuristic information:

The pheromone trails tij in the TSP refer to the desirability of visiting city j directly after i. The heuristic information hij is typically inversely proportional to the distance between cities i and j, a straight-forward choice being hij.

Solution construction:

Each ant is initially put on a randomly chosen start city and at each step iteratively adds one still unvisited city to its partial tour. The solution construction terminates once all cities have been visited.

Applying the above procedure to solve TSP:

INITIALISATION:

1. Global constants alpha, beta, rho and Q are declared and initialized.

alpha = 3. // influence of pheromone in direction

beta = 2. // influence of adjacent node distance

rho = 0.01 // pheromone decrease factor

Q = 2.0 // pheromone increase factor

1. First a graph of n cities is created with random edge lengths in the range 1.0 to 8.0 such that the graph is connected, symmetric and undirected.

The graph is created as a 2-D double array dist[][] where dist[i][j] denotes the distance to move from city i to city j.

1. Artificial ants are created which traverse the graph asynchronously and independently.

To simulate artificial ants a 2-D integer array ants[][] is created where ants[i] is an array of int values that represent the trail or path, from an initial city through all other cities for the ant with index i.

Here the first index indicates the ant number.

The initialization method allocates a row for the trail for each ant picking a random start.

1. A 2-D double array pheromones[][] is created and initialized to 0.01.

UPDATING ANTS:

Here the ant array is updated such that each ant is reinitialized by constructing a new (better) trail based on the pheromone and distance information.

Updating basically determines the next city.

For determining the next city, an array called "tau" is constructed.

tau value = (pheromone[i][j] ^ alpha) \* (1 / d[i][j]) ^ beta)

Larger values of pheromone increase tau.

Larger distances of edge decrease tau.

Now find the probability of the all the cities and select the one with the maximum.

UPDATING PHEROMONES:

Pheromone value is decreased, simulating evaporation and increased, simulating the deposit of pheromones by ants already visited the trail.

pheromones[i][j] = pheromones[i][j] + (Q / d[i][j]) + (rho \* pheromones[i][j]).

**PSEUDO CODE**

main()

{

numCities, numAnts

Creation and Initialization of graph matrix

Initialization of the ant’s trails

Creation and Initialization of pheromone matrix

BestTrail(…) //Finds the best trail among the trails of all the ants

Bestlength = Length(…) //Length of the best trail

Initialize an arbitrary tmax

for ( time = 0 to tmax ) {

UpdateAnts(…) //Starts from a random ant, and builds the trail based on the amount of pheromone deposited on the outgoing paths

UpdatePheromones(…) //Updates pheromone value for each edge present in the trail

BestTrail(…)

currbestlength = Length(…)

If (currbestlength < bestlength)

Update bestlength

Store the corresponding trail

}

}

Output: Best Length and the corresponding trail denoting the order in which cities are to be travelled.

**Network Routing Problem** –

Applying the same Ant Colony Optimization algorithm as applied in the previously solved Travelling Salesman Problem (TSP).

ACO algorithm can be imagined as the interplay of two procedures :

1. ConstructAntsSolutions
2. UpdatePheromones

ConstructAntsSolutions –

//Described previously for the TSP.

UpdatePheromones –

//Described previously for the TSP.

OUTLINE OF ACO METAHEURISTIC -

procedure ACO Metaheuristic

ScheduleActivities

ConstructAntsSolutions

UpdatePheromones

end-ScheduleActivities

end-procedure

SOLUTION TO THE ABOVE PROBLEM –

Transformation of parameters –

|  |  |
| --- | --- |
| **NATURE** | **PROGRAM** |
|  |  |
| Natural Topology | Graph (nodes and edges) |
| Host and Destination | Nodes in the graph: start and destination |
| Packets | Agents, our artificial ants |
| Pheromones | Artificial pheromones, τ |
| Routing behavior | Random walk through graph (guided by pheromones) |

ACO in Network Routing Problems –

ACO algorithms can be applied in the network routing problems to find the shortest path. In a network routing problem, a set of artificial ants (packets) are simulated from a source to the destination. The forward ants are selecting the next node randomly for the first time and the ants who are successful in reaching the destination are updating the pheromone deposit at the edges visited by them. The next set of the ants can now learn from the pheromone deposit feedback left by the previously visited successful ants and will be guided to follow the shortest path.[1]

The main characteristic of the ACO is that the pheromone values are updated by all the ants  
(packets) which have reached the destination successfully. However, before adding the pheromone  
we first must perform the evaporation action.

One more advantage of the application of the ACO in network routing is that when the number of  
packets increases, this algorithm can be applied for controlling congestion also. In static routing  
algorithm all the packets from a starting to the destination node will after some time follow a  
constant path calculated by the algorithm and therefore we can have a problem with congestion. As  
a result, some of the packets will have to wait. However, in ACO as the next node is select randomly,  
with the probability to chose the shortest path more, some packets will some other paths which  
increases the network performance and fights congestion.  
In addition to this, in dynamic circumstances, if after some time the shortest path converges and  
suddenly we get the link failure and disconnection between two nodes which are on the route of the  
shortest path, the ACO will quickly follow some other path and converge on it.

