Intelligent Packets for Dynamic Network Routing Using Distributed Genetic Algorithm

Suihong Liang
abacus@cs.dal.ca

A. Nur Zincir-Heywood
zincir@cs.dal.ca
Faculty of Computer Science
6050 University Avenue,
Halifax, NS, Canada, B3H 1W5

Malcolm I. Heywood
mheywood@cs.dal.ca
+1 902 4942951

Abstract

A distributed GA is designed for the packet switched network routing problem under minimal information. The requirements of such a problem mean that agents are required to possess more intelligence than was previously the case. To this end a distributed GA approach is developed and benchmarked against the AntNet algorithm under the same information constraints.

1 INTRODUCTION

Network information systems and telecommunication in general rely on a combination of routing strategies and protocols to ensure that information sent by a user is actually received at the desired remote location. In addition, the distributed nature of the problem means that multiple users can make requests simultaneously. This results in delayed response times, lost information or other reductions to the quality of service objectives on which users judge network service. Routing is the process used to determine how a packet travels from source to destination. Protocols are used to implement handshaking activities such as error checking and receiver acknowledgements. In this work, we are interested in the routing problem on computer networks.

The routing problem has several properties, which make it particularly challenging. The problem is distributed in nature; hence a solution that assumes access to any form of global information is not desirable. The problem is also dynamic; hence a solution that is sufficient for presently experienced network conditions may well be inefficient under other loads experienced by the network. Moreover, the traffic experienced by networks is subject to widely varying load conditions, making 'typical' network conditions unrepresentative.

Traditionally, routing strategies are implemented through the information contained in routing tables available at each node in the network (Forouzan, 2001). That is, a table details the next 'hop' a packet takes based on the overall destination of the packet. This should not be taken to imply that a routing table consists of an exhaustive list of all destinations – a form of global information. Instead, the table consists of specific entries for the neighboring nodes and then a series of default paths for packets with any other destination – such as OSPF or BGP4 (Halabi, 1997). Application of a classical optimization technique to such a problem might take the form of first assessing the overall pattern of network traffic, and then defining the contents of each routing table such that congestion is minimized. This approach does not generally work in practice as it simply costs too much to collect the information *centrally* on a regular basis, where regular updating is necessary in order to satisfy the dynamic nature of network utilization. We, therefore, see the generic objectives of a routing strategy to be both dynamically reconfigurable and be based on locally available information, whilst also satisfying the user quality of service objectives (i.e. a global objective).

Several approaches have been proposed for addressing these objectives including: active networking (Tennenhouse *et. al.*,1997), social insect metaphors (Di Caro, Dorigo, 1998), (Heusse *et al.*, 1998) cognitive packet networks (Gelenbe *et. al.*,1999), and what might be loosely called other 'adaptive' techniques (Corne *et. al.*, 2000). The latter typically involve using evolutionary or neural techniques to produce a 'routing controller' as opposed to a 'routing table' at each node, where the controller may require knowledge of the global connectivity to ensure a valid route. The global information assumption may be avoided by framing the problem as a reinforcement-learning context (Boyan, Littman, 1994). However, the Q-learning method, on which this is based, results in single path solutions for each destination. Both the social insect metaphor and the cognitive packet approach provide a methodology for routing, without such constraints; by utilizing probabilistic routing tables and letting the packets themselves investigate and report network topology and performance.

All methods as currently implemented, however, suffer from one drawback or another. Cognitive packet networks and active networking algorithms attempt to provide

routing programs at the packet level, hence achieving scalable run time efficiency becomes an issue. To date, implementations of 'adaptive' techniques and social insect metaphors have relied, at some point, on the availability of global information (Liang, *et al*., 2002).

The purpose of this work is to investigate the application of a genetic algorithm (GA) to build on lessons learnt from the social insect metaphor. This represents a major departure from previous works attempting to utilize GAs to solve the dynamic routing problem e.g. (Corne D.W., et al. 2000). In particular, a distributed GA is defined in which populations associated with each node of the network are required to co-evolve to solve the problem as a whole. Moreover, the GA interaction with the environment drives the features measured by the routing tables, as opposed to the tables predefining the features for measurement (a form of *a priori* information). Section 2 introduces the 'ant' based social insect metaphor against which the proposed approach is compared. Section 3 introduces the proposed GA-agent scheme. Section 4 summarizes the network on which experiments are performed. Results are presented in section 5 and conclusions drawn in section 6.

