SWARM INTELLIGENCE FOR ROUTING IN AD HOC WIRELESS NETWORKS

***Milos Stojmenovic*[1]**
School of Computer Science, Carleton University,
Ottawa, Ontario, K1S 5B6, Canada

Abstract

Ad hoc networks consist of autonomous self-organized nodes. Nodes use a wireless medium for communication, thus two nodes can communicate directly if and only if they are within each other's transmission radius. Examples are sensor networks (attached to a monitoring station), rooftop networks (for wireless Internet access), and conference and rescue scenarios for ad hoc networks, possibly mobile. In a routing task, a message is sent from a source to a destination node in a given network. Two nodes normally communicate via other nodes in a multi-hop fashion. Swarm intelligence follows the behaviour of cooperative ants in order to solve hard static and dynamic optimization problems. Ants leave pheromone trails at nodes or edges which increases the likelihood of other ants to follow these trails. Routing paths are then found dynamically on the fly, using this so called notion of stigmergy. In this article we survey existing literature on swarm intelligence based solutions for routing in ad hoc networks. We identified 12 different methods, covering non-position and position based approaches, flooding and path based search methods. Some of the articles consider related problems such as multicasting or data centric routing. All of the articles were published after 2001. The ideas coming from existing swarm intelligence based routing in communication networks are incorporated into the wireless domain, with some new techniques which are typical for the wireless domain (such as flooding, use of position, monitoring traffic at neighbouring nodes) being incorporated. We observed that the experimental data provided by these articles is insufficient to make a firm conclusion about scenarios which show the advantages of the proposed swarm intelligence based methods with respect to other existing methods.

Key Words: Ad hoc, routing, swarm, ant

---

[1] E-mail: knezmilos22@lincsat.com

1. Introduction

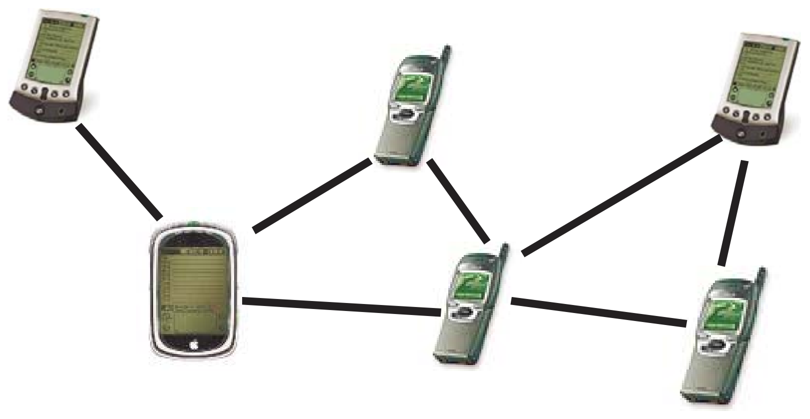


Figure 1. Self-organized ad hoc wireless network

In an ad hoc wireless networks, nodes are self-organized and use wireless links for communication between themselves. Ad hoc networks are dynamically created. Examples are conference, battlefield, rescue scenarios, sensor networks placed in an area to monitor the environment, mesh networks for wireless Internet access etc. Nodes in ad hoc networks can be mobile in many scenarios, or mostly static in other scenarios, as in sensor networks. Nodes may decide to go to sleep mode to preserve energy, and wake up later to rejoin the network. Routing solutions must address the nature of the network, and aim at minimizing control traffic, to preserve both bandwidth and energy at nodes. Ant-colony based algorithms use a number of control traffic, or existing traffic, sets of information to create best routes. It is a challenging task to discover good routes with controlled traffic, so that overall the swarm intelligence approach outperforms existing routing protocols for ad hoc networks.

Swarm intelligence is a set of methods to solve hard static and dynamic optimization problems using cooperative agents, usually called ants. Ant inspired routing algorithms were developed and tested by British Telecomm and NTT for both fixed and cellular networks with superior results [10, 5, 1, 22, 25]. AntNet, a particular such algorithm, was tested in routing for communication networks [5]. The algorithm performed better than OSPF, asynchronous distributed Bellman-Ford with dynamic metrics, shortest path with a dynamic cost metric, the Q-R algorithm and predictive Q-R algorithm [10, 5, 1, 22, 25].

This project will review the literature on swarm intelligence based solutions for routing in ad hoc networks. After an extensive search on http://citeseer.nj.nec.com/cs and www.google.com, 13 different relevant articles (two of the articles were published twice, so the total count is 15) were found. They are all very recent, published in 2001 or later, and they propose some swarm intelligence based routing methods for ad hoc wireless networks. Their list is given in the references section. The goal of the article is to summarize existing solutions, classify them according to assumptions and approaches taken, compare them, report on experimental findings from the article, and to draw some conclusions. It was observed that cross referencing between these articles is poor, which is not surprising since many of them appeared simultaneously and all of them were published within the last two years. Two articles were published in 2001, two were published in 2002, and nine out of these

13 articles were published in 2003. There were some independent discoveries of the same ideas, which was also not surprising. It was observed, however, that a number of summaries of other works given in these articles was incorrect, and that many articles do not clearly state which ideas come from the existing research, and which ideas are new. The approach taken in this article is to first present existing swarm intelligence based methods for routing in communication networks, and existing routing schemes for ad hoc networks (in both cases, we only presented methods that were actually used in the surveyed articles), and then referred to them when ad hoc network scenarios are considered, so that additions and differences between them are underlined.

This article is organized as follows. Section 2 describes swarm intelligence based routing schemes for communication networks. Section 3 presents routing schemes for ad hoc networks, which do not use swarm intelligence, and which are adapted in the surveyed articles by adding ants for enhanced performance. Section 4 summarizes path based routing schemes with swarm intelligence, which are close to the schemes used in communication networks. Section 5 describes routing schemes which use a wireless medium to flood the ants; therefore each initial ant multiplies into a number of ants in the process, which is a non-traditional understanding of what an ant is. Section 6 presents solutions which assume that nodes have position information, that is, they know their geographic coordinates. Two related routing problems, multicasting, and data centric routing in sensor networks, are discussed in sections 7 and 8. Conclusions and references finish this article.

2. Swarm Intelligence for Routing in Communication Networks

2.1. General Principles

We will first describe general principles in all swarm intelligence based solutions. They are used in all of the described solutions, each with particular details starting from this general approach. The ants navigate their designated selection of paths while depositing a certain amount of substance called *pheromone* on the ground, thereby establishing a trail. The idea behind this technique is that the more ants follow a particular trail, the more attractive is that trail for being followed by other ants. They therefore dynamically find a path on the fly, using the explained notion of *stigmergy* to communicate indirectly amongst themselves. In the case of routing, separate pheromone amounts are considered for each possible destination (that is, on each link pheromone trails are placed in a sequence, one trail for each possible destination). An ant chooses a trail depending on the amount of pheromone deposited on the ground. Each ant compares the amounts of trails (for the selected destination) on each link toward the neighbouring nodes. The larger the concentration of pheromone in a particular trail, the greater the probability of the trail being selected by an ant. The ant then reinforces the selected trail with its own pheromone. The concentration of the pheromone on these links evaporates with time at a certain rate. It is important that the decay rate of pheromone be well tuned to the problem at hand. If pheromone decays too quickly then good solutions will lose their appeal before they can be exploited. If the pheromone decays too slowly, then bad solutions will remain in the system as viable options.

Each node in the network has a routing table which helps it determine where to send the next packet or ant. These routing tables have the neighbours of the node as rows, and all of the other nodes in the network as columns. In figure 2, we see an example of a network, and in figure 3 we see the routing table for node *S* in this network.