Parameters -

Construction graph:

The construction graph is identical to the problem graph. The set of nodes correspond to the ‘routers' and the connections correspond to the set of edges, and each connection has a weight which corresponds to the ‘distance metric’ dij between nodes i and j.

Distance metric:

The performance measure that is usually taken into account is the throughput. However, in our implementation we have taken the performance measure as total sum of costs between edges.

Constraints:

The only constraint in Network Routing is that a data packet (an ant) can only visit a router at most once, and must stop as it reaches its destination. This constraint is enforced if a packet (an ant) at each construction step chooses the next city only among those it has not visited yet.

Pheromone trails and heuristic information:

The pheromone trails tij in the Network Routing refer to the desirability of choosing router j directly after i. The heuristic information hij is typically inversely proportional to the ‘distance metric’ between routers i and j, a straight-forward choice being hij.

Solution construction:

Each ant is initially put on a randomly chosen start router (source) and at each step iteratively adds one unvisited (intermediate) router to its partial tour. The solution construction terminates once the destination router have been visited.

Applying the above procedure to solve Network Routing:

INITIALISATION:

1. Global constants alpha, beta, rho and Q are declared and initialized.

alpha = 3. // influence of pheromone in direction

beta = 2. // influence of adjacent node distance

rho = 0.01 // pheromone decrease factor

Q = 2.0 // pheromone increase factor

1. First a graph of n routers is created with random edge lengths in the range 1.0 to 20.0 such that the graph is connected and directed.

The graph is created as a 2-D double array dist[][] where dist[i][j] denotes the distance to move from router i to router j.

1. Artificial ants are created which traverse the graph asynchronously and independently.

To simulate artificial ants a 2-D integer array ants[][] is created where ants[i] is an array of int values that represent the trail or path, from an initial router through all other routers for the ant with index i.

Here the first index indicates the ant number.

The initialization method allocates a row for the trail for each ant picking a random start.

1. A 2-D double array pheromones[][] is created and initialized to 0.01.

UPDATING ANTS:

// Described previously for the TSP.

UPDATING PHEROMONES:

// Described previously for the TSP.

The probability that the ant will select a node j from node i is given by the following formula:

Pij = τijα . ηijβ

∑ τijα . ηijβ

This is of course, if the link between two nodes exists.

Here, τij is the pheromone of each edge which joins the nodes i and j. ηij = (1/dij), where dij is the distance between the nodes i and j. α and β are two parameters which determine the relative influence of the pheromone trail and the heuristic information.

Pheromone evaporation (ρ) on the edge between node i and node j is implemented by the formula:

τij  = (1 - ρ)τij

In addition, the amount of pheromone which some ant k, add to the edge which it has not passed is 0. Otherwise, if the ant k has passed through some edge between the nodes, it will leave the amount of pheromone which is inversely proportional with the total cost of all the edges ant k has passed on its path from the starting node to the destination node. The following formula presents this process:

τij = τij + ∑ ∆τijk , for every (i, j)

Δτijk  is the amount of pheromone an ant k deposits on the edges visit. It is calculated by the following  
expression:

∆τijk = 1/ Ck or 0

Where Ck is the total cost of all the edges an ant k has passed on its path from the starting to the  
destination node.

**PSEUDO CODE**

main()

{

numRouters, numAnts

Creation and Initialization of graph matrix

Initialization of the ant’s trails

Creation and Initialization of pheromone matrix

BestTrail(…) //Finds the best trail among the trails of all the ants

Bestlength = Length(…) //Length of the best trail

Initialize an arbitrary tmax

for ( time = 0 to tmax ) {

UpdateAnts(…) //Starts from a random ant, and builds the trail based on the amount of pheromone deposited on the outgoing paths

UpdatePheromones(…) //Updates pheromone value for each edge present in the trail

BestTrail(…)

currbestlength = Length(…)

If (currbestlength < bestlength)

Update bestlength

Store the corresponding trail

}

}

Output: Best Length and the corresponding trail denoting the order in which routers are to be travelled.

**Advantages of using ACO to find shortest path –**

Ant Colony Optimization, when measuring performance by standard measures such as network throughput and average packet delay, outperforms the current Internet routing algorithm (OSPF), some old Internet routing algorithms (SPF and distributed adaptive Bellman-Ford), and recently proposed forms of asynchronous online Bellman-Ford (Q-routing and Predictive Q-routing).[15]

Even though this algorithm does not always find the shortest route, it definitely brings some more positive aspects to the problem itself as it minimizes the other problems which network routing has to fight with such as congestion, convergence break up, etc.

1. Congestion control

One major advantage of the application of the ACO in network routing is that when the number of packets increases, this algorithm can be applied for controlling congestion also.

