Volume 3, Issue 3, March 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Characteristics Study of Ant Colony Optimization Algorithms for Routing Problems

Bhanu Pratap Singh[1], Sohan Garg[2]
[1]Research Scholar Mewar University-Ghaziabad
[2]Professor-IIMT Management College-Meerut

Abstract: Routing algorithms are often difficult to be formalized into mathematics, they are instead tested using extensive simulation [1]. Early work on volatile network environments experienced in mobile ad hoc networks (MANETs) depends primarily on applying the traditional approaches of routing in wired networks, such as distance vector or link state algorithms. While many optimizations to these algorithms exist, each of them is primarily concerned with finding the minimum hop route from source to destination [2,3,4]. A large amount of work has also been done in the area of energy efficient routing. This approach attempts to maximize network lifetime by routing through paths, which use the least amount of energy relative to each node [5]. In this research paper we will study all ant based optimized routing algorithms to solve routing problems.

Keywords:ACO, ABC, MABR, MANETs,UTC, GPSAL.

I. INTRODUCTION

Recently, more attention has been paid to use specific network parameters when specifying routing metrics. Examples might include delay of the network, link capacity, link stability or identifying low mobility nodes. These schemes are generally based on previous work, which is then enhanced with the new metrics.

In the last years, a great deal of literature has been published in the field of mobile ad hoc networks. There exists relatively little work with regard to biologically inspired algorithms for routing in communications networks. However, there are a number of notable examples which show that these concepts can provide a significant performance gain over traditional approaches. In this chapter, we describe a number of the most famous algorithms in this field and give an overview of using ACO algorithms in routing problems. This paper is intended to serve as an exhaustive guide to all possible and available literature that applying ACO algorithms to solve routing problems, see Figure 1.

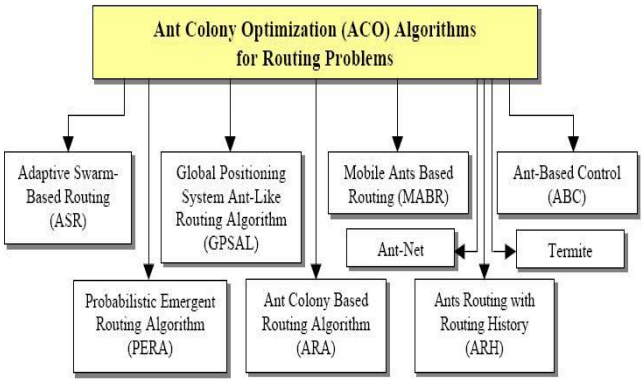

Figure 1. Ant colony optimization algorithms for routing problems

A. Ant-Based Control Algorithm:

One of the earliest work on swarm intelligent routing is done by [6] on the *Ant Based Control* (ABC) algorithm. This algorithm is applied for a wired circuit switched network, such as a telephony network. ABC was later modeled analytically in [7]. The algorithm is adaptive and exhibits robustness under various network conditions. This is accomplished by using many simple agents, called ants, which modify the routing policy at every node in a telephone network by depositing a virtual

pheromone trail on routing table entries. The ants' goal is to build routing tables, and adapt them to load changes in the telephone network so that performance is optimized. Performance here is measured by the rate of incoming calls, which are accepted. A phone call is either accepted or rejected at setup time depending on the available network resources.

Each node has a routing table, where each row represents a destination and each column represents a neighbor. The ants are launched from the sources to various random destinations and eliminated once they reach their destinations. Therefore, the probabilities of the routing tables are updated as the ant visits the node, based on the lifetime of the ant *T* at the time of the visit.

II. ANT NET ALGORITHM

DiCaro and Dorigo[4] introduced an adaptive routing algorithm based on the adaptive learning and ant colonies called AntNet, for routing in packet-switching networks [1,2,4]. This algorithm explores the network with the goal of building (and rebuilding) routing tables and keeping them adapted to traffic conditions. It is presented as a distributed, scalable and responsive algorithm for routing in wired networks. The algorithm is improved upon in [8].

