Journal of Scientific & Industrial Research
Vol. 82, June 2023, pp. 616-628
DOI: 10.56042/jsir.v82i06.1827

QoS Improvement using Enhanced Manhattan Mobility Model on Proposed Ant Colony Optimization Technique in MANETs

Satveer Kour[1]*, Jagpal Singh Ubhi[2] & Manjit Singh[3]

[1]Deptt. of Computer Engineering and Technology, Guru Nanak Dev University, Amritsar 143 005, India
[2]Deptt. of ECE, Sant Longowal Institute of Engineering & Technology, Longowal 148 106, Sangrur, India
[3]Deptt. of Engineering and Technology, Guru Nanak Dev University Regional Campus, Jalandhar 144 009, India

Received 20 July 2022; revised 28 March 2023; accepted 19 May 2023

In the current Mobile Ad-hoc Network (MANET) route discovery procedure, traffic overflow and overhead may pose a major challenge for a path finding between communicating nodes. Swarm intelligence technique has been applied for various routing problems. We suggest an ant-based bottleneck routing method for MANET to identify weak links in the chosen route for routing overhead and delay issues. During data exchange, it selects the route using a swarm based technique called Ant Colony Optimization (ACO). Initially, we have created node movements by Enhanced Manhattan Mobility Model (EMMM) and then a bottleneck value based ant colony optimization technique is applied. The simulation results show the improvement over existing ACO technique in terms of mobility and pause time. The Quality of Service (QoS) performance metrics showed improvement of 66% in drop rate, 141% in the packet delivery ratio, 42% in packet overhead, 171% in throughput, and 34% in the average end-to-end delay for mobility experiment. There is an improvement of 82% in drop rate, 108% in the packet delivery ratio, 45% in packet overhead, 171% in throughput, and 49% in the average end-to-end delay for a pause time experiment.

Keywords: Bonnmotion-3.0.1, Mobility Models, NS-2.35, Optimization, Routing protocols

Introduction

Ad-hoc networks, which lack any permanent architecture, are examples of self-organizing, quickly deployable, and frequently reconfigurable networks. Without any central management or infrastructure, a MANET is made up of a collection of wireless nodes that communicate with one another via wireless links. All nodes can function as a server and a router to forward data because they are mobile and can instantly form a network on demand.[1] To discover the most efficient routing protocol for extremely dynamic topologies, the performance of MANET routing protocols must be examined at node density, node speeds, traffic nodes, and network size. Mobile nodes engage in direct wireless communication with their close neighbors or use a multi-hop method for farther nodes.[2] Mobility affects ongoing transmissions if the mobile node moves out of range while getting and forwarding packets. MANET nodes have limited communication resources and a wide area of applications.[3] It is challenging to determine the network topology that nodes are using at any given moment for routing due to their mobility. The network performance degrades because of node mobility since it causes high packet loss and delay while searching a new route.[1,4] Nodes powered by low power batteries have a shorter transmission range to nodes closest to them. Since, the environment is unpredictable, wireless medium is unreliable, and the mobility may cause different errors.[5] MANET faces difficult problems with regard to bandwidth, electricity, cost, and security. Vulnerable links, dynamic topologies, threats from malicious nodes, a lack of firm boundaries, the need for a centralized entity, a loss of power, and scalability are some of the wanted problems in MANET. Ad-hoc On-Demand Destination Vector (AODV) is one of a distance vector family along with reactive nature which requests a path only on demand. The routing table carries routes of active communicating nodes only.[6] The Quality of Service (QoS) performance measures are system-related, objective traits that cast illumination on how well the delivery service is performing at the network/transmission level. Here, the network performance is analysed over throughput, packet delivery ratio, average end-to-end delay, total dropped packets, and packet overhead QoS

*Author for Correspondence
E-mail: rattansatveer1985@gmail.com

performance metrics. The natural behaviour of ants is followed in Ant Colony Optimization (ACO) algorithm[7] for discovering the shortest path between foods (as a destination) and a nest (as a source). In reality, ants are perfect in the path finding from a source to destination. During movement, a chemical has been released by an ant, called pheromone on their way. Every ant will follow the route which has the maximum pheromone. An ACO[8,9] is based on the real ant behaviour, using pheromone deposited on the routes. The 'forward' and 'backward' ant agents are sent across the routes for depositing pheromone over the route. It is a highly recommended technique to find the shortest path between nodes. By using this concept of pheromone deposited, the researchers have developed meta heuristic algorithm for a wide set of different problems.

Literature Review

In terms of efficiency, the Enhanced Manhattan Mobility Model (EMMM)[10] outperforms the current Manhattan model. The mobile nodes in a Manhattan model[11,12] move at a mean speed, whereas they move at their utmost speed in an Enhanced Manhattan model. When the roads are clear, a mobile node can move at its maximum speed rather than its mean speed (as stated in the prior model). A mobile node's speed can improve network efficiency if it is consistent. It encourages stabling the link and minimizing link expiry rate.[13] The load balancing is the most crucial issue of the network. It can solve by accurate distribution of network load over the links.[14,15] According to Souihli *et al.*[16], load-balancing mechanisms should direct network congestion away from a simulation area's central location. For MANET routing protocols, a novel routing metric was offered to lessen route centrality. There is a slight rise in routing overhead. A novel protocol called QLB-AOMDV (QoS and Load Balancing-AOMDV) was introduced by Tekaya *et al.*[17] by combining a load balancing mechanism with a multipath routing protocol and some QoS. For QoS metrics, it improves traffic balancing. Energy-related QoS measures, however, were not taken into account. Lin & Shao *et al.*[18] proposed a novel routing algorithm relying on enhanced Ant Colony Optimization (ACO) with coordination of ant colony algorithm characteristics. On choosing ant's next hop, selection criterion was added with the router's buffer queue and link usage to modify the global pheromone; a number of paths were chosen for communication. In applications, each path delay is different and varied immediately. Krishna *et al.*[19] suggested a quality of service enabled Ant Colony Based Multiple Routing (QAMR) algorithm that relies on the ant's propensity for path selection and data transfer. The probability of a route being preferred and node stability are the only factors that affect the path finding process. The hop count, bandwidth, delay, number of intermediate nodes, and preference probability of path are the QoS factors taken into account here. Here, there is also a rise in routing overhead. Yang *et al.*[20] proposed a Network Coding-based AOMDV routing algorithm for MANET to enhance the data transmission reliability as well as load balancing. Network coding was applied at a relay node for encoding the packets. The data and control packets generation is increased along with end-to-end delay. In the previous research, Multipath Backbone Routing for a Latency and Energy Reduction (MBRLER) technique of MANET was introduced to create backbone nodes based on link expiration time, energy of node and the speed of a node. The Ant based Multipath Backbone Routing for Load Balancing (AMBRLB) is an extension to an MBRLER technique based on the path preference probability mechanism which is introduced by Selvi *et al.*[7,9] It is estimated through delay, bandwidth, and next-hop availability. The Beetle-Ant Colony Optimization (Be-ACO) algorithm was developed by Zhoujie *et al.*[21] The model first creates a search subspace for an ant colony using a beetle antenna search strategy and optimization service, which is then used to traverse the subspace according to the ant colony algorithm. Rely on the beetle antenna search strategy repeatedly to create the subsequent search subspace in the global scope for the ant colony to travel through and eventually converge to the global optimum solution. An Improved Ant Colony Optimization (IMVPACO) method was proposed by Ping *et al.*[22] The IMVPACO algorithm modifies the pheromone updating rules and adaptive adjustment strategy to better represent the quality of the solution based on the pheromone increment. For a better balance between solving efficiency and solving quality, and to successfully avoid settling for a local optimum that would hasten convergence, the dynamic evaporation factor approach is used. The ant's movement rules are altered to optimize the route, increase search effectiveness, and adapt to large-scale problem solving.

