Dynamic adaptive routing algorithm based on ant algorithm in combination with queuing network analysis

Megha Singh[1], Jitendra Agrawal [2] and Sanjeev Sharma[3]

[123] School of Information Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, M.P. India

Abstract

The field of *Mobile Ad hoc Networks* (MANETs) has expanded an important part of the interest of researchers, hence become very accepted in last few years. The main method for evaluating the performance of MANETs is simulation. This research study introduces a new adaptive and dynamic routing algorithm for MANETs based on the *Ant Colony Optimization* (ACO) algorithms with network delay analysis. Ant colony optimization algorithm helps in finding, if not the shortest, at least a very good path connecting the colony's nest with a source of food. This evaluation of MANETs is based on the estimation of the mean End-to-End delay to send a packet from source to destination node through a MANET. The most important performance evaluation metrics in computer networks is evaluated as mean End-to-End delay.

We scrutinize various simulation set-ups with different node density and pause times. Our new algorithm offers good results under certain conditions such as, increasing the pause time and decreasing node density.

Keywords: *Ant Colony Optimization, Mobile Ad hoc Network, Network Analysis, Routing Algorithms, Mobility Models.*

1. INTRODUCTION

Swarm Intelligence is one of the new approaches which is based on the natural phenomena and commonly known as *Meta-Heuristics* which is a new field in terms of its application to combinatorial optimization problems. The term was first used in [Beni, 1988, Beni and Wang, 1989a,b] to describe the coordination of robots. Since then, two main version of SI have been used for originating schemes to solve optimization problems. One is, *Particle Swarm Optimization* (PSO), and the other is, *Swarm Intelligence* SI is defined to include any scheme which employs the swarm mechanics derived from the behavior of social insects, particulary ants, for solving optimization problems [Bonabeau et al., 1999]. Firstly *Ant Algorithms* was introduced in [Dorigo, 1992] and after that, it has been gained a great success by applying it on both theoretical and practical optimization problems. Ant algorithms are an iterative, probabilistic meta-heuristic for finding solutions to combinatorial optimization problems. They are based on the foraging mechanism employed by real ants trying to find a short path from their nest to a food source. In foraging, the ants communicate circuitously via pheromone, which they use to stain their respective paths and which catch the attention of other ants. In the ant algorithm, artificial ants use virtual pheromone to update their path through the decision graph, i.e. the path that reflects which alternative an ant chooses at certain points. Ants of the later iterations use the pheromone inscription of previous good ants as a means of orientation when constructing their own solutions, which ultimately result in focusing the ants on promising parts of the search space.

Ant algorithms provides adaptation, robustness and decentralized nature, which are appropriate for routing in modern communication networks such as *Mobile Ad hoc Networks* (MANETs). This study commences a new approach for routing in MANETs, which is based on ant algorithm with network delay analysis.

2. Proposed Scheme

2.1 Introduction

The main goal of this study is to present a complete simulation model for dynamic routing in *Mobile Ad hoc Networks* (MANETs) based on *Ant Algorithm* in combination with *network analysis*. The routing algorithm is based on a type of learning algorithm, similar to one described in AntNet, that provides deterministic forwarding of ant packets from source node to destination node. We assume all the links as bi-directional and all the nodes in the network fully collaborate in the operation of the algorithm.

2.2 Mobility Models Used In Simulation

Mobility models play a key role in simulating ad hoc networks. In our simulation process, we will simulate two of these mobility models; *Random Waypoint*

Mobility (RWM) model and *Boundless Simulation Area Mobility* (BSAM) model.

2.2.1 Random Waypoint Mobility (RWM)

This model includes pause times between changes in direction and/or speed [Johnson and Maltz, 1996]. A MN initiate by staying in one location for a certain period of time (i.e., a pause time). Once this time runs out, the MN opt a random destination in the simulation area and a speed that is uniformly distributed between [min_{speed}, max_{speed}]. The MN then travels toward the newly chosen destination with the selected speed. Upon arrival, the MN pauses for a specified time period before starting the process again.

2.2.2 Boundless Simulation Area Mobility (BSAM)

In the *Boundless Simulation Area Mobility* (BSAM) model there exists a relationship between the previous direction θ, velocity v of a MN and its current direction, velocity. This relationship limits the change in direction and speed per time unit that generates more realistic movement patterns. Both the velocity vector $V = (v,\theta)$ and the MN's position (x,y) are updated at every Δt time step according to the following formulas:

$$v(t+\Delta t) = min[max(v(t)+\Delta v, 0), V_{max}];$$
$$\theta(t+\Delta t) = \theta(t) + \Delta\theta;$$
$$x(t+\Delta t) = x(t) + v(t) * \cos\theta(t);$$
$$y(t+\Delta t) = y(t) + v(t) * \sin\theta(t);$$

where V_{max} is the maximum velocity defined in the simulation, Δv is the change in velocity which is uniformly distributed between $[-A_{max}*\Delta t, A_{max}*\Delta t]$, A_{max} is the maximum acceleration of a given MN, $\Delta\theta$ is the change in direction which is uniformly distributed between $[\alpha*\Delta t, \alpha*\Delta t]$ and α is the maximum angular change in the direction travelled by a MN.