In the *AntNet* algorithm, routing is determined by means of very complex interactions of *forward* and *backward* network exploration agents "ants". The idea behind this sub-division of agents is to allow the backward ants to utilize the useful information gathered by the forward ants on their trip from source to destination. Based on this principle, no node routing updates are performed by the forward ants. Their only purpose in life is to report network delay conditions to the backward ants, in the form of trip times between each network node. The backward ants inherit this raw data and use it to update the routing table of each node. The entries of the routing table are probabilities, where the sum of each row in this table must equal 1. These probabilities serve a dual purpose:

- The exploration agents of the network use them to decide the next hop to a destination, randomly selecting among All candidates based on the routing table probabilities for a specific destination.
- The data packets deterministically select the path with the highest probability for the next hop.

The sequence of actions in AntNet is simple and intuitive:

- Each network node launches forward ants to all destinations in regular time intervals.
- The ant finds a path to the destination randomly based on the current routing tables.
- The forward ant creates a stack, pushing in trip times for every node as that node is reached
- When the destination is reached, the backward ant inherits the stack.
- The backward ant pops the stack entries and follows the path in reverse.
- The routing tables of each visited node are updated based on the trip times.

A. Updating Routing Tables:

The update of the routing table is done by using the quantity, *r'*, which derived according to:

$$r' = \begin{cases} \frac{T}{C\mu}, C \geq 1, if \frac{T}{C\mu} < 1 \\ 1, \qquad Otherwise \end{cases}$$

where *T* is the trip time from the current node to the destination, μ is average of *T*, and *C* is a scaling factor, usually set to 2.

Except for the routing table, each node also possesses a table with records of the mean and variance of the trip time to every destination. The ratio of the variance to the mean (σ /μ) is used as a measure of the consistency of the trip times, and accordingly, it alters the effect of the trip time on the routing table. Based on the value of *r'*, we determine the relative goodness of the trip time of an ant. Corresponding strategies of either decreasing or increasing the value of *r'* by a certain amount are then followed, based on setting the threshold for the good/bad trip time to 0.5, and selecting a threshold for the (σ /μ) ratio .

The principle of these updates is that small values of *r'* correspond to small values of *T* and vice versa. By testing, the case in which the consistency is high and the time is good, the algorithm requires the processed value *r'* to be even smaller. Therefore, an exponential quantity is subtracted. This quantity is the exponentially decaying function of the consistency ratio, and it achieves its highest value when the variance is very small. The decay rate can be controlled through parameters *a'* and *a*. Further, positive or negative reinforcement of good or bad routes takes place next, via *positive* and *negative feedback*. Any positive reinforcement of probability should be negatively proportional to current probabilities, and any negative one should be proportional to current probabilities. The effect of this is to prevent saturation of the routing table probabilities to be 0 or 1. The node that receives the positive reinforcement is the one from which the backward ant comes. This is the same node chosen by the forward ant as next-hop on the way to its destination. All the other neighbors of the current node need to be negatively reinforced to preserve the unit sum of all the next-hop probabilities. The reinforcement equations are:

where *Pdf*, *Pdn* are the previous routing table probabilities, *f* is the node from which the backward ant comes, *Nk* is the neighbor of node *k* (current node), and *d* is the destination node. The last step is to update the routing table probabilities using the following rules.

$$P_{df} = P_{df} + r+ \qquad (1)$$

$$P_{dn} = P_{dn} + r- \qquad (2)$$

The packets of the network then use these probabilities in a deterministic way, choosing as next hop the one with the highest probability. The behavior of the AntNet algorithm can be more easily understood with an example, therefore, we will discuss in detail an example for this algorithm.

III. MOBILE ANTS BASED ROUTING

Mobile Ants Based Routing (MABR) algorithm is inspired by social insects and introduced as the first routing algorithm for large-scale mobile ad hoc networks MANETs [10].

