Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Applied Soft Computing xxx (2005) xxx–xxx
Applied Soft Computing
www.elsevier.com/locate/asoc

Adding more intelligence to the network routing problem: AntNet and Ga-agents

S. Liang, A.N. Zincir-Heywood, M.I. Heywood *

NIMS Group, Faculty of Computer Science, Dalhousie University, 6050 University Avenue, Halifax, NS, Canada B3H 1W5

Received 12 November 2003; received in revised form 4 January 2005; accepted 10 January 2005

Abstract

AntNet and GA-agent algorithms are benchmarked against a series of dynamic network routing problems. Performance is characterized using multiple performance metrics on the Japanese backbone (NTTNET). NTTNET is used on account of the elongated topology presenting a more challenging routing problem than in the case of the American backbone, which is basically square. The AntNet scheme is found to provide the best routing ability providing global information is available and network security is not a factor. The GA-agent algorithm is shown to provide routing performance between the AntNet algorithm with global information and that without, whilst avoiding global information requirements and satisfying typical models of network security.

Keywords: Insect metaphor; Genetic algorithms; Packet switched network routing; Decentralized problem solving

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 operation. 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

* Corresponding author. Tel.: +1 902 494 2951; fax: +1 902 492 1517.
E-mail addresses: abacus@cs.dal.ca (S. Liang), zincir@cs.dal.ca (A.N. Zincir-Heywood), mheywood@cs.dal.ca (M.I. Heywood).

1568-4946/$ – see front matter © 2005 Published by Elsevier B.V.
doi:10.1016/j.asoc.2005.01.005

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 [1]. That is, the table consists of specific entries for the neighboring nodes and then a series of default paths for packets with any other destination, for example, OSPF or BGP4 [2]. 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 the measured 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 real-time 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 [3], social insect metaphors [4,5] cognitive packet networks [6], and what might be loosely called other 'adaptive' techniques (e.g. evolutionary computation [7,8], neural networks [9]). 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 in a reinforcement-learning context [9]. 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. Implementations of 'adaptive' techniques or social insect metaphors frequently rely on the availability of global information [10]. Finally, the very nature of the packet routing problem implies that performance should be measured from multiple perspectives simultaneously, where most results currently available characterize performance using one or two parameters alone.

The purpose of this work is, firstly, to investigate the application of a social insect metaphor to solve the dynamic routing problem. This is shown to rely on the availability of a priori global information. Secondly, a distributed genetic algorithm (GA) is introduced. This represents a major departure from previous works attempting to utilize GAs to solve the dynamic routing problem, e.g. [7,8]. In particular, a methodology is detailed for solving the representation problem without recourse to global information. The system is benchmarked under dynamic and static network conditions from the perspective of multiple performance metrics.

In the following, Section 2 introduces the 'ant' based social insect metaphor scheme for packet routing against which this work is compared. Section 3 introduces the proposed alternative scheme based on a distributed genetic algorithm. Results are presented in Section 4 and conclusions are drawn in Section 5.

2. AntNet social insect metaphore

As indicated above, active networking [3] and cognitive packet [6] based approaches emphasize a per packet mechanism for routing. The aforementioned 'adaptive' techniques [7–9] 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 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 [5]). 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 lower stigmergic reinforcement.

This approach has proved to be a flexible framework for solving a range of problems including the traveling sales man problem [11] and the quadratic assignment problem [12]. The work reported here follows the 'AntNet' algorithm of Di Caro and Dorigo, where this was previously demonstrated to perform better than typical approaches to the routing problem including OSPF (as currently employed on the Internet) [4].

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, a route is selected based on the neighbor node probabilities.

- 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;
- 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, $p(j)$; lj 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;
- 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. In this work, 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. By introducing buffers at routers, it is no longer necessary to carry all node and duration information in the packet to the target duration as in the original model [4]. Only the previous node information is, therefore, carried by each ant;
- 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;
- 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 $p(i) = p(i) - rP(i)$

where $r \in (0, 1]$ is the reinforcement factor central to weight the relative significance of path quality (length), congestion and underlying network dynamics.

As indicated above, the reinforcement factor should be a factor of the trip time and the local statistical model of the node neighborhood. To this end [4] recommend the following relationship,

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

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; $I_{\text{inf}} = W_{\text{best}}$; $I_{\text{sup}} = \mu_{kd} + \{\sigma_{kd}/[W(1-\gamma)]^{0.5}\}$.

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,

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

Thus, trip time information is updated incrementally based on the recorded trip duration between current node, k, and ultimate destination, d.

2.2. Global information assumption

Although providing for a robust ant routing algorithm under simulated conditions [4], an assumption is made, which inadvertently implies the use of global information—knowledge of the number of nodes in the network [10]. The definition of routing tables assumes that every node has a unique location in the routing table or a total of l (number of neighboring nodes) by n (number of nodes in the entire network) entries. Hereafter, this is referred to as the GlobalAnt algorithm. 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.

In order to avoid the use of global information, we consider the case of routing tables limited to detailing actions in terms of the neighboring nodes alone, or a total of 2 by l entries. Hereafter referred to as the LocalAnt algorithm. This is equivalent to the tables as used by OSPF or BGP4 protocols currently in use [2]. Such a limitation, therefore, places greater emphasis on the learning capacity of the ant. In Section 4, the AntNet algorithm is benchmarked under both local (LocalAnt) and global (GlobalAnt) routing table configurations.

