Scientific Research and Essay Vol.4 (3), pp. 159-166, March, 2009
Available online at http://www.academicjournals.org/SRE
ISSN 1992-2248 © 2009 Academic Journals

Full Length Research Paper

The effect of AntNet parameters on its performance

R. Vallaie Kebria[1], S. Saffari Aman[2], S. S. Shamshirband[3], H. Shirgahi[4], M. Gholami[5] and B. Kia[6]

[1]Department of Computer Sciences, Islamic Azad University, Sari Branch, Sari, Iran.
[2]Khorasan-e-Razavi Telecom Mashhad, SC1 Center, Iran.
[3]Department of Computer, Islamic Azad University, Chalous Branch, Chalous, Iran.
[4]Department of Computer, Islamic Azad University, Jouybar Branch, Jouybar, Iran.
[5]Iranian academic center for education, culture and research, Mazandaran Branch, Sari, Iran.
[6]Department of Applied Sciences, Islamic Azad University, Chalous Branch, Chalous, Iran.

Accepted 28 January, 2009

AntNet is a routing algorithm which takes advantage of the way ants help each other to find food sources faster. Ants' communication in this respect is indirect by leaving pheromone trails as they move. The richness and thickness of the trail is proportional to the number of ants that have passed. This method of indirect courtesy guidance by affecting the environment is called stigmergy. Many network parameters that affect the performance of AntNet algorithm. However, there is little works on AntNet parameter analysis. In this paper we examine the effect of AntNet parameters on its performance. Simulation is used to estimate performance measures. Results show that changing one specific parameter may not lead to an improvement in performance unless other parameters are also changed, accordingly.

Key words: Routing algorithm, AntNet, AntNet performance parameter, agent.

INTRODUCTION

A wide variety of routing algorithms exists for communication networks. In traditional routing, routing tables are updated within exchanging routing information between different routers. Distance Vector (DV) examples of such algorithms.

A software agent can emulate the behavior of an ant and hence routing algorithms are developed that imitate the food source finding process of ants. The difference is that agents are used to find better paths between the message sending source and message receiver. As ants communicate indirectly by leaving pheromone trail as they move and this trail helps others to follow, agents update routing tables of the nodes which they go through. The routing tables, in turn, are used to better direct packets towards their destinations. AntNet is a hop-by-hop routing algorithm based on stigmergic property taken from ant colonies living style. Stigmergy is a form of indirect communication within modifications of the environment (Theraulaz, 1999; Pasteels, 1987; Buckers, 1992; Deneubourg, 1990).

There are two principal parts to take care of in the implementation of the traditional AntNet algorithm. The exploration for the purpose of discovering shortest path and the routing table update to help others to better find their destinations. These two actions are performed in an intermixed manner. Software agents are periodically generated to find better paths form the node in which the agent is generated to all other nodes of the network. The same agent has the responsibility to update routing tables of the nodes through which it passes. This can help the algorithm to converge faster an in a more efficient way.

In this paper, we investigated the how different network parameters affect the performance of the AntNet routing algorithm. The AntNet algorithm is simulated at the NS2. The rest of this paper is organized as follows. Section 2 discusses the structure of networks. Section 3 is a detail

*Corresponding author. E-mail: shamshirbanda@yahoo.com.

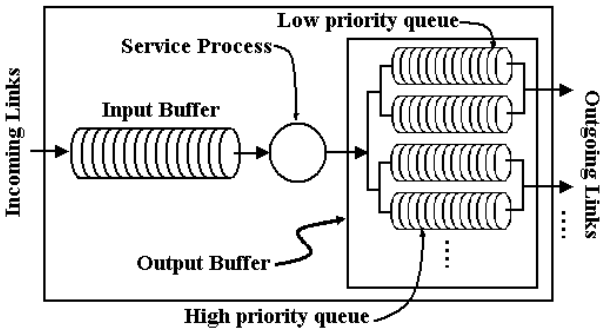

Figure 1. Buffers at the nodes. The input buffer consist of one queue and the output buffer consist of a low priority queue and high priority queue for every outgoing link (every neighbor).

examination of how AntNet works. In Section 4, the algorithm is simulated and parameters are estimate and the effects of variations in the parameters are investigated. Finally, the conclusion of this paper is given in Section 5.

STRUCTURE OF NETWORK IMPLEMENTS

We consider a network as a graph *G*(*N, L*) consisting of *N* nodes and *L* duplex links. The bandwidth and initial delay of every node is known in advance.