In static routing algorithm all the packets from a starting to the destination node will after some time follow a constant path calculated by the algorithm and therefore we can have a problem with congestion. As a result, some of the packets will have to wait. However, in ACO as the next node is select randomly, with the probability to choose the shortest path more, some packets will some other paths which increases the network performance and fights congestion.

1. Link failure

In dynamic circumstances, if after some time the shortest path converges and suddenly we get the link failure and disconnection between two nodes which are on the route of the shortest path, the ACO will quickly follow some other path and converge on it.

**RESULTS**

**PERFORMANCE OF THE TRAVELLING SALESMAN PROBLEM -**

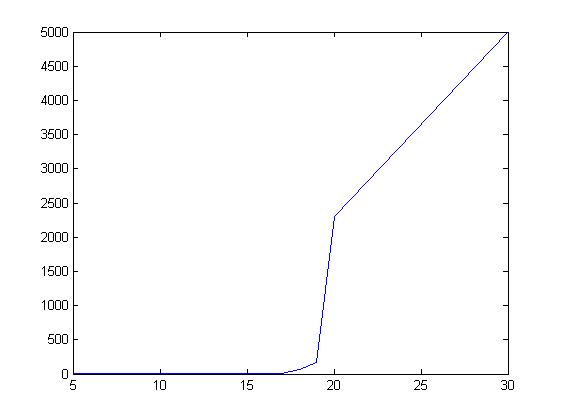
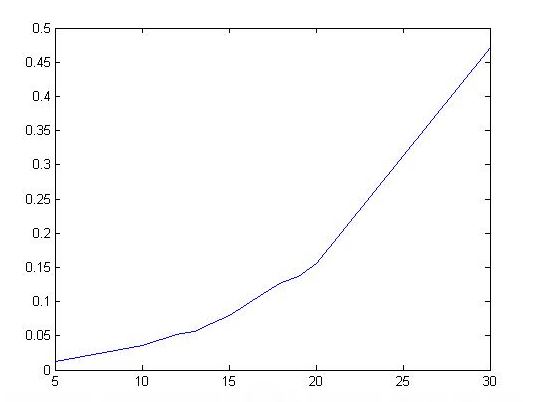
Performance of ACO implementation of TSP vs Branch and Bound implementation on the basis of time complexity–

Considering no. of ants = 4, and No. of iterations = 1000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. Of Cities | Minimum Cost(ACO) | Time(s) (ACO) | Minimum Cost (Branch and Bound) | Time(s)  (Branch and Bound) |
|  |  |  |  |  |
| 5 | 5 | 0.012 | 7 | 0.003 |
| 10 | 20 | 0.036 | 23 | 0.008 |
| 12 | 23 | 0.052 | 22 | 0.028 |
| 13 | 19 | 0.056 | 25 | 0.224 |
| 14 | 17 | 0.068 | 17 | 0.052 |
| 15 | 34 | 0.080 | 31 | 0.403 |
| 16 | 18 | 0.096 | 22 | 0.836 |
| 17 | 23 | 0.112 | 23 | 12.862 |
| 18 | 27 | 0.128 | 34 | 70.459 |
| 19 | 29 | 0.136 | 32 | 170.00 |
| 20 | 30 | 0.156 | 32 | 38min 24 |
| 30 | 39 | 0.472 | - | (Awaited) |

Graph:

x-axis : No. of Cities, y-axis : Time Taken



ACO Branch and Bound

Performance of ACO on the basis of convergence of the solution by varying the number of ants –

Considering no. of cities = 100

|  |  |  |
| --- | --- | --- |
| No. of Ants | Minimum Cost | Time(s) |
|  |  |  |
| 5 | 369 | 15.00 |
| 10 | 360 | 30.00 |
| 15 | 360 | 45.00 |
| 20 | 352 | 61.00 |
| 25 | 352 | 76.00 |
| 30 | 352 | 91.00 |

Considering no. of cities = 80

|  |  |  |
| --- | --- | --- |
| No. of Ants | Minimum Cost | Time(s) |
|  |  |  |
| 5 | 289 | 8.23 |
| 10 | 277 | 16.41 |
| 15 | 277 | 24.50 |
| 20 | 269 | 32.70 |
| 25 | 269 | 41.00 |
| 30 | 269 | 49.00 |

Considering no. of cities = 60

|  |  |  |
| --- | --- | --- |
| No. of Ants | Minimum Cost | Time(s) |
|  |  |  |
| 5 | 68 | 3.75 |
| 10 | 66 | 7.47 |
| 15 | 65 | 11.11 |
| 20 | 66 | 14.78 |
| 25 | 66 | 18.50 |
| 30 | 62 | 22.00 |

**PERFORMANCE OF NETWORK ROUTING PROBLEM –**

On the basis of correctness of solution by comparing ACO implementation with Djikstra algorithm –

Considering Source = 0 and Destination = 10 for every case.