3. Genetic algorithm model

Genetic algorithms (GA) are a class of generic search algorithms that perform a parallel search over a fixed ‘‘population’’ of candidate solutions. To do so, Darwin’s concept of survival of the fittest and observations from genetics are used to guide the general mode of operation. Specifically, a selection operator provides the pressure to improve the contents of the population, examples being generational or tournament based selection. Search operators (crossover and mutation) address the exploitation-exploration trade off associated with manipulating individual members of the population. The algorithm as a whole is iterative in nature with individuals being repeatedly modified such that the overall fitness of the population improves (Holland’s Schema Theorem [13]).

There are three principle inter-related design decisions that have a significant impact on the ability of a GA to efficiently solve problems. Firstly, the representation problem, which is how to efficiently encode candidate solutions into the genotypic string format of a GA. Secondly, the operator problem, or how to define operators such that individuals are always syntactically correct. The third problem is how to succinctly express fitness such that the ‘best’ individuals of the population solve all the properties of the problem of interest.

In the case of this work, we desire a representation that is independent of network connectivity—unlike, for example, the approach of Munetomo [7]. The operator problem naturally has two parts—selection and search. The definition of suitable search operators is rendered straightforward (standard crossover and mutation operators are applicable) if we are able to pose suitable solutions to the representation problem. The case of a suitable selection operator for this work is addressed by utilizing the concept of a static subpopulation model with migration. That is to say, each node of the network has an independent population of candidate solutions and best case solutions are allowed to periodically migrate between neighboring nodes, as in an island model of evolution [14]. Given these general observations, the following subsections detail the specific methodology employed and hereafter referred to as GA-agents.

3.1. Basic GA-agents

Letting individuals from each population travel the network address the objectives of the representation problem. Thus, the genotypic content of any individual expresses the number of nodes visited and routing decision taken at each node. This is similar to the concept of the forward ant in the AntNet algorithm. Likewise, as each ‘GA-agent’ travels the network, previous hop and elapsed time information is

```
Processing GA-agents
if backward agent
then    IF arrives at source
        THEN    IF timeout
                THEN    discard it;
                ELSE    put it into "back" list;
                END IF
        ELSE    IF next hop is down
                THEN    discard it;
                ELSE    forward it to the link;
                END IF
        END IF
else    agent records the trip time info;
        identify gene specifying next hop;
        IF corresponding link is available and no 'loop'
                caused
        THEN    send the agent to the link;
        ELSE    randomly select an available link
                that causes no loop;
        END IF
        IF no such link found
        THEN    convert agent into a backward agent;
        ELSE    set the offset to the new value;
                        send agent to the link;

        END IF
END IF
```

Fig. 1. Processing GA-agents.

```
Updating routing table & population
        (once 4 agents return to the same source)
update the performance table by aging mechanism:
FOR each agent in routing table
   DO   fitness = original_fitness × c2;
        FOR each node in the entry
        DO      trip_time = original_trip_time / c2;
        END FOR
   END FOR
use the fitness function to evaluate backward agents;
select the best two agents as parents;
update the fitness of the 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;
END IF
randomly launch 4 agents from the population to explore
                        the network;
```

Fig. 2. Routing and population update.

recorded (Fig. 1). On reaching the node identified by the last gene, the individual becomes a backward ant and merely retraces its path and waits at the corresponding source node for fitness evaluation (routing table updates are only performed at the source node) (Fig. 1). In the special case of a GA-agent attempting to return down the same link with which the node was entered, 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 (Fig. 1). A genotype, therefore, takes the form of a list of integers—representing next hop offsets, e.g. {1, 5, 0, 4, 2, 3, 5}—over the interval [0, *L*], where '*Z*' is selected to enable indexing of node connectivity.[1] On entering a node, a gene (offset) is used to identify the next link using a clockwise count from the link that the GA-agent entered the node, i.e. the next link is selected modulo (gene % # of links). Such a representation is then independent of the specific network connectivity and directly 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, *L*].

Selection takes the form of a steady-state tournament of size 4. Thus, when four GA-agents return to the same source node, they are ranked in accordance with their fitness, the worst two GA-agents being replaced by the children of the best (Fig. 2). The fitness function itself incorporates 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*' ($\mathrm{NP}_k(i)$) 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 (the time window is set 50 s in this work),

$$\mathrm{NP}_k(i) = \frac{\mathrm{Dest}(i)}{\mathrm{TD}_k}$$

[1] In all the experiments of Section 5, 'Z,' is set to 6.

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'. Fitness now takes the form,

$$\frac{\sum NP_k(i) \times trip\_time_i}{\sum trip\_time_i}$$

Thus, GA-agents that find shortest paths to frequently used destinations are favored.

The routing table in the GA approach consists of a 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 (Table 1, column 3); if such an entry is not found, the entry with the highest fitness (Table 1, column 2) will be selected as the default next node for this data packet (Fig. 3). The first two columns in the routing table are used during ranking and replacement of winning chromosomes (Fig. 2).

3.2. Aging and population initialization

As indicated in the introduction, the general packet switched routing problem of interest here has dynamic properties as a result of different load conditions or network outages. This means that the routing strategy must be able to continuously adapt to new conditions. To provide such a property an incremental aging penalty is applied to each GA-agent entry of the routing table. Thus, fitness is decreased and trip times increased at each update to the routing table entries (Fig. 2).

In addition, each node of the network may naturally have a different degree of connectivity; hence pose a more (less) significant routing problem. Populations (at each node) are, therefore, initialized in proportion to the degree of connectivity of each node; where a square law was empirically found to provide sufficient search capacity (Fig. 4).