Our study's goal is to determine a mobile node's speed using a novel mobility model, then optimize the routing protocol by bottlenecked ACO technique that is the improvement of AMBRLB technique. From the above-said related research work, we found some research gaps like movements of mobile nodes in free areas[10–12], variation in speed calculation formula in Manhattan mobility model, the route finding mechanisms in optimization technique based networks. The primary goal of this strategy is to modify the current mobility model's speed calculation phenomenon. Previous research on ACO[9,23] focused on multipath backbone routing, which was employed in MANET for load balancing. This optimization method was based on the probability of the preferred route. It is used to build the backbone nodes, which are then relocated using energy, expiration time, and speed. These nodes occasionally gather information from the nodes next to them. We added to this study by putting forth an Ant Colony optimization technique based on bottleneck routing. The following are the goals of this study:

1. Implementation of a proposed mobility model[10] for node movements.
2. Apply ant based bottleneck routing technique for finding the strong links.
3. By using an ACO bottleneck method, an algorithm has been designed i.e. independent of reactive routing protocols.
4. Comparison of proposed bottleneck ACO with existing AMBRLB technique[7].

This article makes a real-time scenario because some mobile nodes can move at maximum speed rather than mean speed when the road is clear. Moreover, the mobility model is incorporated with proposed ACO technique to make it optimize. By applying the optimization technique, the proposed work will give better results (in means of performance metrics) in MANET environment.

Materials and Methods

In this paper, we have generated movements of mobile nodes which have already been proposed as Enhanced Manhattan Mobility Model (EMMM).[10,13] A mobile nodes can move at a maximum speed when the roads are free. In the previous Manhattan model, the mobile node moves only in the range of minimum and mean speeds. The randomNextDouble() function is provided more uniformed number instead of uniform numbers created by randomNextGaussian() function (used in previous Manhattan Model). The improvement in performance of network also depends upon the smooth speed of mobile nodes. It aids in link stabilization and lowers link expiration rates.

The various possibilities of movements of node V in a 2D plane are shown in Fig. 1. The node has three different probabilities to move left, right, or straight at each intersection position (Left = 0.25, Right = 0.25, and Straight = 0.5) whereas node V is lying at position $\mathrm{V}(\mathrm{X_p}, \mathrm{Y_p})$ now. The mobile node switches positions from its initial location every instant. The formula for the speed is:

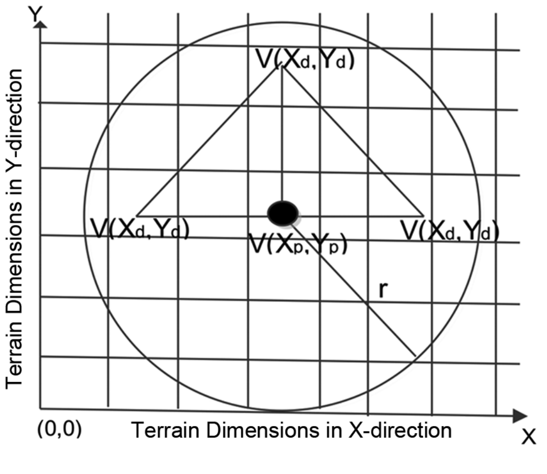
$$\mathrm{speed}\ (\mathrm{V}) = (\mathrm{maxspeed} - \mathrm{minspeed}) \times \mathrm{randomNextDouble}(\,) + \mathrm{minspeed} \qquad \dots (1)$$

where, maxspeed and minspeed are the highest and lowest speeds of a mobile node, and randomNextDouble() is a Java function used to generate a next pseudorandom number, uniformly distributed between the values 0 and 1. Fig. 1 shows the trajectory form of mobile nodes with various possibilities of node movements.[6, 7]

1. In the first option, node V moves the straight line along Y-axis. If speed of node V is known, and radius, r is a transmission range of node, $\mathrm{V}(\mathrm{X_p}, \mathrm{Y_p})$ is the present position of node, $\mathrm{V}(\mathrm{X_d}, \mathrm{Y_d})$ is the destination position chosen by V, then the value of $(\mathrm{X_d}, \mathrm{Y_d})$ is measured as:

$$\mathrm{X_d} = \mathrm{X_p}, \text{ and } \mathrm{Y_d} = \mathrm{Y_p} + \mathrm{speed}\ (\mathrm{V}) \times (\mathrm{t_d} - \mathrm{t_p})$$

Here, speed (V) is the speed of a mobile node, $\mathrm{t_d}$ is the time when it will reach the destination and $\mathrm{t_p}$ is the time when it started to travel.


Y
V(Xd,Yd)
V(Xd,Yd)
V(Xp,Yp)
V(Xd,Yd)
r
Terrain Dimensions in Y-direction
(0,0)
Terrain Dimensions in X-direction
X
Fig. 1 — A type of node's movement in trajectory

2. In the second option, node V gets the right direction along X-axis. Then the value of (X_d, Y_d) is measured as:

$$X_d = X_p + \text{speed}\ (V) \times (t_d - t_p), \text{and}\ Y_d = Y_p \quad \ldots (3)$$

3. In the third option, node V goes to the left direction along X-axis. Then the value of (X_d, Y_d) is measured as:

$$X_d = X_p - \text{speed}\ (V) \times (t_d - t_p), \text{and}\ Y_d = Y_p \quad \ldots (4)$$

Consider a link between the nodes N1 and N2 as an illustration:

N1 might be traveling at a speed of m1 to the place indicated by (X1, Y1) at time t. The position coordinates for N2 traveling at speed m2 are, for example, (X2, Y2). The nodes in MANETs can use GPS to retrieve location information. To node N2, node N1 transmits a request message. They can calculate their distance from one another at this time using the Euclidean algorithm:

$$S(N1, N2) = \sqrt{(X1 - X2)^2 + (Y1 - Y2)^2} \quad \ldots (5)$$

$S(N1, N2)$ is the separation (distance) between nodes $N1$ and $N2$. Let's say that r represents the two nodes' greatest communication range. As a result, the connection between these two nodes is severed when the distance between them exceeds $(r - S(N1, N2))$. Here, r is the maximum communication range of the mobile nodes. When the mobile nodes are moving according to their speeds, their communication range, r, moves with them. And both nodes are moving in the same direction. As a separation goes more than (r- S (N1-N2)), means negative resultant value, the link get broken. In other words, the link breaks, when the separation becomes larger than communication range, r.

The longest period that a link between these nodes can be maintained is because they are moving in the same direction at speeds of m1 and m2, respectively. It can be expressed as:

$$\text{LinkSustainTime(LST)} = \frac{(r - S(N1,N2))}{(m1 - m2)} \quad \ldots (6)$$

Link Sustain Time (LST) is the time up to the link remains sustained. The Enhanced Manhattan Mobility Model[6,7], states that every node travels at the rate specified in equation 1.

As a result, the mobility differential between two nodes, (m1-m2), can be roughly expressed as :

$$(m1 - m2)(\text{maxspeed} - \text{minspeed}) \times (\text{randomNextDouble1}() - \text{randomNextDouble2}()) \quad \ldots (7)$$

where, randomNextDouble1() is the random number generated for a first node, and randomNextDouble2() is the random number generated for a second node. As per Random Waypoint mobility model[10], speed (either m1 or m2) can take any value from $[\text{maxspeed}, \text{minspeed}]$. Say, N1 moves at maxspeed, and N2 moves at minspeed.