2.3 Network Configuration

MANET is plotted on a directed graph with N nodes and M links. All the links are viewed as bit pipes exemplify by a bandwidth (bits/sec) and a transmission delay (sec). For this purpose, every node of type store-and-forward holds a buffer space where the incoming and the out coming packets are stored. Packets are queued and served on the basis of a First-In-First-Out (FIFO) policy. A packet reads from the routing table the information about which link to use to follow the path toward its target node. On packet's arrival, if there is not enough buffer space to hold it, the packet will be discarded. The links between any two mobile nodes i and j in our network are bi-directional links.

2.4 A Complete Scenario of the Simulation Algorithm

2.4.1 Simulation Environment and Assumption Initialization

• Topology Initialization

The simulation environment consists of mobile nodes that are constantly moving on a simulation rectangular area 1000m×1000m that is chosen to have longer distances so packets are sent over more hops. MNs are moving according to one of the mobility models; Random Waypoint Mobility (RWM) model and Boundless Simulation Area Mobility (BSAM) model.

• Routing Table Initialization

A routing table is a locally stored for every node of the network with rows and columns. The routing table columns represent the neighbor nodes that can be reached from that node. For every node of the network, except the node itself, the routing table contains rows which represent the destinations. The table entries are numeric values between 0 and 1 which show the probability to reach other nodes represented by the rows along the neighbor nodes in shortest time. The probabilities in each row sum up to 1. Its entries are initialized with probabilities 1/N for each neighbor as the next hop for the respective destination, where N is the number of neighbors of the node. The probability value P_{in} which expresses the goodness of choosing n as next node when the destination node is i, is stored for each pair (i, n) with the constraint:

$$\sum_{n \in N_k} P_{in} = 1 \quad \in [1, N_k],\ N_k = \{\text{neighbors}(k)\};$$

The uniform probabilities assigned to all the neighbors signifies indefinite state of the network. The routing table at each node is organized in the form (*Destination, Next hop, Probability*) on a per-destination basis. It contains the goodness values for a particular neighbor to be selected as the next hop for a particular destination. A notation is used to give each entry a unique name: $P_{destinationNode,\ neighborNode}$. The first variable, *destinationNode*, defines the row in the routing table and the second variable, *neighborNode*, defines the column. The probability $P_{destinationNode,\ neighborNode}$, is the probability to reach the destination node in shortest time by going along the neighbor node from the current node.

• *Parameter Initialization*

The *arrival rate* λ, which is the average number of packets arrived in each unit of time. Another parameter is the *service rate* μ_{ij} (Bandwidth) between each two nodes i and j in the network,

which is the average number of packets serviced in each time unit. This parameter μ_{ij} will be given in a matrix called service rate table.

• *Timer Initialization*

At the beginning of the simulation process, we initialize two different timers. Ant timer T_{ant} is the time for the ant algorithm to work on a given network topology (termination condition for ant algorithm). The Simulation timer T_{simul} is the whole time for the simulation process in which the ant algorithm will be repeated each T_{ant} time. For example, if T_{simul} =180 sec. and T_{ant} =30 sec. then, the ant algorithm will be repeated 6 times on 6 different network topologies.

2.4.2 Queuing Network Analysis

In simulation, we will use *Kleinrocks's Independence Assumption* to calculate the delay of sending a packet.

End-to-End delay = $T_{destination\ receives\ packet} - T_{source\ wants\ to\ send\ packet}$.

Time between the moment in which the source wants to send a packet and the moment the packet reaches its destination is defined as End-to-End delay.

Today many applications (e.g. IP telephony) need a small latency to deliver usable results hence this delay is essential, because it shows the aptness of the protocol for these applications. The expected response time delay to send packets from a source node S to a destination node D is the sum over the response times at all links and nodes visited along the way [Haverkort 1998]:

$$E[R(S,D)] = \Sigma E[R_{ij}] + \Sigma E[Ri] \quad (1)$$

Where $E[R_{ij}] = \frac{1}{\mu C_{ij} - \lambda_{ij}}$, is the expected delay at all links, μc_{ij} is the number of packets that can be transmitted over a link ij (packets/sec.), and λ_{ij} is the arrival rate over a link ij.