Table 1
Example GA-agent routing table

Agent ID	Fitness	Trip time (ms) and node ID
95	0.32	(3, J), (9, C) (21, W)
234	0.355	(1, B), (7, A), . . ., (432, Y)
. . .	. . .	. . .
31	0.71	(5, C), (9, K), . . ., (871, X)

```
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 explored the desired destination
        THEN    choose a agent with best performance;
        END IF
END IF
IF no route is found
THEN    discard the packet;
END IF
```

Fig. 3. Routing data packets.

```
Initialize Population
initialize first generation of agents;
#agents = #links^2 × c1;
clear routing table;
clear flow pattern statistics;
send out half population of individuals/chromosomes;
```

Fig. 4. GA-agent initialization.

4. Evaluation

For the purposes of investigation and comparison, a discrete event simulation (DES) is developed (C++, UNIX system) for modeling the action of the GA-agent and AntNet algorithms on a network configured to represent the Japanese backbone (NTTNET) (Fig. 5). Such a configuration is of particular interest due to the long thin topology in comparison to other networks (e.g. in the box like topology of the US

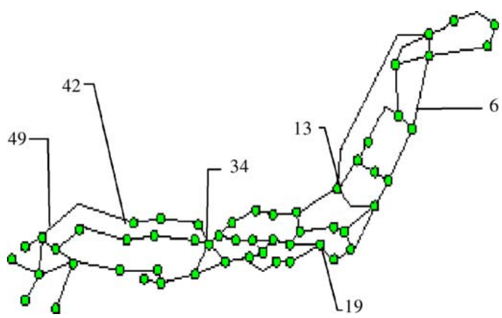

Fig. 5. Japanese NNTNET topology.

Table 2
Parameter values

AntNet		GA-agent	
α	0.3	*P*(crossover)	0.9
c_1	0.7	*P*(mutation)	0.1
c_2	0.3	#Agents/link^2	32
η	0.005	Aging rate	0.9
γ	0.654	Prop. ratio (%)	3
		Prop. freq. (ms)	500
		Flow clear freq. (s)	50

Table 3
Static parameters—scenario 1

Algorithm	GA-agent	GlobalAnt	LocalAnt
Finish time (s)	1252	1253	1267
Routing packets (%)	48	10	11
Arrived packets (%)	85.3	99.7	45.5
Dead packets (%)	14.7	0.3	54.5
AP avg. trip time (ms)	1171	566	398

backbone nodes tend to provide a high degree of connectivity across the network as a whole). This property of NTTNET makes it more difficult to identify alternative routes or increases the number of pathologically bad routes. The DES models each node as an incoming buffer, a memory space for processing packets, and an outgoing buffer for each neighboring link. Both AntNet 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 router availability. The following are the parameters used in the simulation,

- Network topology takes the form of the Japanese backbone (Fig. 5);
- Forward ants are launched every 300 ms;
- Data packets are generated by Poisson distribution (mean of 35 ms);
- AntNet and GA-agent algorithms are given 5 s 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;
- Any packets that are routed down links representing a fault condition are distinguished as *lost* packets. In addition, packets may also be *killed*. In this case any packets, including data packets, are terminated should they encounter a previously visited node. Given the probabilistic nature of the routing tables this represents a rather harsh constraint, but is utilized to emphasize the properties of different routing strategies. In the following results, lost and killed packets are collectively referred to as *dead* packets.

Simulations are ran for the duration of 1250 s, as a result 1,985,536 data packets are generated. The queue length is the total number of waiting packets per se-

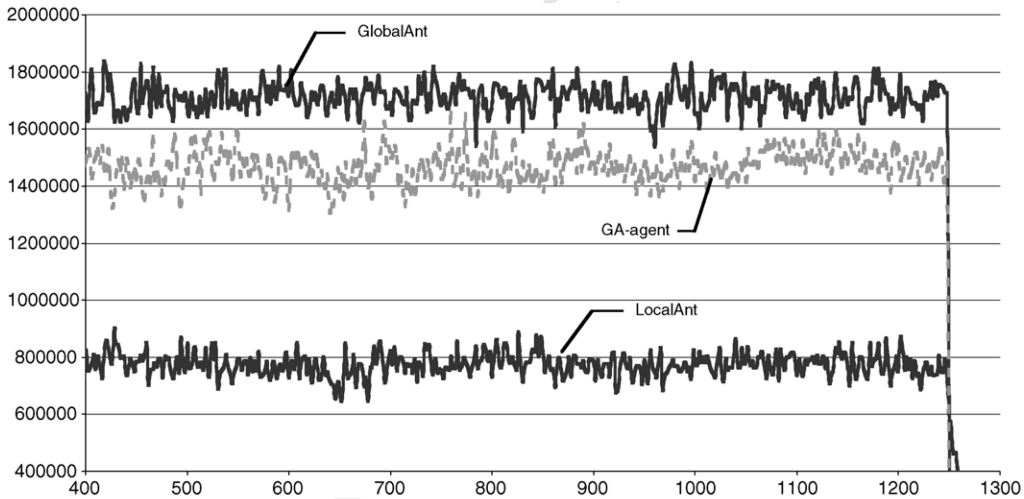

Fig. 6. Throughput (bytes) vs. time (s)—no network failure.