Every node has one buffer for incoming messages and one set of double buffers for outgoing messages. Each double buffer is composed of one queue for low priority outgoing messages and one for high priority outgoing messages. The number of double buffers is equal to the number of neighbors of this node (Figure 1).

There are two types of network packets, data and overhead packets, in the system.

Data packets are for exchanging data between different nodes. On the other hand, overhead messages are agents that travel within the network and have the responsibility of updating the routing tables.

There are two types of overhead messages (that is, mobile agents), forward ants and backward ants. The priority of forward ants is low while the priority of back-ward ants is high. In this research, both incoming and outgoing messages are processes in a first come first served (FCFS) manner (that is, all incoming and outgoing queues are FIFO). The length of all queues are considered unlimited and hence there will be no queue overflow.

When a node receives a packet from its neighbor, the packet is first stored in the input buffer. Packets in the input buffer are taken one at a time and the routing process will designate which output buffer should this packet be sent.

To be able to do the routing process AntNet needs. To use two data structures that are explained in the followings.

AntNet data structures

Mobile agents communicate with each other in an indirect form using two data structures T_k and M_k, in each node *k*.

Routing table T_k is organized as a matrix of probabilities as shown in Figure 2. The rows of this matrix are destination nodes and the columns are the neighbors of this node. For each destination *d* and every neighbor node *n*, T_k stores a probability value p_{nd} that is the probability of choosing *n* as the next hop.

$$\sum_{n \in N_k} p_{nd} = 1 \qquad N_k = \{neighbors\ (k)\} \qquad d \in [1, N]$$

For each destination *d,* table $M_k(\mu_d, \sigma_d^2, W_d)$ contains a moving observation window W_d, of maximum size W_{max}. W_d is used to compute the agents' best trip time t_{best_d}. The average μ_d and variance σ_d^2 represent the mean and variance of the trip times experienced by forward ants to move from node *k* to destination node *d*.

In (1) and (2) $t_{k \to d}$ represents the newly trip times of ants that come recently from node *k* to destination node *d* and the factor η weights the number of most recent samples that will really affect average and W_{max} is the maximum allowed size of the observation window (Gianni Di Caro, 1998).

$$\mu_d \leftarrow \mu_d + \eta(t_{k \to d} - \mu_d) \tag{1}$$

$$\sigma_d^2 \leftarrow \sigma_d^2 + \eta((t_{k \to d} - \mu_d)^2 - \sigma_d^2) \tag{2}$$

Di Caro and Dorigo in [Gianni Di Caro, 1998] prefer the formula (3) for relation between η and maximum size of moving observation W_{max}. We explain more about it later.

$$W_{\max} = \frac{5c}{\eta} \quad where \quad c < 1 \tag{3}$$

AntNet algorithm

AntNet can be described as follows (Gianni Di Caro, 1998):

i.) At regular intervals, from each node *s*, a forward ant $F_{s \to d}$ is launched randomly to each network node *d*.

ii.) The forward ants store their paths and traffic information to their stack $S_{s \to d}$, while traveling to destination nodes.

iii.) At each node *k*, each forward ant chooses the next node as follows:

a.) If all the neighboring nodes have not been visited, the next hop is one of the nodes that are not already visited.

b.) The selection is based on the routing table and the size of queue of the neighbor using (4) and (5).

