Improving AODV Routing Protocol For Mobile Ad-Hoc Networks Using Swarm-Based Algorithms

Clodomir J. Santana Jr.[1] and E inst1

[1] University of Pernambuco, Recife-PE, Brazil
Email: cjsj@ecomp.poli.br, emilly.alves@poli.br, emnf@ecomp.poli.br, mgmm@ecomp.poli.br, pjbls@ecomp.poli.br, carmelofilho@ieee.org
[2] Federal University of Technology - Parana, Ponta Grossa-PR, Brazil
Email: hugosiqueira@utfpr.edu.br
[3] Illinois State University, Normal-IL, USA
Email: aagokhale@ilstu.edu

Abstract. This paper investigates the application of bio-inspired Swarm-based algorithms - Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) - to improve the capability of the Ad Hoc On-Demand Distance Vector (AODV) routing protocol to recover broken connections that occur due to the mobility of nodes in Mobile ad-hoc networks (MANETs). We use the algorithms as mentioned earlier to calculate the weight of a function based on the number of hops traveled by the package, the number of hops to reach the destiny, source node connectivity and the predecessor node connectivity. We compare the obtained results with the standard version of the AODV. The results show that the proposed approach with both ABC and PSO algorithms are capable of achieving better results than the conventional AODV protocol concerning routing overhead, delay, and packet delivery ratio.

Keywords: MANETs, Swarm Algorithms, AODV

1 Introduction

Ad hoc networks are communication systems in which there is no infrastructure in the environment to distribute the tasks or to realize the communications between the distributed nodes of the network. In this case, the coordination of the communication is decentralized and distributed among the nodes [15] [20].

Nowadays, we observe the improvement of the devices which communicate via wireless technology. It increases the portability and flexibility of this kind of platform, allowing the development of many types of equipment, such as mobile sensors [16]. Mobile ad-hoc networks (MANETs) were developed to implement the two former ideas. The possible applications are several: ambient monitoring, security monitoring, military proposes, among others [4].

The main characteristic of these networks is the information is transmitted through multiple hops between a source node - from where it comes up - to the destination node. However, if the nodes are allowed to move, their spatial location may change dynamically and arbitrarily. In this case, the routes previously defined can be interrupted, which

creates the necessity to repair the communication from the source to destiny nodes. To solve this task, some protocols have been proposed [19] [2]. The protocols are divided in three main categories [5] [1] [16]:

i) proactive: these protocols regularly evaluate the routes sending control packages to the nodes. In this case, the routes are currently known and available. This characteristic avoid communication breakdown but implicates in an additional overhead to the network and elevate energy expenditure;

ii) reactive: this kind of protocol creates routes just when the source node needs to communicates to other. After the end of the process, a route is established, and the protocol starts its maintenance procedure. Since these nodes do not send control packages, they spend less energy than the table-driven ones and do not increase the communication capability of the network. However, the latency to send messages increases, occasioning delays;

iii) hybrid: this kind of protocols was developed using the main characteristics of those as mentioned earlier, being adequate to ad hoc networks whose behavior changes regularly. The nodes define a service area called zone according to their transmission power rating. Then, the nodes work as a table-driven inside their zone (intra-zone), and if a route request is not found, the protocol initiates a discovery process out of the zone (inter-zone) on-demand.

Indeed, during the maintenance of the routes, the protocols have to perceive the failure and to decide which is the best recovery mechanism to be used to restore the routes [15]. In this sense, one of the most popular algorithms is the Ad Hoc On-Demand Distance Vector (AODV) algorithm [19], which utilizes the connectivity concept to decide where is better to proceed repair when a link break occurs, i.e., on the source node or in the predecessor node regarding the failure. It is an on-demand method, which means that a new route is discovered when a source node needs to send data to a destination node [4].

In 2012, Pereira et al. [16] used the particle swarm optimization (PSO) approach [12] to determine the coefficients of a new cost function elaborated based on four elements of the network: packet Forward, source Connectivity, predecessor Hop Count and predecessor Connectivity. This approach was named AODV-PSO.