Since,

$$(\text{maxspeed} - \text{minspeed}) = (\text{maxspeed} - \text{minspeed}) \times (\text{randomNextDouble1}() - \text{randomNextDouble2}()) \quad \ldots (8)$$

This implies,

$$\text{LST(Enhanced Manhattan Mobility Model)} > LST\text{(Random Waypoint Model)} \quad \ldots (9)$$

This demonstrates that, in comparison to the Random Waypoint mobility model, the use of the Enhanced Manhattan Mobility Model will enable the links between the nodes to be maintained for extended periods of time. Because, in Random Waypoint Model, the existence of randomNextDouble() function is missing during speed calculation. In RWP, the node travels with the speed chosen between 0 and maximum. These generated movements are further collaborated with proposed ACO technique.

Mathematical Model and Working of Ant Colony Optimization

Ant Colony Optimization (ACO) is a technique to find the shortest path between the mobile nodes during the communication. ACO technique is used for QoS improvement by selecting only the strong links instead of packets travelled on vulnerable links. Some other techniques for QoS enhancement are Genetic Algorithms, Particle Swarm Optimization, Numerical methods etc. ACO is the best technique for finding the shortest path between the communicating nodes and in order to increase the effectiveness for convergence time, ACO uses the centralise update method for pheromone updates and only concentrates the search within a neighbourhood of the best solution so far discovered. As considering the mathematical model of ACO, it mainly works in two phases:

1. Pheromone calculation (model and vaporization)
2. Decision to choose a path (on probability based)

For understanding the concept of path-finding, a numerical example is considered here. A graph is mentioned here with 4 vertices and 6 respective edges. The cost of each edge is mentioned on it.

Vertices= {tree, car, tent, pond}

Edges= {tree<->car, tree<->tent, tree<->pond, car<->pond, tent<->pond, tent<->car}

The graph and its corresponding cost matrix are shown in Fig. 2.

If the k^{th} ant travels on the edge (i,j) then the deposited pheromone on respective edge is computed as

$$\Delta\tau_{i,j}^{k} = \begin{Bmatrix} \frac{1}{L^k} & if\ k^{th}\ ant\ travels\ on\ the\ edge\ (i,j) \\ 0 & otherwise \end{Bmatrix} \quad \ldots (10)$$

Here, L is the length of a path. The deposited pheromone by an ant over the link can be non-vaporized or vaporized with respect to time. If the deposited pheromone remains same over the link during the simulation, the equation will be written as

$$\tau_{i,j}^{k} = \sum_{k=0}^{m} \Delta\tau_{i,j}^{k} \quad \ldots (11)$$

Here, k is the ant, and m are the consecutive total ants travelling on the same edge (i,j). According to equation 11, the total pheromone over an edge (i,j) is the pheromone deposited by the ants travelling over that edge.

If the deposited pheromone over the link is vaporizing over time, then the equation will be written as

$$\tau_{i,j}^{k} = (1-\rho)\tau_{i,j} + \sum \Delta\tau_{i,j}^{k} \quad \ldots (12)$$

Here, ρ is a constant that shows the evaporation rate. If ρ is 0, then there is no evaporation of pheromone over the link with respect to time. When ρ is 1, then the evaporation rate is at the maximum level.

By considering the same example; two ants (L1 and L2) are travelling over the network:

Here, an ant (of Blue color), denoted as $L1$, is traveling over the network. $L1$ starts its route from tent to car with cost 4, from car to tree with cost 5, from tree to pond with cost 4, and from pond to tent with cost 1. The total cost, paid by an ant L1 is 14. So, the deposited pheromone over the respective edges by an ant L1 is $\frac{1}{14}$ by using equation 11. The computation is shown in Fig. 3.

Same as, for ant $L2$ (of Green color), this value is $\frac{1}{31}$. $L2$ is traveling from tent to car, car to pond, pond to tree and tree to tent. If the same edge is traveled by the consecutive ants, then the deposited pheromone over the edge is the sum of deposited pheromones by respective ants. As shown in Fig. 4(a), the common edges traveled by $L1$ and $L2$ are: from tent to car $(\frac{1}{14}+\frac{1}{31})$ and tree to pond $(\frac{1}{14}+\frac{1}{31})$. Now, the deposited pheromones over edges are shown in Fig. 4 (a) and its respective computation is depicted in Fig. 4 (b).

The link which has the highest deposited pheromone will be chosen by ant. In this case, the highest value of pheromone is 0.1 which lies from tent to car. So this path will be the most recommendable path by the next coming ant.

Suppose the initial deposited pheromone over edges is equal to 1 (by travelling ants) as shown in Fig. 5(a). If the pheromone is evaporating with time then computed pheromone will be computed by using equation 12. If ρ = 0.5, it means that the 50% of the current pheromone level evaporates every time is depicted in Fig. 5(b). According to the equation 12, the computed values of pheromone with 0.5 evaporation rate is depicted in Fig. 5(c).

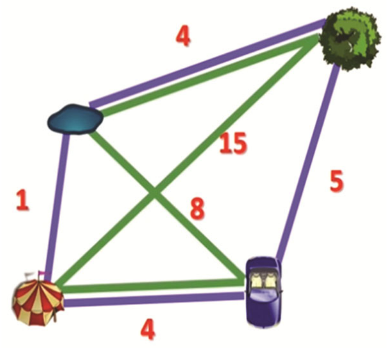


Fig. 3 — Traveling ants (L1 and L2) over the network

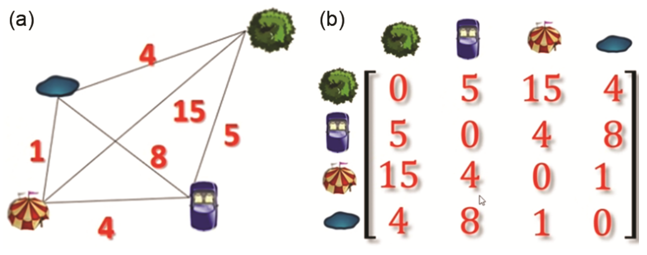


Fig. 2 — (a) Graph (b) Cost matrix

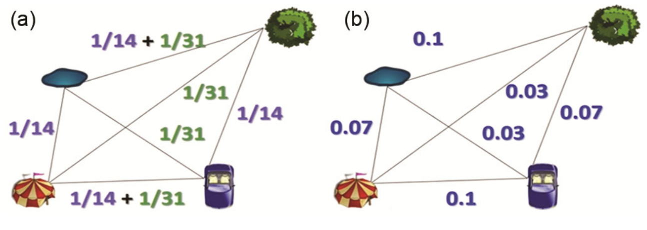


Fig. 4 — (a) Pheromone deposited over links (b) Computed values of pheromone

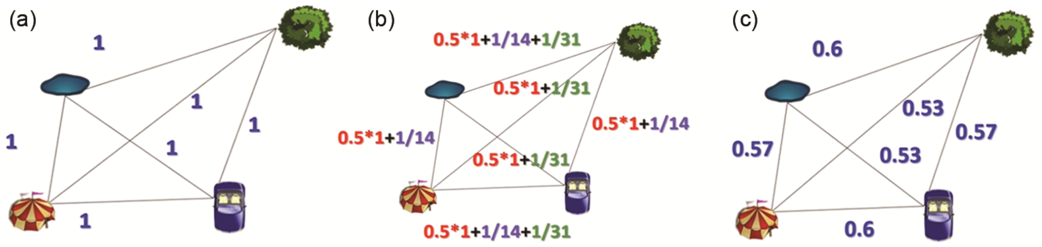


Fig. 5 — (a) Initial pheromone distribution (b) Pheromone level with ρ = 0.5 evaporation rate (c) Computed pheromone values

Now, the probability of choosing a path $P_{i,j}$ (here i,j is the edge) can be calculated as

$$P_{i,j} = \frac{(\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}}{\Sigma\left((\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}\right)} \quad \dots (13)$$

where, $\eta_{i,j} = \frac{1}{L_{i,j}}$ is the quantity of edge (i, j) and $\tau_{i,j}$ is the pheromone level over the edge. The values of constants α, β can be increased or decreased. This probability is used for all the connecting edges connected from the current node, i, and in the number interval of 0 and 1. For choosing a shortest path, this computed probability will be used.