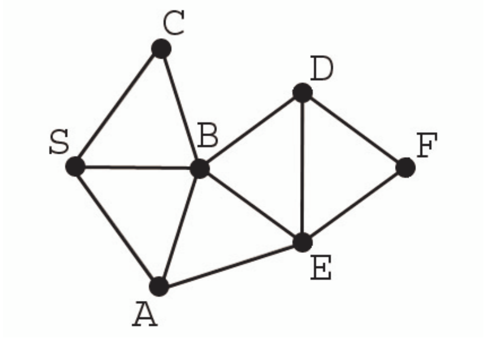


Figure 2 network

	A	B	C	D	E	F
A	0.9	0.1	0.1	0.4	0.5	0.5
B	0.1	0.8	0.2	0.6	0.4	0.4
C	0	0.1	0.7	0	0.1	0.1

Figure 3 routing table for node *S*

An ant or message going from node *S* to node *F*, for example, would consider the cells in column *F* to determine the next hop. Ants and messages can determine the next hop in a variety of ways. The next hop can be determined uniformly; which means that any one of the neighbours has an equally likely probability of being chosen. It can be chosen probabilistically, that is, the values in the routing table in column *F* are taken as the likelihoods of being chosen. Taking the highest value in the column of *F* could be another way of choosing the next hop. It could also be chosen randomly, which means choosing uniformly if there is no pheromone present, and taking the highest value if there is. There is also an exploratory way of choosing the next hop, which means taking a route with a value of 0 if one exists.

There are a few swarm intelligence (ant-based) routing algorithms developed for wired networks, and the most well known of which are AntNet [5] and Ant-Based Control (ABC) [22]. The fundamental principle behind both AntNet and ABC is similar – they use ants as exploration agents. These ants are used for traversing the network node to node and updating routing metrics. A routing table is built based on the probability distribution functions derived from the trip times of the routes discovered by the ants. The approaches used in AntNet and ABC are, however, dissimilar – in AntNet, there are *forward* and *backward* ants, whereas in ABC, there is only one kind of ant. Another difference between AntNet and ABC is in the routing front. In ABC, the probabilities of the routing tables are updated as the ants visit the nodes, and are based on the life of the ant at the time of the visit; while in AntNet, the probabilities are only updated when the backward ant visits a node.

2.2. Ant-Based Control (ABC) Routing

Schoonderwoerd, Holland, and Bruten [22] proposed the Ant-Based Control (ABC) scheme for routing in telephone networks. In the ABC routing scheme [22], there exist two kinds of routing tasks: exploratory ants which make probabilistic decisions, and actual calls which made deterministic decisions (that is, choosing the link with the most pheromone in the column corresponding to the destination). Exploratory ants are used for source updates. Each source node S issues a number of exploratory ants. Each of these ants goes toward a randomly selected destination D (the ant is deleted when it reaches D). The routing table at each node contains neighbours as rows and all possible destinations as columns, and each entry corresponds to the amount of pheromone on the link towards a particular neighbour for a particular destination. These amounts are normalized in each column (the sum is one), so that they can be used as probabilities for selecting the best link. At each current node C, the entry in the routing table at C corresponding to the source node S is updated. Exploratory ants make the next node choice by generating a random number and using it to select a link based on their probabilities in the routing table. The amount of pheromone left on a trail depends on how well the ant performs. Aging is used to measure performance. In each hop, the delay depends on the amount of spare capacity of the node, and is added to the age. Both ants and calls travel on the same queue. Calls make a deterministic choice of a link with the highest probability, but do not leave any pheromone. The pseudo code of the ABC algorithm is presented below. RT[S][X][Y] is the probability of going from node *S* to node *Y* via node *X*. Coming back to figure 3, for example, the value of RT[S][A][C]=0.

```
Each ant chooses source $S$ and destination $D$ at random; $C=S$; $T=0$
    While $C \neq D$ do {
        Choose next node $B$ using probabilities from $RT[C][B][D]$
        $D = c\ e{-}d*sparecapacity(B);\ T=T+D;\ \delta= a/T + b$
        $RT[B][C][S]= (RT[B][C][S] + \delta)/(1+ \delta)$
        $RT[B][X][S]= RT[B][X][S]/(\delta +1)$ for $X \neq C$
        $C=B$
    }
```

The variables *a*, *b*, *c* and *d* are parameters with empirically determined values. There is an exploration threshold, *g*, as well. The threshold *g*, if crossed determines the next hop uniformly instead of consulting the routing table. This *g* value is used to ensure that not only one path is used. It is there to make sure that other routes are tried from time to time.

Guerin proposed an all column update enhancement to the ABC scheme. While moving forward, the ABC algorithm only updates routing tables corresponding to source S. Guerin [7] proposed updating the routing tables for all other nodes visited in the route. For example, let the route be: SABCD. In ABC, the routing tables for S are updated at nodes A, B, C and D as an ant moves toward D. The all column update scheme [7] adds updating routing tables for A at B, C and D, routing tables for B at C and D, and routing table for C at D.

2.3. AntNet and Other Schemes

In the AntNet scheme [5], each node periodically sends a forward ant packet to a random destination. The forward ant records its path as well as the time needed to arrive at each intermediate node. The timing information recorded by the forward ant, which is forwarded with the same priority as data traffic, is returned from the destination to the source by means of a high priority backward ant. Each intermediate node updates its routing tables with the information from the backward ant. Routing tables contain per destination next hop biases. This way, faster routes are used with greater likelihood.

Subramaniam, Druschel, and Chen [19] described a method which has characteristics of both the AntNet and ABC schemes, and applied it to packet switching networks. Routing tables are probabilistic and are updated as in ABC [22]. They [19] introduce *uniform ants* that uniformly randomly choose the next node to visit (all neighbours have the same probability of being selected). Ants accumulate cost as they progress through the network. Their method is called Ants-Routing. Only backward exploration is used to update routing tables.

White [24, 25] suggested another routing algorithm for circuit switched networks. The approach is based on three kinds of ants. The first class collects information, the second class allocates network resources based on the collected information and the third class makes allocated resources free after usage.

3. Routing in Ad Hoc Networks without Swarm Intelligence

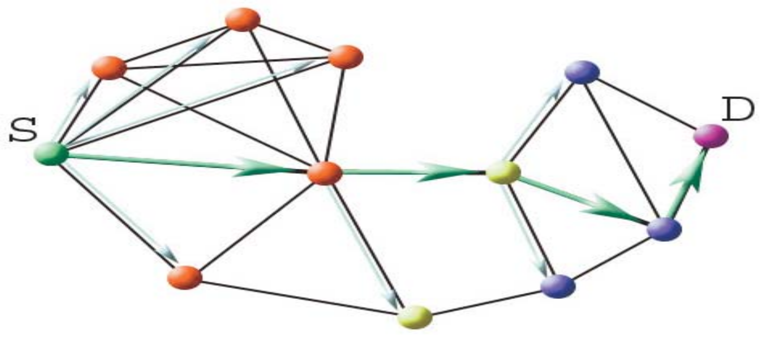


Figure 4: Route discovery form S to D

Routing methods in literature are divided into two groups based on the assumptions made on the availability of position information. There exist non-position and position based approaches. In position based approaches, it is assumed that each node knows its geographic coordinates, the coordinates of its all neighbours, and is somehow informed about the position of the destination. Location based systems have recently been making rapid technological and software advances, and there are cheap solutions with tiny hardware already available. Non-position based solutions assume no knowledge of position information.