In addition, we consider (PSO) [12] [14], to generate the appropriate composition of weights for all considered parameters in AODV route recovery, to improve the choice in local repair decision aiming to reduce data packet delivery delay [6].

This paper proposes extending the work from Pereira et al. [16]. We evaluate the performance of the PSO and the Artificial Bee Colony (ABC) [10] [11] [8], aiming to optimize the four coefficients of their proposal. We compare the results to the standard AODV.

The remainder of the article is organized as follow: Section II presents the AODV routing protocol and those proposed in [16]; Section III discusses the optimization methods - PSO and ABC; Section IV shows the computational results obtained through simulations. In Section V, we present the conclusions.

2 Ad Hoc On-Demand Distance Vector (AODV)

Ad Hoc On-Demand Distance Vector (AODV) is an on-demand algorithm which the primary goal is to self-adapt quickly and dynamically to the variations of the conditions of the links of the networks. It is an important issue since these communication links can break due to a change in the position of the nodes [3] [7].

New routes must be discovered when source nodes need to communicate with destination nodes using route request messages (RREQs). The source spreads RREQs and waits for a predetermined time to receive a route reply (RREP) confirming the route. At the moment the node receives a RREQ, it creates or updates the input in the routing table of the predecessor node. If this node does not present a valid sequence number, this indicates that the route is not available. Next, the node checks if it already received a requisition containing the same broadcast identifier and the corresponding address of the source. If this happens, the RREQ message is discarded because it was already spread to its neighbors. In case the node receives a new requisition but it is not the destination and do not have a valid route to the destination, the hop counter contained in the RREQ is incremented, and the route requisition is forwarded to its neighbors [16] [15].

While the requisition is transmitted in the network, many backward routes between the source and the intermediary nodes are stored. Thus, having the possibility to reach the destiny, the network has the information how the answer it has to return to the source. To make it possible, every intermediary node which propagates a RREQ store the address of the neighbor which received the first RREQ in an input or table. In this table, we stored the sequence number of the source and the expiration time of the backward route. An intermediary node also can respond to a route requisition if it has in its routing table one active input to the destiny presetting sequence number higher or equal to the last known by the source. Before sending the RREP, the destination node updates its sequence number to the maximum between its current value and the value that was in the RREQ [4].

In the AODV algorithm, there are two mechanisms of recovery: from the source and locally. The recovery is needed when occurs a break in a link which was being used. When a link break is detected, this information is communicated to the other nodes which use this link in some active route by a route error message (RERR) or a local repair is done [17] [18] [5].

In the source repair mechanism, the error message sent by the predecessor node to break (RERR) have to be disseminated throughout the network to reach all the nodes which use the broken link. Then, each node which receives the RERR decides if it continues or not to pass the message. Besides, it has to update its routing table, recording if the specified destination in the message is unreached and then refresh the sequence number of the nodes [5].

In the local repair mechanism, when a link break occurs in an active route, the predecessor node to break increase the sequence number and spread the RREQ aiming to find the destination node. The local repair attempt is imperceptible to the source node, and it continues sending the data packages which are stored in the predecessor node regarding the failure until the desired destination node are found. If we exceed the time limit to find a new route, we transmit a RERR message, and the packages are discarded. While the data stored predecessor node to break do not exceed this limit,

they can be delivered if a new route is found. When it happens, delays and package losses may occur. But, if the repairing node receives a RREP, it ensures lower overhead and delays [15].

2.1 Route recovery using the connectivity concept

The most known approach implemented in the AODV uses both source and local repair. The decision regarding the method to be used depends on the number of hops involved on the path to the destination. However, the number of neighbors of the source and predecessor nodes to repair the failure can indicate the possibility of finding another good route.

Pereira et al. [16] utilize this assumption to propose a new decision function to route recovery using the connectivity concept to improve the performance of the AODV algorithm. Firstly, they define the connectivity concept as the number of neighbors in the transmission range of the node. Every time a node sends a RREQ, it includes its connectivity in the packet so that every node that receives the request, stores or updates the source node connectivity information before forwarding or replying the RREQ. When the link between nodes breaks, then we update the node connectivity. When a node sends a packet using a route, it sends information regarding the connectivity as well. Therefore, every node in this route can update the source node connectivity information. In this case, the authors mention the necessity to get the connectivity of the predecessor and source nodes to elaborate the decision function. This function associates weights to the number of hops on the path to the destination and with the connectivity.