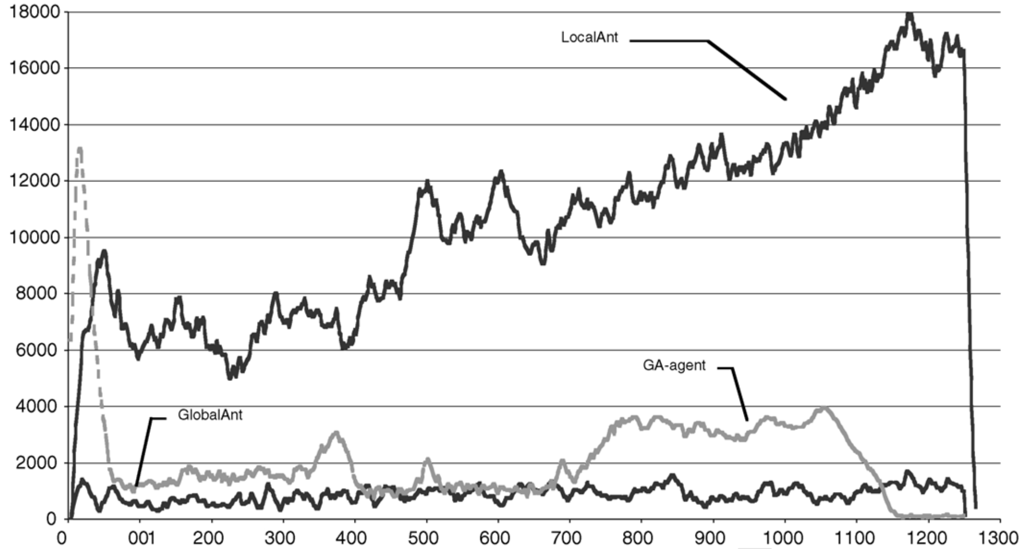

Fig. 7. Queue length vs. time (s)—no network failure.

cond, 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.

4.1. Algorithm parameterization

Parameter selection in the case of the AntNet algorithm follows the recommendations of Di Caro and Dorigo [4]. Two versions of the AntNet algorithm are considered. LocalAnt represents the case of a routing table without the capacity to represent global information [10], whereas GlobalAnt represents the original ‘‘full’’ routing table scenarios [4]. In the case of GA-agents, there are five basic parameters, summarized as follows

1. Rates of crossover and mutation;
2. #Agents/link[2]—a constant c_1, which determines the population of chromosomes on every node;
3. Aging—a constant $c_2 \in (0.0, 1.0)$, rate by which fitness of individuals currently populating the routing tables decay;
4. Propagate ratio—the number of chromosomes exchanged between populations, expressed as a %node population size;
5. Propagate freq—constant rate/frequency of exchange of chromosomes between populations;
6. Flow clear freq—a constant c_3, time interval over which data packet destination statistics are collected.

Default values for GA-agent were established in [15]. Table 2 summarizes parameter values employed in the following experiments for both AntNet and GA-agents.

4.2. Network scenarios

A total of four simulation scenarios are considered for the AntNet and GA approaches, all of which utilize the Japanese backbone network topology (Fig. 5). Moreover, unlike the original study, we concentrate on network reconfiguration properties [4]. In the first case, all routers remain available, scenario 1. The remaining experiments investigate plasticity of the agents by introducing fault conditions. First, router

Table 4
Static parameters—scenario 2

Algorithm	GA-agent	GlobalAnt	LocalAnt
Finish time (s)	1507	1668	1369
Routing packets (%)	58.9	10	11
Arrived packets (%)	70.6	92.3	41
Dead packets (%)	29.4	7.7	59
AP avg. trip time (ms)	356	998	2899

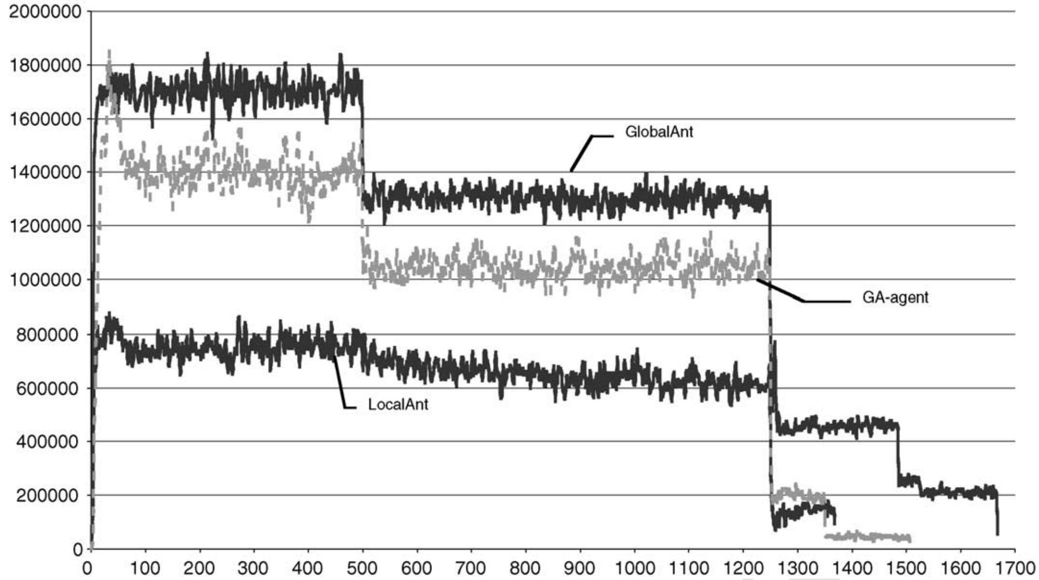

Fig. 8. Throughput (bytes) vs. time (s)—node 34 lost at 500 s.