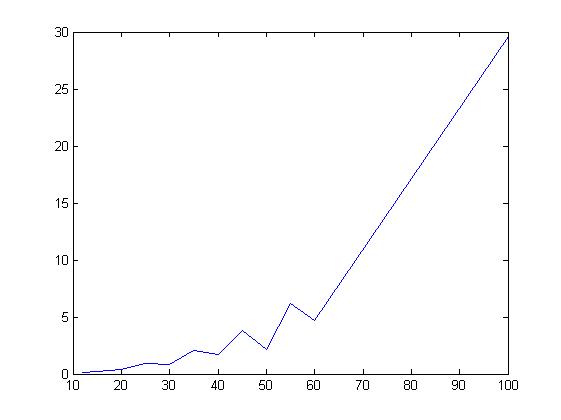
For ACO implementation,

Considering no. of ants = 4

|  |  |  |  |
| --- | --- | --- | --- |
| No. Of Routers | Cost of shortest path  (ACO) | Time(s)  (ACO) | Cost of shortest path  (Djikstra Algorithm) |
|  |  |  |  |
| 12 | 19 | 0.15 | 19 |
| 15 | 23 | 0.20 | 23 |
| 20 | 13 | 0.436 | 13 |
| 25 | 12 | 0.92 | 12 |
| 30 | 21 | 0.816 | 10 |
| 35 | 18 | 2.09 | 7 |
| 40 | 10 | 1.75 | 10 |
| 45 | 33 | 3.84 | 23 |
| 50 | 4 | 2.17 | 4 |
| 55 | 30 | 6.20 | 17 |
| 60 | 20 | 4.69 | 16 |
| 100 | 15 | 29.60 | 10 |

Graph:

x-axis : No. of Routers, y-axis : Time Taken



Performance of ACO On the basis of time complexity -

Considering no. of Ants = 10

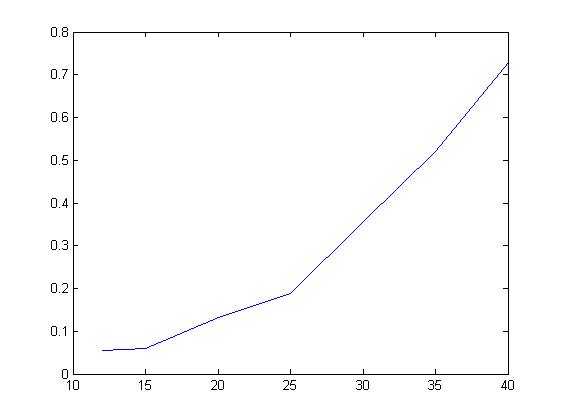
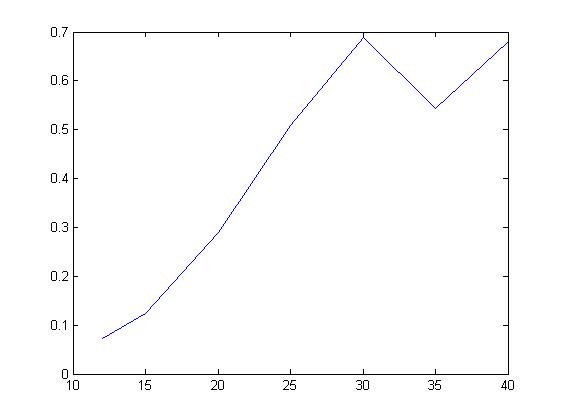
|  |  |  |
| --- | --- | --- |
| No. of Routers | Shortest Path | Time(s) |
|  |  |  |
| 12 | 0 – 9 – 10 | 0.056 |
| 15 | 0 – 3 – 10 – 17 | 0.060 |
| 20 | 0 – 10 | 0.132 |
| 25 | 0 – 22 – 10 | 0.188 |
| 30 | 0 – 22 – 10 | 0.356 |
| 35 | 0 – 22 – 2 – 10 | 0.520 |
| 40 | 0 – 26 – 10 | 0.728 |

Considering no. of Ants = 20

|  |  |  |
| --- | --- | --- |
| No. of Routers | Shortest Path | Time(s) |
|  |  |  |
| 12 | 0 – 5 – 10 | 0.073 |
| 15 | 0 – 12 – 10 | 0.123 |
| 20 | 0 – 17 – 10 | 0.290 |
| 25 | 0 – 12 – 10 | 0.508 |
| 30 | 0 – 10 | 0.688 |
| 35 | 0 – 7 – 10 | 0.544 |
| 40 | 0 – 10 | 0.681 |

Graph :