In algorithm 1 we present the steps to perform it. The variables $source$ and $local$ are functions of the number of hops on the path to the destination and their connectivities. The coefficients, A, B, C and D are real numbers defined in the interval [-1;+1] and weighted the influence on the choice between one and other repair action [16].

Algorithm 1 Route recovery using the connectivity concept

1: *source = (A × packet Forward) + (B × sourceConnectivity)*
2: *local = (C × predecessorHopCount) + (D × predecessorConnectivity)*
3: **if** (source $<=$ local) **then**
4: Local Repair;
5: **else**
6: Source Repair;
7: **end if**

The last step of the algorithm is to define suitable values to the weights A, B, C and D. In the original work from Pereira et al. [16], they used the Particle Swarm Optimization (PSO), a bio-inspired swarm-based methodology for optimization. In the next Section, we present the PSO and the Artificial Bee Colony Algorithm (ABC) to solve this task.

3 Swarm-Based Algorithms for Optimization

3.1 Particle Swarm Optimization

In 1995, Kennedy and Eberhart [12] developed one of the most known representatives of the bio-inspired swarm-based algorithms: the Particle swarm optimization (PSO). The technique was developed to tackle optimization problems with continuous variables. The socio interactions observed in animals swarms, such as birds flocks and fish schools were the biological inspiration for these algorithms.

In the PSO algorithm, a particle represents a candidate solution which changes its position moving in the search space according to their individual experience and the experience of the others members of the group. This idea allows a complex emergent global performance from simple individual behavior [14].

We highlight the primary step in the development and application of the PSO is to define a fitness function as the performance index. In the case of optimization problems, it is necessary to investigate which is the best proposition to perform search processes better [12].

Considering a swarm of particles (swarm) with size *P*. Each particle *i* is in the *D*-dimension space presents current position $\mathbf{x}_p = (x_{p,1}, x_{p,2}, ..., x_{p,D})$ and velocity $\mathbf{v}_p = (v_{p,1}, v_{p,2}, ..., v_{p,D})$. In this section the specification of the dimensions are omitted to simplify the equations. The movement of a particle *p* occurs according Equation 1.

$$\mathbf{x}_p^{t+1} = \mathbf{x}_p^t + \mathbf{v}_p^{t+1} \tag{1}$$

The velocity is updated according to Equation 2.

$$\begin{aligned} \mathbf{v}_p^{t+1} = \omega \mathbf{v}_p^t + \mathbf{c_1}\mathbf{rand_1} \otimes (\mathbf{pbest}_p^t - \mathbf{x}_t^p) + \\ \mathbf{c_2}\mathbf{rand_2} \otimes (\mathbf{gbest}_p^t - \mathbf{x}_p^t) \end{aligned} \tag{2}$$

where *t* denotes the current iteration of particle $p = 1, ..., P$, ω is the inertia weight, $\mathbf{rand_1}$ and $\mathbf{rand_2}$ are random vector in which the dimensions are generated uniformly in the interval [0,1], $\mathbf{c_1}$ and $\mathbf{c_2}$ are the cognitive and social rates respectively; $\mathbf{pbest}_p^t$ is the best position found by the current particle *p* until time *t* (the best individual experience or the position that led to the best performance index) and $\mathbf{gbest}_p^t$ is the best position found by a predefined neighbors.

The initialization of the particles positions is done randomly and follow a uniform distribution. The same process is applied to generate the initial velocities, but it is possible to initiate them as zero. The velocity must be in a range $[-v_{max}, +v_{max}]$. The inertia weight ω is used to controlling the particles' explosion [14].

Each particle's position $\mathbf{x}_p$ is codified as a set of possible values to the weights A, B, C and D in its structure, as showed in Equation 3.

$$\mathbf{x}_p = (A_p, B_p, C_p, D_p) \tag{3}$$

Therefore, the particles are composed of four vectors.