R34 is removed at a time step of 500 s, scenario 2, where this effectively cuts the network in two, with only one path linking the two halves. In scenario 3, two routers (R49, R13) are removed, whereas in scenario 4, the same two routers (R49, R13) are taken *down* asynchronously, but return later synchronously. Scenario 4 is, therefore, of particular interest because it requires three different reconfigurations—once in the introduction of each fault and again when all the faults are restored.

In all cases the performance of routing algorithms is measured from multiple perspectives,

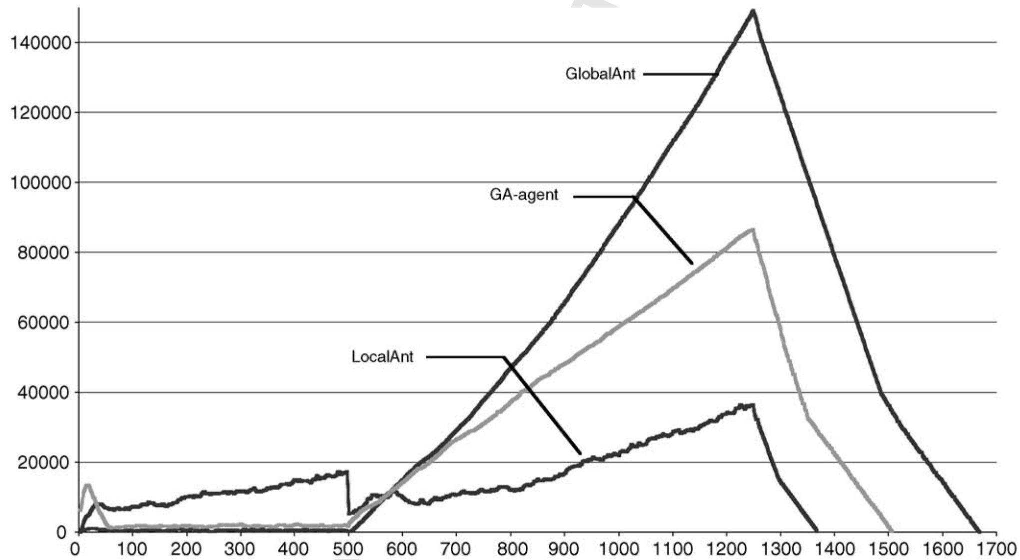

Fig. 9. Queue length vs. time (s)—node 34 lost at 500 s.

- Network throughput, which is defined as the number of data packet bytes successfully received at their destination per two second window;
- Average queue length, where this is the average of the number of packets—data and routing—per two second interval over the network as a whole;
- Total time to deliver all the data packets that are not lost or killed (finish time);
- Number of arrived data packets (AP) as a percentage of the number of data packets generated;
- Average trip time of arrived data packets, and;
- Number of routing packets created during the course of the simulation, again expressed as a percentage of the number of data packets generated.

In the case of throughput and queue length, we are interested in capturing the temporal characteristics; hence plots are used over the duration of the simulation. All other parameters are summarized by a single numerical value.

4.2.1. Scenario 1—no network failure

Table 3 summarizes the static parameters over a network experiencing no failure conditions, whereas Figs. 6 and 7 represent throughput and queue length, respectively.

Table 5
Static parameters—scenario 3

Algorithm	GA-agent	GlobalAnt	LocalAnt
Finish time (s)	1252	1466	1300
Routing packets (%)	51.6	10	11
Arrived packets (%)	71.4	94.3	41.7
Dead packets (%)	28.6	5.7	58.3
AP avg. trip time (ms)	861	1325	1617

The linearly increasing queue length property (and low throughput) of the LocalAnt algorithm indicates that a good routing strategy has not been identified (Figs. 7 and 6). Moreover, from Table 3, it is evident that more packets are lost than successfully reach their destination. That is to say, without the global information, the LocalAnt algorithm is unable to stop packets from revisiting nodes more than once. In effect, the data structure used to support (global) positive feedback is no longer available. The GA-agent algorithm delivers twice as many packets, but with a significant overhead in the number of routing packets utilized. This property will be revisited in the discussion.

After an initial configuration period (typically 100 s for the GA scheme) GA-agent and GlobalAnt control queue length effectively (Fig. 6). GlobalAnt

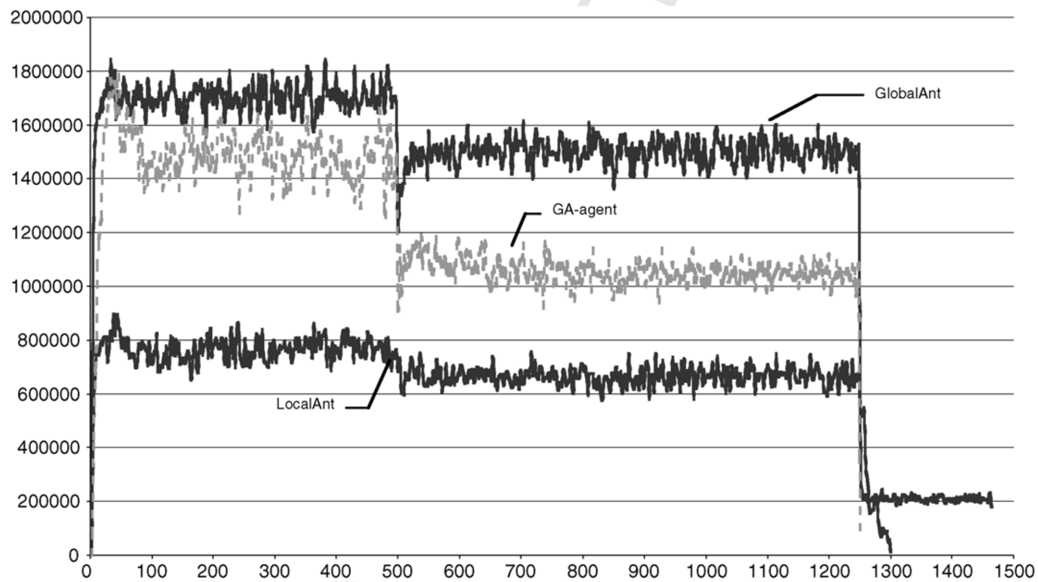

Fig. 10. Throughput (bytes) vs. time (s)—nodes 13 and 49 lost at 500 s.