By taking the same example of Fig. 2 (a) and Fig. 5(a), the computed probabilities of path from tent to car, pond and tree is shown in the Fig. 6. So, probabilities of paths by using equation 13 from tent are:

P (tent to pond) = 76%

P (tent to tree) = 5%

P (tent to car) = 19%

If the pheromone deposited over an edge from tent to car is replaced with value 5, then the corresponding calculation of deposited pheromone is shown in Fig. 7 (a–d).

The conclusion from above-said examples is that the path chosen by an ACO technique depends upon the quality of the path as well as the pheromone deposited over the path. As example showed the probability of car is increased from 19% to 54% (by varying the quantity of deposited pheromone).

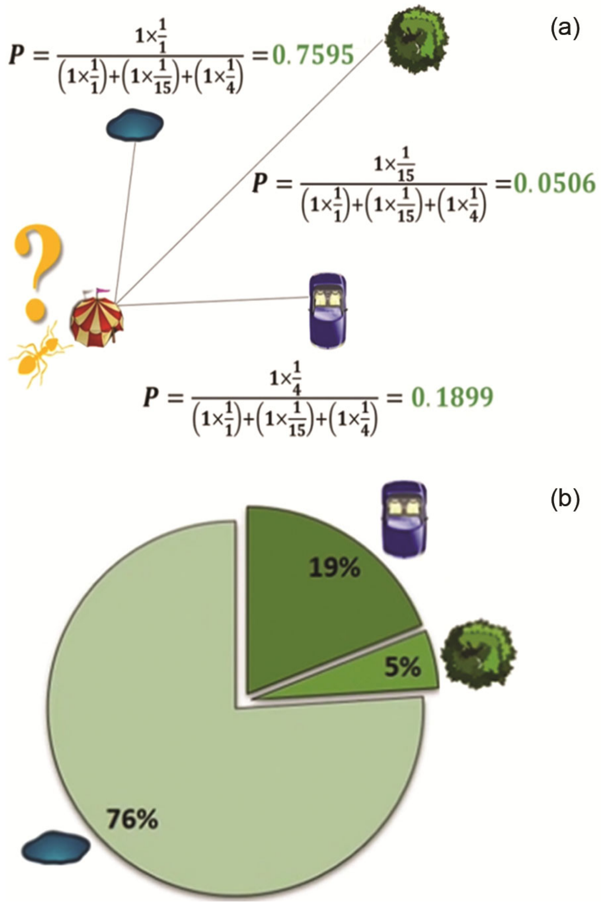

(a)
$P = \frac{1\times\frac{1}{1}}{(1\times\frac{1}{1})+(1\times\frac{1}{15})+(1\times\frac{1}{4})} = 0.7595$
$P = \frac{1\times\frac{1}{15}}{(1\times\frac{1}{1})+(1\times\frac{1}{15})+(1\times\frac{1}{4})} = 0.0506$
$P = \frac{1\times\frac{1}{4}}{(1\times\frac{1}{1})+(1\times\frac{1}{15})+(1\times\frac{1}{4})} = 0.1899$
(b)
19%
5%
76%

Fig. 6 — (a) Pheromone computation (b) Roulette wheel

Proposed Ant Colony Based On Bottleneck Routing (ACBBR)

In the previous research[9], ACO algorithm is based on path preference probability, stability of nodes, and number of hops factors. The multipath backbone routing used in MANET for load balancing was the subject of the ant based multipath backbone routing for load balancing (AMBRLB)[7] study. This optimization method was based on the probability of the preferred route. It is used to build the backbone nodes, which are then relocated using energy, expiration time, and speed. These nodes occasionally gather information from the nodes next to them. The neighbor nodes were proposed using a method. Backbones save all the information they gather in a database called local topology. The ideal routing routes passed by this table. In the proposed work, named as ACBBR, we find two categories of links on the basis of pheromone deposited over the link. When a source forwards a Forward Ant (FANT) message to another node, it deposits a certain amount of pheromone over it. In the starting of route discovery process, the pheromone deposited by ant is:

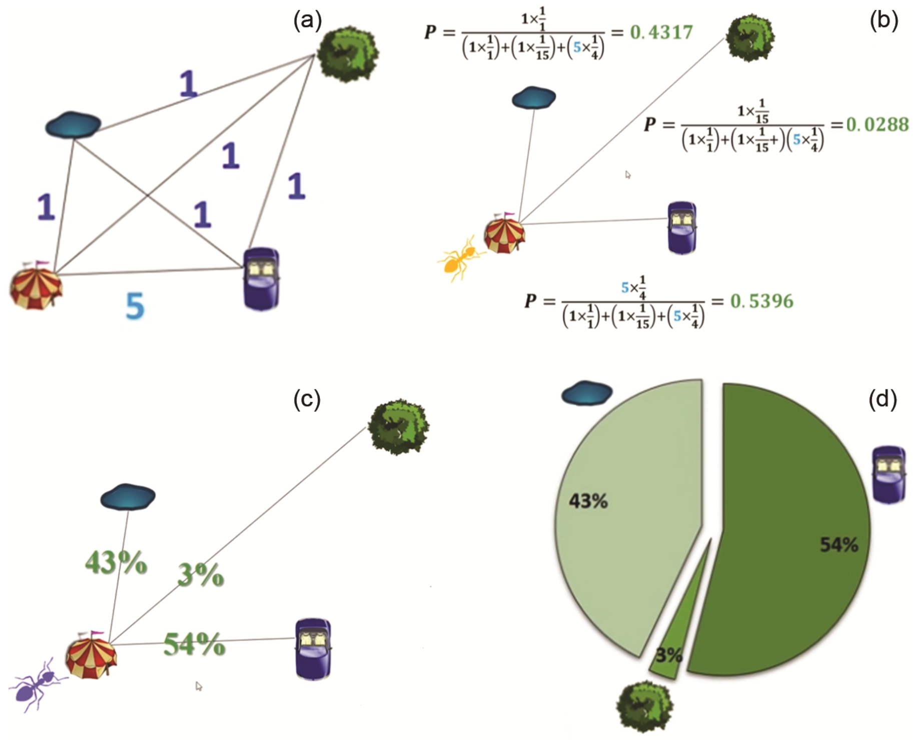

(a)
1
1
1
1
1
5
(b)
$P = \frac{1\times\frac{1}{1}}{(1\times\frac{1}{1})+(1\times\frac{1}{15})+(5\times\frac{1}{4})} = 0.4317$
$P = \frac{1\times\frac{1}{15}}{(1\times\frac{1}{1})+(1\times\frac{1}{15}+)(5\times\frac{1}{4})} = 0.0288$
$P = \frac{5\times\frac{1}{4}}{(1\times\frac{1}{1})+(1\times\frac{1}{15})+(5\times\frac{1}{4})} = 0.5396$
(c)
43%
3%
54%
(d)
43%
54%
3%

Fig. 7 — a) Update pheromone graph b) Probability computation c) Probabilities over edges d) Roulette wheel

$$\text{PheromoneDeposited}\left(\Delta\tau_{i,j}^{k}\right) = \frac{1}{\text{L}} \qquad \dots (14)$$

If PheromoneDeposited $< \frac{\text{Q}}{0.8\text{L}}$, then link falls into the category of a vulnerable link, otherwise link is considered as a strong link. If the pheromone left on link is below 80% of the pheromone deposited, then the link will be a vulnerable link. The deposited pheromone is disseminated over the link continuously. Here, Q is a constant variable with value 1 and the L is the length of a link between two nodes. The receiving party would drop the packet and would not participate in the subsequent broadcasting of the FANT packets if the FANT packet is received over a weak connection, and vice versa. According to its rate of evaporation, all the strong connections will weaken. If β is the pheromone evaporation rate per second varies from 0 to 1, then a strong link will become vulnerable in time:

$$\text{Vulnerable Time (V\_T)} = \beta \times \left(\text{Pheromone Deposited} - \frac{\text{Q}}{0.8\text{L}}\right) \qquad \dots (15)$$

So, VulnerableTime (V_T)is the time when the link goes into a vulnerable state with respect to the time.