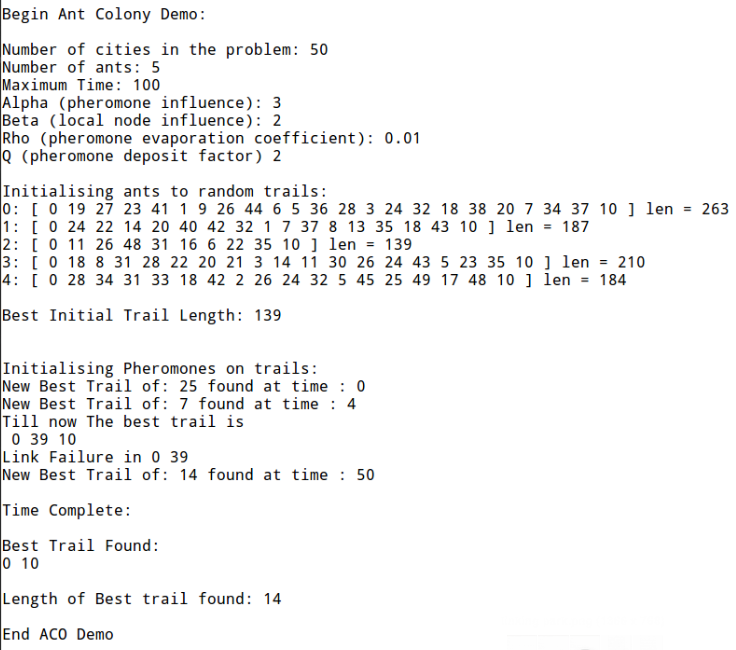
x-axis : No. of Routers, y-axis : Time Taken

** **

10 Ants 20 Ants

Showing handling of link failure –

In dynamic circumstances, if after some time the shortest path converges and suddenly we get the link failure and disconnection between two nodes which are on the route of the shortest path, the ACO will quickly follow some other path and converge on it.

****

**SHORTCOMINGS**

1. This algorithm does not always find the shortest route.
2. Time complexity increases when the number of routers increase to a very high number.
3. In our implementation we are checking that an ant is not making any cycles in its path whenever it is going to select a new node in its journey. So, for any number of nodes present in the network, it has been found that this implementation ACO is best from the point of view of the throughput.[15]

However, if we consider from the point of view of the number of packets that has been reached successfully in its destination, our implementation lags behind.

Whenever an ant is following a path, that has no other solution except to form a cycle, we are canceling the journey of the ant and hence reducing the number of successful packets which is a serious drawback of our ACO when all the packets in the network hold equal importance.

**FUTURE IMPROVEMENTS**

The Ant Colony Optimization algorithm can be modified to fulfill our requirements. Few versions of the ACO are given below –

1. **ACO1** (Our Implementation):

In the first method, no ants (packets) are allowed to visit a node that is already visited by it in its journey, that is no ants (packets) are allowed to make a loop in its path.

While selecting the next node, it is checking that the node has already been visited by it or not, if the node is an already visited one, then just discard the node and check for the other available nodes. If a packet reaches in a state such that it has no other way except to select an already visited node, then stop the packet and mark the packet as unsuccessful.[15]

1. **ACO2** :

In the second method, the packets are allowed to make loops in their paths and visit an already visited node provided that it will not visit only the last node visited by it.

However, this method may give rise to a situation where an ant (packet) may fall in an infinite loop. To avoid this situation, after a certain interval of time if an ant (packet) has not reached the destination, the packet is marked as unsuccessful and its journey is forcefully stopped.[15]

1. **ACO3**:

When the network model is large enough, the restriction of not visiting the last visited node only can be little modified with the restriction that the ants (packets) will not visit the last number of nodes already visited by it. This modification is done to reduce the chances of forming loops in the path of the journey and hence reducing the overhead of removing the cycles from the path of the successful packets after the completion of its journey. Here also the journey of the ants (packets) are suspended after a certain interval of time and the packets failed to reach the destination are marked as unsuccessful.

In this method, a Tabu list is maintained to keep the list of the last n number of nodes visited by a packet so that the last n number of nodes are not selected while selecting the next node for going to the destination. Here experiments are carried out for large networks by varying the size of this Tabu list to find the optimum size of the Tabu list with the given number of nodes in the network model.[15]

**WORK DONE**

TILL MID-SEM

* Understood the concept of the Travelling Salesman Problem -

Type of problem: NP – Hard.

General solving technique: Brute force method, Branch and Bound approach.

* Implemented the Travelling Salesman Problem using Branch and Bound approach.

Works better than the Brute force method (i.e., exhaustive search for the results obtained from all the possible combinations and comparing the results).

Problems encountered with the above technique - well only for solving the problems with no more than 40-80 nodes. For the practical relevance, it is necessary to solve the larger-scale problems with the help of heuristics.

* Familiarized with the basic elements of Ant Colony Optimization (ACO) technique of solving combinatorial problems.

Heuristic methods vary from exacts methods in that they give no guarantee to find the optimal solution to the given problem (so that solution is called suboptimal), but in many cases this is the solution of good quality and we can obtain it in acceptable time. Heuristic methods are usually focused on solving the special type of problems.

* Implemented the Travelling Salesman Problem using Ant Colony Optimization (ACO)

meta-heuristics.