3.1. Non-Position Based Routing

In AODV [16], the source node floods a route discovery message throughout the network. Each node receiving the message for the first time retransmits it, and ignores further copies of the same message. This method is known as blind flooding. The destination node replies back to the source upon receiving the first copy of the discovery message using the memorized hops of the route. The source node then sends the full message using the recorded path. The method may easily provide multipaths for quality of service, and each node may introduce forwarding delays which may depend on the energy left at the node, or is imposed by a queuing delay. Local route maintenance methods are developed for mobile ad hoc networks. The expanding ring search is also considered to reduce the overhead coming from blind flooding. An adaptive distance vector (ADV) routing algorithm for mobile, ad hoc networks is proposed in [2], where the amount of proactive activity increases with increasing mobility.

The zone routing protocol [11] applies a combination of proactive and reactive routing. Proactive routing is applied for nodes within the same zone, while reactive on-demand routing (such as AODV) is applied if the source and destination are not in the same zone. Within the zone, routes can be proactively maintained using one of several options. One option is to broadcast local topological change within the zone so that shortest paths can be computed. The other option is to periodically exchange routing tables between neighbours, so that each node can refresh its route selection using new information from its neighbours.

3.2. Position Based Routing

Finn [6] proposed a position based localized greedy routing method. Each node is assumed to know the position of itself, its neighbours, and the destination. The source node, or node currently holding the message, adopts the greedy principle: choose the successor node that is closest to the destination. The greedy method fails when none of the neighbouring nodes are closer to the destination than the current node. Finn [6] also proposed a recovery scheme from failure: searching all *n*-hop neighbours (nodes at a distance of at most *n* hops from the current node) by limited flooding until a node closer to the destination than C is found, where *n* is a network dependent parameter. The algorithm has nontrivial details and does not guarantee delivery.

4. Path Based ant Routing for Ad Hoc Networks

Our literature review will begin with swarm intelligence based routing methods which do not use the geographic positions of nodes, and which follow the well known traditional definition of an ant, as a single entity that travels through the network, creating a path, possibly travels back to its source, and eventually disappears. There are three protocols described in this category, by Matsuo and Mori [13] in 2001, Islam, Thulasiraman and Thulasiram [12] in April 2003, and by Roth and Wicker [18] in June 2003. The following section will cover an alternative notion of an ant as an entity that can multiply itself.

4.1. Accelerated Ants Routing

Matsuo and Mori [13] apparently described the first ant based routing scheme for ad hoc networks, called *accelerated ants routing* in 2001. It appears that it is a straightforward adaptation of a well known scheme for communication networks, with two additions which themselves do not appear to be novel. They followed the Ants-Routing method [19] and added a ‘no return’ rule which does not allow ants to select the neighbour where the message came from. They also added an ‘N step backward exploration rule’. This is identical to the all column update scheme proposed by Guerin [7]. In [13], it is applied when an ant moves backward (and consequently routing entries toward the destination are updated). Performance evaluation showed that the new ants routing algorithm achieves good acceleration for routing table’s convergence with respect to the Ants-Routing method, even if network topology was dynamically changed.

The accelerated ants routing scheme [13] uses both probabilistic and uniform ants. Uniform ants are important in ad hoc networks because of link instabilities. When a link on a favourite route is broken, uniform ants may quickly establish an alternative route. The whole algorithm is illustrated in the following figures.

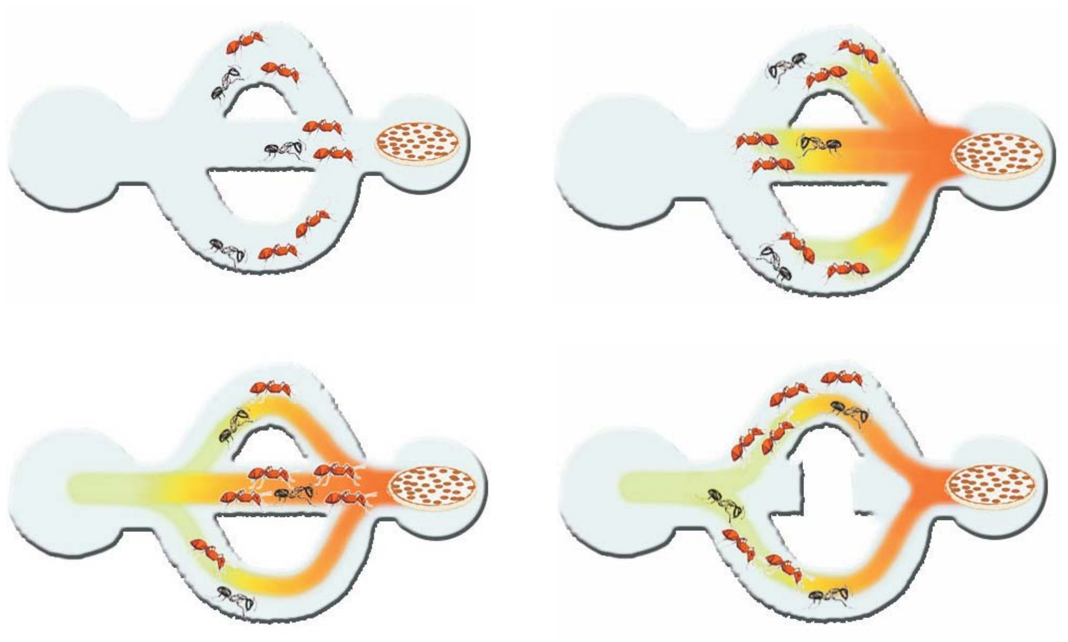


Figure 6 shortest path is most reinforced

Figure 4 illustrates both the probabilistic (red) and the uniform (black) ants choosing the paths uniformly since there is no pheromone present in the network. Figure 5 shows the returning ants marking the path with pheromone. The path in the middle is the shortest, and therefore has the highest concentration of pheromone. This is why most of the probabilistic ants in figure 6 follow this trail. Figure 7 shows that the ants will adapt if a path disappears. The top path is shorter than the bottom one; therefore, the probabilistic ants have a higher chance of choosing it.

4.2. Source Update Routing

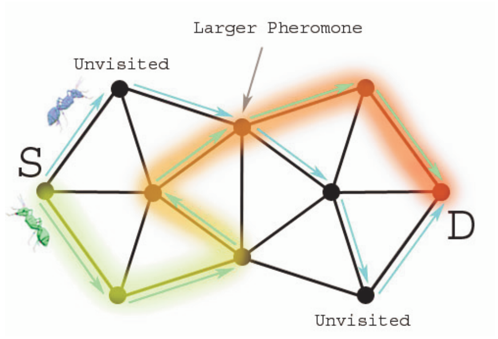
Islam, Thulasiraman and Thulasiram [12] recently proposed an ant colony optimization (ACO) algorithm, called *source update*, for all-pair routing in ad hoc networks. ‘All pair’ routing means that routing tables are created at each node, for all source-destination pairs, in the form of a matrix with neighbours as rows and destinations as columns, so that the table assists in any randomly chosen source-destination pair. The algorithm is claimed to be scalable, but apparently this is with respect to the number of processors on a parallel computer, not the number of nodes in an ad hoc network. The authors also claim that it is an on-demand routing algorithm for ad hoc networks; this is true if ants are launched just before data traffic. They, [12], develop a mechanism to detect cycles, and parallelize this algorithm on a distributed memory machine using MPI.