2 ROUTING USING A SOCIAL INSECT METAPHORE

As indicated above, active networking (Tennenhouse *et. al.*,1997) and cognitive packet (Gelenbe *et. al.*,1999) based approaches emphasize a per packet mechanism for routing. The aforementioned 'adaptive' techniques (Corne *et. al.*, 2000) tend to emphasize adding 'intelligence' to the routers leaving the packets unchanged. A social insect metaphor provides a middle ground in which the concepts of a routing table and data packet still exist, but in addition, intelligent packets – *ants* – are introduced that interact to keep the contents of the routing tables up to date. To do so, the operation of ant packets is modeled on observations made regarding the manner in which worker ants use chemical trails as a method of indirect *stigmergic* communication. Specifically, ants are only capable of simple stochastic decisions influenced by the availability of previously laid stigmergic trails. The chemical denoting a stigmergic trail is subject to decay over time, and reinforcement proportional to the number of ants taking the same path. Trail building is naturally a bi-directional process, ants need to reach the food (destination) and make a successful return path, in order to significantly reinforce a stigmergic trail (Forward only routing has also been demonstrated (Heusse *et al.*, 1998)). Moreover, the faster the route, then the earlier the trail is reinforced. An ant on encountering multiple stigmergic trails will *probabilistically* choose the route with greatest stigmergic reinforcement. Naturally, this will correspond to the 'fastest' route to the food (destination). The probabilistic nature of the decision, however, means that ants are still able to investigate routes with a lower stigmergic trial.

This approach has proved to be a flexible framework for solving a range of problems including the traveling sales man problem (Dorigo *et al*., 1996) and the quadratic assignment problem (Maniezzo *et al.*, 1999). The work reported here follows the 'AntNet' algorithm of Di Caro and Dorigo (Di Caro, Dorigo, 1998), and is informally summarized as follows,

- Each node in the network retains a record of packet destinations as seen on data packets passing through that node. This is used to periodically, but asynchronously, launch 'forward' ants with destinations stochastically sampled from the collected set of destinations;
- Once launched, a forward ant uses the routing table information to make probabilistic decisions regarding the next hop to take at each node. While moving, each forward ant collects time stamp and node identifier information where this is later used to update the routing tables along the path followed;
- If a forward ant re-encounters a node previously visited before reaching the destination, it is killed;
- On successfully reaching the destination node, total trip time is estimated and the forward ant converted into a backward ant;
- The backward ant returns to the source using exactly the same route as recorded by the forward ant. Instead of using the data packet queues, however, the backward ant uses a priority queue;
- At each node visited by the backward ant the corresponding routing table entries are updated to reflect the relative performance of the path;
- When the backward ant reaches the source, it 'dies'.

Although providing for a robust ant routing algorithm under simulation conditions, an assumption is made, which inadvertently implies the use of global information - knowledge of the number of nodes in the network (Di Caro, Dorigo, 1998). The definition of routing tables is, such that it is assumed that every node has a unique location in the routing table, see Table 1, or a total of L (number of neighboring nodes) by K (number of nodes in the entire network) entries. In practice, this is never the case. To do so would assume that it is first feasible, and secondly, should the network configuration ever change, then all nodes should be updated with the new configuration information. Moreover, as forward ants propagate across the network, the amount of information they need to 'carry' also increases (node identifier and time stamp).

Table-1 Original Routing Table at any Network Node k on the NTTnet

	All Network Nodes (Possible Destinations)			
Outgoing Links (Each node has L neighboring links)	$P_{1,1}$	$P_{1,2}$	------	$P_{1,55}$
	$P_{2,1}$	$P_{2,2}$	------	$P_{2,55}$
	------	------	------	------
	$P_{L,1}$	$P_{L,2}$	------	$P_{L,55}$

In order to avoid the use of global information, the authors modify the information provided at the routing tables such that the routing tables only detail the neighboring nodes, see Table 2, or a total of 2 by L entries. Such a limitation, therefore, places greater emphasis on the learning capacity of the ant. This is particularly significant during step (2) of the ant forward pass (section 2.1). Tables 1 and 2 illustrate the difference in available information for a node in the commonly used Japanese benchmark backbone (NTTNet) routing problem.

Table-2 Proposed Routing Table at any Network Node on the NTTnet

	Neighbor Node	If used for other nodes
Outgoing Links (Each node has L neighboring links)	$P_{1,1}$	$P_{1,d}$, $d \neq 1$
	$P_{2,2}$	$P_{2,d}$, $d \neq 2$
	------	------
	$P_{L,L}$	$P_{L,d}$, $d \neq L$

The following section summarizes the AntNet algorithm.