$$p'_{nd} = \frac{p_{nd} + \alpha l_n}{1 + \alpha(|N_k| - 1)} \tag{4}$$

$$l_n = 1 - \frac{q_n}{\sum_{n'=1}^{|N_k|} q_{n'}} \tag{5}$$

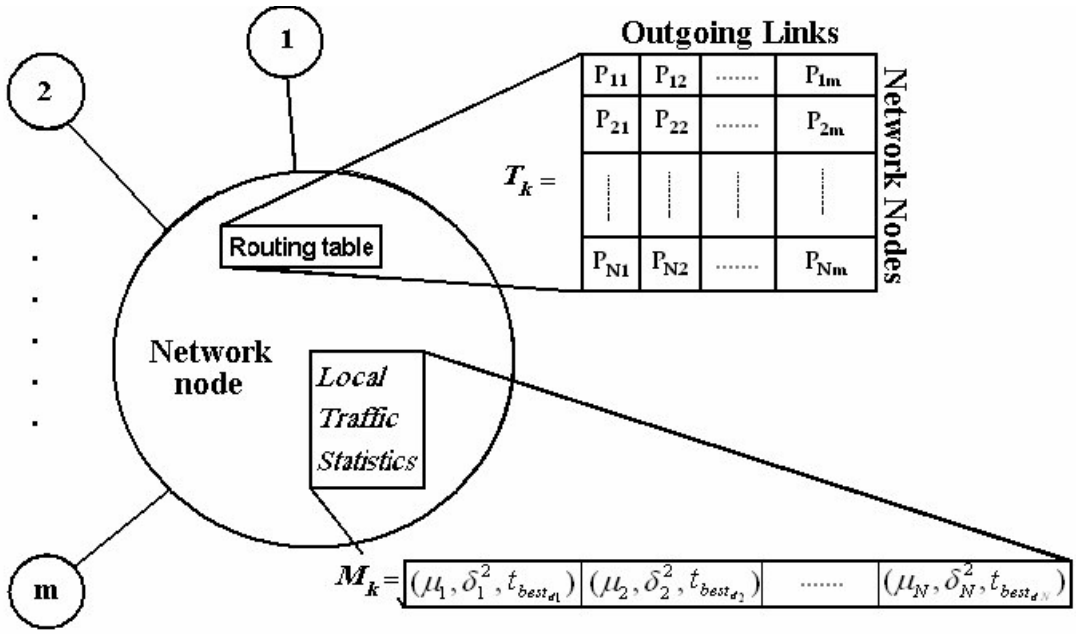

Figure 2. The data structures of node *k* with neighbors *x*, *y* and *z* and a network with *N* nodes: routing table (T_k) and statistics table (M_k).

N_k is the set of neighbors of the node *k* and l_n is the availability factor which is calculated according to Equation (5). The q_n in Equation (5) is the length of the queue of messages to be forwarded from node *k* to its neighbor node *n*. The value α in (4) weighs the importance of the instantaneous state of the node's queue with respect to the probability values stored in the routing table.

c.) If all the neighbors have been already visited, then next node is selected with equal probability.

iv.) If a cycle is detected, all nodes composing the cycle are popped from the ant's stack.

v.) After forward ant $F_{s\to d}$ is reached to node *d*, it produces backward ant $B_{d\to s}$. The backward ant returns to the source node by using the same path as the forward ant in opposite direction.

vi.) When the backward ant receives from neighbor *h* to node *k*, it can update two data structure T_k and M_k of node *k*.

a.) The mean μ_d and variance σ_d^2 entries of the local model of traffic M_k are modified using (1) and (2). The best value t_{best_d} of the forward ants trip time from node *k* to the destination *d* stored in the moving observation window W_d is also updated by the backward ant. If the newly observed forward ant's trip time $t_{k\to d}$ is less than t_{best_d}, then t_{best_d} is replaced by $t_{k\to d}$.

b.) The routing table T_k is updated by increasing the probability $p_{hd'}$ (the probability of choosing neighbor *h* when destination is d'). The probability $p_{hd'}$ of the selected neighbor is increased by the reinforcement value *r* as shown in (6). The *r* value will be computed later.

$$p_{hd'} \leftarrow p_{hd'} + r(1 - p_{hd'}) \tag{6}$$

The probabilities $p_{hd'}$ of other neighbors of node *n* for destination d' are decreased by negative reinforcement values computed in (7):

$$p_{hd'} \leftarrow p_{hd'} - rp_{hd'}, \forall n \neq h, n \in N_k \tag{7}$$

Hence, in AntNet, every path found by the forward ants receives a positive reinforcement value.

c.) The reinforcement value *r* used in Equations (6) and (7) is a dimensionless value in the range of (0, 1) and it is computed as (8):

$$r = c_1 \frac{t_{best_d}}{t_{k\to d}} + c_2 \frac{t_{\sup} - t_{best_d}}{(t_{\sup} - t_{best_d}) + (t_{k\to d} - t_{best_d})} \tag{8}$$

