IDENTIFYING M-SHORTEST OPTIMAL PATHS AND INCORPORATING LOAD BALANCING OVER THESE PATHS WITH NEGLIGIBLE OVERHEAD.

[1]SHUCHITA UPADHYAYA , [2]RICHA SETIYA

[1]Department Of Computer Sc & Applications, Kurukshetra University, Kurukshetra, India
[2]Department Of Computer Sc & Engg., N.C.College Of Engineering, Israna, India

ABSTRACT

Antnet is an agent based routing algorithm based on real ants' behavior. Ants are able to find shortest path to food source .In real life, ants drop some kind of chemical substances to mark the path that they used. Then on their way back they choose the path with the highest pheromones which becomes the shortest path. But Antnet Algorithms may cause the network congestion and stagnation. Here, M-shortest optimal paths are proposed with negligible overhead in spite of single optimal path in Antnet routing algorithm and consequently the load is distributed over those paths with some strategy.

Keywords: *Routing, Antnet, agent, swarm Intelligence, Adaptive*

1. INTRODUCTION

Today with the fast growth of Internet everybody wants to get connected to the internet. Millions of people use the Internet for daily business all over the world. Today the Internet has become very large, complex and dynamic. Failures and challenges occur at every step because of traffic flow from one part of network to another part. Routing is the process of selecting paths in a network to send traffic. Routing is an important aspect of network communication which affects the performance of any network, since other characteristics of the network like throughput; reliability and congestion depend directly on it. In packet switching networks when a packet travels from a source to destination, it has to pass through a number of networks with varying characteristics. So an ideal routing algorithm is one which is able to deliver the packet to its destination with minimum amount of delay. It must be adaptive and intelligent enough to make the decisions. The growing size and increasing demands of Internet impelled the study of more powerful routing algorithms which can optimize the flow of traffic.
The routing algorithms currently in use (e.g. OSPF, RIP, and BGP) are not sufficient to tackle the increasing complexity of such networks. They are not adaptive, intelligent and fault intolerant. The routing tables in them are updated by exchanging routing information between the routers. Different routing protocols use different approaches to exchange the routing information. There are mainly two approaches for routing algorithms, distance-vector algorithms and link-state algorithms. Distance vector algorithms use the Bellman algorithm. This approach assigns a number, the cost, to each of the links between each node in the network. Nodes will send information from one point to another point via the path that result in the lowest total cost. In link-state algorithms for example, Open Shortest Path First (OSPF), the routers exchange link-state information by flooding the link state packets. The link state updates are generated only when the link status changes. Once a node has obtained topology information of the entire network, Dijkstra's algorithm is generally used to compute the shortest path.

The two main performance metrics that are affected by the routing algorithm are throughput and average packet delay. Coordination is needed between nodes. Nodes and links can fail, and congestion can arise in some areas. Thus, the routing algorithm needs to modify its routes,

Vol6. No1. (pp 009 - 016)

redirecting traffic and updating databases very quickly and adaptively.

In recent years, a new kind of routing protocols influenced by software agents called Ant Based routing is developed. S. Appleby and S. Steward were the first ones to introduce the concept of software agents used for control in telecommunication networks [1].This approach of software agents was modified for routing problem by R. Schoonderwoerd[2].The research process continued and it was applied to connection oriented networks [3]. Ant based routing was then applied to packet based connection less systems [4].This agent based approach was further researched and was modified for adaptive routing [5].Swarm intelligence provides a promising alternative to traditional routing algorithms by utilizing mobile software agents for network management.

Although, an ant [1],[2] is a simple and unsophisticated creature, collectively a colony of ants can perform useful tasks such as building nests, and foraging (searching for food)[1],[2],[3].What is interesting is that ants are able to discover the shortest path to a food source and to share that information with other ants through stigmergy [1]-[5]. Stigmergy is a form of indirect communication used by ants in nature to coordinate their problem-solving activities. Ants achieve stigmergic communication by laying a chemical substance called pheromone.

Ant Colony Optimization (ACO) [5]-[9] is a family of optimization algorithms based on real ants' behavior. ACO is inspired by the foraging behavior of ant colonies, where in they are able to find shortest path to food source. It has been observed that of available routes, ants find shortest route to food source. In real life, ants deposit some kind of chemical substances to mark the path that they used. Then on their way back they choose the path with the highest pheromones which becomes the shortest path.