2.1 ANTNET ALGORITHM

It is assumed that routing tables, T_k, exist at each node, k, in which a routing decision is made. Tables consist of 'n' rows, one row for each neighboring node/link. As far as a normal data packet is concerned, if the destination, d, from current node, k, is a neighbor then the routing is still a stochastic decision. In all other cases, a route is selected based on the neighbor node probabilities.

1. New forward ants, F_{sd}, are created periodically, but independently of the other nodes, from source, s, to destination node, d, in proportion to the destination frequency of passing data packets. Forward ants travel the network using the same priority structures as data packets, hence are subject to the same delay profiles.
2. Next link in the forward ant route is selected stochastically, $p'(j)$, in proportion to the routing table probabilities and length of the corresponding output queue.

$$p'(j) = \frac{p(j) + \alpha l_j}{1 + \alpha(|N_k| - 1)}$$

where $p(j)$ is the probability of selecting node j as the next hop; α weights the significance given to local queue length verses global routing information, $p(j)$; l_j is proportional to the inverse of queue length at destination 'j' normalized to the unit interval; and N_k is the number of links from node k.

3. On visiting a node different from the destination, a forward ant checks for a buffer with the same identifier as itself. If such a buffer exists, the ant must be entering a cycle and dies. If this is not the case, then the ant saves the previously visited node identifier and time stamp at which the ant was serviced by the current node in a buffer with the forward ant's identifier. The total number of buffers at a node is managed by attaching "an age" to buffer space and allowing backward ants to free the corresponding buffer space.
4. When the current node is the destination, $k = d$, then the forward ant is converted into a backward ant, B_{ds}. The information recorded at the forward ant buffer is then used to retrace the route followed by the forward ant.
5. At each node visited by the backward ant, routing table probabilities are updated using the following rule,

   IF (node was in the path of the ant)
   THEN $p(i) = p(i) + r\ \{1 - p(i)\}$
   ELSE $p(i) = p(i) - r\ P(i)$
   where $r \in (0, 1]$ is the reinforcement factor central to expressing path quality (length), congestion and underlying network dynamics.

As indicated above, the reinforcement factor should be a factor of trip time and local statistical model of the node neighborhood. To this end (Di Caro, Dorigo, 1998) recommend the following relationship,

$$r = c_1\left(\frac{W_{best}}{t_{ant}}\right) + c_2\left\{\frac{I_{\sup} - I_{\inf}}{(I_{\sup} - I_{\inf}) + (t_{ant} - I_{\inf})}\right\}$$

where W_{best} is the best case trip time to destination d over a suitable temporal horizon, W; t_{ant} is the actual trip time taken by the ant; $I_{inf} = W_{best}$; $I_{sup} = \mu_{kd} + \{\sigma_{kd} / [W\ (1 - \gamma)]^{0.5}\}$.

The estimates for mean, μ_{kd}, and variant, σ_{kd}, of the trip time are also made iteratively, using the trip time information, o_{kd}. Thus,

$$\mu_{kd} = \mu_{kd} + \eta(o_{kd} - \mu_{kd})$$
$$(\sigma_{kd})^2 = (\sigma_{kd})^2 + \eta\ \{(o_{kd} - \mu_d)^2 - (\sigma_{kd})^2\}$$

Trip time information is now updated incrementally based on the recorded trip duration between current node, k, and ultimate destination, d. This means that it is no longer

necessary to carry all node and duration information as a 'stack' to the target duration as in the original model (Di Caro, Dorigo, 1998). Only the previous node information is therefore carried by each ant.

3 GENETIC ALGORITHM BASED SCHEME TO ROUTING

Simulation of the above AntNet scheme has been shown to provide a robust alternative to six standard routing algorithms – OSPF, SPF, BF, Q-R, P-QR and Daemon (Di Caro, Dorigo, 1998). However, this is not without utilizing routing tables in the AntNet algorithm, which provide entries for all nodes on the network. In practice, such (global) information is not actually available. In (Liang *et al*. 2002), the AntNet algorithm is benchmarked with routing tables configured with information regarding their neighbors alone; Table 2 as opposed to Table 1. The performance of such a system is then deemed unacceptable. Specifically 55% of packets are lost where 'lost' in this work is defined as *any* packet (data or ant) that visits the same node more than once. In order to address this problem, we are, therefore interested in the ability of route finding packets learning to find paths independently from the routing table information. By doing so, we do not rely on the capacity of the routing tables alone to retain information regarding all nodes in the network.