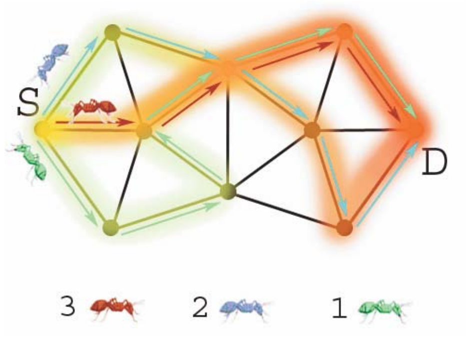
In the source update technique [12], each ant memorizes the whole path to its destination and uses it to return back to the source**.** While the ant is searching for the destination, the routing table updates are performed to form a trail that leads back to the source. During the backward move, updates are made with respect to the selected destination D (with D as the starting point in the route, thus erasing the accumulated weight first), which then in fact serves as the source of the new message, therefore the procedure for the backward move is algorithmically identical to the one used in the forward move. Backward routing is needed so that S finally places some pheromone in its routing table for D. The amount of pheromone placed at each selected edge is not constant in [12]. It depends on the weight, which can be a function of distance, transmission time (delay), congestion, interaction time or other metrics ([12] used the delay as weight). Note that the amount of new pheromone left on a traveled link is inversely proportional to the cumulative weight from S to the current node, so that longer paths are less enforced. The amount of pheromone in other entries is decreased by a certain fixed percentage. The authors do not normalize the total pheromone count (that is, the sum is not equal to 1), which is done in some traditional approaches such as [22]. Comparing several different ants going toward the same destination, longer created paths obviously evaporation more and accumulate less pheromone, and shorter path therefore have a higher chance of be selected.

To memorize the path, ants in [12] use a stack data structure containing all of the nodes along the path from S (these nodes are called *stack* nodes). The same stack is used in [12] for loop detection and avoidance. This is achieved by ignoring neighbours which are already in the stack when deciding the next hop. Therefore, a loop is never created. If a node has no neighbour which is not already in the stack (such a node becomes a *visited* node), the search backtracks to the previous node. The authors do not discuss the possible reappearance of such visited nodes in the stack later on, which could lead to infinite loops. However, this can be avoided by keeping such nodes in a separate list of visited nodes, so that it does not reappear on the route (and loop creation is avoided). The algorithm, therefore, is a simple *depth-first search* scheme, which the authors [12] do not note.

Exploratory ants [12] apply the following semi-deterministic scheme when deciding the next node to continue the depth first search with. If there is any link toward *unseen* neighbours (unseen neighbours are nodes which are neither stack nodes nor visited nodes) that also has not yet been tried by any other ants, it is selected (if there are a few such links, one at random is selected). The reason is that the quality of all path candidates needs to be tested. This is important for ad hoc networks, since a newly created edge may provide good quality path. If there is no such unseen node, the ant searches for the next hop by considering the pheromone concentration. It selects the neighbour whose pheromone trail in the column corresponding to destination D is the largest.

The experimental results in [12] concentrate only on the parallel implementation for the algorithm, and discuss issues like parallel speed up, scalability with respect to number of processors used, and time versus number of ants. The only comparison is with a basic technique without source update, which is a technique where ants make random decisions at each node, without leaving any pheromone behind. There is no discussion on the impact of various parameters. Since ad hoc networks are self-organized networks where each node makes independent decisions (generally following the pre-agreed protocol), parallel implementations (aiming at speedup optimization), where one processor simulates the work of several nodes from the ad hoc network, do not provide the needed insight into the performance of a particular routing protocol. The insight provided by the authors [12] is only on the quality of their parallelization.





Figures 8 and 9 illustrate the source update routing algorithm presented by [12]. Figure 8 shows how ants prefer unvisited nodes in their path to the destination. They pick the node with the highest concentration of pheromone if no unvisited nodes exist in their path. The arrows in both figures depict the forward movement of the ants, and the pheromone trails depict the backward movement. In figure 9, the brown ant was last to move, and it found a path that is shorter than that of its predecessors.

4.3. Random Walk Based Route Discovery

Roth and Wicker [18] presented the scheme called ‘Termite’ which expands on the ABC algorithm [22], but does away with the idea that only specialized packets may update routing tables. In the Termite protocol [18], data traffic follows the largest pheromone trails, if any exist on any link. If there are no pheromone trails on any link, a route request is performed by a certain number of ants. Each ant performs a random walk over the network. In the random walk, ants and packets uniformly randomly choose their next hop, except for the link they arrived on. During the random walk, pheromone trails with respect to the source are left. If an ant cannot be forwarded, it is dropped. Any number of ant packets may be sent for each route request; the exact number of which may be tuned for a particular environment. An ant is not looking for an explicit route to the destination. Rather it is searching for the beginning of a pheromone trail to the destination. The route will be strengthened by future communications. Once an ant reaches a node containing pheromone to the requested destination, a route reply packet is returned to the requestor. The message is created such that the source of the packet appears to be the requested destination and the destination of the packet is the requestor. The reply packet extends pheromone for the requested destination back to the requestor without any need to change the way in which pheromone is recorded at each node. The reply packet is routed normally through the network probabilistically following a pheromone trail to the requestor. Intermediate nodes on the return path automatically discover the requested node. Hello packets are used to search for neighbours when a node has become isolated. Proactive seed packets are used to actively spread a node's pheromone throughout the network. Seeds make a random walk through the network and serve to advertise a node's existence. They can be useful for reducing the necessary number of explicit route request transactions. All routing decisions in Termite are random. A time to live field is used to prevent propagation of bad routes. The size of the pheromone table may be reduced by implementing a clustering scheme.

Termite can take advantage of the wireless broadcast medium, since it is possible for each node to promiscuously listen to all transmissions. Routing information can be gained from listening to all traffic, rather than only to specifically addressed traffic. New nodes can quickly be detected when their transmissions are overheard. Also, a great deal of information about the network can be gained from the destinations that neighbours are forwarding to. While promiscuity can boost the performance of Termite, it also creates some problems. The same packet overheard a few times shall not be processed more than once, to avoid misleading pheromone gradients. In order to prevent the double counting of packets, a message identification field is included in Termite packets. Another problem is that energy consumption increases when traffic at neighbouring nodes is monitored. Finally, Termite assumes bidirectional links. This article therefore presented a number of novel ideas for ant based routing. However, the experimental data only presented the performance of the Termite protocol, without comparing it with any other routing scheme.

The Termite scheme [18] differs from the source routing [12] by applying pheromone trails or random walks instead of a stack based depth first search. Therefore it allows loops. It differs from the accelerated ants routing [13] by applying random walk ants rather than uniform or probabilistic ones. Random walk ants differ from uniform ants since they follow pheromone trails, if any. Termite [18] also does not apply all column updates. Finally, the Termite scheme applies monitoring traffic at neighbouring nodes, which is not present in [13] and [12].

Figures 10 and 11 illustrate the Random walk based route discovery algorithm. The red ant in figure 10 has left a pheromone trail from its current location to destination *D*. The blue ant makes a random walk (labelled by the numbered blue arrows) along the network until it reaches the pheromone trail left by the red ant to the destination. As it searches for a trail to the destination, it leaves a trail which leads back to the source. It then turns around, and lays a second pheromone trail (which leads to the destination) from this node back to the source, as seen in figure 11. This forms a trail that leads from source to destination.

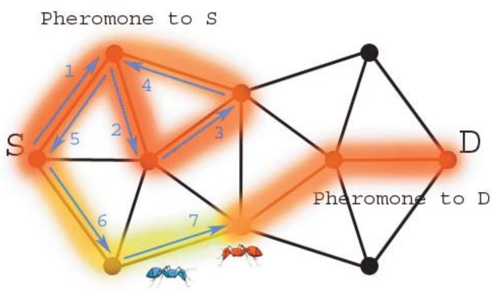


Figure 10 Blue ant searches for trail

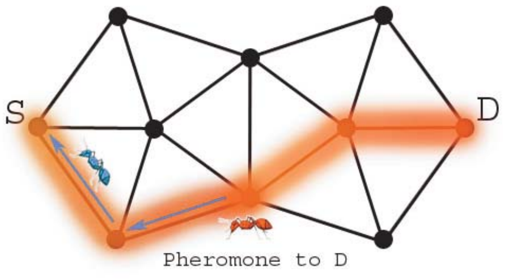


Figure 11 Blue ant returns to source

5. Flooding Based Ant Routing

Half, (that is, six out of 12) of the published articles that we surveyed fall into this category. Two such methods are proposed in 2002, by Marwaha, Tham, and Srinivasan [14, 15], and by Gunes, Sorges and Bouazizi [9]. This later method was improved by Gunes, Kahmer and Bouzazizi [KBB] in June 2003. Baras and Mehta [3] added a method in March 2003. Finally, in May 2003, Rajagopalan, Jaikaeo and Shen [17] applied flooding in the context of their zonal routing scheme.