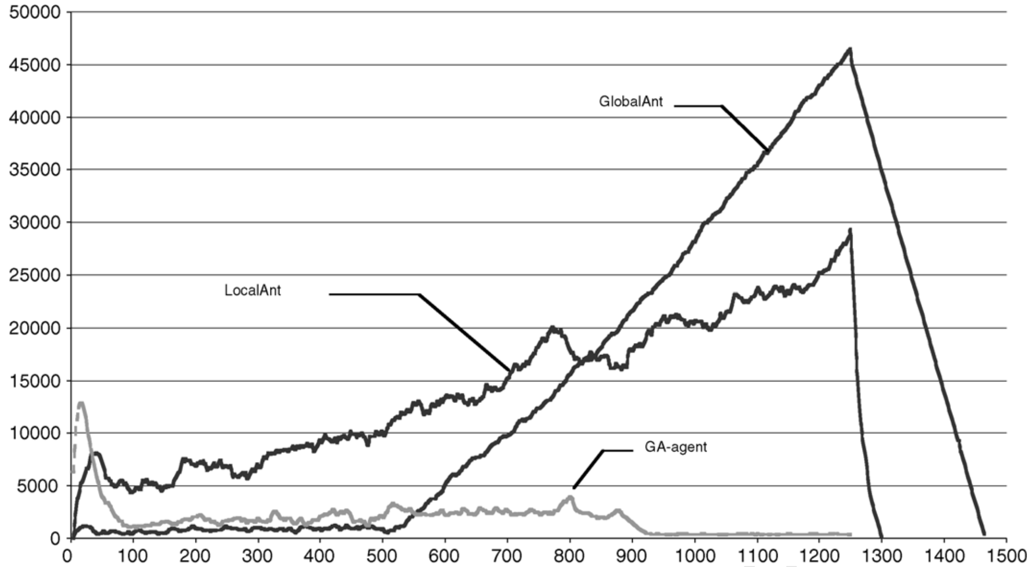

Fig. 11. Queue length vs. time (s)—nodes 13 and 49 lost at 500 s.

''loses'' the least packets (0.3% as opposed to 15% for GA-agent) (Table 3) and maintains the highest levels of throughput (Fig. 7). LocalAnt returns the shortest average trip time for a delivered packet, but this is most likely a reflection of the low number of packets actually delivered (Table 3).

4.2.2. Scenario 2—node 34 lost at 500 s

In this scenario, node 34 is removed at time step 500, where node 34 represents a critical node for connectivity (Fig. 5). Table 4 and Figs. 8 and 9 summarize the performance. The LocalAnt algorithm continues to loose more packets than it delivers (implying that more packets attempt to revisit nodes than find a direct path) and in addition returns the longest trip time for those packets that are delivered (Table 4). On account of the reduction in the number of packets delivered, the LocalAnt queue length profile is now better than GlobalAnt (Fig. 9) whereas throughput is still the worst (Fig. 8). The linear increase in GlobalAnt queue length is an indication of the significance of node 34, where the same property is observed by GA-agent. That is to say, in order to avoid loosing packets down paths previously available in node 34, it is necessary to queue packets waiting for the low number of alternative routes.

4.2.3. Scenario 3—node 13 and 49 lost at 500 s

In this case, we are interested in the case of multiple network failures, with Table 5 and Figs. 10 and 11 summarizing performance. LocalAnt is still losing far more packets than it is delivering (Table 5) which naturally results in low throughput and queue length profiles (Figs. 10 and 11). The GlobalAnt algorithm still loses the least number of packets (Table 5). Performance of GA-agent again appears to fall between that of Local and Global versions of the AntNet algorithm. Moreover, the GA-agent in this case appears to have identified alternative routes that have minimal impact on queue lengths (Fig. 11).

4.2.4. Scenario 4—asynchronous removal of two nodes

Here, the effect of different routers going down at different times and recovering (at the same time) is

Table 6
Static parameters—scenario 4

Algorithm	GA-agent	GlobalAnt	LocalAnt
Finish time (s)	1252	1252	1289
Routing packets (%)	54.5	10	11
Arrived packets (%)	78.8	96.5	43.6
Dead packets	21.2	3.5	56.4
AP avg. trip time (ms)	1012	677	3259