AntNet is an Ant Colony Optimization (ACO) meta heuristic for data network routing proposed by Gianni Di Caro and Marco Dorigo [6]-[9]. In this network routing algorithm, a group of mobile agents (or artificial ants) build paths between pair of nodes, exploring the network concurrently and exchanging obtained information to update the routing tables. This information is also used to direct the data packets towards their destination.

2. ANTNET ROUTING ALGORITHM

In antnet [6]-[9] software agents explore the network to find the optimal paths from the randomly selected source destination pairs. Moreover while exploring the network ants update the probabilistic routing tables and build statistical models of the nodes local traffic. Ants use these tables to communicate with each other.

2.1 Data Structure maintained at each node

Routing table T_k is a local data-base that helps router to decide where to forward data packets. It contains the information which specifies the next (neighbor) node that should be taken by a data packet to get to any possible destination in the network. Each routing table is organized as a set of

- All the possible destinations (all the nodes in the network).
- The probabilities to reach these destinations through each of the neighbors of the node.

T_k stores a probability value P_{nd} which express the goodness of choosing n as next node when the destination node is d

$$\Sigma P_{nd} = 1,\ d \in [1,N], N_k = neighbors(k)$$
$$n \in N_k$$

Probability value P_{nd} represents pheromone concentration along the link from node k to neighbor node n for destination node d.

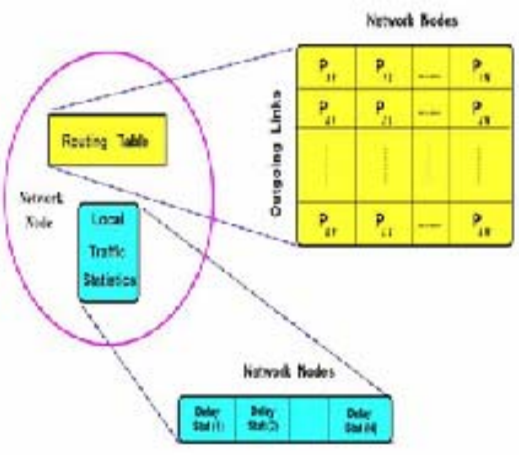

Figure 1

Local traffic statistics is a second data-structure that each node has. The main task of this structure

is to follow the traffic fluctuations as seen by local node i. For each destination d in the network, an array M_i (μ_d, σ_d^2, W_d) contains an estimated mean μ_d, an estimated variance σ_d^2 computed over the times experienced by the artificial ants, and a moving observation window W_d.

2.2 The AntNet algorithm

The operation of AntNet as proposed by Di Caro and Dorigo is based on two types of agents:

• Forward Ants who gather information about the state of the network, and
• Backward Ants who use the collected information to update the routing tables of routers on their path.

The AntNet algorithm is described as follows.

1. At regular time intervals Δt from every node s, forward ant $F_{s\to d}$ is launched toward a destination node d to discover a possible, low-cost path to that node and to explore the load status of the network. The forward ants make use of the same priority queues as used by data packets.

Destinations are locally selected according to the data traffic patterns generated by the local workload: if f_{sd} is a measure (in bits or in number of packets) of the data flow $s \to d$, then the probability of creating at node s a forward ant with node d as destination is

$$p_{sd} = \frac{f_{sd}}{\sum_{i=1}^{N} f_{si}} \quad -$$

.

2.The forward agents create a stack where they store memory of their paths and of the traffic conditions found. The identifier of every visited node i and the time elapsed since the launching time to arrive at this i-th node is pushed onto the memory stack.

3. At each node i, each traveling agent moving towards its destination d selects the node j to move to, choosing among the unvisited neighbor nodes, or over all the neighbors in case all of them had been previously visited. The neighbor j is selected with probability (goodness) P_{ijd}, computed as normalized sum of the pheromone τ_{ijd} with a heuristic value η_{ij} taking into account the state (the length) of the j-th link queue of the current node i:

$$\tau_{ijd} + \alpha\eta_{ij}$$

$$P_{ijd} = \frac{}{1+\alpha\,(|N_i| - 1)}$$

The heuristic value η_{ij} is a [0, 1] normalized value proportional to the length q_{ij} (in bits waiting to be sent) of the queue of the link connecting the node i with its neighbor j.

$$\eta_{ij} = 1 - \frac{q_{ij}}{\sum_{i=1}^{N} q_{i1}}$$

The value of α weighs the importance of the heuristic value with respect to the probability values stored in the routing table. η_{ij} reflects the instantaneous state of the node's queues, and assuming that the queue's consuming process is almost stationary or slowly varying, η_{ij} gives a quantitative measure associated with the queue waiting time.