As mentioned previously, it is necessary to define an evaluation criterion to perform *gbest* and *pbest*. For each simulated scenario, the fitness function is the average data packet delivery delay for each network, which has to be minimized.

Algorithm 2 shows the steps to implement the PSO algorithm [12].

Algorithm 2 PSO Pseudocode.

Initialize all particle's position $\mathbf{x}_p$ with a uniformly distributed random vector ($rand[0, 1]$)
Initialize the particle's best known position to its initial position
Set ω, ϕ_1 and ϕ_2
Be $f(\mathbf{x}_p)$ the fitness function of the particle p;
if $f(\mathbf{x}_p) < \mathbf{gbest}$ **then**
    update $pbest$ the swarm's best known position: $\mathbf{gbest} = \mathbf{x}_p$;
end if
Initialize the particle's velocity;
while a termination criterion is not met **do**
    for each particle $p = 1, ..., N$ **do**
        Generate the numbers $rand_1$ and $rand_2$ with a uniformly distributed random vector ($rand[0, 1]$)
        Update the particle's velocity using Equation 2
        Update the particle's position by Equation 1
        Update particle's fitness $f(\mathbf{x}_p)$
        if $f(\mathbf{x}_p) < \mathbf{pbest}$ **then**
            Update the particle's best known position: $\mathbf{pbest} = \mathbf{x}_p$
        end if
        if $f(\mathbf{x}_p) < \mathbf{gbest}$ **then**
            Update the swarm's best known position: $\mathbf{gbest} = \mathbf{x}_p$
        end if
    end for
end while

3.2 Artificial Bee Colony

In 2005, Karaboga introduced the Artificial Bee Colony algorithm (ABC) [10], which was developed for optimization of continuous problems. The biological inspiration is based on the foraging process of honey bees. The ABC has an essential difference to the majority of the swarm-based algorithms: the candidate solutions are the food sources instead of the agents (bees) [11].

Here, a swarm or a hive is a set of artificial bees which aim to found the best food sources. This is a metaphor to the optimum points of an objective function. In this sense, the quality of the source is measured by as nectar amount, directly related to the fitness associated with each source.

According to Karaboga [11]. The swarm divides the bees into three groups: employed, onlookers and scouts. There are SN employed bees, and each one explores each food source ($\mathbf{x}_i, i = 1, ..., SN$). During the iterative process, used go to their respective food source, come back to the hive and perform a dance. Onlooker bees wait

for the employed in the dance area to decide which source they will explore depending on the dance.

Onlooker and employed bees choose food sources depending on the experience of themselves and their nest mates. On the other hand, scout bees perform random searches movements aiming to discover new unknown food sources.

Sometimes, a food source is abandoned if their nectar quantity became low. In this case, the respectively employed bee becomes a scout and starts to search for finding a new one.

ABC algorithm combines local search methods, developed by employed and onlooker bees, with global search potential, performed by onlookers and scouts, giving to the algorithm exploration and exploitation capability.

The algorithm is initialized generating a group of SN food sources randomly within the range of the boundaries of the variables. Equation 4 presents this procedure (we highlight the dimensions).

$$x_{i,d} = x_{d_{min}} + rand(0,1)(x_{d_{max}} - x_{d_{min}}) \tag{4}$$

where $i = 1, .., SN$ is the number of food sources, $d = 1, ..., D$ are the dimension of the problem (the number of parameters to be optimized), $rand(0,1)$ is a random number generated according to a uniform distribution in the interval [0;1], ($x_{d_{max}}$ and $x_{d_{min}}$) are the lower and upper bounds of the d-th parameter (dimension).

Then, each employed bee has to go to its food source and finds a new one in its neighborhood, as shown in Equation 5.

$$v_{i,d} = x_{i,d} + rand(-1,1).(x_{i,d} - x_{j,d}) \tag{5}$$

being $j, j \neq i$ a randomly selected food source and $d = 1, ..., D$ a random selected dimension (parameter), $rand(-1,1)$ a random number generated following to a uniform distribution in the interval [-1;1]. One can observe that the new food source v is found changing just one dimension of x. This operation works as a local search.