In (8), the reinforcement value *r* is adjusted using the squash function *s*(*x*) discussed later in Equations (11) and (12), $t_{k\to d}$ is the newly observed forward ant's trip time from node *k* to *d* and t_{best_d} is the best trip time experienced by the forward ants traveling to *d* over the observation window W_d. The value of $t_{\sup}$ is computed as (9):

$$t_{\sup} = \mu_d + \frac{\sigma_d}{\sqrt{1-\gamma}\sqrt{|W_{\max}|}} \tag{9}$$

Where γ is the confidence level. Equation (9) represent the upper limit of confidence interval for the mean μ_d, assuming that the mean μ_d and the variance σ_d^2 are estimated over W_{max} samples (Leon-Garcia, 1994).
In Equation (8), c_1 and c_2 are real values in the interval (0, 1) so that $c_1 + c_2$=1. On the other hand, it is suggested that c_1 be greater than c_2, because the first term of (8) is more important than second term and γ in Equation (9) is confidence level (Gianni Di Caro, 1998).

For more information see Kaeilbling (1996) and our earlier work (Saffari-Aman, 2008).

THE EFFECT OF PARAMETERS ON ANTNET PERFORMANCE

We have focused on some principal parameters that they have high effectiveness on AntNet implementation as below:

Inspecting AntNet complexity

One of the most important goodness criteria of every algorithm is its time complexity. At this part, we present the complexity of primary operations of a forward ant to travel between a source and a destination node.

At every node along the path from source toward destination, a forward ant needs to search its stack to find the next hop to be chosen among the node's neighbors. The worst-case complexity of searching through the stack is $O(1)$, if the stack is implemented as a combination of linked list and an additional array or a hash table. Moreover, the complexity of (4) is $O(N_k)$, because the value of probabilities is computed for every neighbor. There are some other computations for each ant to do. It is pushing the current node identifier and the current time. The complexity of these operations is also $O(1)$. Then the ant refers to the queue for one of the output links with the complexity of $O(1)$. If the maximum hopcount of the forward ant be M, in the worst-case, the forward ant should repeat all the above computations at everyone of M nodes. Thus, the worst-case complexity for a single ant to travel from a source to a destination node in AntNet is $O(MN_k)$.

The worst-case complexity for a backward ant to travel from its source to its destination is also $O(MN_k)$ to compute three operations at every node along its path. At first, it needs to pop the stack to know the next hop to travel. The pop operation in the stack is $O(1)$. Second, the backward ant must enter the one of output queues with complexity $O(1)$. Third, it should update routing table for every neighbor N_k for destination d which is $O(N_k)$. Furthermore, it has to do all computations for each M of the path.

If the total number of ants (forward and backward) that are generated is $\lambda > 1$ and the worst-case complexity of single forward or backward ant to travel from a given source to a given destination node is $O(MN_k)$, the worst-case complexity of AntNet is $O(\lambda MN_k)$.

To compute the complexity of AntNet for finding the shortest path in the worst-case, suppose a forward ant searches for one distinct path and a forward ant or backward ant pair updates only the routing tables at the source node for the given node. There is an equal probability of creating a forward ant with any one of the $N-1$ nodes as the destination. Hence, only one of $N-1$ forward ants is use to search for the shortest path between source and destination. Consider the number of paths between a source and a destination is m. The complexity of AntNet to search for shortest path (C_{AntNet}) with $m=m_{max}$ is $C_{AntNet} = O(mMN_kN)$, where m_{max} is upper bound of the maximum number of paths between source node and destination node in $G(N,L)$. The upper bound m_{max} on the total number of paths between source and destination node in $G(N,L)$ is m_{max} = [e(N-2)!] (Mieghem, 2005 Dhillon, 2006) e is a constant value equal to 2.718281828459.

Performance of AntNet

The computation of time $T_{i\to j}$ of a network packet (data packet/overhead packet) to travel from node i to node j is given in (10).

$$T_{i\to j} = \frac{q_i + size_{packet}}{C_{i\to j}} + D_{i\to j} \quad (10)$$

In (10) q_i is length of low priority queue at the i for link $i\to j$, the size of the packet is also $Size_{Packet}$, the capacity of link $i\to j$ is $C_{i\to j}$ and $D_{i\to j}$ is transmission time of link $i\to j$. The part of $Size_{Packet} / C_{i\to j}$ is the transmission time of a packet and the part of $q_j / C_{i\to j}$ is queuing delay that experienced by the packet while waiting at node i for the link $i\to j$.

For implementing AntNet, we have used the node model of the network according to Figure 1. At this model the queuing delay in (10) and parameter I_n in (4) is computed by the number of packets waiting in the low priority queue for a specified link (in FIFO order). Assume that every node is able to remove one packet from its high and low priority queue in the output buffer (for any output link) at a rate of 0.01 *ms*. Hence, we assume it negligible, in our simulations.