$E[R_i] = \frac{\rho_i E[S_i]}{1-\rho_i} + E[S_i]$ is the expected delay for node i with $E[Si]=1/\mu_i$ and $\rho_i=\lambda_i/\mu_i$, $i=1,\ldots,N$.

2.4.3 Ant Algorithm

Here we are explaining the agents that are used in the ant algorithm. On a computer the pheromone has been substituted by *artificial stigmergy*, which are the probabilities in the routing tables. Some intelligent agents are introduced to replace the ants for calculating and updating the probabilities. The *forward agents* and the *backward agents* are the two main agents. These agents receive percepts from the environment for doing certain actions. For expressing what action should be selected under which circumstances we define some condition action rules. The current situation is defined by the percept and the stored internal state. A description of both agent types is given in Table 2-1. The condition action rules of the agents are defined in Table 2-2.

Table 2-1 Description of forward and backward agents

Agent Type	***Percepts***	***Actions***	***Goals***	***Environment***
Forward agent	-ID of current node -IDs of neighbor nodes and links that lead to them -routing table of current node	-update memory -remove cycle from memory -determine and go to next node -transform to backward agent	Go to destination node, store the route and the travel times	Network consisting of nodes and link connecting the nodes, only local information available at every node
Backward agent	-ID of current node -IDs of neighbor nodes and links that lead to them	-update routing table -go to next node -kill the agent	Go back to the source node along the stored route of the forward agent and update the routing tables	As forward agent

Table 2-2 Condition action rules of the agents

Agent type	***Condition action rules***
Forward agent	IF current node already exists in memory THEN remove cycle from memory IF current node ≠ destination node THEN update memory And go to next node IF current node = destination node THEN update memory And transform to backward agent
Backward agent	IF current node ≠ source node THEN update routing table AND go to next node IF current node = source node THEN update routing table AND kill the agent

The algorithm can be explained now with the help of depiction of these agents.

2.4.3.1 Forward Ants

A forward ant i.e. forward agent $F_{s\rightarrow d}$ is periodically sent to a randomly chosen destination node *d by* Each node *s* all over the network. The charge of the forward ant is to find out a feasible, low-cost path to the destination and to gather valuable information on its trip.

Working of forward ant at every visited node *k* on the way to the destination node are as follows:

- The forward ant checks its stack memory whether node *k* has already been visited before or not. If it has been visited, there will be a cycle in the ant's path and this cycle will be deleted from the memory. The cycle's nodes are popped from the ant's stack and all the memory about them is destroyed.
- The forward ant updates its memory by adding a new (k,t_k) pair to the memory.
- If node *k* is not the destination node then the forward ant determines the next node to set out by using the probabilities in the row of the routing table of node *k* which represents the destination node. The node where the ant has just come from is clean out to avoid that the ant directly returned to that node. The next link is randomly selected with the generated probabilities. The forward ant goes to the next node along that link.
- If node *k* is the destination node then the forward ant converts to a backward ant $B_{d\rightarrow s}$.. By using the information stored in the routing table each forward ant selects the next hop node. The route is selected by following a random scheme, proportional to the probability of each path and to the local queues status. If selected link is not currently available, the forward ant waits its turn in the low-priority queue of the data packets, where it is served on the basis of a FIFO policy.

2.4.3.2 Backward Ants

The backward ant inherits the memory from the forward ant. The task of the backward ant is to go back to the source node *s* along the same path as the forward agent but in the opposite direction and to update the routing tables on this path.

At every visited node *k* on the way back to the source node a backward agent does the following:

• The backward agent updates the routing table of node *k* by using the travel times stored in its memory.

• If node *k* is not the source node then the backward agent uses its memory to determine the next link of the path back to the source node. The backward agent goes to the next node along that link.

• If node *k* is the source node then the backward agent is killed.

2.4.3.3 Updating Routing Tables

The updation process for routing tables at each node will be done by backward ants using ants trip times. At every visited node *k* on the way back to the source node, a backward ant updates some of the probabilities in the routing table of node *k* by using the travel information in its memory which was collected by the forward ant. The backward ant retraces the path of the forward ant by popping the stack, making in the routing tables at each intermediate node according to the following learning rules:

IF (node was in the path of the ant)

THEN $p^{new}(i) = p^{old}(i) + r\,[1 - p^{old}(i)]$ (2)

ELSE $p^{new}(i) = p^{old}(i) - r\,p^{old}(i)$ (3)

Where $r \in (0,1)$ is the *reinforcement factor* which is central to express path quality. The reinforcement factor should be a factor of trip time of the node neighborhood. This factor is given by the relationship $r=t_1/t_2$ where t_1 is the minimum trip time of all the forward ants, and t_2 is the trip time of the current forward ant from a node to the destination node. $p^{new}(i)$ is the new probability and $p^{old}(i)$ is the old probability for a node *i* in the routing table.