AFTER MID-SEM

* Comparative Study of the performances of the Ant Colony Optimization (ACO) implementation and the Branch and Bound approach of solving the Travelling Salesman Problem on the basis of time complexity analysis.
* Implementation of the Roulette Wheel Selection algorithm to improve the performance of Travelling Salesman Problem.
* Implementation of the Network Routing Problem to study the real life application of Ant Colony Optimization.
* Study of the Ant Colony Optimization (ACO) implementation of our problem (Network Routing).
* Found some future improvements which can be done in the current version to improve the performance.

**TOOLS USED**

System:

Linux Ubuntu

Version: 12.04 LTS

Language used:

C++

Version: 4.6.3

Libraries: Standard Template Library

Matlab

Version: R2010a

**ACTIVITY TIME CHART**

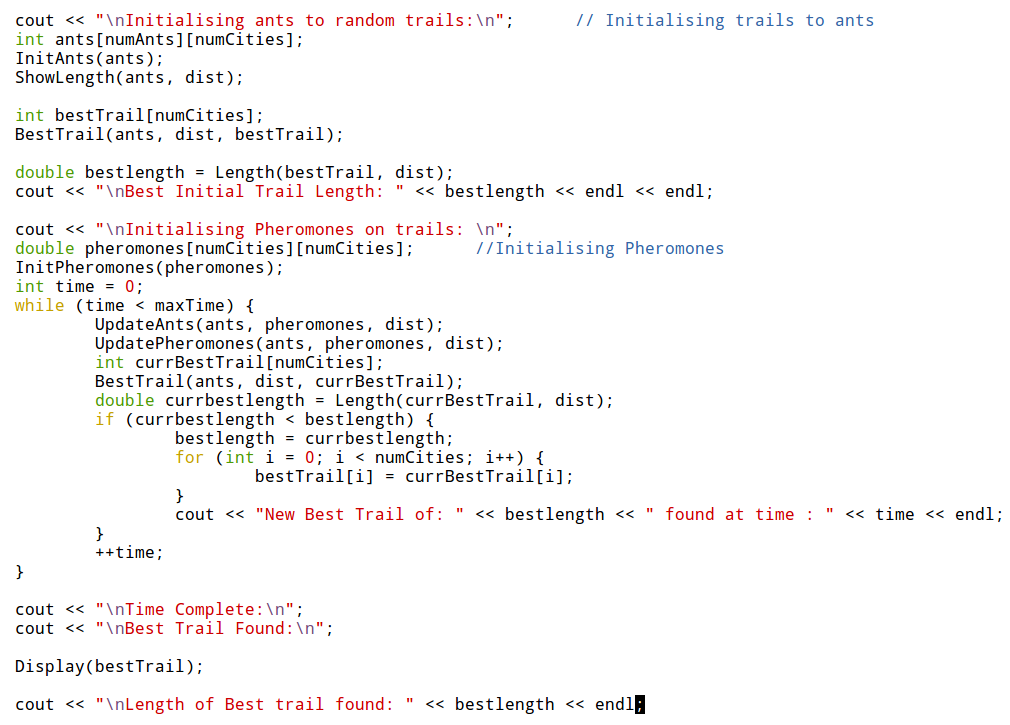
|  |  |  |
| --- | --- | --- |
| **TASK** | **STATUS** | **DURATION** (Estimated) |
|  |  |  |
| Study of the Ant Colony Optimization | Completed | 10th Aug. – 15th Aug. |
| Figuring out the problem on which to apply the ACO | Completed | 17th Aug. – 19th Aug. |
| Study of the Travelling Salesman Problem | Completed | 20th Aug. – 21st Aug. |
| Implementation of TSP using Branch and Bound approach | Completed | 22nd Aug. – 23rd Aug. |
| Identifying the Base Research Paper | Completed | 23rd Aug. – 27th Aug. |
| Analysis of the Research Paper  (Ant Colony Optimization: A New Meta-Heuristic by Marco Dorigo and Gianni Di Caro) | Completed | 28th Aug. – 2nd Sept. |
| Implementation of TSP using the ACO meta-heuristic | Completed | 5th Sept. – 18th Sept. |
| Preparation of the project presentation | Completed | 2nd Oct. – 5th Oct. |
| Preparation of the project report | Completed | 3rd Oct. – 6th Oct. |
| Comparative study between the results obtained from Branch and Bound approach and ACO | Completed | 10th Oct. – 13th Oct. |
| Finding out a real life problem on which to apply the ACO | Completed | 15th Oct. – 17th Oct. |
| Study of the Network Routing Problem | Completed | 1st Nov. – 2nd Nov. |
| Study of a few Research Papers | Completed | 3rd Nov. – 5th Nov. |
| Implementation of Network Routing using ACO | Completed | 8th Nov. – 11th Nov. |
| Implementing the Roulette Wheel Selection algorithm | Completed | 12th Nov. – 13th Nov. |
| Study and Complexity analysis of the ACO implementation of the network routing problem | Completed | 23rd Nov. – 24th Nov. |
| Preparation of the project report | Completed | 24th Nov. – 25th Nov. |
| Preparation of the project presentation | Completed | 25th Nov. – 26th Nov. |