$$r_+ = (1 - r') \cdot (1 - P_{df})$$

$$r_- = -(1 - r')P_{dn}, n \neq f, n \in N_k$$

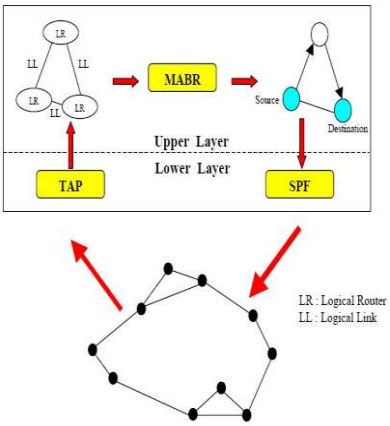

Figure 2. The interactions between TAP, MABR and SPF protocols

The approach presented in AntNet is extended to ad hoc networks by abstracting the dynamic, irregular topology of a MANET into a topology with *logical routers* and *logical links*, this is done by using TAP (Topology Abstracting Protocol). The term logical router represents a collection of mobile hosts, which all together build and share among each other the same routing tables. A logical link represents a path along a roughly straight line to a distant logical router over possible multiple hops. To build these logical routers, nodes geographically close to each other are grouped together. Logical links are established between selected logical routers. An optimized greedy routing algorithm is used to forward messages between logical nodes. On top of this abstract topology, the actual routing protocol MABR (Mobile Ants Based Routing) will be run. This ants-based protocol is responsible for updating the routing tables of logical routers and determining logical paths for routing packets over this abstract topology. Finally, the SPF (Straight Packet Forwarding) protocol is applied in order to transmit packets over a logical link. Therefore, it forwards packets along this logical link in a greedy manner. An overview of th.e architecture with the interactions between the protocols is depicted in Figure2.

IV. ANT COLONY BASED ROUTING ALGORITHM

This Ant-Colony Based Routing (ARA) algorithm presents a detailed routing scheme for MANETs [11]. It is similarly constructed as many other routing approaches and consists of three phases.

A. Route Discovery:

Route discovery is responsible for creating new routes. The creation of new routes requires the use of a *forward ant* (FANT) which is initiated at the source and a *backward ant* (BANT) which is initiated at the destination. The FANT is a small packet with a unique sequence number. A forward ant is broadcasted by the sender and will be relayed by the neighbors of the sender as shown in Figure 3.

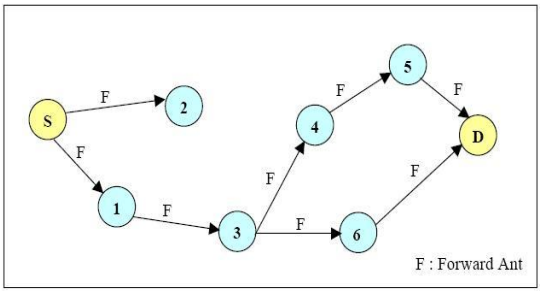

Figure 3. Route discovery phase by forward ant

A node receiving a FANT for the first time, creates a record in its routing table. A record in the routing table consists of (*destination address, next hop, pheromone value*). The node interprets the source address of the FANT as destination address, the address of the previous node as the next hop and computes the pheromone value depending on the number of hops the FANT needed to reach the node. The node forwards the FANT to its neighbors. If the FANT reaches the destination, a BANT will be returned to the source. The BANT will take the same path as the FANT but in the opposite direction as indicated in

Figure 4. When the sender receives the BANT from the destination node, the path is established and data packets can be sent. Figures 3 and 4 schematically depict the route discovery phase.

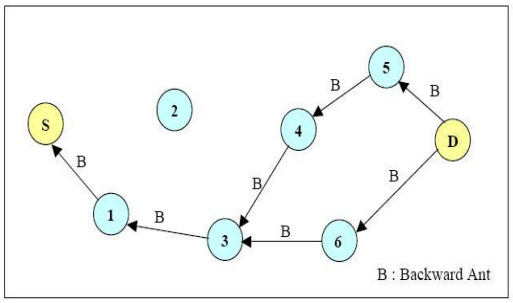

Figure 4. Route discovery phase by backward ant

B. Route Maintenance:

Route maintenance is responsible for the improvement of the routes during the communication. ARA does not need any special packets for route maintenance. Once the FANT and BANT have established the pheromone tracks for the source and destination nodes, subsequent data packets are used to maintain the path. Similar to the nature, established paths do not keep their initial pheromone values forever. A high decay rate will quickly reduce the pheromone value. If a regular time interval of one second is taken, the decreasing formula will be $\phi i,j = (1- q).\ \phi i,j$, where $\phi i,j$ is the pheromone concentration and $q \in (0,1]$. Every time data passed by the pheromone value in the routing table will be increased by $\phi i,j = \phi i,j + \Delta\phi$ where $\Delta\phi$ is a constant amount of pheromone. ARA prevents loops by a very simple method, which is also used during the route discovery phase. Duplicated FANTs can be identified through the unique sequence number and are removed. If a node receives a duplicated packet, it will set the DUPLICATE_ERROR flag and send the packet back to the previous node. The previous node deactivates the link to this node so that packets cannot be sent anymore to this direction.