While broadcasting a process, all nodes will include this susceptible time in their request packet. All of the original solutions or paths that FANTs used to travel from one node to another are acquired at the destination node. The bottleneck value for each route will be discovered by the destination node as local minima of vulnerable time from each path.

$$\text{BottleneckValue (B\_V)} = \text{Min(VulnerableTime)} \qquad \dots (16)$$

The target will respond to the Backward Ant (BANT) message sent by the source node for all paths where the bottleneck value is higher than the average value of the local minima received. The sender once more spreads a certain quantity of pheromone over the network nodes that are moving during the route return. Pheromone values are first changed at the source node. For the bottleneck values of vulnerable time, the source node discovers global peaks. The best route for sending the data to the target node will be the one for which global maxima are discovered. A link may become a vulnerable link while it is in the data transmission period. If it does, a period of route maintenance will begin. The global maxima of the path are used to find the bottleneck connections. Fig. 8 illustrates the idea of strong and vulnerable connections.

Path Finding Process

The path finding process between the origin and a target is shown in algorithm given below. This algorithm is discovered the strong links on the basis of pheromone

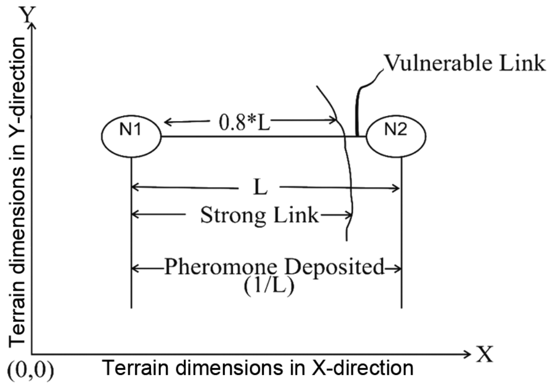


Fig. 8 — Strong and vulnerable links

deposited by ants. For the safety consequences, we take a static value as a threshold value, i.e., 80% of the pheromone. This value of threshold has chosen because this will be the hardest value for selecting a strong link. It can be dynamic by changing pheromone deposited parameter's value in the coding section. If pheromone deposited by ants is below the threshold value, i.e. 80% of pheromone deposited, the link will be considered as a vulnerable link and drop the Forward Ant (FANT) packet sent by the sender. Otherwise, the link will be marked as a strong link. After computing all the strong links, a Link Sustain Time (LST) will be calculated. Then, the vulnerable time and bottleneck value is calculated. On the basis of bottleneck value, route reply procedure is executed. In route reply procedure, a Backward Ant (BANT) packet is sent back to the sender by those links whose bottleneck value is greater than average bottleneck value. The space complexity of the algorithm is the status storage capacity of all the nodes with queue buffer (5 packets storage). The computation time of algorithm depends upon the quantity of FANT messages broadcast over the internet with respect to a simulation time. The complete algorithm is written as below:

Algorithm: Path_Finding_Process ( )

```
N: Node List, Neighbour: Neighbour List
1.  For i=1 to N
2.  For j=1 to Neighbour
3.  Broadcast 'FANT' packet to Neighbour[j].
4.  If Neighbour[j]==Destination
5.  Go to Route_Reply_Process( )
6.  Else
7.  Compute PheromoneDeposited
8.  If PheromoeDeposited<Threshold
9.  Mark Link as vulnerable
10. Drop 'FANT' packet
11. Else
12. Mark link as 'strong link'
13. End If
14. For all the 'strong links' do
15. Compute LST
16. Goto step (3)
17. End For
18. End If
19. End For
20. End For
```

Procedure: Route_Reply_Process( )

```
1.  Fori=1 to P, P indicates the collection of paths
    from a Source to Destination
2.  Find bottleneck_value of LST
3.  End For
4.  Compute average_bottleneck_value
5.  If ottleneck_value>average_bottleneck_value
6.  Send 'BANT' packet to source node
7.  Else
8.  Drop 'BANT' packet
9.  End If
10. Choose path with global maxima for
    bottleneck_value at a source node
11. Send data over the chosen path
12. For all links in chosen path
13. If distance (Ni, Ni+1) > 0.8R
14. Mark a link as a broken link
15. The source fetches another path from cache
    memory
16. Else
17. Continue data transmission
18. End If
19. End For
20. Return
```

Results and Discussion

The NS-2.35 simulator is used to implement the simulation environment in this part. It is running on an Ubuntu 12.04 system with 8GB of RAM and a 1TB hard drive. An application interface has generated for simulation experiment named as AMBRLB[7] and ACBBR. The Bonnmotion-3.0.1 mobility tool generates the node motions. After creating mobility with the Bonnmotion tool,[24] an NSFile is made and added to the NS2 Tcl file. Instead of using the mean speed parameter, the maxspeed parameter is added to the previous model to boost network efficiency. Since some mobile nodes may move faster than the recommended minimum speed when the roads are clear, the change from meanspeed to maxspeed was made in order to establish a real-time trace-based scenario. The evaluated MANET frame design is depicted in Fig. 9. The input parameters are fed

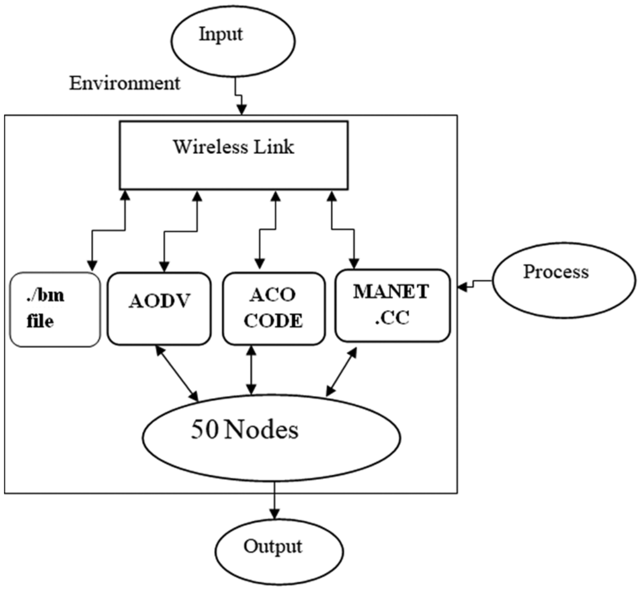


Fig. 9 — Evaluated MANET Frame Design

into the experiment through input phase. The wireless routing protocol is set over the NS2.35 by value 802.11b. The ./bm file is created by Bonnmotion tool for creating the node movements. The complete embedded file is named as MANET.cc file with a set AODV routing protocol and ACO code. The external process involves running of file using ns command. The final outcome is shown in output phase. The initial node formation and their movements during simulation are shown in Fig. 10.