5.1. AntAODV Reactive Routing

Marwaha, Tham and Srinivasan [14, 15] studied a hybrid approach using both AODV and reactive Ant based exploration. Their technique is called AntAODV. Routing tables in AntAODV are common to both ants and AODV. If the sender node (or node currently holding the message) S has a fresh route toward the destination, it uses it to forward the packet. The authors claim that this is different from AODV which starts route discovery first, but there are modifications of AODV in literature that use fresh routes in the same way. Otherwise (no fresh route available) it will have to keep the data packets in its send buffer until an ant arrives and provides it with a route to that destination. Each ant follows a blind flooding approach and therefore multiplies into several ants. If an ant reaches a node with a fresh route, it stops the advance and converts into a backward ant to report the route to S. Note that again, a similar provision already exists in AODV variations. Ants take a ‘no return’ rule, meaning that they never return to the node they came from. Overall, it appears that the only difference between AODV and its variants, and AntAODV, is that routing tables are larger, listing all neighbors with their trail amounts for each destination instead of simple routing tables used in AODV, listing only the best choice. This allows a random selection of the next hop, based on pheromone trails. The definition of a fresh route is similar in the two schemes. In the experimental section, comparing a new scheme against AODV (without the mentioned variations), the authors added a proactive component to AntAODV. If no ant visited a node within a certain visit period, the node would generate a new ant and transmit it to one of its neighbours selected randomly. This article does not discuss pheromone trails (that is, what they mean by ‘fresh’ routes) and therefore does not sufficiently underline how the ant based approach really works compared to already existing equivalent AODV variants.

5.2. ARA Reactive Routing

Gunes, Sorges and Bouazizi [9] presented a detailed routing scheme, called ARA, for MANETs, including route discovery and maintenance mechanisms. Route discovery is achieved by flooding forward ants to the destination while establishing reverse links to the source. Their approach uses ants only for building routes initially and hence is a completely reactive algorithm. A similar mechanism is employed in other reactive routing algorithms such as AODV. Routes are maintained primarily by data packets as they flow through the network. In the case of a route failure, an attempt is made to send the packet over an alternate link. Otherwise, it is returned to the previous hop for similar processing. A new route discovery sequence is launched if the packet is eventually returned to the source. The scheme also uses a notion of reinforcement of currently used routes. A forward ant establishes a pheromone track back to the source, while a backward ant establishes a pheromone track to the destination. ARA prevents loops by memorizing traffic at nodes. If a node receives a duplicate packet, it will send the packet back to the previous node. The previous node deactivates the link to this node, so that the packet cannot be sent in that direction any longer. This loop prevention mechanism is problematic, since further backtracking, if needed, is not resolved, and is based on traffic memorization. Regular data packets are used to maintain the path. In case of link failure, the pheromone trail is set to 0, and the node will send the packet on the second best link. If that link also fails, the node informs the source node about the failure, which then initiates a new route discovery process. Their algorithm is implemented in the ns-2 simulator and compared with AODV. The algorithm, however, is inherently not scalable. The protocol is similar to the AntAODV [14, 15] but gives more specific ant behaviour by discussing pheromone use and updates. It also additionally memorizes past traffic and applies pheromone table values instead of ‘fresh’ link indicators.

5.3. Enhanced ARA Protocol: Prioritized Queue, Backward Flooding and Tapping

Gunes, Kahmer, and Bouzazizi [8] presented some extensions and improvements to their previous article [9]. Probabilistic routing is used instead of selecting the path with the maximal pheromone trail. Pheromone values decrease continually rather than in discrete intervals. Ant packets use a prioritized queue rather than handling them as ordinary data packets. Backward ants use the same type of flooding as forward ants instead of returning on the constructed path. For several packets on the same connection, only one forward ant is created. Finally, similarly as in [18], MAC-Tap extracts information from packets from the neighbourhood. Experimental data shows improvements, however the need to flood the network is a big disadvantage in mobile ad hoc networks. A flooding technique with less overhead is desirable.

5.4. PERA: Proactive, Stack and AODV Based Routing Protocol

Baras and Mehta [3] described two ant-based routing schemes for ad hoc networks. One scheme only uses one-to-one or unicast communications where a message sent by one node is only processed at one neighbouring node, while the other utilizes the inherent broadcast one-to-all nature of wireless networks to multicast control and signalling packets (ants), where a message sent by one node is received by all its neighbours. Both algorithms are compared with the well known ad hoc reactive routing scheme, AODV [16].

The first algorithm in [3] is similar to the swarm intelligence algorithm described in [5, 19]. It uses regular forward, uniform forward and backward ants. Regular forward ants make probabilistic decisions based on pheromone trails, while uniform forward ants use the same probability of selecting each neighbour. Forward ants use the same queue as data packets. When a forward ant is received at a node, and that node is already in the stack of the ant, the forward ant has gone into a loop and is destroyed. Backward ants use the stack which memorized the path to return to the source, using high priority queues. Only backward ants leave pheromones on the trails. Newly created edges are assigned a small amount of pheromone, while broken edges are followed by the redistribution of pheromone to other nodes with normalization.

The second algorithm [3] is called PERA (Probabilistic Emergent Routing Algorithm). The algorithm applies a route discovery scheme used in AODV to proactively establish routes by the ants. This is a very similar type of route discovery, used reactively in AODV, the difference being that metrics other than hop count may be used. If hop count is used, forward and backward ants travel on high priority queues. If delay is used as metric, they use data queues, so that routes with less congestion are preferred. Multipath routes are established. Each initial forward ant (only regular forward ants are used) creates multiple forward ants. Only backward ants change the probabilities in the routing tables (pheromone trails are placed using a different reinforcement model than in other articles). Data packets can be routed probabilistically, or deterministically (using the neighbour with the highest probability for the next hop). The simulation was performed on the ns-2 with 20 nodes, and PERA was compared with AODV. The authors observe that end-to-end delay for swarm based routing is low compared to AODV, but the goodput (ratio of data to control packets at each node) is worse (lower) than in AODV. The later conclusion is due to heavy proactive overheads in situations with heavy topological changes. We also note that AODV is used with the hop count as a metric which is unfair when delay is used for comparison (AODV schemes with other metrics are already proposed in the literature).

5.5. ANSI: Zone, Flooding and Proactive/Reactive Routing

Rajagopalan, Jaikeo, and Shen [17] described the ANSI (Ad hoc Networking with Swarm Intelligence) protocol. Route discovery and maintenance in ANSI is a combination of proactive and reactive activities. Proactive ants are broadcast periodically to maintain routes in a local area. Whenever other routes are required, a forward reactive ant is broadcast. The outline of the process of ANSI routing is as follows:

- Every node periodically broadcasts *proactive ants* which reach a number of nodes in its local area. Each ant is allocated a certain maximum energy, which is reduced by the energy needed to transmit to a given node. The zone of each node is equal to the transmission radius used in the broadcast. Each receiving neighbour decides to retransmit with a certain fixed probability.
- When a route to a destination *D* is required, but not known at source *S*, *S* broadcasts a *forward reactive ant* to discover a route to *D*. The number of hops that ant can travel is limited.
- When *D* receives the forward reactive ant from *S*, it source-routes a *backward reactive ant* to the source *S*. The backward reactive ant updates the routing table of all the nodes in the path from *S* to *D*.
- When a route fails at an intermediate node *X*, ANSI buffers the packets which could not be routed and initiates a route discovery to find *D*. Additionally, *X* sends a route error message back to the source node *S*.