**APPENDIX**

Code Snippet –

Travelling Salesman Problem –

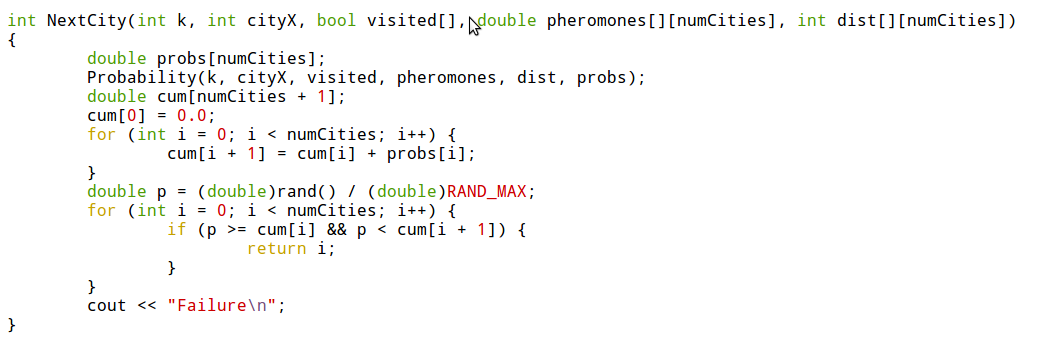
Basic construct –



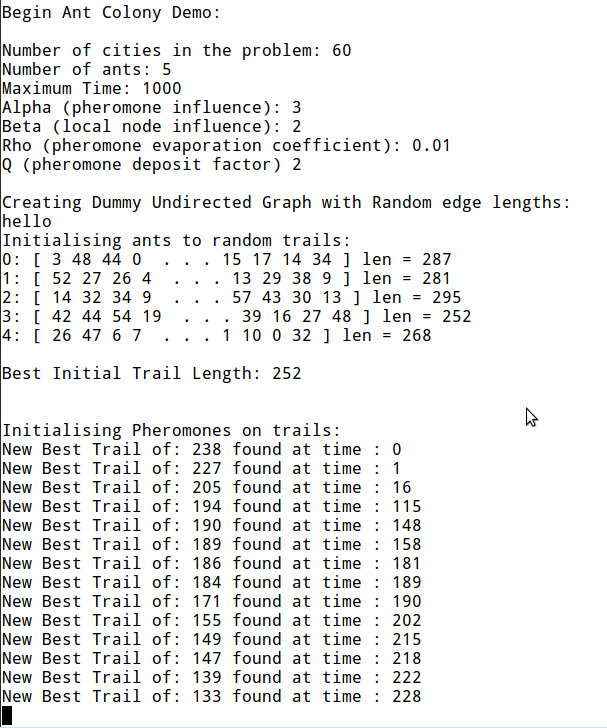
Roulette Wheel Selection –

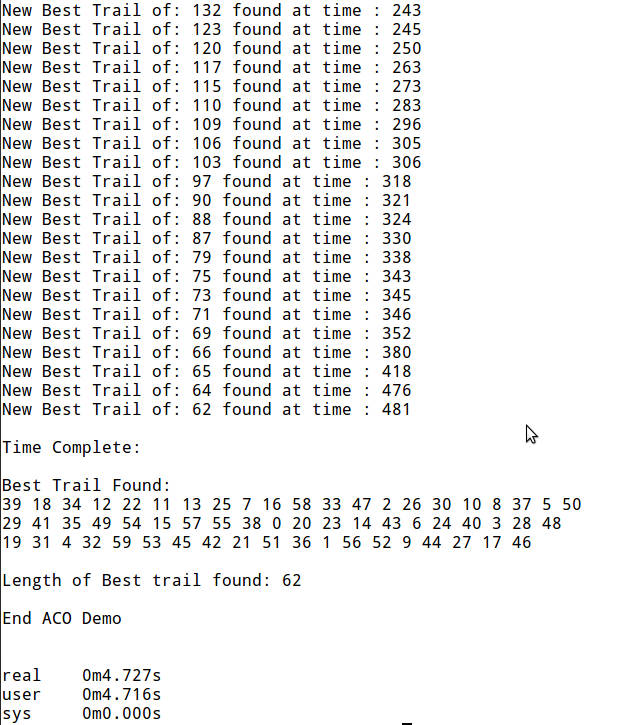
Roulette Wheel Selection ensures that each individual is given a chance to become a parent in proportion to its fitness.

Here in TSP, the next city is selected using Roulette Wheel Selection algorithm.