RESULTS

To inspect the AntNet implementation parameters, we compute some of criteria for our network model such as End-to-End delay, the average hopcount of the paths that

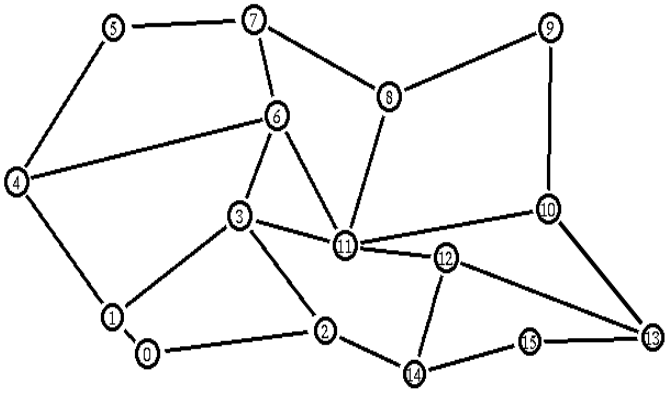

Figure 3. The 16-node network simulation model.

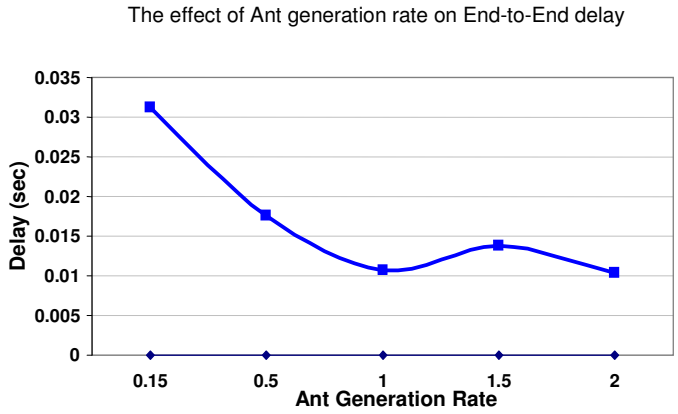

Figure 4. The effect of ant generation rate on End-to-End delay.

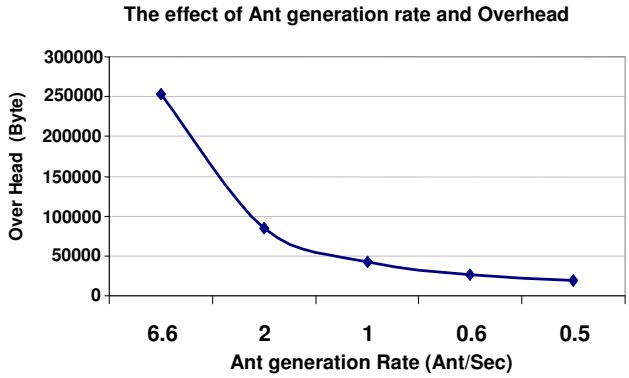

Figure 5. The effect of ant generation rate on routing overhead.

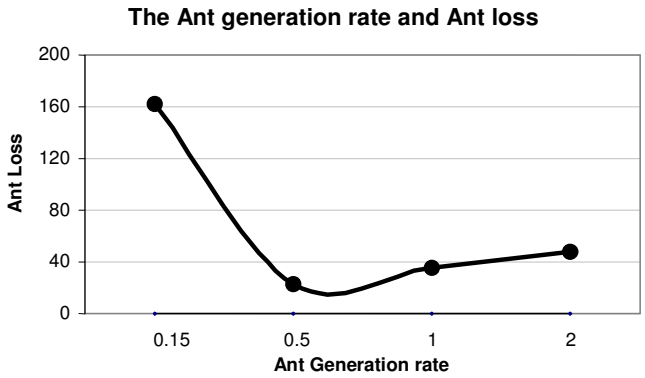

Figure 6. The effect of ant generation rate on ant loss.

the data packets use to travel between source and destination node, receiving throughput and overhead in 16 nodes network as shown in Figure 3.

The simulation is divided in two parts, training period and testing period. Ant packets generate along the training period, however both overhead packets and data packets generate along the test period. The total time of our simulations is 25 s. We have considered 15 s for training period. Node 0 is the source node and node 9 is the destination node. The size of data packets is 512 bits and bandwidth of links varies from 1.5 to 6 MBps. We have also tried the AntNet with the several of ant generation rate.