The simulation is performed using Qualnet with up to 30 nodes, and comparison is made with AODV. ANSI consistently performed better than AODV with respect to delay characteristics, but the packet delivery rate in ANSI needs to be improved. The scalability of ANSI remains to be investigated. If the zone size remains limited, and hop count for reactive ants becomes unlimited, the performance is expected to be close to that of AODV. If zone size is increased, a comparison with ZRP becomes more appropriate.

Hybrid routing protocols like ZRP [11], ADV [2], and AntAODV [14, 15] have leveraged the power of proactive routing with the flexibility and scalability of purely reactive routing. ZRP has a fixed zone radius, while ANSI has a flexible implicit zone radius, which can adapt itself to changing network requirements. This adaptive model resonates with the approach used in ADV [2], where the amount of proactive activity increases with increasing mobility. Furthermore, the timeout period (equivalent to the beacon timeout in ZRP [11]) in ANSI can also be adaptive to reflect the routing needs as the mobility and route errors in a network increase.

6. Ant and Position Based Routing in Large Scale Ad Hoc Networks

6.1. Proactive, Zone Grouping, Logical Link Based Routing

Heissenbüttel and Braun [10] described a proactive position and ant based routing algorithm for large and possibly mobile ad hoc networks. The plane is divided into geographical areas (e.g. squares) with all nodes within the same area belonging to the same logical router (LR). All the nodes within a LR share and use the same routing tables. Every logical router has its own set of logical links (LLs). A set of LRs is considered as a communication endpoint for the LLs. For that purpose, a LR groups the other LRs into zones depending on their position relative to it (as shown in Fig. 12). More LRs are grouped together as they are located farther away. It is not a pure hierarchical approach since these zones look different for different LRs. LLs are now established from a specific LR to all its zones. The routing table at each LR has a row for every outgoing LL and a column for every zone. Therefore this is a table with zones as both rows and columns. For a given row zone entry, the table gives probabilities to select column zone entries as the next logical hops, if the destination is located in a row zone. The link costs of incoming LLs are stored in another table. This information will be used to determine the quality of the followed path by the ants.

Ants and data packets are both marked in the header fields with source and destination coordinates. Further, they keep track of the followed path by storing the coordinates of each intermediate relaying node. The followed path can be approximated by a sequence of straight lines. Data packets and ants are routed basically in the same way. The LR determines in which zone the destination coordinates are located and then selects an outgoing LL for that zone with the probability given in the routing table. Multipath routing and load balancing are therefore achieved with this approach. Forward ants are launched periodically from every LR to a random destination. After reaching the destination, the ant becomes a backward ant, and returns to the source node over the recorded path. Pheromone trails are left both ways (whose amount depends on path costs), which evaporate over time. The reason for using a different LL from the zone LL itself when routing is that perhaps there is an obstacle on the direct line, thus greedy routing along exact directions may fail. Ants are supposed to go around such obstacles, and their path is then decomposed into several straight line segments. Each such straight line segment represents a path between two zones, which can be achieved using any existing position based routing scheme (examples are the greedy scheme and greedy-face-greedy).

The authors did not present any experimental data on the performance of the proposed scheme, which appears very interesting and appealing. Division into zones requires network pre-processing, and for large networks with *n* nodes, the number of zones is $O(log^2 n)$. For n nodes, there are therefore $O(n \log^2 n)$ searches for table entries, and each of them needs a number of ants before the best neighbouring zone is selected. If a constant number of ants is used to test most of the candidate zones, there are $O(n \log^4 n)$ ants generated. In a network where topologies change frequently, the overhead of doing proactive routing may far overweight the benefits of doing so.

Figures 12 and 13 demonstrate the main steps in the proactive, zone grouping algorithm presented by Heissenbüttel and Braun [10]. The transmission radius of the nodes in the network is seen on the bottom left of the figure. Assume that all nodes that are within the transmission radius of each other can communicate directly. These links are not drawn in order to simplify the diagram. The large scale ad hoc network is divided into logical regions, as seen in figure 12. The partitioning of the networks is only depicted for logical region *X*, however. The routing table of LR *X* contains the next hops toward all of the logical regions. Only a few of these logical links are drawn in figure 13 for the purposes of clarity. The actual, physical routing between nodes is done using a greedy algorithm. There exists a direct logical link from LR *X* to LR *Y*, sine the greedy algorithm between them works. On the other hand, three logical links are necessary to reach LR *D*. As seen in figure 13, each logical link requires its own greedy algorithm. Therefore, the messages may be routed via other logical regions.