After this updation, the probabilities in routing tables are altered a little bit and then, the delay is recalculated using equation (1). If the new delay is better, it definitely becomes the starting point for the next iteration and each forward ant will move by applying *roulette wheel selection* on the new probabilities after updating. If the delay is worse, each forward ant will move by applying r*oulette wheel selection* on the old probabilities before updating. This has the effect that nearly every new solution is adopted, while over time it becomes more and more likely that only better solutions are accepted.

This comparison between the delay of each successive iteration repeats until the ant timer T_{ant} finishes, then if the simulation timer T_{Simul} hasn't finished yet, a new topology of the network will be established according to the used mobility model and repeats again the same procedure until the simulation timer expires.

3. Simulation and Results

Here we present the performance results of the simulation experiment. The simulation program has been executed on Network Simulator 2. The simulation scenario consists of a number of nodes that are initially placed randomly and constantly moving in a simulation rectangular area 1000×1000

m^2 according to RWM or BSAM models. During simulation, nodes are free to move anywhere within this area. Each node travels toward a random spot, and then takes a pause time in second. After this pause time, the node travels toward another randomly selected spot. This process repeats throughout the simulation, causing continuous changes in the topology of the underlying network, followed by a simulation of the ant behavior yielding an improvement of the routing tables, which is evaluated by Kleinrock´s delay analysis. The model parameters used are summarized in Table 3-1.

Table 3-1 Simulation parameters

Parameter	***Value***
Number of nodes	10,20,….150
Arrival Rate	150Kbps
Transmission Range	250m
Velocity/Direction	10m/sec & 45 degree
Packet size	64byte
Pause time (Ant time)	10,20,….100sec
Simulation Time	180sec
Link Bandwidth	1Mbps
Simulation Area	1000m×1000m
Mobility Model	RWM & BSAM Model
Routing Protocol	Ant Algorithm

3.1 Experimental Results

Here we are showing various experimental results for different scenarios.

3.1.1. Number of nodes vs. mean delay

By remaining the ant time constant at 30 sec, we use different number of nodes (10, 20,…140, 150) and get the relation between increasing the number of nodes and the End to End delay. The result shows that increase in number of nodes achieve an increase in the delay due to congestion and time consuming discovery of routes.

3.1.2. Ant time vs. mean delay

This experiment investigates the relation between increasing the ant time and the End to- End delay in case of using different pause times (10,20, …,100 seconds) and 60 nodes which shows that increasing the pause time leads to a decrease in the delay, because the ant algorithm performs more iterations that help to approach the minimum delay.

3.1.3. Mean delay through the different topologies

We use randomly distributed 60 nodes and an ant time of 60 seconds for this experiment. The whole simulation time is 180 seconds and we observe the packet delays over this time period. The experiment is done with three topologies for finding the mean delay and the result shows that there is no significant effect on the delay between RWM and BSAM models, but the performance can vary extensively with other mobility models.

After a change in the topology, we observe a degradation of the performance (i.e. an increase of the delay) followed by a performance improvement due to the work of the ant algorithm which converges step-by-step toward the minimum possible delay.

4. References

1. E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm Intelligence. Oxford University Press, 1999.s
2. G. Beni and J. Wang. Swarm Intelligence.
3. M. Dorigo. Optimization, Learning and Natural Algorithms (in Italian). Ph.D.thesis, DEI, Politecnico di Milano,
4. A. Tarek and M. Sampels. Optimization of Circular Networks byGenetic Algorithms. Third Middle East Symposium on Simulation and Modelling (3rd MESM'01), Amman, Jordan, 2001
5. D. Johnson, D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. Mobile Computing, edited by Tomasz Imielinski and Hank Korth (Kluwer Academic Publishers), chapter 5.
6. C. Perkins, E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications, 1999.
7. C. E. Perkins. Ad Hoc Networking. Addison-Wesley, ISBN 0-201-30976-9, 2001.
8. T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for AdHoc Network Research, In Wireless Communication and Mobile Computing (WCMC), Vol. 2
9. B.Haverkort Performance of computer communication system, A model based approach, John Wiley & Sons, Ltd. 1998.
10. B. Barán, R. Sosa. A New Approach for AntNet Routing. Ninth International Conference on Computer Communications and Networks IEEE ICCCN- 2000
11. G. DiCaro, M. Dorigo. AntNet: A mobile Agents Approach to Adaptive Routing Technical Report IRIDIA/97-12, Universite Libre de Bruxelles, Belgium, 1997.