Note that, the AntNet algorithm implementation is dynamic, that is, links can be broken and reconnected during the simulation.

The effect of ant generation rate

To study ant generation rate of AntNet algorithm, we used different rates in our simulations, like 0.15, 0.5, 1, 1.5, 2 (for example, 0.15 means that every 0.15 s one ant is generated).

The average of End-to-End delay shows convergence rate of algorithm. According Figure 4 when we decrease ant generation rate, total delay is improved. However the improvement is not very notable.

It is clear that if we decrease the number of agents created, a higher bandwidth is left for transmitting actual data.

According to Figure 5, when ant generation rate is decreased the routing overhead is decreased, too. The other problem is finding the optimal rate of generating agents, in order for the agents not to be lost. In Figure 6 we have shown the effect of the ant generation rate on ant generation rate, the ant loss rate decreases at the same rate.

One more question is the effect of ant generation rate on packet loss ratio. In Figure 7, we have shown packet loss rate versus ant generation rate.

In the following, the effect of ant generation rate is studied on the number of hops from source to destination. It is desirable to reduce the number of hops as there are

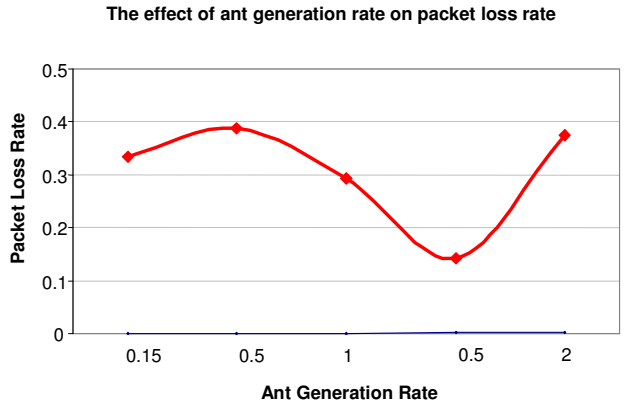

Figure 7. The effect of ant generation rate on packet loss rate.

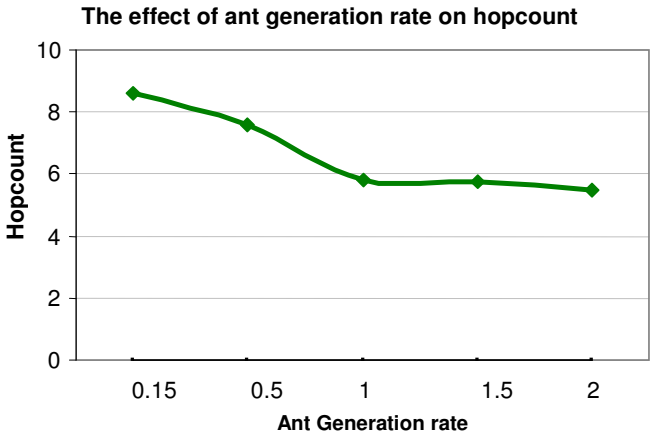

Figure 8. The effect of the ant generation rate on hopcount.

some computations to perform for each hop. Figure 8 shows the results of simulation for this purpose. Although, the figure doesn't show a large difference in hopcounts between lower number and higher number of ant generation rates, but as the number of packets are high the total hopcount difference becomes noticeable. size of the network, AntNet performance decrease, because the ants have to travel longer distances. As a result of traveling longer distances, ant's carrying information is outdated and there is a higher possibility for every ant to be lost. Our simulation results showed that when we increase the size of network from N = 16 to N = 25, almost all Quality of Service (QoS) criteria is declined. On the other hand, as Figure 9 shows, increasing the bandwidth of links decreases the End-to-End delay and increases the convergence rate.

According to Figure 10, by increasing the bandwidth the receiving throughput is increased. Of course, this

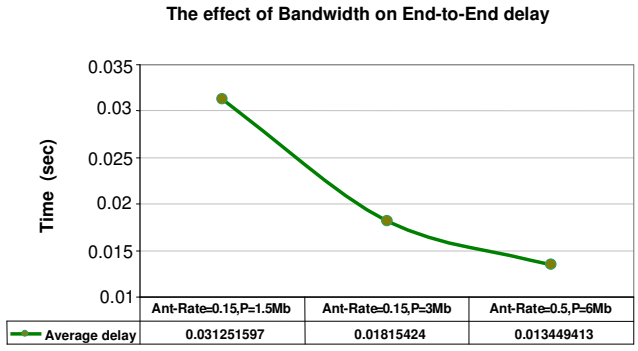

Figure 9. The effect of bandwidth on End-to-End delay.