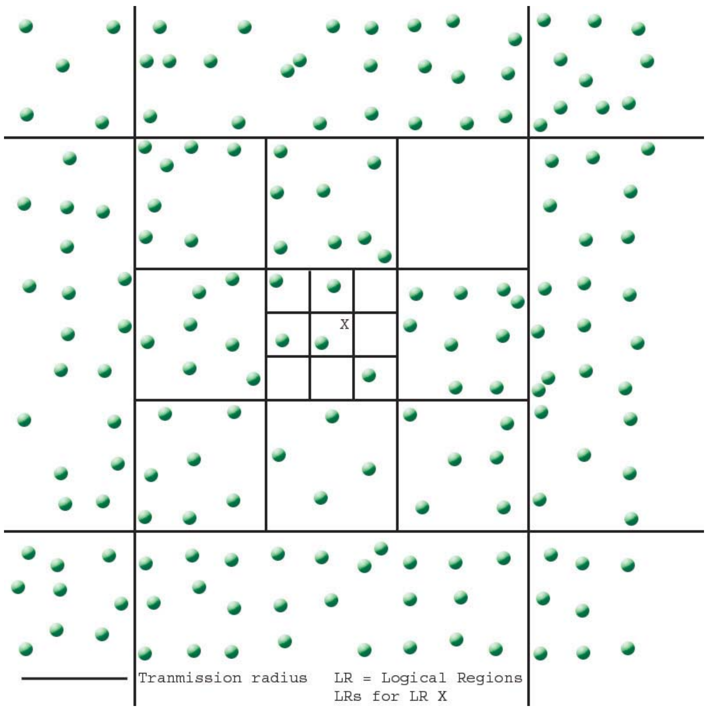


Figure 12 Logical regions as seen from *X*

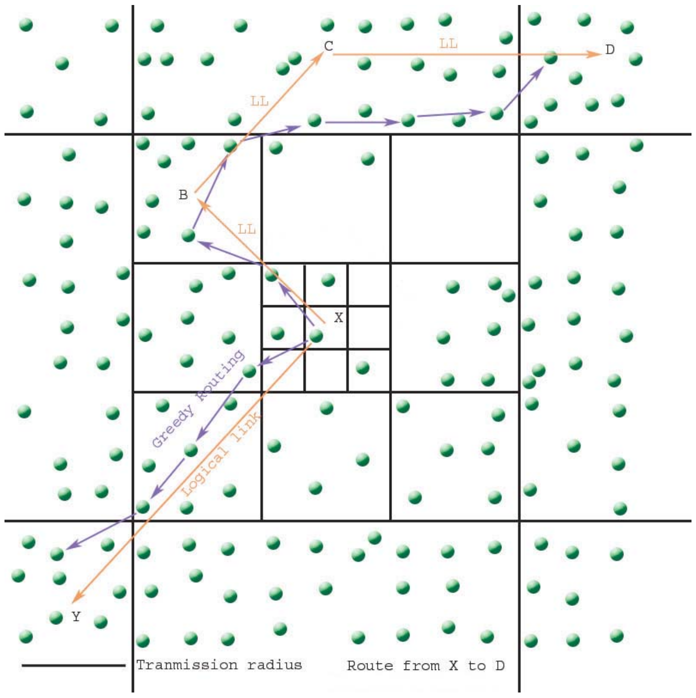


Figure 13 Logical links for *X*

6.2. Ant-Based Location Updates

Camara and Loureiro [4] proposed the GPSAL protocol which employs ants only to collect and disseminate information about node's locations in an ad hoc network. The destination for an ant could be the node with the oldest information in the routing table. Routing tables contain information about previous and current locations and timestamps of each node, and whether each node is fixed or mobile. When a host receives an ant it compares the routing table present in the ant packet with its routing table and updates the entries that have older information. The protocol, therefore, does not make use of the 'auto-catalytic' effect for finding shortest paths. Furthermore, a shortest path algorithm is applied to determine the best possible route to a destination. Therefore, the protocol assumes that a node knows a lot about the links currently present in the network, and a lot about the positions of other nodes, which certainly will not be true for large scale ad hoc networks. However, once location information is available, localized routing algorithms can be applied, such as greedy [6] or greedy-face-greedy. The algorithm is compared with a position and flooding based algorithm, and decreasing routing overhead is reported. However, the algorithm selected for comparison has significant and unnecessary communication overhead.

7. Multicasting in Ad Hoc Networks

Shen and Jaikaeo [23] described a swarm intelligence based multicast routing algorithm for ad hoc networks. In the multicasting problem, a source node sends the same message to several destination nodes. The sender and its recipients create a multicast group. There could be several multicasting groups running in the same network. In their algorithm each source starts its session by using shortest paths to each recipient (group member), which is obtained by flooding the message to the whole network, with each group member responding using a reverse broadcast tree (forwarding nodes are decided in this step). Ants are then used to look for paths with a smaller overall cost, that is, to create a multicast core. The cost of multicasting will be reduced if the number of forwarding nodes is reduced. This is achieved by using common paths to several members as much as possible, before splitting into individual or subgroup paths. In addition, each member which is not in the core periodically deploys a small packet that behaves like an ant to opportunistically explore different paths to the core. This exploration mechanism enables the protocol to discover new forwarding nodes that yield lower forwarding costs (the cost represents any suitable metric, such as number of retransmissions, total energy for retransmitting, load balancing, security level etc.). When a better path is discovered, a backward ant (using the memorized path) returns to its origin and leaves a sufficient amount of pheromone to change the route. To avoid cross cutting the 'roads', forwarding nodes keep the highest ID of the nodes that use it to connect to the core, and only the link to a higher ID forwarding node is allowed. Adaptation to ad hoc network dynamics is achieved by cancelling appropriate information whenever a link is broken, and using the best current pheromone trails to continue the multicast. Exploratory ants or periodic core announce messages will restore the connectivity if pheromone trails do not lead toward all group members. The experiments [23] are performed on the Qualnet simulator with 50 nodes, and the ant-based protocol is compared with a similar multicasting scheme that does not use ants, and with a simple flooding scheme. The new method performed better, however there exist other multicasting schemes (such as the one that constructs the core based tree first) which are not taken for comparison.

8. Data Centric Routing in Sensor Networks

Singh, Das, Gosavi, and Pujar [20, 21] proposed an ant colony based algorithm for data centric routing in sensor networks. This problem involves establishing paths from multiple sources in a sensor network to a single destination, where data is aggregated at intermediate stages in the paths for optimal dissemination. The optimal path amounts to a minimum Steiner tree in the sensor network. The minimum Steiner tree problem is a classic NP-complete problem that has numerous applications. It is a problem of extracting a sub-tree from a given graph with certain properties. The algorithm makes use of two kinds of ants, forward ants that travel from the sources to the destination, exploring new paths and gathering information, and backward ants that travel back to the sources from the destination to update the information in each sensor node as they move. A Steiner tree is obtained when the paths traced by forward ants merge into each other or reach the destination. This Steiner tree defines the paths along which data is to be transmitted from the sources to the destination. Because the forward ants move from the sources to the destination, they can also carry packets of data. In the proposed algorithm [20, 21], each sensor node *i* contains two vectors, the pheromone trails *ph*, and the node potential *pot*, with one entry per each of its neighbours. This node potential is a measure of the proximity of the node to the Steiner tree. The pheromone trails are all initialized to a sufficiently high value to make the algorithm exploratory, and the initial node potentials are based on heuristic estimates. Each sensor node also maintains a variable *tag*, which is initialized to zero, and contains information about how many ants have visited the node.

The total number of forward ants is equal to the number of source sensors, and each ant begins its path from a source sensor. Each such forward ant *m* maintains the tabu list *T* of nodes already visited, as well as a variable *pCost* that indicates the partial cost contributed by the ant's path to the Steiner tree. The list T is initialized to the source sensor where the ant is located, while *pCost* is set to zero. The probability of an ant moving from the current node *i* to its neighbour *j* is proportional to pheromone trail *ph*, and inversely proportional to potential *pot*. In order to prevent the formation of cycles, nodes in *T* that are already visited are excluded. The next location for ant *m* is chosen based on this probability, the new location *j* is pushed into *T*, and *tag* is examined. If *tag* is zero, indicating that location *j* is previously unvisited, the cost of the path *i*, is added to *pCost*. A non-zero value indicates that another ant has already visited the node, and therefore the cost of the path *i* is already incorporated in another ant's *pCost*. Under these circumstances, the forward ant *m* has already merged into an already existing path. It simply follows the previous ant's path to the destination node. The destination node, *d* contains a variable *cost*, the total cost of the Steiner tree path from the sources to *d*. When a forward ant enters the destination node, *d* it increments *cost* by an amount *pCost*. In the present version of the online algorithm, it is assumed that the total number of source nodes is known by the destination at the beginning of the computation. When all forward ants have arrived at the destination, backward ants are generated at the destination. There is a one-to-one correspondence between the forward and the backward ants, and a backward ant, also indexed as *m* acquires the list T of the corresponding forward ant *m*.

Each time a backward ant moves, it pops *T* to obtain the next destination. The backward ants carry a copy of the destination variable *cost*. This information is used to update the pheromones. Updating the tables of node potentials is somewhat more complex. A node's potential is considered low if it is either close to the destination, or brings a forward ant closer to the rest of the Steiner tree. In order to detect the cost of a node to *d*, each backward ant *m* maintains a variable *pCost* similar to a forward moving one, initially zero at the destination d, that gets incremented by an amount equal to *dis(i,j)* whenever a backward ant moves from *j* to *i*. When a backward ant is in any node, *pCost* is the cost of the path joining the node to *d*. In order to compute the cost of joining a node to another route, i.e. only a branch of the Steiner tree, another variable *rCost* is used by backward ants that are updated in the same manner. However, *rCost* is reset to zero each time a backward ant detects a split in a path leading to more than one branch of the Steiner tree. A split, leading to another branch is detected by examining the *tag* variable of a node *i*. If the previous node of the backward ant was *j*, then node *i* is a separate branch if *tag(i)<tag(j)*. A backward ant *m* leaving node *j* decrements the *tag(j)* variable. Backward ants travel back to the sources in *S* and reset these tag variables to zero for future ants. The updating rule for the potential is a linear combination of *rCost* and *pCost*. This updating is carried out only if the node potential gets lowered.

The experimental data showed that the ant based algorithm performed significantly better than the address-centric one, where shortest paths are used from each source sensor to the destination.