C. Route Failure Handling:

Route failure handling is responsible for handling routing failures that caused especially through node mobility which is very common in mobile ad hoc networks. ARA recognizes a route failure through a missing acknowledgment. The ROUTE_ERROR is set when the node misses an acknowledgment. When this occurs, first, the link will be disabled and then it will check if there exists any other route to the destination node. If no route exists to the destination, it will inform its neighbors, hoping that they can forward the packet toward the destination. If none of the neighbors can forward the packet toward the destination, the packet is sent back to previous nodes until the source has been reached. Control bandwidth overhead is minimized and remains constant across several degrees of the network volatility.

V. TERMITE

Termite is a distributed routing algorithm for mobile wireless ad hoc networks [Roth and Wicker, 2003]. It is designed using the swarm intelligence framework in order to achieve better adaptivity, lower control overhead and lower per-node computation. The algorithm is inspired by the hill building behavior of termites. The Termite algorithm may be described simply as follows, each node in the network has a specific pheromone scent. As packets move through the network links between nodes, they are biased toward moving in the direction of destination pheromone gradients. Packets follow this gradient while laying pheromone on the links that they traverse toward their source. The amount of pheromone deposited by a packet on a link is equal to the utility of its traversed path. Using this method of pheromone updating, consistent pheromone trails are built through the network. Changes in the network environment, such as topological or path quality changes, are accounted for by allowing pheromone to decay over time. This requires paths to be continuously reinforced by new traffic, new information about the network is added to links. Each node records the amount of pheromone that exists for each destination on each of its links. This creates a routing table similar to those found in traditional link-state routing algorithms.

***A**. Pheromone Update:*

The pheromone update equation is a function which updates the pheromone of the packet source at a node upon its arrival. The update function shown here is the traditional approach and is known as the *Pheromone Filter*. Pheromone on all links decays simultaneously upon packet arrival, and proportionally to packet interarrival times. This is known as continuous pheromone decay and was originally presented in [10]. This idea is extended in this work also by decaying pheromone when it is checked to send a packet. Each packet maintains the total utility of the path it has traversed (updated at each node visited)

and deposits this amount of pheromone on each link. The *Pheromone Filter* update equation is,

Where $P_i^n s$ is the amount of pheromone from source node *s*, on the link from neighbor node *i*, at node *n*. The previous hop of the packet is node *r*. The variable γ is the amount of pheromone carried by the packet, which will vary from packet to packet depending on its path. The time $t_s n obs$ is the last instant that the pheromone from source node *s* at node *n* was observed, either due to packet sending or packet receiving. τ is the pheromone decay rate.

B. Forwarding Equation:

The forwarding equation is used to determine the probability of using a link, based on the amount of pheromone on it.

$$\forall i \quad P_{i,s}^n = P_{i,s}^n \cdot e^{-(t-t_{i,obs}^n)\tau} \quad \text{and} \quad P_{r,s}^n = P_{r,s}^n + \gamma$$

$$P_{i,d}^n = \frac{(P_{i,d}^n + K)^F}{\sum_{j=1}^{N_n}(P_{j,d}^n + K)^F}$$

Where *Pi,d* is the probability of using neighbor node *I* in order to get to destination node *d*, at node *n*. N_n is the number of neighbors of node *n*. $K \geq 0$ is the pheromone threshold and $F \geq 0$ is the pheromone sensitivity.