4. If a cycle is detected, that is, if an ant is forced to return to an already visited node, the cycle's nodes are popped from the ant's stack and all the memory about them is destroyed. If the cycle lasted longer than the lifetime of the ant before entering the cycle, (that is, if the cycle is greater than half the ant's age) the ant is destroyed.

5. When the destination node d is reached, the agent $F_{s\to d}$ generates backward agent $B_{d\to s}$, transfers to it all of its memory, and dies.

6. The backward ant takes the same path as that of its corresponding forward and, but in the opposite direction. At each node i along the path it pops its stack to know the next hop node. Backward ants also don't share the same link queues as data packets; they use higher priority queues, because their task is to quickly propagate to the routing tables the information accumulated by the forward ants.

7. Arriving at a node i coming from a neighbor node f, the backward ant updates the two main data structures of the node, the local model of the traffic M_i and the routing table T_i, for all the entries corresponding to the (forward ant) destination node d.

2.3 Update traffic model M_i

M_i is updated with the values stored in the stack memory. The time elapsed to arrive (for the forward ant) to the destination node d starting from the current node i is used to update the mean μ_d,

variance estimates σ^2_d and the best value over the observation window W_d. The moving observation window $W_{d,}$ of size W_{max} , represents an array containing the trip times of last W_{max} forward ants that travel from the node i to the destination d. The moving observation window W_d is used to compute the best trip time t_{bestd} i.e., the best trip time experienced by a forward ant travelling from the node i to the destination d among the last W_{max} forward ants that travel from the node i to the destination d.

The mean value of this time and its dispersion can vary strongly, depending on the traffic conditions: a poor time (path) under low traffic load can be a very good one under heavy traffic load. The statistical models had to be able to capture this variability and to follow in a robust way the fluctuations of the traffic.

The estimated mean and variance are updated as follows:

$$\mu_{id} \leftarrow \mu_{id} + \square\ (o_{i\rightarrow d} - \mu_{id})$$

$$\sigma^2_{id} \leftarrow \sigma^2_{id} + \square\ ((o_{i\rightarrow d} - \mu_{id})^2 - \sigma^2_{id})$$

where $o_{i\rightarrow d}$ is the observed ant's trip time from node i to destination d. The factor $\square$ weighs the number of most recent samples that will really affect the average.

2.4 Update pheromone matrix T_i

The reinforcement $r = r(T,Mi)$ is defined to be a function of the goodness of the observed trip time as estimated on the basis of the local traffic model. r is a dimensionless value, $r \in (0,1]$, used by the current node i as a positive reinforcement for the node f the backward ant $B_{d\rightarrow s}$ comes from. Reinforcement value r takes into account some average of the so far observed values and of their dispersion to score the goodness of the trip time T, such that the smaller T is, the higher r is.

The backward ant $B_{d\rightarrow s}$ moving from node f to node i increases the pheromone values $\tau_{ifd'}$

$$\tau_{ifd'} \leftarrow \tau_{ifd'} + r.(1 - \tau_{ifd'})$$

Pheromones $\tau_{ifd'}$ for destination d' of the other neighboring nodes j, $j \in N_i$, $j \neq f$, evaporate implicitly by normalization. That is, their values are reduced so that the sum of probabilities will still be 1.

$$\tau_{ifd'} \leftarrow \tau_{ifd'} + r.(\tau_{ifd'}), j \in N_{i\,,}\ j \neq f$$

2.5 Limitations

Although Experiments of AntNet have shown very promising results, antnet has outperformed under different experimental conditions with respect to other dynamic routing algorithms e.g. RIP, OSPF. Still there are some problems with this adaptive algorithm. One of the major problems is that the network gets trapped because a node prefers a link with higher probability to a destination when choosing an outgoing link say n_o to send a packet. If link n_o keeps good condition for a long time, its probability to that destination will be very high. Such a condition may cause congestion of n_o; it also reduces probability of selecting other paths. Hence the node will stuck to this outgoing link and loose its adaptive ability. This is called the problem of "stagnation". Stagnation is reached when a node reaches its convergence. Stagnation is a very critical problem for any network because

1) n_o may lose its optimality if it gets congested;
2) If the network gets fails at any time then the most preferred path n_o may become unavailable
3) Other non-optimal paths may become optimal due to changes in network topology

Many researchers [11]-[13] have tried to provide the solution for the same. These methods are

- Evaporation
- Aging
- Limiting and Smoothing Pheromone
- Privileged Pheromone Laying
- Pheromone-Heuristic Control

But all the methods are very complex and needed extra overhead. Here, multiple optimal paths are identified so that the congestion and stagnation over single optimal path can be avoided.

