Co-evolutionary Agents for Telecommunication Network Restoration

Sajjad H. Shami and Mark C. Sinclair

Dept. of Electronic Systems Engineering,
University of Essex, Wivenhoe Park,
Colchester, Essex CO4 3SQ, UK.

Abstract. A novel system of simple mobile software agents is described which, in the event of node or span failure in an arbitrary transport network, quickly spread out through reproduction, cooperate with other agents and gather useful restoration information on the way. On encountering their destinations, or other critical situations, they return to the affected nodes in real-time. The algorithm is capable of handling multiple span failures or up to one node failure at a given time and provides ample restoration information within one second.

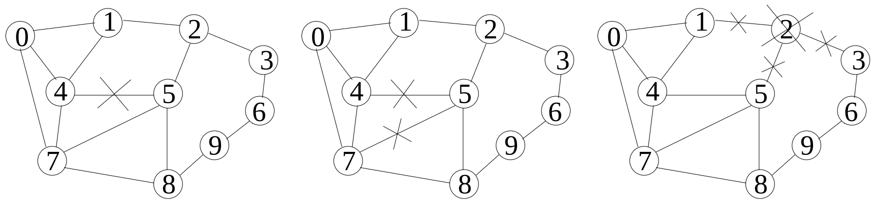
Keywords. Telecommunication networks, restoration, software agents

1 Introduction

Agent-based solutions to complex issues in telecommunications are increasingly being introduced. Whereas stand-alone software agents present an effective model for modern computing, their power becomes significantly enhanced when they are given the ability to cooperate with each other [1]. Within telecommunication networks, restoration is a multi-constraint optimisation problem that has to adapt and respond to a changing environment. These factors make it difficult for reliable distributed network control software to be developed by hand. This has led researchers to find alternative approaches to tackling the issue, including the use of agent-based software. With the widespread deployment of high-capacity fibre networks, advanced methods of restoration have become necessary. The agent system we describe was inspired by Grover's Self-Healing Network (SHN) [2] and Liu *et al.*'s co-evolutionary agent system for image feature extraction [3]. Grover's algorithm covers only one span cut in the network at a given time and does not accommodate node failure, whereas our algorithm can handle multiple span failures, as well as single-node failure.

1.1 Span and Node Failure Restoration

The individual agents in the system have minimal knowledge of the network topology and work on rules designed for very localised situations. A simple example depicting the single span failure scenario is shown in Fig. 1(a).



(a) Single-span (b) Two-span (c) Single-node

Fig. 1. Failure types

Span (4-5), connecting Nodes 4 and 5, fails and then the first restoration route found (described as a node sequence) is 4-7-5. The next three routes would be 4-0-7-5, 4-1-2-5 and 4-7-8-5. In Fig. 1(b) Spans (4-5) and (5-7) have failed simultaneously. The algorithm has to find restoration routes for two sets of affected nodes. In comparison, Fig. 1(c) shows the failure of Node 2, which is equivalent to the simultaneous failure of all three connected spans. Node 2 now cannot communicate in any way. What can be done, however, is that traffic previously on path 1-2-5 can now be sent on path 1-4-5. Similarly, traffic previously on path 3-2-5 can now be sent on path 3-6-9-8-5. The proposed algorithm picks up possible restoration paths and hence partial restoration is realised for single node failure situations.

2 The Algorithm

When a span is cut, at each of the two affected nodes a *forward agent* is created whose aim is to seek out restoration routes for that cut. At the start, the forward agent is given certain initial energy that is used up as the agent moves. On its way, the individual forward agent stores the route it traverses and thus does not move to nodes which it has already visited. When the destination node is encountered, or some other critical information is obtained, the agent transforms into a *return agent* which moves backwards along the stored route to the originator, and delivers the acquired full or partial restoration information. The originator, during a very short time period, may receive several return agents. For a particular destination the return agents may provide anything from zero to several restoration routes.

Each node structure in the simulation maintains its node id; an alarm status which is turned on when any connected span is cut; its original degree; and the number of connected spans cut (if any). Each span structure maintains its span id and span status (intact or cut). Each mobile agent carries with it its type (forward or return); the node ids of the originator, the destination and the previous node; a *route* array, that stores the ids of each node traversed from the originator; a time field that tracks agent time *wrt.* the global time; the present energy of the agent; and finally a *cut-node* array which stores the ids of all nodes encountered that have a span cut from the destination node. At present the system simulation employs synchronous time. We have assigned message processing time to be 35 ms and message transmission time to be 15 ms so that one time step, *i.e.* the time to transfer from any node to an adjacent one, to be exactly 50 ms.