After v is determined, then a greedy selection is applied to v, and x and those with better fitness value are the assumed to be the current source.

Next, the onlooker bees are updated. They receive information about the quality of the food sources shared by the employed bees and choose one of them to explore. The probability of selecting a source is proportional to its fitness (amount of nectar). This probability is calculated using Equation 6.

$$p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_j} \tag{6}$$

For each food source $i = 1, ..., SN$, the onlooker draw a number $r = rand(0,1)$. For each $r < p_i$ the onlooker bee explores the corresponding source $\mathbf{x}_i$. Then, it calculates of a new source by 5 and performs the greedy selection.

The last stage of the ABC is the scout bee's phase. It just occurs if some food source is exhausted. This means that some employed bee exceeds the maximum number of tries

($limit$) to improve the fitness of a source. In this case, a new food source is generated using 4 and $limit$ is set as zero.

After the complete cycle, the best solution is always stored.

The Algorithm 3 shows all the steps to follow [11].

Algorithm 3 ABC Pseudocode.

Initialize randomly all food sources positions $\mathbf{x}_i^0$, according to the same scheme of Equation 1;
Evaluate the nectar amount (fitness) of them;
Send the employed bees to food sources;
while stop criterion is not reached **do**
    -Employed Bees Phase
    for each employed bee $i = 1, ..., SN$ **do**
        Find a new food source in its neighborhood using Equation 5;
        Evaluate the nectar amount (fitness) of the new food source;
        Compare the fitness of the original food source and the new one and choose those which presents the best fitness (greedy selection);
    end for
    Calculate the probability p_i for each food source utilizing Equation 6;
    -Onlooker Bees Phase
    for each onlooker bee **do**
        for each food source i **do do** Draw $r = rand(0, 1)$
            if $r < p_i$ **then**
                Send the current onlooker bee to the i-th food sources;
                Find a new food source in its neighborhood using Equation 5;
                Compare the fitness of the original food source and the new one and choose those which presents the best fitness (greedy selection);
            end if
        end for
    end for
    -Scout Bees Phase
    if any employed bee $\mathbf{x}_i$ abandoned its food source **then**
        Find a new food source randomly produced using scout bee and Equation 4;
        Send the employed bee $\mathbf{x}_i$ to the new food source;
    end if
    Memorize the best solution achieved so far
end while
Output the best solution achieved

4 Experiments and Results

4.1 Network Simulation Environment

We chose the network simulator NS-2, version 2.35 [9] to perform the simulations using the AOVD proposals analyzed in this paper. NS-2 is a discrete event simulator developed to be used in network analysis. The platform is suitable to deal with TCP protocol,

routing, multicast protocols to wireless and wired network [13]. The NS-2 are widely used by many researchers since it is an open source tool. This characteristic becomes essential if the user needs to insert some change in the source code.

To analyze the performance of the proposals using the AODV protocols, we set a network containing 50 mobile nodes distributed in an area with 1500m x 300m. The transmission range was set to 250m, and the duration of each simulation was 900s and pausing time of 30 seconds. The radio propagation model was the Two-Ray Ground, and The physical (PHY) and medium access (MAC) layers follow the IEEE 802.11 at a bit rate of 2 Mbits/s.

It is important to observe that the data and the routing packets sent by the network layer are put in a row at the interface queue until the MAC layer can transmit them. Each node has a queue for packets waiting for transmission by the network interface, which holds up to 50 packets. The maximum speed (V_{max}) is 20 m/s, and the traffic pattern is CBR with the following parameters: 30 traffic sources, packet rate of 4 packets/s with a packet size of 512 bytes. We used the User Datagram Protocol (UDP).

4.2 Performance Evaluation

To perform the PSO and ABC, we used 20 particles (or bees) to determine the values of the weights A, B, C and D, which minimizes the Routing Overhead. The search space is [-1, 1] and was defined after previous experiments. The number of iterations was set as 200, resulting in 4000 calls to the simulator per algorithm. The simulation scenario was executed 10 times and we executed 200 iterations in each case. A single execution takes 6 days to be completed. Additionally, previous validations were necessary to ensure the accuracy of the code, and experiments lasted over a month to be completed.