C. Termite Summary:

Termite has been shown to perform well, especially in regions of high node mobility [11]. However, packets often take substantially longer paths than necessary ones. In part, this is required in order to maintain current pheromone through the network, but this behavior can also generate significant resource inefficiencies. AntNet is a related algorithm, although its forward/backward ant feature is not used. Termite and ARA share nearly the same data routing mechanism, although they differ significantly with regards to route discovery and failure recovery. Ants Routing with routing History (ARH) algorithm is a stochastic routing algorithm that chooses a suitable route and efficiently acquires the information of a route, when the nodes communicate man-to-man in ad hoc communication environments [12]. ARH algorithm considers each node as agent, and copes with the dynamic environments by learning the route. Additionally, ARH can perform a robust routing by selecting stochastically the good route, and efficiently acquire the information of a route by using routing history. It is based on two algorithms Ants-Routing and AntNet that improve learning efficiency by recording routing history in a message packet. It is based on two main processes, selecting next route and updating processes.

VII. ANTS ROUTING WITH ROUTING HISTORY & NO RETURN RULE

Ants-Routing with routing History and no return rule (ARHnr) is a modified version of the previous algorithm ARH that adds enhanced "*no return rule"* to ARH to solve the *routing-lock* problem. Routing-lock is a phenomenon in which a route is fixed because a routing probability converges in the neighbor of one route during the learning progress. When change of topology takes place and it becomes impossible to use the routes according to this phenomenon, a lot of time is required to discover the new one. In addition, selecting the conventional route will be continued without the ability of discovering a new route, although a new efficient one may be available. "*No return rule*" function is to eliminate return rule (back track path) while selecting next node. It is firstly introduced into *Accelerated Ants Routing* and enhanced in this algorithm.

VIII. GLOBAL POSITIONING SYSTEM ANT-LIKE ROUTING ALGORITHM

In 2000, a author present a novel routing algorithm for MANET called Global Positioning System Ant-Like Routing Algorithm (GPSAL) which is based on the physical location of a destination node and mobile software agents modeled on ants [12]. Ants are used to collect and disseminate information about nodes location in the MANET. GPSAL works with datagram packets and assumes that all mobile hosts participating in a MANET have a GPS unit, which provides the host its approximate three-dimensional position (latitude, longitude, and altitude), velocity and accurate time in Universal Time Coordinate (UTC) format. The proposed algorithm assumes that mobile hosts are moving in a plane, and therefore, the altitude information is not used. The algorithm makes use of location information of a mobile host to reduce the number of routing messages. All hosts in the MANET have a routing table, where each entry represents a known host *d* and has the following information; current location of *d*, previous location of *d*, time-to-live of current location, time-to-live of previous location and whether *d* is a mobile or fixed host. Whenever a mobile node wants to join the MANET, it listens to the medium to find out a neighbor node *n*. Once a neighbor node *n* is identified, the mobile host sends a request packet to *n* asking for its routing table which is sent back to the host. From this moment on, the new mobile host can start routing and sending packets in the MANET. The routing protocol is based on the physical location of a destination host *d* stored in the routing table. If there is an entry in the routing table for host *d*, the best possible route is chosen using the shortest path algorithm. The route, comprised of a list of nodes and the corresponding TTL's, is attached to the packet which is sent to first host in the list. If host *d* is not found in the routing table, the mobile node sends a message to the nearest fixed node that tries to find the destination node.Note that the information in the routing table, which was used to route the packet, probably reflects a snapshot in the past, and the current network configuration may be different. Therefore, each host upon receiving a packet compares the routing information present in the header with the information in its routing table. The entries that have

older information than that in the packet received are updated. This is performed by comparing the TTL field in the packet received and in the routing table. Furthermore, each intermediate node can change the route to a destination node when there is a better route.