The objective of this work is to investigate a scenario in which the entries themselves are identified dynamically. This will be a first step towards a co-evolutionary model capable of evolving solutions to the packet switched routing problem. The ants, in this case, take the form of individuals from a distributed Genetic Algorithm (GA), hereafter referred to as GA-agents. Individual chromosomes travel the network using a string of next hop offsets, e.g., {1, 5, 0, 4, 2, 3, 5} over the interval [0, [0, L], where 'L' is selected to enable indexing of node connectivity. In all the experiments of section 5, 'L' is set to 6. On entering a node, genes (offsets) are used to identify the next link using a clockwise count, with respect to the port the GA-agent entered the node i.e. the next link is selected as a modulus of (gene % # of links). Such a representation is then independent of the specific network connectivity, unlike say the GA approach in (Munetomo *et al*., 1997). For each node encountered, the gene, used to select the next link, is incremented and a record is made of the node ID. The process naturally continues until the GA-agent executes its last gene, at which point it becomes a backward agent, returning to its original source node. In the special case of a GA-agent attempting to return down the same link as it entered a node, the router randomly selects the next hop from the available links, and changes the gene to the new value (deterministic mutation). If no next hop is available, then the chromosome is truncated, and the GA-agent becomes a backward agent (see the algorithm "processing agents"). Note, unlike the AntNet algorithm, modification of routing tables only takes place once the GA-agents have returned to their original source, and modifications only affect the source node routing table. The above representation supports single point crossover, resulting in variable length individuals. Mutation randomly selects a gene and adds/ subtracts an integer such that the new gene is still in the interval [0, 6].

Table 3 – GA-agent Routing Table

Agent ID	Agent Fitness	Trip Time (ms) and node ID
95	0.32	(3, J), (9, C), (21, W)
234	0.39	(1,B), (7, A),…, (432, Y)
…	…	…
31	0.71	(5,C), (9, K), …, (871, X)

At initialization, a router sends out half of the population of GA-agents to explore the network. Once the number of returned GA-agents reaches four, the fitness of the four agents is evaluated; the best two agents are then chosen – as in a steady state tournament (See algorithm "updating routing table & population").

The fitness function measures the popularity of nodes visited as well as the time taken to reach nodes encountered by GA-agents. Both of these properties are measured with respect to the original source node. Popularity of destination 'i' at node 'k' ($NP_k(i)$) is a dynamic property, measured at the original source node by recoding the frequency of different data packet destinations as seen by the source node over a fixed time window (50 seconds in this case), or

$$NP_k(i) = Dest(i) / TD_k$$

Where TD_k is the total number of data packets passing through node 'k'; and $Dest(i)$ is the number of data packets with destination 'i'.

Chromosomes, which find shortest paths to frequently used destinations, are therefore favored. The ensuing fitness function taking the form,

$$\frac{\sum_{\text{for each explored node } i} NP_k(i) \times trip\_time_i}{\sum_{\text{for each explored node } I} trip\_time_i} \quad (1)$$

The routing table in the GA approach consists of a short list of returned agents, every entry corresponds to an evaluated returned agent path. On routing a data packet, the router checks the table for a path that had experienced shortest trip time to the desired destination (third column of Table 3); if such an entry is not found, the entry with the highest fitness, Table 3 column 2, will be selected as the default next node for this data packet. The first two columns in the routing table are used during ranking and replacement of winning chromosomes.

The above constitutes our basic GA-agent approach. In addition, three further concepts are introduced. The first is that of demes. This provides a mechanism for passing useful chromosomes between *neighboring* nodes. To do so, every node will propagate best-case chromosomes to

neighboring nodes every 500 or 700ms (tunable parameter, see "propagate freq" in section 5). Secondly, in order to avoid stagnation in the routing tables, an incremental penalty is applied to each entry of the routing table (see the algorithm "updating routing table & population"). The motivation for such an aging mechanism is to ensure that routing tables remain sensitive to the dynamic nature of the environment (e.g., changes to network topology, network node/link failure, network congestion). Such a mechanism is introduced during updates to routing tables: making every routing table entry a bit smaller in fitness, and a bit longer in trip time, or

$$\text{fitness}_{\text{agent in routing table}} = \text{original fitness} \times c2;$$
$$\text{trip\_time}_{\text{every node of every entry}} = \text{original\_trip\_time} / c2;$$
$$\text{where } c2 \text{ is a constant} \in (0.0, 1.0) \qquad (2)$$

Finally, when initializing the populations of chromosomes at each node, nodes with a higher connectivity naturally represent a larger search problem. Thus, the number of chromosomes per population is initialized in proportion to the square of the number of neighbors.