3. WHY MULTIPLE PATHS?

Most of the studies of Antnet Routing have been aimed at producing a single optimum path. With this approach, only a single path is established between a source-destination pair, even if there exist some alternative, possibly suboptimal paths. With multiple –path routing, the traffic between the same source and destination is assigned to different paths and load is divided over the selected paths according to proposed strategy. Hence, in case of congestion and stagnation, the proposed approach is likely to perform better than other routing schemes.

4. FINDING MULTIPLE PATHS

According to Original Antnet Algorithm, **Routing table** is a local data-base that helps router to decide where to forward data packets. It contains the information which specifies the next (neighbor) node that should be taken by a packet to get to any possible destination in the network. The columns of the table represent the neighbor nodes and rows represent the possible destinations.

- The column wise values in the table are picked up and a sorting algorithm is executed on these values.
- The sorted values ranging from higher to lower are stored in a temporary array.
- The difference d, amongst the adjacent values is calculated and is compared to some threshold value say p_m.
- If the difference d is less than p_m then those values are selected and comparison amongst the adjacent values is continued until difference is greater than p_m.
- Otherwise at the very first occurrence of difference greater than p_m , the comparison is stopped and the corresponding value(s) in the array is(are) selected.
- The interfaces corresponding to these values are stored in a new routing table which will have the same structure as the original one, but obviously, the new table will have less number of rows.

5. EXAMPLE NETWORK

Fig.2 Shows a simple network which will be used to show how the routing algorithm works.

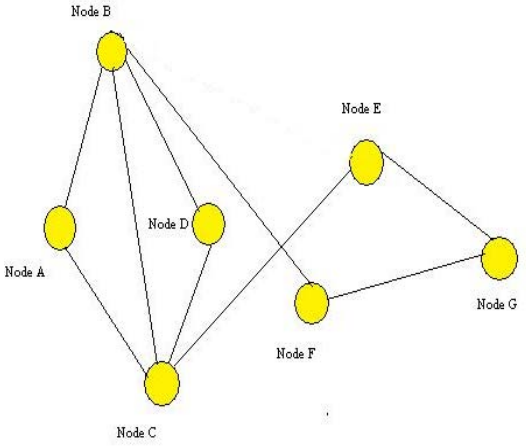

Fig. 2 Example Network

Consider a node B in the network having 4 neighbors. The routing table of node B, for every destination k will have probabilities. The probabilities are assumed as below in the table 1.

Table 1 AntNet routing table at node B

	All Network Nodes(Possible Destinations)					
Outgoing Links	P_{AE}=.25	P_{AG}=.39	P_{AF}=.8	P_{AA}=.34	P_{AD}=.56	P_{AC}=.47
	P_{CE}=.45	P_{CG}=.32	P_{CF}=0	P_{CA}=.34	P_{CD}=.23	P_{CC}=.43
	P_{DE}=.17	P_{DG}=.11	P_{DF}=.1	P_{DA}=.20	P_{DD}=.01	P_{DC}=.03
	P_{FE}=.13	P_{FG}=.28	P_{FF}=.1	P_{FA}=.12	P_{FD}=.20	P_{FC}=.07

The value of p_m = 0.1(say)

Procedure:

Ist Column

- The values of Ist column in the table are picked up and a sorting algorithm is executed.
- The sorted values ranging from higher to lower are stored in a temporary array, say A_i, i.e. A_i ={0.45,0.25, 0.17,0.13}
- Difference d =0.45-0.25=0.2
- Then comparison is made if $(d < p_m)$ i.e. if (0.2<0.1) ,which is false, hence {0.45} is selected.
- The interfaces corresponding to 0.45 are stored in a new proposed routing table.