2.1 Basic Features

This agent system has certain basic characteristic features that jointly comprise the core behaviour. Intelligent and stochastic features (described later) were added subsequently to make the whole operation more efficient.

Firstly, *agent energy* is an estimate of the relative likelihood that a particular agent will find the first restoration route. To start with, a newly created forward agent is given initial energy equal to, say, $K_i = 100$ times the number of spans, m, in the network. Then for each move to an adjacent node, a fixed amount, say $K_s = 5$, is deducted. However the deductions do not apply to return agents. Next, *forward travel and reproduction*, whereby a forward agent travels ahead by copying itself to all adjacent nodes subject to the condition that the adjacent node has not been visited already. Conservation of energy is implemented whenever agent reproduction occurs. *Agent death* is defined to be the situation where a forward agent does not move, reproduce or generate a return agent. If a forward agent reaches a node and finds that all connected adjacent nodes have already been visited, it is said to experience blockage. In this eventuality, the agent dies. Another situation that results in death of an agent is when its energy level reaches (or goes below) a preset minimum level, typically zero units. *Return travel* is a characteristic feature of return agents. A forward agent transforms into a return agent either on reaching the destination or a cut-node and starts a return journey on the exact path that it has traversed. When a return agent from the destination reaches its originator, the originator node spots the destination as the last member of the agent's route array and declares the query to be *routed.* However, when a return agent reaches its originator from a cut-node, the originator declares that the destination *may not be connected.* Lastly, *node failure detection and restoration* is realised as follows. When all return agents have reached the originator and none contains the destination in its route array *i.e.* all are returning 'may not be connected' messages, the originator concludes decisively that the destination is not connected to the network and that it is a case of node failure. Upon establishing node failure, the route and cut-node arrays of all return agents can now be utilised to provide partial restoration.

2.2 Initial Results

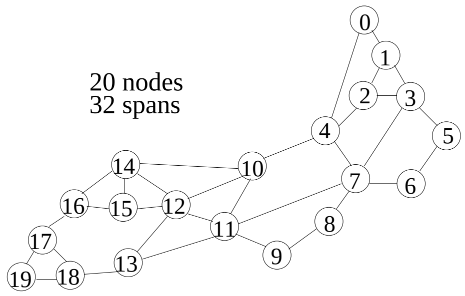
The system was tested on three hypothetical transport networks, namely the UK national, US long-haul and the Japanese national. The Japanese network is shown in Fig. 2(a). The plot in Fig. 3(a) illustrates the growth in agent numbers over time following failure of one span, namely Span (7-11), in the Japanese network using four different initial energy values. Clearly, lower energy values, and the resulting agent deaths, are acting both to limit the growth of agent numbers and the time for the last agent to return.

3 Intelligent and Stochastic Features

In realistic networks, the first (and second and further) restoration route(s) following a span failure would be found relatively quickly and there would be hundreds of agents still roaming about in the network wasting computing and communication resources. By introducing limited initial energy and agent death, these numbers can be contained. However, there is the possibility that a potential restoration solution is not found, because the agents die before discovering it. To tackle this, we have incorporated several stochastic elements in the agent system. *Early-route completion* (ERC) occurs when two forward agents from the opposite ends of the same query, meet at an intermediate node, in the same time step. They complete their desired routes at that node by cooperating and generate return agents right there, instead of going all the way to their respective destinations and then generating return agents. *Early-info completion* (EIC) takes place when two forward agents from different originators, but seeking the same destination, meet at an intermediate node, in the same time step. They complete their desired routes and cut-node sets at that intermediate node and send return agents as in ERC. As ERC and EIC are possible only if the path for the counterparts comprises an odd number of nodes, we introduce *probabilistic waiting* (PW) so as to give some chance of success to situations where the path has an even number of nodes. PW enables an agent to wait for one time step at an arbitrary node on its forward route. This is carried out stochastically. By incorporating *probabilistic death* (PD) we introduce a range of agent energy above and below zero where an agent is allowed to live or die stochastically. When *probabilistic splitting* (PS) is incorporated, some offspring may be stochastically killed off at the time of forward reproduction. In addition, more efficient utilisation of the parent agent's energy is realised as energy splitting is quantised to multiples of K_s. This ensures that most small fragments ($< K_s$) are combined to allow additional moves to one or more offspring agents.