Some of the properties of scenario are covered in Table 1. The packet size is of 512 bytes. Here, packet size, mobility speed, pause time, and duration time are treated as an input and the QoS performance metrics are treated as an output.

QoS Performance Metrics

QoS in MANET is universally a growing area. According to the definition of QoS, the main

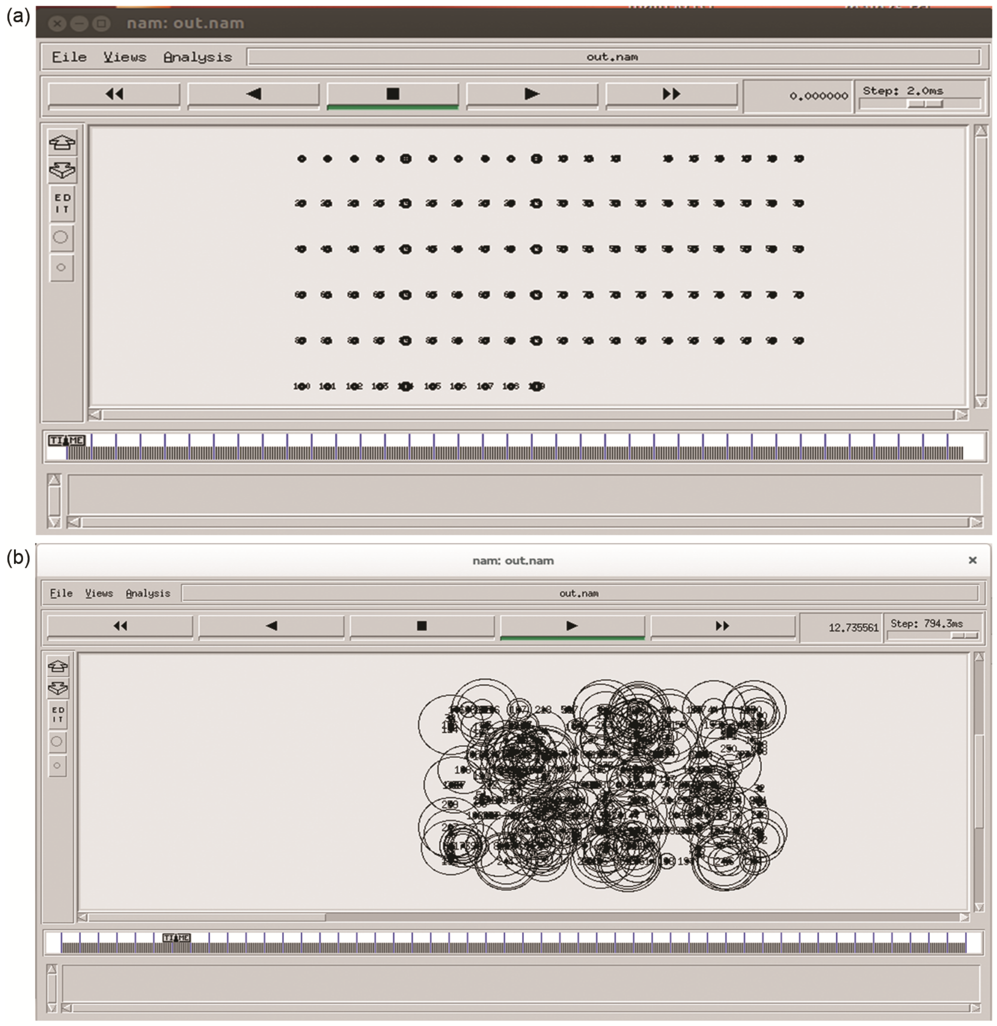


Fig. 10 — (a) Initial node formation (b) Nodes movements during simulation

Table 1 — Simulation parameters[6,7]

Experiment variable	Variable value
Simulator	NS-2.35
Mobility generator	Bonnmotion-3.0.1
Simulation time	50 seconds
Terrain dimension	1200 ×1200 m^2
No. Of mobile nodes	50
Mobility speed	[5–10, 10–20, 20–40, 40–60, 60–80] meter per second
Pause time	1,2,3,4,5 seconds
Application traffic	User datagram protocol (UDP), Constant bit rate (CBR)
Mobility model	Enhanced manhattan
Routing protocol	AODV, DSDV, DSR
Mac protocol	802.11
Channel	WirelessChannel
Propagation method	TwoRayGround
Antenna	OmniAntenna
Queue	DropTail/PriQueue
Queue length	5

objective is to give networks and organizations the ability to prioritize traffic. This is accomplished by providing dedicated bandwidth, managed jitter, and reduced latency. Wide-Area Networks (WANs), service provider networks, and corporate applications can all benefit from the technologies used to make this possible. Organizations can use QoS in networking to improve the performance of various apps running on their network and gain insight into its bit rate, delay, jitter, and packet rate. By doing so, they can control network traffic and alter how packets are transmitted to the internet or other networks to prevent transmission delays. Additionally, this guarantees that the company provides services with the anticipated level of quality for applications.[25]

Here, proposed optimized algorithm is compared with existing ACO[7] on the basis of Packet Delivery Ratio (PDR), throughput, dropped packets, average end-to-end delay, and packet overhead QoS performance metrics.

Throughput: During the simulation time, the ratio of packets received is called throughput.[26]

Packet Delivery Ratio (PDR): Packet Delivery Ratio (PDR) indicates the data packets received over data packets sent.

Average End-to-End Delay: It is the time taken by a data packet to reach to target node from the origin.

Total Dropped Packets: The total packets dropped because of unavailability of paths and collisions.

Packet Overhead: The total of data and control packets is known as packet overhead is the packet overhead.

The real enhancement of the Enhanced Manhattan Mobility model over Manhattan model is the smooth movement provided by a randomNextDouble() java function which makes it more realistic. Moreover, maximum speed can be gained by moving vehicles in case of free roads. We have compared the existing AMBRLB technique with a proposed ACBBR technique in two experiments: mobility and pause time. In the first section of each experiment, the various mobility speeds of mobile nodes are divided into five categories: 5–10 m/s, 10–20 m/s, 20–40 m/s, 40–60 m/s, and 60–80 m/s. The common pause period for all the mobility category experiments is 1 second. The first mobility-related experiment, shown in Fig. 11 (a–e), compares the performance of AMBRLB and ACBBR at various mobility rates in terms of drop packets, PDR, throughput, packet overhead, and average end-to-end delay. We have plotted the average value of each ACO method that corresponds to mobility and pause time in each figure. In Fig. 11(a) a 65% reduction in packet loss rate when comparing ACBBR to AMBRLB is recorded. The rationale behind this is to always choose the strong connection during data transmission over the vulnerable link. As per Fig. 11(b) the PDR of ACBBR over the existing AMBRLB algorithm demonstrates an improvement of 114%. The smooth motions produced by EMMM enhance the PDR. The ACBBR algorithm's 171% throughput increase over the AMBRLB algorithm is illustrated in Fig. 11(c). Using the randomNextDouble() java method increases throughput. A built-in function called randomNextDouble() generates a random number between 0 and 1. The foundation of EMMM's speed creation is the number this function generates. The network's total performance is impacted by the smooth movements produced by EMMM. Due to the reactive nature of the AODV routing algorithm, the packet overhead ratio of ACBBR is 39% to 42% lower than AMBRLB in Fig. 11(d). The average end-to-end delay of ACBBR is 24% to 34% longer than AMBRLB, which correlates to mobility. Sending packets over only strong connections reduce the typical end-to-end delay is depicted in Fig. 11(e).