2nd column

- The values of 2nd column in the table are picked up and a sorting algorithm is executed.
- The sorted values ranging from higher to lower are stored in a temporary array, say
- A_i, i.e. A_i ={0.39,0.32,0.28,0.11}
- Difference d =0.39-0.32=0.07
- Then comparison is made if $(d < p_m)$ i.e. if (0.07<0.1) ,which is true, hence

{0.39,0.32} are selected and the process is continued until ($d > p_m$)

- d=0.32-0.28=0.04
- if $(d < p_m)$ i.e. if (0.04<0.1) ,which is true, hence {0.39,0.32,0.28} are selected.
- d=0.28-0.11=0.17
- $d > p_m$,The process is discontinued.
- The interfaces corresponding to {0.39, 0.32, 0.28} are stored in a new proposed routing table.

3rd column

- The values of 3rd column in the table are picked up and a sorting algorithm is executed.
- The sorted values ranging from higher to lower are stored in a temporary array, say $A_{i,}$ i.e. A_i ={ 0.8,0.1,0.1,0.0}
- Difference d =0.8-0.1=0.7
- Then comparison is made if $(d < p_m)$ i.e. if (0.7<0.1) ,which is false, hence {0.8} is selected. The process is discontinued.
- The interface corresponding to 0.8 is stored in a new proposed routing table.

4th column

- The values of 4th column in the table are picked up and a sorting algorithm is executed.
- The sorted values ranging from higher to lower are stored in a temporary array, say $A_{i,}$ i.e. A_i = {0.34, 0.34, 0.20, 0.12}
- Difference d =0.34-0.34=0.0
- Then comparison is made if $(d < p_m)$ i.e. if (0.0<0.1) ,which is true, hence {0.34,0.34} are selected and the process is continued until $(d > p_m)$.
- d=0.34-0.20=0.14
- if $(d < p_m)$ i.e. if (0.14<0.1),which is false, hence {0.34,0.34} are selected and the comparison is discontinued.
- The interfaces corresponding to {0.34, 0.34} are stored in a new proposed routing table.

5th column

- The values of 5th column in the table are picked up and a sorting algorithm is executed.
- The sorted values ranging from higher to lower are stored in a temporary array, say $A_{i,}$ i.e. A_i ={ 0.56,0.23,0.20,0.01}
- Difference d =0.56-0.23=0.33
- Then comparison is made, if $(d < p_m)$ i.e. if (0.33<0.1) ,which is false, hence {0.56} is selected and the process is discontinued
- The interface corresponding to 0.56 is $P_{AD.}$
- The interface corresponding to 0.56 are stored in a new proposed routing table.

6th column

- The values of 6th column in the table are picked up and a sorting algorithm is executed.
- The sorted values ranging from higher to lower are stored in a temporary array, say $A_{i,,}$ i.e. A_i = { 0. 47,0.43,0.07,0.03}
- Difference d =0.47-0.43=0.04
- Then comparison is made if ($d < p_m$) i.e. if (0.04<0.1) ,which is true, hence {0.47,0. 43} are selected and the process is continued until (d>pm).
- d=0.43-0.07=0.36
- if ($d < p_m$) i.e. if (0.36<0.1) , which is false, hence {0.47,0.43} are selected and the comparison is discontinued.
- The interfaces corresponding to {0.47, 0.43} are stored in a new proposed routing table.

Table 2. The routing table at node B which have identified multiple paths

Destination	(Next-Hop, Probability)
E	(A,.45)
G	(A,0.39),(C,0.32),(F,0.28)
F	(A,0.8)
A	(A,0.34),(C,0.34)
D	(A,0.56)
C	(A,0.47),(C,0.43)

Result

As seen from the *table 2* , more than one optimal paths if exist, are identified**.**

6. INCORPORATING LOAD BALANCING (THE PROPOSED STRATEGY)

After getting *table 2* for network node B, the load will be distributed amongst the selected neighbor nodes according to the probability of the selected links.

Load = (Probability of the link * Total Load)/ (Total Probability of all the identified optimal paths)

Example:

Total Load=100 (Let us say)

1. According to *table 2*, at network node *B*, when the destination node is *E*, there exists only one optimal path *A*, hence, all the traffic will be forwarded to this route only.
2. When the destination node is *G*, there exist three optimal paths *A, C* and *F* with probabilities 0.39, 0.32 and 0.28 respectively. Load will be distributed according to

Load on Link *A*= (0.39 * 100)/ (0.39+0.32+0.28) =39.39

Load on Link *C*= (0.32 * 100)/ (0.39+0.32+0.28) =32.32

Load on Link *F*= (0.28 * 100)/ (0.39+0.32+0.28) =28.28

Hence ,40%, 32% and 28% of total load will be distributed to Link *A,C* and *F* respectively.