Conclusions

The dynamic and wireless nature of ad hoc networks has led to some modifications and new ideas in ant based routing schemes. The frequent edge creation and breakage has added the portion of exploratory ants that behave at random or with uniform probability, so that new paths are quickly discovered and reinforced, or new edges incorporated quickly into the path. Most articles exploit the one-to-all nature of message transmission, which gave the opportunity to multiply an ant and flood it throughout the network instead of simply following a path as in other considered communication networks. It also allowed nodes to overhear transmissions from neighbouring nodes and use them to update their pheromone tables.

While new opportunities for ad hoc networks are exploited in the proposed solutions, their experimental evaluation apparently was not done properly. Most authors only compare their methods with other weaker ant based methods, or with the standard version of the AODV protocol, without considering existing AODV improvements that might prove competitive. Also, position based schemes and routing schemes were not compared with the best existing position based methods. Therefore future articles are expected to provide a realistic evaluation of ant based routing in ad hoc networks, with emphasis on the primary question, whether the communication overhead imposed by using ants is worthwhile for obtaining gains in paths, especially in dynamic scenarios.

The need for improved accuracy of simulations exists also in ant based routing for communication networks. The routing problem becomes more challenging if constraints are added, for example to achieve quality of service. Flow control and admission control in routing are also important to incorporate. The existing reported simulation results that are encouraging are not done in real networks with real equipment. The primary concerns for routing are about convergence to a steady state, adaptation to changing environments, and oscillation [24, 25]. One of the interesting challenges for ant based routing is in their applications for routing and searching in Internet networks.

Further ant based methods can be expected soon; especially for position based routing. The recovery scheme proposed by Finn [6] is based on flooding up to *n* hops, hoping that a node closer to the destination than the current node will be found. This introduces a lot of flooding but still does not guarantee delivery. We believe that is worthwhile to consider the application of ants in search for such a node. A certain number of ants can be sent, each with a certain limited distance from the current node. The distances traveled by the ants could be set incrementally so that, if a closer node is not found by a certain time, new ants with a longer search range are sent. This is a preliminary idea, and obviously extensive simulation and modification is needed to get an acceptable version.

References

[1] E. Bonabeau, F. Henaux, S. Guerin, D. Snyers, P. Kuntz, G. Theraulaz, *Routing in telecommunication networks with 'smart' ant-like agents*, Proc. 2nd Int. Workshop on Intelligent Agents for Telecommunication Applications, Paris, France, July 1998.

[2] R.V. Boppana, S.P. Konduru, *An adaptive distance vector routing algorithm for mobile, ad hoc networks*, IEEE INFOCOM, Anchorage, Alaska, 2001.

[3] J. S. Baras and H. Mehta, *A Probabilistic Emergent Routing Algorithm for Mobile Ad Hoc Networks*, Proc. WiOpt, Sophia-Antipolis, France, March 2003.

[4] D. Camara, A. Loureiro, GPS/Ant-like routing algorithm in ad hoc networks, *Telecommunication Systems*, **18**, 1-3, 2001, 85-100.

[5] G. DiCaro, M. Dorigo, AntNet: Distributed stigmergetic control for communication networks, *J. of Artificial intelligence Research*, **9**, 1998, 317-365.

[6] G.G. Finn, Routing and Addressing Problems in Large Metropolitan-Scale Internetworks, *Research Report ISU/RR-87-180*, Inst. for Scientific Information, Mar. 1987.

[7] S. Guerin, *Optimisation multi-agents en environment dynamique: Application au routage dans les reseaux de telecommunications*, DEA Dissertation, University of Rennes I, France, 1997.

[8] M. Günes, M. Kähmer, and I. Bouazizi, *Ant-routing-algorithm (ARA) for mobile multi-hop ad-hoc networks - new features and results*, Proceedings of the 2nd Mediterranean Workshop on Ad-Hoc Net- works (Med-Hoc-Net'2003), Mahdia, Tunisia, 25-27, June 2003.

[9] M. Gunes, U. Sorges, and I. Bouazizi, *ARA - the ant colony based routing algorithm for manets*, Proc. ICPP Workshop on Ad Hoc Networks IWAHN, 2002, 79-85.

[10] M. Heissenbüttel and T. Braun, *Ants-based routing in large scale mobile ad-hoc Networks, Kommunikation in Verteilten Systemen KiVS03*, Leipzig, Germany, February 25-28, 2003.

[11] Z. J. Haas, M. R. Pearlman, and P. Samar. *The zone routing protocol (ZRP) for ad hoc networks*, July 2002. IETF Internet Draft, draft-ietf-manetzone-zrp-04.txt.

[12] M.T. Islam, P. Thulasiraman, R.K. Thulasiram, *A parallel ant colony optimization algorithm for all-pair routing in MANETs*, Proc. IEEE Int. Parallel and Distributed Processing Symposium IPDPS, Nice, France, April 2003.

[13] H. Matsuo, K. Mori, *Accelerated ants routing in dynamic networks*, 2nd Int. Conf. Software Engineering, Artificial Intelligence, Networking & Parallel Distributed Computing, Nagoya, Japan, 2001.

[14] S. Marwaha, C. K. Tham, D. Srinivasan, *A novel routing protocol using mobile agents and reactive route discovery for ad hoc wireless networks*, Proc. IEEE ICON, 2002.

[15] S. Marwaha, C. K. Tham, D. Srinivasan, *Mobile agent based routing protocol for mobile ad hoc networks*, Proc. IEEE GLOBECOM, 2002.

[16] C. Perkins, E. M. Royer, *Ad hoc on demand distance vector routing*, Proc. IEEE Workshop on Mobile Computing Systems and Applications, WSMA, Feb. 1999.

[17] S. Rajagopalan, C. Jaikaeo, and C.C. Shen, *Unicast Routing for Mobile Ad hoc Networks with Swarm Intelligence*, Technical Report #2003-07, University of Delaware, May 2003.

[18] M. Roth and S. Wicker, Termite: *Emergent ad-hoc networking*, Proceedings of the 2nd Mediterranean Workshop on Ad-Hoc Net- works (Med-Hoc-Net'2003), Mahdia, Tunisia, 25-27, June 2003.

[19] D. Subramaniam, P. Druschel, J. Chen, *Ants and reinforcement learning: A case study in routing in dynamic networks*, Proc. IEEE MILCOM, Atlantic City, NJ, 1997.

[20] G. Singh, S. Das, S. Gosavi, S. Pujar, "Ant Colony Algorithms for Steiner Trees: An Application to Routing in Sensor Networks", *Recent Developments in Biologically Inspired Computing*, Eds. L. N. de Castro, F. J. von Zuben, 2003, under preparation.

[21] S. Das, G. Singh, S. Gosavi, S. Pujar, "Ant Colony Algorithms for Data-Centric Routing in Sensor Networks", *Proceedings*, *Joint Conference on Information Sciences*, Durham, North Carolina, 2003.

[22] R. Schoonderwoerd, O.E. Holland, J.L. Bruten, Ant-like agents for load balancing in telecommunication networks, Proc. First ACM Int. Conf. on Autonomous Agents, Marina del Ray, CA, USA, 1997, 209-216.

[23] C.C. Shen, and C. Jaikaeo, Ad hoc Multicast Routing Algorithm with Swarm Intelligence, Technical Report #2003-08, DEGAS Group, Dept. of CIS, University of Delaware, 2003.

[24] T. White. Swarm intelligence and problem solving in telecommunications, *Canadian Artificial Intelligence Magazine*, Spring 1997.

[25] T. White and B. Pagurek, Towards multi-swarm problem solving in networks, Proc. Third Int. Conf. Multi-Agent Systems ICMAS, July 1998, 333-340.