The algorithm is outlined as follows: (c_1, c_2, and c_3 are constants.)

init

initialize first generation of agents;
#agents = #links^2 × c_1;
string of offsets of an agent - {3, 1, 4, 5, 2, …}
clear routing table;
clear flow pattern stats;
send out half population of individuals;

processing agents

if (case of backward agent)
then if (agent arrives at the source)
then if (agent timeout)
then (kill agent);
else (put into "back" list);
end if
else if (next hop is down)
then (kill agent);
else (forward to the link)
end if
end if
else agent records the trip time info;
retrieve offset from the next unused gene position;
if (corresponding link is available and no loop caused)
then (send the agent to the link);
else (randomly [each available link has equal probability] select an available link [without entering a loop]);
end if
if (no such link found)
then (convert the agent into a backward agent)
else (set the offset to the new value);
(send agent to the link);
end if
end if

updating routing table & population (once 4 agents return to the same source, i.e. steady state tournament**)**

update the performance table by aging mechanism:
fitness of agent = original fitness × c2;
trip time to every node of every entry = original trip time / c2;
use the fitness function to evaluate the fitness of backward agents;
select the best two agents as parents;
put/update the fitness of parent agents in the routing table;
delete the entries of the worst two agents in the routing table;
use standard crossover and mutation on the parents to generate two children;
put the children into the population;
delete the worst two agents from the population;
if (current time > last clear time + c3)
then (clear flow statistics)
randomly launch 4 agents from the population to explore the network;

routing data packets

if (routing table is empty)
then (randomly choose a link to forward)
else (search the routing table for the shortest trip time to the desired destination)
if (no entry found for the desired destination)
then (choose fittest entry);
end if
end if
if (no route is found)
then (discard the packet)
end if

3.1 DATA STRUCTURES

Every agent consists of a string of next hop offsets, and time stamp records. Every router consists of an incoming buffer, a processing buffer (stores a packet at a time), and an outgoing buffer for each neighboring router. For the GA approach, every router has a population of chromosomes, a routing table, a flow pattern statistics table, and a fitness table. The number of chromosomes per population is in direct proportion to the square of number of neighbors. The routing table, which is updated whenever four chromosomes return, consists of current fittest individuals, c.f (1). The flow pattern estimates the popularity of data packets passing through the node, c.f. (2). The fitness table stores the fitness of every chromosome, currently a member of the routing table.

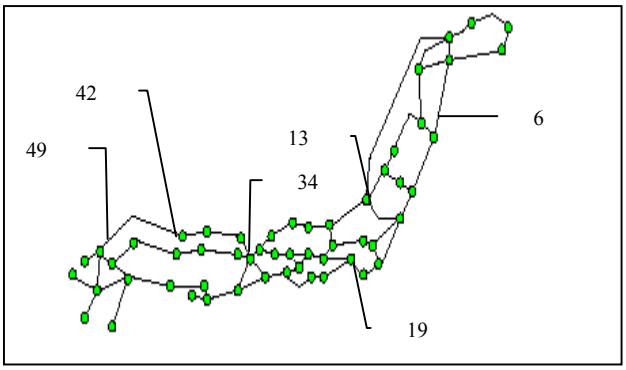

Figure-1: Japanese Backbone - NTTnet (55 nodes)

To simulate and test the GA-agent algorithm, an event driven simulation environment is developed (C++ on UNIX system). Specifically, the Japanese Internet backbone (NTTNET – see figure 1) is modeled, where this represents a narrow long configuration in which the degree of connectivity is low (from 1 to 5), when compared to the US backbone. Hence the Japanese network provides a more demanding configuration for testing routing algorithms, as higher degrees of connectivity lower the possibility of packet loss due to loops, timeouts, i.e., in a narrow long shaped network, once a packet is forwarded in a wrong direction, it might never have the chance to be routed to the desired destination.

4 SIMULATION ENVIRONMENT

The event driven simulation models the network as routers (nodes) and links. Every router has an incoming buffer, a memory space for processing packets, and an outgoing buffer for each link to its neighboring routers. A priority queue is used to store the events. Both AntNet (local routing table information, Table 2) and GA-agent algorithms are simulated under the same environmental conditions. That is, an event generator is used to generate the events, such as new packet time of generation, or routers availability. The following are the parameters used in the simulation,

- Network topology takes the form of the Japanese backbone, figure 1;
- Forward ants are launched every 300ms;
- The AntNet algorithm is given 5 seconds at the beginning of the simulation to converge the initial routing tables, during this period, routing packets (ants or GA-agents) are the only packets traversing the network;
- Data packets are generated by Poisson distribution (mean of 35ms);
- The parameters for the GA based scheme are given as the first 5 rows of tables 4 - 7, where 4 (columns 2 - 5) different GA based agent structures are simulated;
- Any packets, including data packets, are *killed* should they encounter a previously visted node. Given the probabilistic nature of the routing tables this represents a rather harsh constraint, but in doing so is utilized to emphasize the properties of different routing strategies. In addition packets that are routed down links representing a fault condition are distinguished separately as *lost* packets.