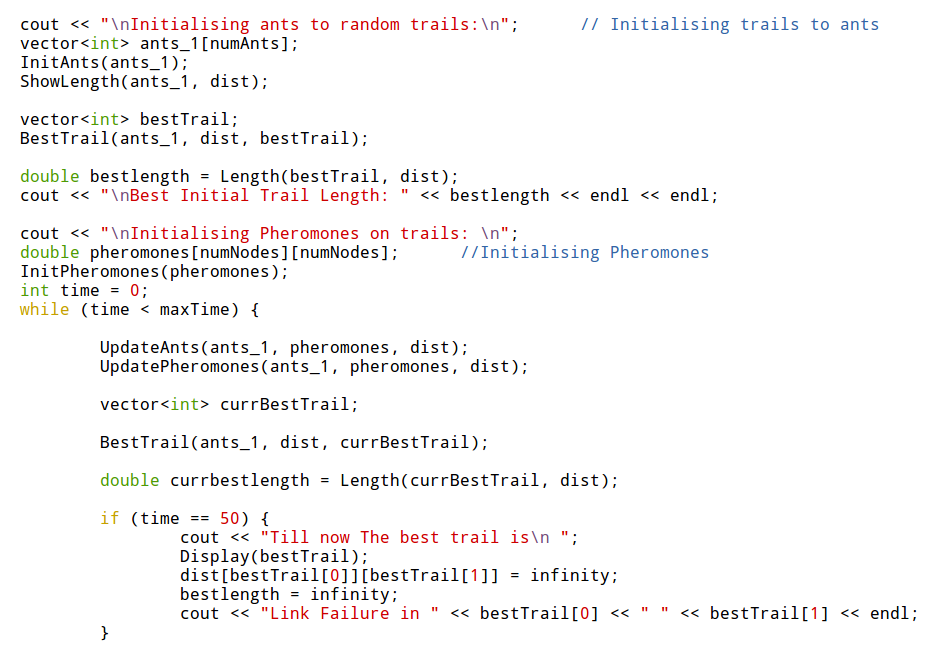
Processing the output –

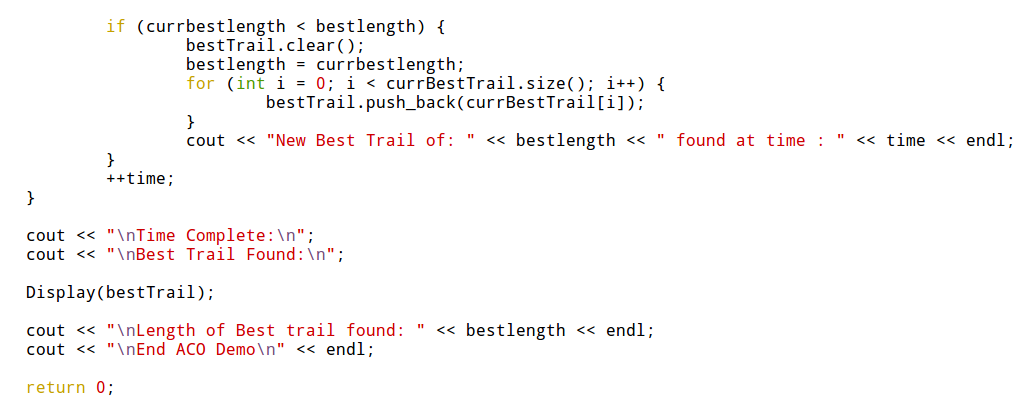




Network Routing Problem –

Basic construct –





Roulette Wheel Selection algorithm is applied in the same way as in TSP.

**REFERENCES**

[1] Marco Dorigo and Thomas Stutzle, Ant Colony Optimization. Cambridge, MIT Press, 2004.

[2] Marco Dorigo and Gianni Di Caro, Ant Colony Optimization: A New Meta-Heuristic, IEEE Press.

[3] Dorigo M. & L.M. Gambardella (1997). **Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem.** IEEE Transactions on Evolutionary Computation, 1, 1, in press.

[4] Wikipedia(<http://en.wikipedia.org/wiki/Travelling_salesman_problem>)

[5] Wikipedia(<http://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms>)

[6] Wikipedia(<http://en.wikipedia.org/wiki/Branch_and_bound>)

[7] Wikipedia(<http://en.wikipedia.org/wiki/Metaheuristic>)

[8] Wikipedia(<http://en.wikipedia.org/wiki/Combinatorial_optimization>)

[9] Wikipedia(<http://en.wikipedia.org/wiki/Fitness_proportionate_selection>)

[10] Wikipedia(<http://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle>)

[11] <http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/AproxAlgor/TSP/tsp.htm>

[12] <http://www.scholarpedia.org/article/Ant_colony_optimization>

[13] Wikipedia(<http://en.wikipedia.org/wiki/Routing>)

[14] Wikipedia(<http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm>)

[15] Debasmita Mukherjee and Sriyankar Acharyya, Ant Colony Optimization Technique Applied in Network Routing Problem. International Journal of Computer Applications (0975 - 8887).

**COMMENTS AND SUGGESTIONS**