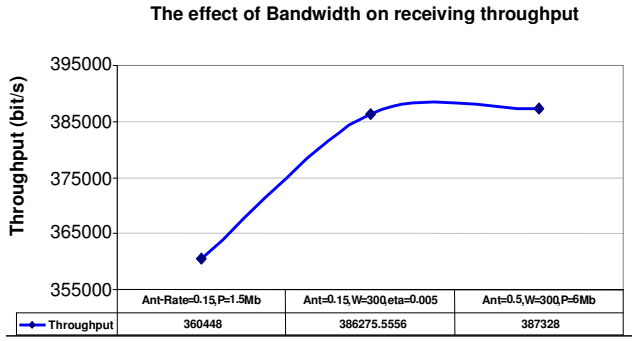

Figure 10. The effect of bandwidth on receiving throughput.

increase in throughput continues up to a certain value of the bandwidth. Further increases in the bandwidth do not have a positive effect on the throughput.

The effect of bandwidth and number of nodes

AntNet has a scalability problem, that is, by increasing the size of the network, AntNet performance decrease, because the ants have to travel longer distances. As a result of traveling longer distances, ant's carrying information is outdated and there is a higher possibility for every ant to be lost. Our simulation results showed that when we increase the size of network from *N=16* to *N=25*, almost all Quality of Service (QoS) criteria is declined. On the other hand, as Fig. 9 shows, increasing the bandwidth of links decreases the End-to-End delay and increases the convergence rate.

According to Figure 10, by increasing the bandwidth the receiving throughput is increased. Of course, this increase in throughput continues up to a certain value of the bandwidth. Further increases in the bandwidth do not have a positive effect on the throughput.

Note that by increasing bandwidths both throughput and

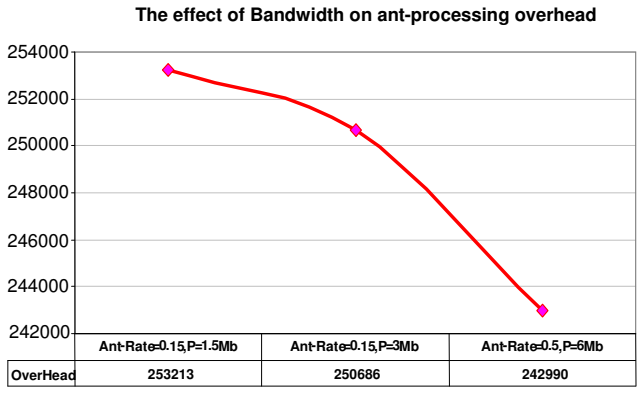

Figure 11. The effect of bandwidth on ant-processing overhead.

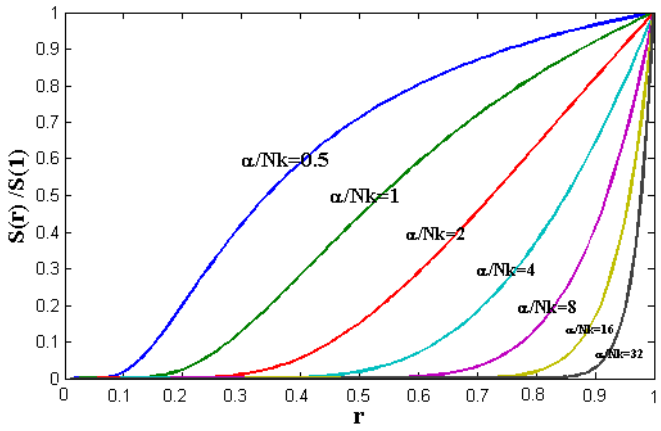

Figure 12. The squash function $s(x)$ for different values of the coefficient $\frac{a}{|N_k|}$

delay are improved. However, this causes fewer ants to be lost and hence increases the overall ant-processing overhead (Figure 11).