To assess the performance of the algorithms, the following metrics were used:

- End-to-end Delay (Delay): It is the average time that a data packet takes to arrive at the destination. This includes all additional delays caused by forwarding, queuing, route discovery and route repair processes. It is calculated using Equation 7.

$$delay = \frac{\sum_{i=1}^{NR} ArriveTime - SendTime}{\sum_{i=1}^{N} NumberOfConnections} \tag{7}$$

  Where N is the number of nodes, NR is the number of received data packets, $SendTime$ is the time when the packet was sent, $ArriveTime$ is the time in which the packet was received and $NumberOfConnections$ is the number of active connections.
- Routing Overhead (RO): It is the sum of all routing and control packets used during network communication.
- Normalized Routing Overhead (NRL): It is the ratio between the sum of all routing and control packets and the sum of all packets sent (data, control and routing packets).
- Packet Delivery Ratio (PDR): It is the ratio of successfully received data packets and the total of sent data packets.

Table 1. Performance Comparison considering the Mean Values and the Standard Deviation.

Algorithm	Delay	RO	NRL	PDR
AODV	1214.0560 (0.0)	224307.0 (0.0)	2.8446 (0.0)	73.6990 (0.0)
ABC	704.4434 (323.5753)	173484.2 (28494.15)	2.02275 (0.4912)	80.9588 (4.0348)
PSO	608.3896 (21.5576)	164566.3 (1921.564)	1.8686 (0.0214)	82.2246 (0.0319)

As can be seen in Table 1, we expected to minimize the routing overhead resulting in the improvement in the rest of the metrics. This can be explained by the fact that when a network is full of routing and control packets, data packets tend to get dropped or take longer to get to the destination.

Both PSO and ABC presented better results than the traditional AODV. When comparing the two swarm approach, the Wilcoxon test was applied considering a significance level of 0.05, and the result shows that there is no statistical difference between the PSO and the ABC population. This might be due to the low complexity of the problem. In fact, during the experiments was observed the both algorithms converged with around one hundred iterations.

In Table 2, we observe a variation in the coefficients. The experiments showed that the same result could be obtained using different coefficients values. This is an indication that this is a multimodal problem and the algorithms were capable of finding multiple solutions (different local optima).

Regarding the values found for the coefficients, as Table 2 illustrates for both algorithms, the weights of the *source* equation (A and B) are greater if compared to their correspondent on the *local* equation. In other words, the source repair coefficients have more weight which makes us conclude that AODV obtains better results when the source repair is performed more frequently than the local repair. This behavior was also observed in [18]. The *AODV-Source* achieved better overhead results than the *AODV-Local* in a similar scenario with 30 nodes with the velocity of 20 m/s and environment size of 1500 by 300 meters. This may occur because when the local repair fails, the predecessor node drops all cached packets which should be sent via the broken route. Moreover, when the network topology changes constantly, one route can be broken into several points, and when this occurs, one source repair is more efficient than several local repair attempts.

5 Conclusions

The computational results show that using a swarm-based algorithm is possible to improve the performance of the AODV routing protocol regarding routing overhead, end-to-end delay, and packet delivery ratio. Both metaheuristics used in this work were able to find a set of values to the coefficients which improve the performance of the standard

Table 2. Best Coefficient Values Found by ABC and PSO in Each Execution.

Algorithm	A	B	C	D
ABC	0.5086	-0.5995	0.1534	-0.7524
	-0.3069	0.9975	-0.9942	1.0
	0.7989	0.8289	-0.5289	-0.1556
	-0.3069	0.9975	-0.9942	1.0
	-0.5047	0.3388	-1.0	-0.5122
	1.8562	0.7877	-0.9783	-0.5471
	0.4656	0.2470	-0.9651	-0.6262
	-0.5047	0.3388	-1.0	-0.5122
	1.8562	0.7877	-0.9783	-0.5471
	1.8562	0.7877	-0.9783	-0.5471
PSO	-0.1130	0.9664	-0.8479	0.0558
	-0.5047	0.3388	-1.0	-0.5122
	0.0335	-0.1985	-0.7352	-0.7070
	-0.2187	0.3984	-0.6421	-0.4665
	-0.1698	-0.5439	-0.5302	-0.8053
	1.8562	0.7877	-0.9783	-0.5471
	-0.5047	0.3388	-1.0	-0.5122
	1.8562	0.7877	-0.9783	-0.5471
	1.8562	0.7877	-0.9783	-0.5471
	-0.1698	-0.5439	-0.5302	-0.8053