The simulation length is 1250s. As a result, 1985536 data packets are generated within 1250s. The queue length is the total number of waiting packets per second, which includes the data packets and the routing packets. In this paper, the routing packets refer to the ants in the AntNet algorithm, and to the GA-agents in the GA approach.

5 RESULTS

On measuring the performance of a routing algorithm, we focus on:

- Network throughput, which is defined as number of data packet bytes successfully received at their destination in a two second window;
- Total time to deliver all the data packets (finish time);
- Number of arrived data packets;
- Number of 'killed' and 'lost' packets;
- Average trip time of arrived data packets.

Two sets of experiments are conducted, in both cases using a series of network scenarios designed to investigate operation under changing network conditions. The first set of experiments investigates parameters associated with the distributed GA. The second of experiments takes one set of these parameters and reduces the degree of exploration/ exploitation (mutation/ crossover respectively).

There are a total of 4 scenarios in each set of experiments, in the first case all routers remain available. The remaining scenarios investigate plasticity of the network by removing different router combinations. First, router 34 is removed at a time step of 500s. From figure 1, it is apparent that router 34 represents a significant node in the topology, although alternative paths certainly exist. In the third scenario, two routers are removed, whereas in scenario four the same two routers are removed but asynchronously.

5.1 PARAMETERIZATION OF DISTRIBUTED GA

In the case of routing using GA-agents, there are six basic parameters,

1. Agents / link^2 – c_1, determines the population of chromosomes per node. The implication being that there are $O(L^2)$ locations in each routing table, where L is the number of neighboring nodes;

2. Aging – c_2, rate by which fitness of individuals currently populating the routing tables decay;
3. Propagate ratio – the number of chromosomes exchanged between populations, expressed as a % of node population size;
4. Propagate freq – rate of exchange of chromosomes between populations;
5. Flow clear freq – c_3, time interval over which data packet destination statistics are collected;
6. Crossover and Mutation – the results in this section utilize maximum crossover and mutation rates in order to encourage continuous investigation of alternative routes. Section 5.2 considers the case of a more classical section of crossover and mutation thresholds.

These are initially selected to enable qualification of the sensitivity to population size, rate of aging etc. and remain the same across all experiments; Tables 4 to 7, columns 2 - 5. Table 8 summarizes the same information for the AntNet algorithm under a 'local' routing table configuration. Thus, Local AntNet utilizes tables of length O(L), significantly less than the GA-agent case.

Table 4 – No Network Failure.

# Agents / link^2	32	32	40	48
Aging	0.8	0.9	0.9	0.9
Propagate ratio	5%	3%	3%	2%
Propagate freq	500ms	500ms	700ms	700ms
Flow clear freq	50s	50s	50s	50s
Finish time (s)	1,252	1,253	1,252	1,267
Arrived Packets (AP) (×1000)	1,619	1,585	1,583	1,560
Avg. trip time for AP (ms)	742	905	678	1,236
# killed packets	366,533	400,351	402,517	385,750
# lost packets	0	0	0	0
# Agents (×10^3)	1,690	1,028	801	475

Table 5 – Router 34 is Down at 500s.

# Agents / link^2	32	32	40	48
Aging	0.8	0.9	0.9	0.9
Propagate ratio	5%	3%	3%	2%
Propagate freq	500ms	500ms	700ms	700ms
Flow clear freq	50s	50s	50s	50s
Finish time (s)	1,417	1,307	1,444	1,494
Arrived Packets (AP) (×1000)	1,346	1,298	1,333	1,373
Avg. trip time for AP (ms)	2,014	2,613	3,156	2,668
# killed packets	617,064	665,188	630,479	590,732
# lost packets	21,922	21,922	21,923	21,918
# Agents (×10^3)	1,801	1,087	966	552

Table 6 – Routers 49 & 13 are Down at 500s.

# Agents / link^2	32	32	40	48
Aging	0.8	0.9	0.9	0.9
Propagate ratio	5%	3%	3%	2%
Propagate freq	500ms	500ms	700ms	700ms
Flow clear freq	50s	50s	50s	50s
Finish time (s)	1,254	1,445	1,258	1,520
Arrived packets (AP) (×1000)	1,317	1,369	1,402	1,504
Avg. trip time for AP (ms)	947	1,301	850	1,759
# killed packets	623,539	571,390	539,747	438,378
# lost packets	44,466	44,882	43,658	43,496
# Agents (×10^3)	1,543	973	754	514

Table 7 – Router 13 is down at 300s, Router 49 is down at 500s, and both are up at 800s.