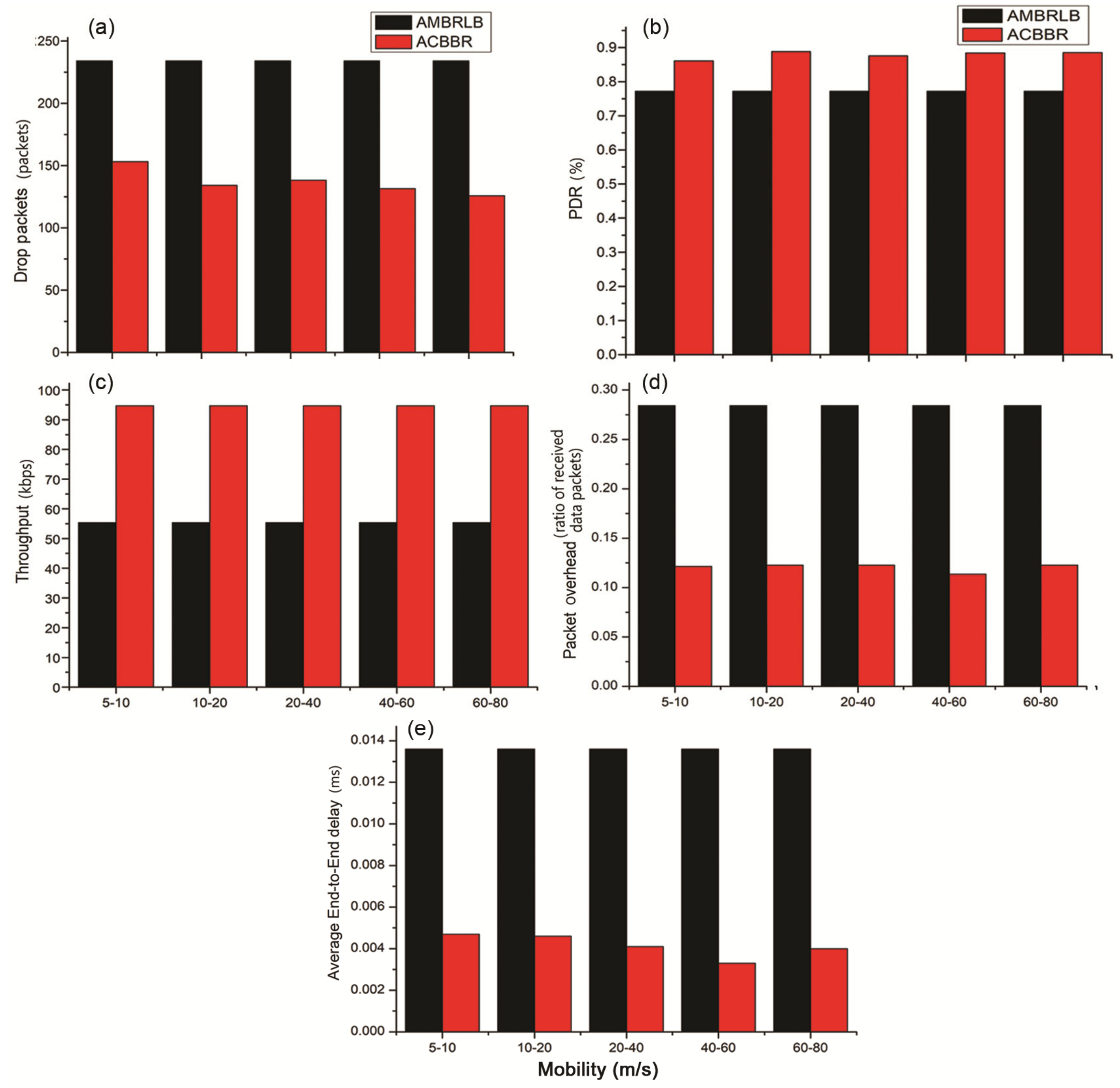


Fig. 11 (a) — Mobility vs. Drop packets, (b) Mobility vs. Packet delivery ratio, (c) Mobility vs. Throughput, (d) Mobility vs. Packet overhead, (e) Mobility vs. Average end-to-end delay

We have taken into account pause times of 1, 2, 3, 4, and 5 seconds in our second trial. Upon reaching the destination, the node will stop for the pause duration (in seconds). The drop packets, PDR, throughput, packet overhead, and average end-to-end latency for the ACBBR and AMBRLB algorithms, respectively, are shown in Fig. 12 (a–e). Due to the smooth movements produced by EMMM, Fig. 12(a) demonstrates a 59% to 82% improvement in the drop rate of packets in ACBBR compared to AMBRLB. Similarly, Fig. 12 (b) shows an enhancement of 108% in PDR. As per Fig. 12(c) there is 171% improvement of throughput in case of ACBBR than AMBRLB because of the uniform number created by randomNextDouble() function. According to Fig. 12(d) there is 41% to 45% less routing overhead in ACBBR than AMBRLB because it concentrates on a single link only rather than finding multiple routes. Every time, a backward ant message travels on to strong links only. According to Fig. 12(e), ACBBR performs 30% to 49% better than AMBRLB for a typical end-to-end delay. The packets follow the link's bottlenecked value, which again relies on the minimum vulnerable time, so the average end-to-end delay is lower.