3. When the destination node is *F*, there exists only one optimal path *A*, hence, all the traffic will be forwarded to this route only.
4. When the destination node is *A*, there exist two optimal paths *A* and *C* with equal probabilities 0.34, 0.34 respectively. Load will be distributed equally.
5. When the destination node is *D*, there exists only one optimal path *A*, hence, all the traffic will be forwarded to this route only.
6. When the destination node is *C*, there exist two optimal paths *A* and *C* with probabilities 0.47 and 0.43 respectively

Load on Link *A*= (0.47 * 100)/ (0.47+0.43) =52.222

Load on Link *C*= (0.43 * 100)/ (0.47+0.43) =47.777

Hence, 52% and 48% of total load will be distributed to Link *A* and *C* respectively.

7. CONCLUSION AND FUTURE WORK

In this paper an improved version of the AntNet algorithm is proposed. In the improved version, M-optimal outgoing interfaces are identified and load is distributed amongst the selected paths according to some strategy, which are supposed to provide higher throughput and will be able to explore new and better paths even if the network topologies gets changed very frequently. This will distribute the traffic of overloaded link to other optimal links. Hence, the throughput of the network will be improved and the problem of stagnation will be rectified. In the future work, we intend to simulate the same using ns simulator so that exact results can be found.

REFERENCES

[1] S. Appleby and S. Steward, "Mobile software agents for control in telecommunication networks," *BT echnol.* vol. 12, no. 2, 1994.

[2] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz,"Ants for Load Balancing in Telecommunication Networks," Hewlett Packard Lab., Bristol, U.K., Tech. Rep. HPL-96-35, 1996.

[3] E. Bonabeau ,D.Suyers,"Routing in telecommunication networks with smart like agents ",Proceedings of LATA,1998.

[4] D.Subbramanian,P.Druschel,and J.Chen,'Ants and reinforcement learning: A case study in routing in dynamic networks",Proces of IJCAI-97,Palo Alto,Morgan Kauffman,1997,pp 832-838

[5] M. Dorigo and G.D.Caro,"Antnet : A mobile agents approach to adaptive routing ",Tech Report, University Libre de Bruxelles,IRIDIA,1997.

[6] G. D. Caro and M. Dorigo, "AntNet: Distributed stigmergetic control for communications networks," Journal of Artificial Intelligence Research, 9, 1998 vol. 9, pp. 317–365.

[7] G. D. Caro and M. Dorigo, "Ant colonies for adaptive routing in packet-switched communications networks," in Proc. 5th Int. Conf. Parallel Problem Solving from Nature, Amsterdam, The Netherlands, Sept. 27–30, 1998.

[8] G. D. Caro and M. Dorigo, "Two ant colony algorithms for best-effort routing in datagram networks," in Proc. 10th IASTED Int. Conf. Parallel Distributed Computing Systems, 1998, pp. 541–546.

[9] M. Dorigo, G. D. Caro, and L. M. Gambardella,"Ant algorithms for discrete optimization,"Arti.Life,vol5,no.2,pp.137–172,1999

[10] B. Baran and R. Sosa, "A new approach for AntNet routing," presented at the Proc. 9th Int. Conf. Computer Communications Networks, Las Vegas, NV, 2000.

[11] T. Stuzle and H. H. Hoos, "MAX-MIN ant system," Future Gener. Comput. Syst. J., vol. 16, no. 8, pp. 889–914, 2000.

[12] Kwang Mong Sim and Weng Hong Sun, "Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions ",IEEE

transactions on systems and humans,Vol.33,No.5 Sept 2003.

[13]Marco Dorigo and Krzysztof Socha," An Introduction to Ant Colony Optimization"IRIDIA- Technical Report Series Technical Report No. TR/IRIDIA/2006-010 April 2006, Last revision: April 2007.

[14]Marco Dorigo, Mauro Birattari, and Thomas Stutzle,"Ant Colony Optimization" IRIDIA– Technical Report Series, Technical Report No. TR/IRIDIA/2006-023 , September 2006.