The effect of squash function

As (Gianni Di Caro, 1998) suggests, all paths are discovered by forward ants that receive the positive reinforcement value r. r is computing by using Equation (8) and it is then adjusted using the squash function $s(x)$ given in [11]. The squash function $s(x)$ is defined in AntNet to be used in computing the r value by using Equation (12). The purpose of function $s(x)$ is to decrease the smaller values of reinforcement r in order to lessen its effect on routing table update and increase the larger values of r in order to increase its effect on the routing table update (Gianni Di Caro, 1998).

$$s(x) = \frac{1}{1+\exp(\frac{a}{x|N_k|})}, where\ x \in (0,1], a \in R^+ \quad (11)$$

$$r \leftarrow \frac{s(r)}{s(1)} \quad (12)$$

The coefficient $\frac{a}{|N_k|}$ identifies the dependence of squash function $s(x)$ on the number of neighbors N_k of the node k. Figure 12 shows the effect of the coefficient $\frac{a}{|N_k|}$ on r. If its value is less than 1, then even low values of r increases based on $s(x)$. Hence, the value of parameter a should be chosen such that the coefficient $\frac{a}{|N_k|}$ is greater than 1.

The effect of moving observation window size and η

The value of W_{max} should be large enough to store the trip times of all forward ants generated by the node as well as the trip times of forward ants received.

The parameter η is used to estimate the mean μ_d and the variance σ_d^2 by using the exponential model as shown in (1) and (2). This parameter represents how many of the previous forward ants trip times have noticeable effect on the mean and average value. Furthermore, the value of parameter η is chosen to be very small say η = 0.002 so that large number of sample trip times of forward ants are used to calculate the mean and variance in (1) and (2). In our experiments W_{max} took the values 10, 20, 50, 300, 500, and 700.

As mentioned above, Di Caro and Dorigo in (Gianni Di Caro, 1998) presented the Equation (3) for the relation between parameters η and the maximum size of moving observation window, W_{max}. Our simulation results showed that this formula can be interpreted with respect to the number of possible different paths in the network. In addition, our simulation results show that with a small moving observation window, even if ant generation rate increases, the performance of the algorithm remains the same or may decrease. However, the AntNet algorithm is robust to changes on above mentioned parameters and other effecting parameters.

Detailed study of Figures 4 - 12, reveals that changing one specific parameter may not lead to an improvement in performance unless other parameters are also modified, accordingly.

Conclusion

AntNet is recently used as a method for solving optimization problems. In fact, the coupling of various parame-

ters is the inherent cause of complexity in the AntNet algorithm and changing one specified parameter may not lead to an improvement in performance until other parameters are also modified.

Based of our simulation, the performance of AntNet degrades as the network size is increased. The AntNet algorithm is robust against changes during the training and converges to a good solution even at low ant generation rates. The performance of the algorithm can be improved by using a large value of the parameter *a* in the squash function *s*(*x*).

The complexity of AntNet algorithm can be reduced by a number of methods, but this generally comes at the expense of AntNet performance.

REFERENCES

Buckers R (1992). Trails and U -turns in the selection of the shortest paths by the ant Lasius Niger, J. Theor. Biol. 159: 397–415.

Deneubourg J-L (1990). The selforganizing exploratory pattern of the argenttine ant, J. Insect Behavior, 3: 159–168.

Dhillon SS (2006). Performance analysis of the AntNet algorithm, Computer Network, (2006), doi:10.1016/j.comnet, 11: 002.

Gianni Di Caro (1998), "AntNet: distributed stigmergetic control for communication networks", J. Art. Intelligence Res. 9: 317–365.

Kaeilbling LP (1996). Reinforcement learning: a survey, J. Art. Intelligence Res. 4: 237–285.

Leon-Garcia A (1994). Probability and Random Process for Electrical Engineering, Addison-Wesley Publishing Company, ISBN 0-201-50037-X.

Pasteels JM (1987). Self-organization mechanisms in ant societies (I): trail recruitment to newly discovered food sources, *Experientia Supplementum* 54: 155–175.

Saffari-Aman S (2008). A Novel Approach to Distributed Routing by Super-AntNet", IEEE Congress on Evolutionary Computation, Hong Kong, Page(s): 2151–2157.

Theraulaz G (1999). A brief history of stigmergy, Artificial Life, *Special* Issue on Stigmergy, pp. 97–116.

Mieghem VP (2005). Performance Analysis of Computer Systems and Networks, Cambridge university Press.