An important aspect of any routing algorithm for MANETs is how the routing table is updated. It is clear that better routes can be determined whenever a host has a more recent information about the network configuration. Routing information can be obtained both locally and globally. Local information is obtained from a neighbor node that periodically broadcasts only the changes occurred since the last time (this interval is a configuration parameter). Global information can be disseminated more rapidly using mobile software agents modeled on ants. Ants are agents sent to random nodes on the network. In a MANET, when a mobile computer is powered on, it may not know the physical location (current or past) of other hosts, however, this information can be gathered when a computer receives a message to be forwarded to another node or as a result of its own route discovery. All messages carry the routing information that can be used by intermediate hosts in the routing process. The route discovery can be accelerated using mobile software agents modeled on ants which are responsible for collecting and disseminating more up-to-date location information of mobile hosts. Any node can send an ant to any other node on the MANET. When an intermediate or destiny host receives an ant, it compares the routing information present in the packet received with its routing table, and updates the entries that have older information as explained above. The main goal of the ant agent is to exchange routing information among the network nodes. This process is fundamental for the good performance of our algorithm, since newer information can be disseminated without waiting table exchange. Of course there is an overhead associated with this process that can be controlled by the number of ants in the MANET, but it makes the neighbors' tables exchange more interesting. The information about node movement can now flow more quickly through the entire network. Another important point is how to determine a destination to where an ant should be sent in order to collect more up-to-date information. One possibility is to choose the node with the oldest information in the routing table or the farthest known node in the MANET or just any node.

IX. PROBABILISTIC EMERGENT ROUTING ALGORITHM

A Probabilistic Emergent Routing Algorithm for mobile ad hoc networks (PERA) which presented in [13] is a routing algorithm for mobile ad hoc networks based on the swarm intelligence paradigm and similar to the swarm intelligence algorithms described in [1,4] and [15]. The algorithm uses three kinds of agents - *regular forward ants*, *uniform forward ants* and *backward ants*. Uniform and regular forward ants are agents (routing packets) that are of *unicast* type. These agents proactively explore and reinforce available paths in the network. They create a probability distribution at each node for its neighbors. The probability or goodness value at a node for its neighbor reflects the likelihood of a data packet reaching its destination by taking the neighbor as a next hop.

Backward ants are utilized to propagate the information collected by forward ants through the network and to adjust the routing table entries according to the perceived network status. Nodes proactively and periodically send out forward regular and uniform ants to randomly chosen destinations. Thus, regardless of whether a packet needs to be sent from a node to another node in the network or not, each node creates and periodically updates the routing tables to all the other nodes in the network.The algorithm assumes bidirectional links in the network and that all the nodes in the network fully cooperate in the operation of the algorithm.

X. ADAPTIVE SWARM BASED ROUTING IN COMMUNICATION NETWORKS

Adaptive Swarm-based Routing (ASR) algorithm increases the speed of convergence and has quite good stability by using a novel variation of reinforcement learning and a technique called *momentum* [14]. The momentum technique is used to increase the speed of convergence and avoid high oscillation. It is a very effective mechanism that seeks to incorporate a memory in the learning process, increases the stability of the scheme and helps to increase the learning efficiency of the network. ASR algorithm uses two types of ants called *Forward Ant*, denoted F_{ant}, which will travel from the source node *s* to a destination *d* and *Backward Ant*, denoted B_{ant}, will be generated by a forward ant F_{ant} in the destination *d*. It will come back to *s* following the same path traversed by F_{ant}, with the purpose of using the information already picked up by F_{ant} in order to update routing tables of the visited nodes.

ASR algorithm can react and deal with changes of network. Reported results showed that ASR is better than AntNet in performance and in behavior. ASR is converging rapidly toward a good stable delay value after an initial transitory phase.

XI. ANTS ROUTING ALGORITHMS

Ants-Routing Algorithms are distributed routing algorithms for data networks based on biological ants that explore the network and rapidly learn good routes, using a novel variation of reinforcement learning [13]. The framework of reinforcement leaning is shown in Figure 5. The basic reinforcement learning procedure is stated as follows,

Step1: An agent decides an action according to observed condition in time *t*, and performs selected action *at*.

Step2: Environment is changed from *st* to *st+1*, and an agent acquires reward *rt* corresponded with transition environment.

Step3: $t \leftarrow t+1$, and go to step 1.