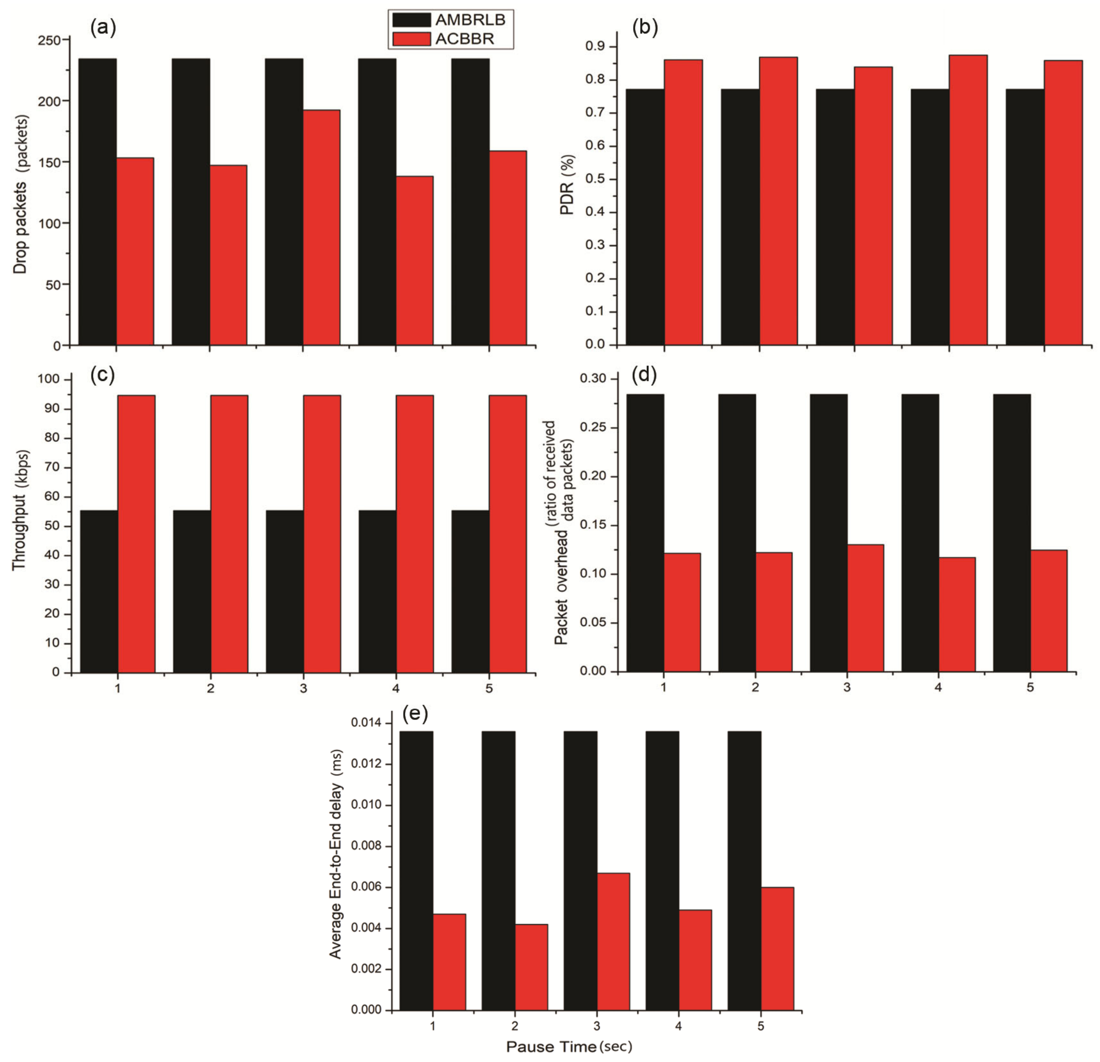


Fig. 12 (a) — Pause time vs. Drop packets, (b) Pause time vs. Packet delivery ratio, (c) Pause time vs. Throughput, (d) Pause time vs. Packet overhead, (e) Pause time vs. Average end-to-end delay

Conclusions and Future Challenges

In this article, we suggest a bottleneck routing algorithm for strong links in MANET based on ant colonies. During data transmission, the originator uses the ant colony optimization method to choose the route based on the amount of pheromone remaining and the bottleneck value of the link. We have taken into account pheromone left behind, link length, vulnerable time, and link bottleneck value when determining a path. The local minima of vulnerable time from each route will be found by the destination node and will serve as each path's bottleneck value. The nodes once more spread out a certain quantity of pheromone during the path reply. Pheromone values are first changed at the source node. For the bottleneck values of vulnerable time, the source node discovers global peaks. The optimal route for sending data to the target node will be determined by the path for global maxima. The maximum speed can be gained by moving vehicles in case of free roads. We have shown that the proposed technique improves the performance of the QoS performance metrics. This technique will be useful in defence mechanisms where highly reliable links are needed for communication. We may also apply this technique on multipath ACO algorithms and try to balance the network load by using this technique. An empirical method to select the threshold value will due for future work.

Acknowledgements

We are thankful to Guru Nanak Dev University, Amritsar and Sant Longowal Institute of Engineering and Technology, Longowal for providing us with infrastructure and facilities to do this research.

Conflict of Interest

The authors have no conflict of interest to declare.

References

1 Eissa T, Razak S A, Khokhar R H & Samian N, Trust based routing mechanism in MANET: design and implementation, *Mob Netws Appl,* **18** (2013) 666–677.

2 Das S R, Castañeda R & Yan J, Simulation-based performance evaluation of routing protocols for mobile ad hoc networks, *Mob Netw Appl,* **5** (2000) 179–189.

3 Zhang Y & Ren K, On address privacy in mobile Ad Hoc networks, *Mob Netw Appl,* **14** (2009) 188–197.

4 Pentikousis K, Agüero R, Sargento S & Aguiar R L, Mobility and network management in virtualized networks, *Mob Netw Appl,* **17** (2012) 431–434.

5 Yu Z, Yu Z & Chen Y, Multi-hop mobility prediction, *Mob Netw Appl,* **21** (2016) 367–371.

6 Kaur S & Ubhi J S, A simplified approach to analyze routing protocols in dynamic MANET environment, *Int J Soft Comput Eng*, **5** (2015) 19–23.

7 Selvi P F A & Manikandan M S K, Ant based multipath backbone routing for load balancing in MANET, *IET Commun*, **11** (2017) 136–141.

8 Nallusamy C & Sabari A, Particle swarm based resource optimized geographic routing for improved network lifetime in MANET, *Mob Netw Appl,* (2017) 1–11.

9 Selvi P F A & Manikandan M S K, Latency and energy aware backbone routing protocol for MANET, *J Theor Appl Inf Technol,* **64** (2014) 63–73.

10 Kour S & Ubhi J S, A novel approach to predict mobility pattern of mobile nodes in mobile Ad-hoc networks, *J Sci Ind Res*, **77** (2018) 629–632.

11 Roy R R, *Group mobility extending individual mobility models-handbook of mobile Ad hoc networks for mobility models* (Springer, Boston, MA) 2011.

12 Martyna J, Simulation study of the mobility models for the wireless mobile Ad hoc and sensor networks, in *Computer Networks: 19th Int Conf* (Springer Berlin Heidelberg) 2012, 324–333

13 Kour S & Ubhi J S, Performance Analysis of mobile nodes in mobile ad-hoc networks using enhanced manhattan mobility model, *J Sci Ind Res*, **78** (2019) 69–72.

14 Sangwan S, Goel N & Jnagra A, AELB: Adaptive and efficient load balancing schemes to achieve fair routing in Mobile Ad hoc Networks (MANETs), *Int J Comput Sci Techol,* **2** (2011) 11–15.

15 Kaur R, Dhillon R S & Sohal H S, Load balancing of ant based algorithm in MANET, *Int J Comput Sci Techol,* **1** (2010) 173–178.

16 Souihli O, Frikha M & Hamouda M B, Load-balancing in MANET shortest-path routing protocols, *Ad Hoc Netw,* **7(2)** (2009) 431–442.

17 Tekaya M, Tabbane N & Tabbane S, Multipath routing with load balancing and QoS in Ad hoc network, *Int J Comput Sci Netw Secur,* 10 (2010) 280–286.

18 Lin N & Shao Z, Improved ant colony algorithm for multipath routing algorithm research, *Int Symp Intel Info Proces Trusted Comput* (IEEE) 2010, 651–655.

19 Krishna P V, Saritha V & Vedha G, Quality-of-service-enabled ant colony-based multipath routing for mobile ad hoc networks, *IET Commun,* **6** (2010) 76–83.

20 Yang F, Ling S & Xu H, Network coding-based AOMDV routing in MANET, *IEEE Int Conf Info Sci Technol* (Wuhan, Hubei, China) 2012, 337–340.

21 Zhoujie D & Huaikou M, An optimization method based on Be-ACO algorithm in service composition context, *Comput Intell Neurosci*, (2022) 1–10. https://doi.org/10.1155/2022/5231262.

22 Ping D & Yong A I, Research on an improved ant colony optimization algorithm and its application, *Int J Hybrid Inf Technol*, **9** (2016) 223–234. http://dx.doi.org/10.14257/ijhit.2016.9.4.20

23 Karimpour R, Khayyambashi M R & Movahhedinia N, Applying ant colony optimization for load balancing on grid, *J Chin Inst Eng,* **39** (2015) 49–56.

24 Aschenbruck, N, Ernst R, Gerhards-Padilla E & Schwamborn M, BonnMotion - A mobility scenario generation and analysis tool. SIMUTools, in *3rd International ICST Conference on Simulation Tools and Techniques* 2010, 16 May. 10.4108/ICST.SIMUTOOLS2010.8684.

25 https://www.fortinet.com/resources/cyberglossary/qos-quality-of-service (Accessed, July 2022)

26 Johnson D & Lysko A, Comparison of MANET routing protocols using a scaled indoor wireless grid, *Mob Net Appl*, **13** (2008) 82–96.