# Agents / link^2	32	32	40	48
Aging	0.8	0.9	0.9	0.9
Propagate ratio	5%	3%	3%	2%
Propagate freq	500ms	500ms	700ms	700ms
Flow clear freq	50s	50s	50s	50s
Finish time (s)	1,535	1,261	1,496	1,437
Arrived packets (AP) (×10^3)	1,410	1,334	1,441	1,458
Avg. trip time for AP (ms)	2088	1202	470	2018
# killed packets	551,218	627,596	520,989	503,873
# lost packets	23,953	23,426	23,401	23,085
# Agents (×10^3)	1,447	1,043	896	648

Performance is qualified in terms of two basic parameters, time taken for all packets to be received (or lost) and the number of packets successfully received. Naturally, the former should be minimized and the latter maximized. In the case of experiment 1 – no network failures – the time for all packets to be accounted for is essentially the same,

irrespective of parameter or algorithm. An immediate difference is recognized, however, in the number of arrived packets. The AntNet algorithm can only successfully route 55% of those in the GA-agent approach. This observation is repeated across all the remaining scenarios. Moreover, in terms of 'killed' packets this means that less than 50% of the packets in the local version of the AntNet algorithm revisit sites previously encountered.

Table 8 – AntNet with Local Information Only

No network failure	
Finish time (s)	1,267
Arrived packets (AP)	903 ($\times 10^3$)
Avg. trip time of AP (ms)	398
# killed packets	1,082,652
# lost packets	0
# of Ants	218 ($\times 10^3$)
Router 34 down at 500s	
Finish time (s)	1,369
Arrived packets (AP)	813 ($\times 10^3$)
Avg. trip time of AP (ms)	2,899
# killed packets	1,138,860
# lost packets	32,763
# of Ants	219 ($\times 10^3$)
Routers 49 & 13 down at 500s	
Finish time (s)	1,300
Arrived packets (AP)	827 ($\times 10^3$)
Avg. trip time of AP (ms)	1,617
# killed packets	1,114,729
# lost packets	43,682
# of Ants	219 ($\times 10^3$)
Routers 13 down at 300s, Router 49 down at 500s, both up at 800s	
Finish time (s)	1,272
Arrived packets (AP)	863 ($\times 10^3$)
Avg. trip time of AP (ms)	1,254
# killed packets	1,099,283
# lost packets	23,209
# of Ants	219 ($\times 10^3$)

In terms of specific parameter settings, the GA-agent approach appears to consistently route the most packets successfully when the number of agents per link is highest and propagation ratio least. (Investigation of GA-agents without demes, however, performs very badly.) It is also noticed that although a maximum allowable length of 30 genes per individual is permitted, chromosomes never reach this limit. Instead a preference of chromosome lengths of 10 or less genes is found for nodes with a low level of connectivity and 15 to 25 for individuals with a connectivity of 3 or more.

5.2 PARAMETERIZATION OF CROSSOVER AND MUTATION

As a final experiment, one instance of the distributed parameter set is investigated under a classical crossover and mutation rate of 90% crossover and 10% mutation. As identified in section 5.1, lower agent per link counts result in less packets being delivered. Table 9 reports the case of 32 agents/ link, an aging factor of 0.9, a propagation ration of 3% and a frequency of 500ms (column 3 in tables 4 to 7).

On comparison with the same parameterization under 100% crossover and mutation, the number of 'killed' or 'lost' packets decreases by 33% to 8%, and the trip time improves in each scenario other than no network failure. Moreover, the case of 90% crossover and 10% mutation betters all combined 'killed-lost' packet counts of any of the distributed GA parameters investigated in section 5.1. The implication being that more data packets are routed to the destination without either encountering a faulty link or a previously visited node. The principle penalty, however, appears to be an increase in the number of GA-agents introduced. Future work will naturally investigate whether this trend holds for other distributed GA parameterizations (the case of 48 agents per link appears to utilize less GA-agents).