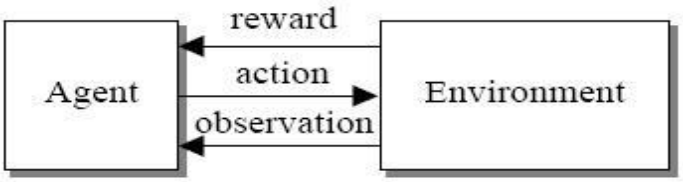

Figure 5. The framework of reinforcement learning

An agent selects an action for the purpose of the maximum reward acquisition. There are two algorithms called *Regular Ant Algorithm* and *Uniform Ant Algorithm* both are fully adaptive to topology changes and link costs changes in the network. The Regular Ant Algorithm is based on earlier work by [15] for call routing in telephone networks. It is the single shortest path algorithm and is only applicable to networks with symmetric path costs. Uniform ant algorithm is a natural multi-path routing algorithm that is applicable to data networks with or without path cost symmetry.

XII. ACCELERATED ANTS ROUTING

Accelerated Ants-Routing algorithm uses ant-like agents that go through the network randomly, without a specific destination, updating pheromone entries pointing to their source. Converging speed of the routing table is the most important factor to evaluate this routing algorithm, due to frequent and unpredictable changes in ad hoc network topology. Accelerated Ants-Routing algorithm is a modified Ants-Routing algorithm that makes convergence speed accelerated compared with other reinforcement base algorithms as Q-Routing, DRQ-Routing and Ants-Routing algorithms [12].

REFERENCES

[1] A. Colorni, M. Dorigo et V. Maniezzo, *Distributed Optimization by Ant Colonies*, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.
[2] M. Dorigo, *Optimization, Learning and Natural Algorithms*, PhD thesis, Politecnico di Milano, Italie, 1992.
[3] S. Goss, S. Aron, J.-L. Deneubourg et J.-M. Pasteels, *Self-organized shortcuts in the Argentine ant*, Naturwissenschaften, volume 76, pages 579-581, 1989
[4] M. Dorigo et L.M. Gambardella, *Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem*, IEEE Transactions on Evolutionary Computation, volume 1, numéro 1, pages 53-66, 1997.
[5] D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck, B. Baesens, *Classification with Ant Colony Optimization*, IEEE Transactions on Evolutionary Computation, volume 11, number 5, pages 651—665, 2007.
[6] B. Pfahring, "Multi-agent search for open scheduling: adapting the Ant-Q formalism," Technical report TR-96-09, 1996.
[7] T. Stützle, "An ant approach to the flow shop problem," Technical report AIDA-97-07, 1997.
[8] A. Baucer, B. Bullnheimer, R. F. Hartl and C. Strauss, "Minimizing total tardiness on a single machine using ant colony optimization," Central European Journal for Operations Research and Economics, vol.8, no.2, pp.125-141, 2000.
[9] A. V. Donati, V. Darley, B. Ramachandran, "An Ant-Bidding Algorithm for Multistage Flowshop Scheduling Problem: Optimization and Phase Transitions", book chapter in Advances in Metaheuristics for Hard Optimization, Springer, ISBN 978-3-540-72959-4, pp.111-138, 2008.
[10] P. Toth, D. Vigo, "Models, relaxations and exact approaches for the capacitated vehicle routing problem," Discrete Applied Mathematics, vol.123, pp.487-512, 2002.
[11] Y. C. Liang and A. E. Smith, "An ant colony optimization algorithm for the redundancy allocation problem (RAP)," IEEE Transactions on Reliability, vol.53, no.3, pp.417-423, 2004.
[12] D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck, B. Baesens, "Classification with Ant Colony Optimization", IEEE Transactions on Evolutionary Computation, volume 11, number 5, pages 651—665, 2007.
[13] Shmygelska, R. A. Hernández and H. H. Hoos, "An ant colony algorithm for the 2D HP protein folding problem," Proceedings of the 3rd International Workshop on Ant Algorithms/ANTS 2002, Lecture Notes in Computer Science, vol.2463, pp.40-52, 2002.
[14] J. ZHANG, H. Chung, W. L. Lo, and T. Huang, "Extended Ant Colony Optimization Algorithm for Power Electronic Circuit Design", IEEE Transactions on Power Electronic. Vol.24,No.1, pp.147-162, Jan 2009.
[15] L. Bianchi, L.M. Gambardella et M.Dorigo, *An ant colony optimization approach to the probabilistic traveling salesman problem*, PPSN-VII, Seventh International Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science, Springer Verlag, Berlin, Allemagne, 2002.