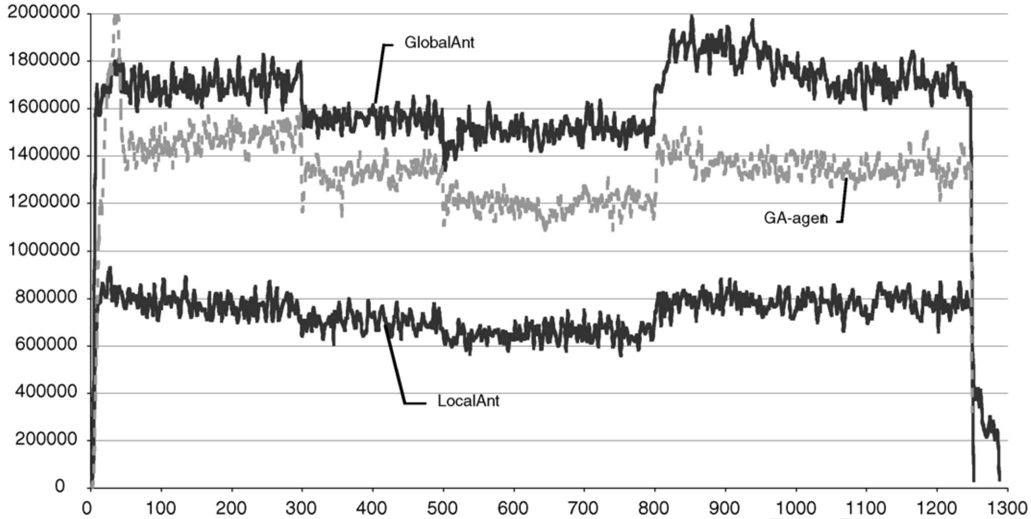

Fig. 12. Throughput (bytes) vs. time (s)—asynchronous removal of two nodes.

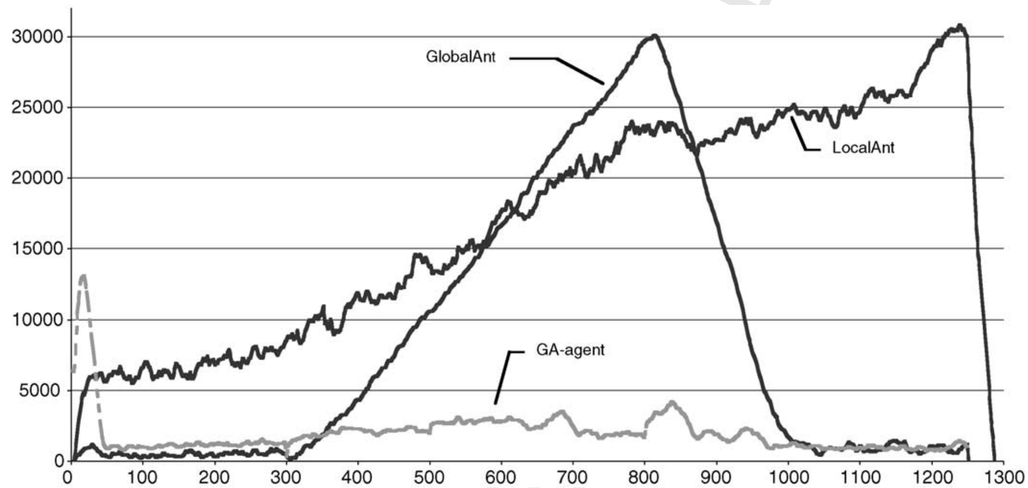

Fig. 13. Queue length vs. time (s)—asynchronous removal of two nodes.

investigated: Table 6 and Figs. 12 and 13. Both the AntNet algorithms make use of queues (Fig. 13) possibly implying the utilization of a small number of preferred alternative routes. The GA-agent strategy appears to minimize queue lengths at the expense of higher dead packet counts with respect to GlobalAnt (Table 6). Throughput profiles follow the same general pattern as previously encountered—GlobalAnt consistently has the highest throughput, with GA-agent performance midway between Global and Local AntNet (Fig. 12). It is also interesting to note that, once all network connections are re-established, all three algorithms successfully return to throughput levels each identified before any faults were introduced.

4.3. Discussion

By way of an overall ranking, it is clear that the utilization of global information in the AntNet algorithm plays a central role in its performance. Without this—LocalAnt—more dead packets occur than packets delivered, irrespective of whether there are missing links or not. GA-agents clearly perform far better than LocalAnt, typically delivering twice as many packets, irrespective of the scenario. Moreover, it was also apparent that GA-agents evolved different strategies depending on the location of the population. Populations associated with nodes at the periphery of the network tended to have relatively short chromosomes (around five genes or less). Those associated with nodes having higher degrees of connectively tended to have much longer chromosomes (around 10 or more genes). Future work will investigate this property further within the context of co-evolutionary strategies.

The sizes of the routing tables are significantly different. For a router with l neighbors in a network with n routers, we make the following comments. A GlobalAnt router has l records, each has $(n - 1)$ fields for each node in the network. Thus, the size of the routing table is $l(n - 1)$, i.e. $\Theta(l \times n)$, where usually $l \ll n$, so, the size of the routing table is $\Theta(n)$. Since the routing table is a two-dimensional array, the next hop look up time is only $\Theta(1)$. A LocalAnt router has l records (number of neighboring links), each has only two fields, one for the neighbor, one for the rest of the network. Thus, the size of the routing table is $l \times 2$, i.e. $\Theta(l)$; the next hop look up time is also only $\Theta(1)$. A GA-agent based router has a population of $c_1 \times l^2$ chromosomes, thus the routing table has $O(l^2)$ records, and each represents an explored route. According to the statistics of the experiments, routes have approximately 2–12 genes; this fits a $\Theta(l)$ relation. Thus the size of a routing table is $O(l^3)$. Sequential search of the routing table will take $O(l^3)$ time.