Table 9 – GA-agent with Crossover of 90%, Mutation 10%

No network failure	
Finish time (s)	1,252
Arrived packets (AP)	1,693 ($\times 10^3$)
Avg. trip time of AP (ms)	1,171
# killed packets	292,723
# lost packets	0
# of Agents	961 ($\times 10^3$)
Router 34 down at 500s	
Finish time (s)	1,507
Arrived packets (AP)	1,401 ($\times 10^3$)
Avg. trip time of AP (ms)	356
# killed packets	562,751
# lost packets	21,924
# of Agents	1,170 ($\times 10^3$)

Routers 49 & 13 down at 500s	
Finish time (s)	1,252
Arrived packets (AP)	1,417 ($\times 10^3$)
Avg. trip time of AP (ms)	861
# killed packets	523,673
# lost packets	44,658
# of Agents	1,025 ($\times 10^3$)
Routers 13 down at 300s, Router 49 down at 500s, both up at 800s	
Finish time (s)	1,252
Arrived packets (AP)	1,555 ($\times 10^3$)
Avg. trip time of AP (ms)	1,012
# lost packets	406,840
# killed packets	23,861
# of Ants	1,083 ($\times 10^3$)

6 CONCLUSIONS

As indicated in the introduction, network routing problems force an interesting set of constraints, which present a suitable test-bed for problem solving using co-evolutionary techniques. In this work, we emphasize the case in which routing table features, as well as content, are evolved. Thus, we are not privy to *a priori* knowledge regarding the number of nodes in the network. The AntNet algorithm (Di Caro *et al*., 1998) does not perform efficiently and the GA representation cannot make use of global knowledge of network connectivity, as has been the case in the past (Munetomo *et al*., 1997). Such an environment implies that packets responsible for updating network connectivity requires more autonomy than were previously acknowledged to solve packet switched routing problems. As a first attempt at addressing these problems directly, we utilize a representation that is independent of specific network connectivity patterns and distributed in its operation (multi-population model with chromosomes physically traveling the network). Such a system improves on the AntNet algorithm when constrained to a 'local' table representation, Table 2 (see (Liang *et al*., 2002) for a detailed discussion of AntNet under 'local' and 'global' routing table constraints), or be it whilst utilizing larger routing tables. The principle drawback for the GA-agent is the search efficiently of the ensuing routing table where a search as opposed to an indexing process is now necessary. Future work will expand the interaction between chromosomes to facilitate a more co-evolutionary approach to the development of routing policies and develop a better organization to the routing table structure. Moreover, the relationship between routing table size and performance requires further investigation.

Acknowledgments

A. Nur Zincir-Heywood and Malcolm I. Heywood gratefully acknowledge the support of the Individual Research Grants from the Natural Sciences and Engineering Research Council of Canada.

References

Boyan J.A., Littman M.L., "Packet Routing in Dynamically Routing Networks: A Reinforcement Learning Approach," *Advances in Neural Information Processing Systems*, Volume 6, pp 671-678, 1994.

Corne D.W., Oates M.J., Smith G.D., Telecommunications Optimization: Heuristic and Adaptive Techniques. John Wiley & Sons, isbn 0-471-98855-3, 2000.

Di Caro G., Dorigo M., "AntNet: Distributed Stigmergetic Control for Communications Networks," Journal of Artificial Intelligence Research, 9, pp 317-365, 1998.

Dorigo M., Maniezzo V., Colorni A., "Ant System: Optimization by a Colony of Cooperating Agents," IEEE Transactions on Systems, Man and Cybernetics – B, 26(1) pp 29-41, Feb 1996.

Forouzan B. A., "Data Communications and Networking", Mc-Graw Hill, ISBN 0-07-232204-7, 2001.

Gelenbe E., Xu Z., Seref E., "Cognitive Packet Networks," Proceeding of 11th IEEE International Conference on Tools with Artificial Intelligence, pp 47-54, 1999.

Halabi B., Internet Routing Architectures, Cisco Press, ISBN 1-56205-652-2, 1997.

Heusse M., Snyers D., Guerin S., Kuntz P., "Adaptive Agent-driven Routing and Load Balancing in Communication Networks," Advances in Complex Systems, 1, pp 237-254, 1998.

Liang S., Zincir-Heywood A.N., Heywood M.I., "The Effect of Routing under Local Information using a Social Insect Metaphor," IEEE International Congress on Evolutionary Computation, May 2002.

Maniezzo V., Colorni A., "The Ant System Applied to the Quadratic Assignment Problem," IEEE Transactions on Knowledge and Data Engineering, 11(5), pp 769-778, Sept/ Oct 1999.

Munetomo M., Takai Y., Sato Y., "An Adaptive Network Routing Algorithm Employing Path Genetic Operators," Proceedings of the 7th International Conference on Genetic Algorithms, pp 643-649, 1997.

Tennenhouse D., Smith J., Sincoskie W., Wetherall D., Minden G., "A Survey of Active Network Research," IEEE Communications Magazine, 35(1), pp 80-86, Jan 1997.