3.1 Experimental Results

For all three model networks, we recorded the change in the number of agents as restoration proceeded, plus the times to find the first, second, third and



(a) Japanese network (after [4])

Scenario	ERC	EIC	PW	PD	PS
1					
2	√				
3	√	√			
4	√	√	√		
5	√	√	√	√	
6	√	√	√	√	√

(b) Agent system scenarios

Fig. 2. Experimental setup

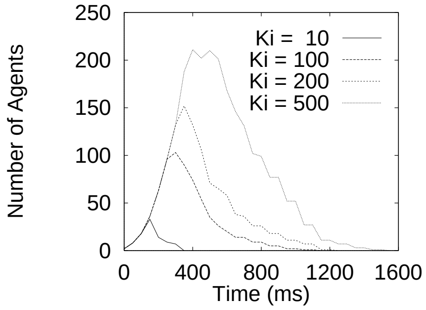
further restoration routes for arbitrary span(s) and node failures. We experimented with six scenarios and we monitored improvement relative to Scenario 1 (which has no intelligent features). The six scenarios are defined in Fig. 2(b). For Scenarios 4, 5 and 6, ten runs were carried out with different random seeds for the stochastic features and the differences analysed using non-parametric median tests [5]. Fig. 3(b) shows very marked reduction in agent numbers for Scenario 2 after a four span failure in the UK network. Fig. 3(c) shows agent numbers for Scenarios 1, 2 and 3 after a seven span failure in the US long-haul network. Fig. 3(d) shows agent numbers for Scenarios 1, 3, 4, 5 and 6 after a node failure (Node 11) in the Japanese network (Scenario 2 does not affect node failure).

4 Conclusion & Further Work

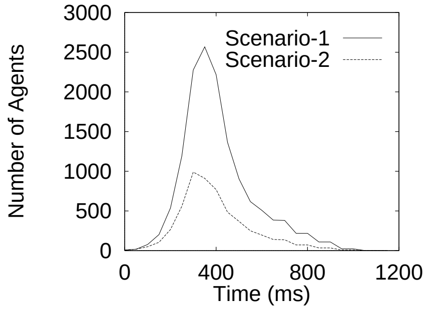
Overall the agent system appears quite viable based on the results obtained. In future work, we intend to (i) use genetic algorithms to tune agent parameters (such as initial energy, agent-death probabilities, *etc.*) to individual networks, thereby enhancing both the efficiency and accuracy of the agent system; (ii) make the system work in asynchronous time conditions; and (iii) enhance the algorithm to include explicit establishment of restored paths.

Acknowledgements

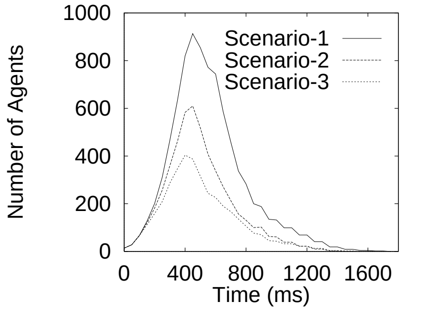
The research described here was undertaken by the first author (supervised by the second) as part of a Ph.D. project sponsored by the Government of Pakistan. The authors are grateful to Joshua Knowles (University of Reading) for the suggestion of agent waiting.



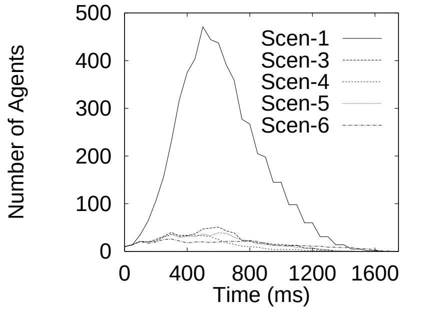
(a) Japanese network after one span failure with various K_i



(b) UK network after four span failure with $K_i = 200$



(c) US long-haul network after seven span failure with $K_i = 200$



(d) Japanese network with single node failure with $K_i = 200$

Fig. 3. Results

References

1. Bradshaw, J.M. (Ed.) (1997) Software Agents, AAAI Press/MIT Press
2. Grover, W.D. (1987) The self-healing network: A fast distributed restoration technique for networks using digital cross-connect machines, Proc. IEEE Global Telecom. Conf. (GLOBECOM'87), Tokyo, 1090–1095
3. Liu, J., Tang, Y.Y., Cao, Y.C. (1997) An evolutionary autonomous agents approach to image feature extraction, IEEE Trans. on Evolutionary Computation, **1**, n2, 141–158
4. Munetomo, M., Takai, Y., Sato, Y. (1997) An adaptive network routing algorithm employing genetic operators, Proc. 7th Intl. Conf. on Genetic Algorithms (ICGA'97), Michigan, USA, 643–649
5. Sprent, P. (1989) Applied Non-parametric Statistical Methods, Chapman and Hall, UK, 7–9, 86–87