Finally, the relationship between routing agents and routing tables also differs significantly between the two approaches. The AntNet algorithm currently updates all routing tables along the return path of an ant. GA-agents in its current formulation only update the table of the source node. This means that for each routing packet (Ant or GA-agent) more routing tables are updated per agent in the case of the AntNet algorithm. Modifying the GA-agent scheme along the lines of the AntNet update process would significantly reduce the number of routing packets necessary. However, this approach also represents a serious security issue from the perspective of network management. In effect, any node is able to modify the routing table of another node, so opening the door to malicious modification of the network routing.

5. Conclusion

Dynamic/adaptive routing has become a research topic of significant interest over the last five years. The growing size and increasing demands placed on packet switched networks has pushed their application into areas not considered at their conception. As a consequence routing techniques currently in operation are increasingly being shown to be ineffective [16]. Extensive simulation of the AntNet algorithm against routing algorithms currently in use—OSPF, SPF, BF, Q-R, P-QR and Daemon—has demonstrated the superiority of AntNet under dynamic load conditions [4]. In this work, we show that such performance comes at a cost. Global information is necessary and network security may be compromised. Removing access to global information is shown to compromise the ability of the AntNet algorithm to find suitable routes, even in the case of a static network configuration (no faults).

In order to reduce the significance of these drawbacks, we investigate the utilization of a genetic algorithm based on a static multi-population model. To do so, the GA representation problem is addressed, such that agents do not require global information regarding network topology. In addition, the network security problem is reduced as agents may only modify the routing table of their source. The penalty paid for this is a reduction in routing capacity, with performance falling between that of the AntNet algorithm with and without global information. We believe, however, that this establishes a baseline of performance for a routing algorithm that does conform to all the constraints of a network routing problem in practice. Future work will concentrate on two general topics. Firstly, the investigation of more advanced co-evolutionary techniques to promote the sharing of information between routing agents whilst minimizing the potential for compromises in network security. Secondly, the organization of routing table information should be addressed such that identifying a required route is much more efficient than is currently the case. Other opportunities for improvement might concentrate on the optimization of the GA search operators. Specifically, the indirect encoding scheme has been improved by introducing biases into the selection of crossover and mutation points [17]. Moreover, multi-population models based on the island model abound [14], from which we might learn of improved schemes for parameter adaptation and migration.

Alternatively, the problem of information sharing could be addressed by letting data packets carry routing information with them (similar to the case of adaptive routing [3]). Thus, it is no longer necessary for each routing table to provide routes to all destinations and data packets maximize the utilization of routing information when it is provided (the current implementation only uses the source to destination path specified in a GA-agent routing table to select the next hop). Finally, instances of the AntNet algorithm should be investigated in which there are more than two columns, but less than the number of nodes in the entire network. In this case, the objective is to dynamically identify what destinations each column should correspond to.

References

[1] B.A. Forouzan, Data Communications and Networking, McGraw Hill, 2001 ISBN 0-07-232204-7.

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

[3] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, G. Minden, A Survey of Active Network Research, IEEE Commun. Mag. 35 (1) (1997) 80–86.

[4] G. Di Caro, M. Dorigo, AntNet: distributed stigmergetic control for communications networks, J. Artif. Intell. Res. 9 (1998) 317–365.

[5] M. Heusse, D. Snyers, S. Guerin, P. Kuntz, Adaptive agent-driven routing and load balancing in communication networks, Adv. Complex Syst. 1 (1998) 237–254.

[6] E. Gelenbe, Z. Xu, E. Seref, Cognitive packet networks, in: Proceeding of the 11th IEEE International Conference on Tools with Artificial Intelligence, 1999, 47–54.

[7] M. Munetomo, The Genetic Adaptive Routing Algorithm, in: D.W. Corne, M.J. Oates, G.D. Smith (Eds.), Telecommunications Optimization: Heuristic and Adaptive Techniques, John Wiley & Sons, 2000, pp. 151–166 (Chapter 9), ISBN 0-471-98855-3.

[8] S.H. Shami, I.M.A. Kirkwood, M.C. Sinclair, Evolving simple fault-tolerant routing rules using genetic programming, Electron. Lett. 33 (17) (1997) 1440–1441.

[9] J.A. Boyan, M.L. Littman, Packet routing in dynamically routing networks: a reinforcement learning approach, Adv. Neural Inf. Process. Syst. 6 (1994) 671–678.

[10] S. Liang, A.N. Zincir-Heywood, M.I. Heywood, The effect of routing under local information using a social insect metaphor, in: Proceedings of the IEEE International Congress on Evolutionary Computation, May 2002.

[11] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B 26 (1) (1996) 29–41.

[12] V. Maniezzo, A. Colorni, The ant system applied to the quadratic assignment problem, IEEE Trans. Knowl. Data Eng. 11 (5) (1999) 769–778.

[13] D.E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison Wesley, 1989 ISBN 0-201-15767-5.

[14] N. Toshine, N. Iwauchi, S. Wakabayashi, T. Koide, I. Nishimura, A parallel genetic algorithm with adaptive adjustment of genetic parameters, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), 2001, pp. 679–686.

[15] S. Liang, A.N. Zincir-Heywood, M.I. Heywood, Using distributed genetic algorithm intelligent packets for dynamic network routing, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002, New York, Morgan Kaufmann, 9–13 July, 2002, pp. 88–96.

[16] G. Huston, Analyzing the Internet BGP routing table, The Internet J. 4 (1) (2001) 2–15.

[17] H.-L. Fang, P. Ross, D. Corne, A promising genetic algorithm approach to jop-shop scheduling, rescheduling, and open-shop scheduling problems, in: Proceedings of the 5th International Conference on Genetic Algorithms, 1993, pp. 375–382.