AODV. However, since PSO algorithm has lower computational cost than ABC and uses only one fitness evaluation per iteration, while ABC uses two, we can conclude that the PSO represents a more appropriated solution if the usage of computational resources is taken into account.

As future work, we intend to compare the algorithms using different environment sizes, the number of nodes, the velocity of the nodes and pausing times. We also plan to include new metrics such as energy efficiency to achieve better solutions. We will apply a multi-objective algorithm to obtain the best trade-off considering all the considered metrics.

References

1. A review of current routing protocols for ad hoc mobile wireless networks. IEEE Personal Communications 6(2), 46–55 (July 1999)
2. Dsr: the dynamic source routing protocol for multihop wireless ad hoc networks. Ad Hoc Networking pp. 139–172 (2001)
3. On-demand multipath distance vector routing in ad hoc networks. In: in Proc. International Conference on Network Procotols (ICNP) (2001)
4. Ad hoc on-demand distance vector routing. Request for Comments:3561 8(1) (July 2003)
5. The effects of local repair schemes in aodv - based ad hoc networks. IEICE Transactions on Communications E87-B(9), 2456–2466 (July 2004)

6. Label switched multi-path forwarding in wireless ad-hoc networks. In: Proc. of IEEE International Conference on Pervasive Computing and Communications Workshops (2005)
7. Local repair mechanisms for on-demand routing in mobile ad hoc networks. In: Proc. of Pacific Rim International Symposium on Dependable Computing (2005)
8. Boussad, I., Lepagnot, J., Siarry, P.: A survey on optimization metaheuristics. Information Sciences 237, 82–117 (2013)
9. Issariyakul, T., Hossain, E.: Introduction to Network Simulator NS2. Springer Publishing Company, Incorporated, 1st edn. (2010)
10. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report, Erciyes University, Engineering Faculty, Computer Engineering Department TR06 (2005)
11. Karaboga, D., Basturkl, B.: A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm. Journal of Global Optimization 39(3)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Network. vol. IV, p. 19421948 (1995)
13. Kubinidze, Ninoand Ganchev, I., O'Droma, M.: Network Simulator NS2: Shortcomings, Potential Development and Enhancement Strategies
14. Van der Merwe, D., Engelbrecht, A.: Data clustering using particle swarm optimization. In: Proceedings of IEEE Congress on Evolutionary Computation. pp. 215–220 (2003)
15. Murthy, C.S.R., Manoj, B.S.: Ad Hoc Wireless Networks: Architectures and Protocols. Prentice Hall (2006)
16. Pereira, N.C.V.N., Bastos-Filho, C.J.A., de Moraes, R.M.: Improving aodv route recovery mechanisms with connectivity and particle swarm optimization. Journal of Communications and Information Systems 1(27), 15–21 (2012)
17. Pereira, N.C.V.N., de Moraes, R.M.: A comparative analysis of aodv route recovery mechanisms in wireless ad hoc networks. In: IEEE Latin-American Conference on Communications (LatinCom). pp. 1–6 (May 2009)
18. Pereira, N.C.V.N., de Moraes, R.M.: Comparative analysis of aodv route recovery mechanisms in wireless ad hoc networks. IEEE Latin America Transactions 8(4), 385–393 (2010)
19. Perkins, C.E., Belding-Royer, E.M.: Ad hoc on-demand distance vector routing. In: Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications. pp. 095–100 (1999)
20. Perkins, C.E., E. M. Belding-Royer, S.R.D., Marina, M.K.: Performance comparison of two on-demand routing protocols for ad hoc networks wireless networks. IEEE Personal Communication 8(1), 16–28 (2001)