Optimising a Rule Based Agent using a Genetic Algorithm

Evaggelos Nonas[1]

Abstract. Software intelligent agents and more specifically BDI (beliefs, desires intentions) agents have been widely used in self adaptive systems. Their common attributes with active databases have also been studied. This paper assumes that the knowledge of an agent is expressed in the form of event-action rules and proposes a method for optimising the rule base of such an agent using a genetic algorithm. We also describe how this technique is applied to the problem of optimising the performance of a self adaptive network.

1 Introduction

BDI agents have been around for some years now and have been extensively studied ([5],[9],[11],[2]). A BDI agent is composed of beliefs, desires, Plans and Intentions.

Active databases are also based upon an Event-Action architecture ([4]). An active database system consists of the "traditional" components of a database system plus a component that is concerned with the firing of ECA (event, condition, action) rules. The meaning of an ECA rule is: "when an event occurs check the condition and if it is true execute the action".

Similarities and differences between BDI agents and active databases are extensively discussed in [1],[12]. One can easily see that the most important of their common characteristics is the way actions are taken: upon a certain event if a condition holds a rule is fired. But there are cases where more than one rule applies for the same event and condition. The system will then either fire a rule randomly or just execute the first rule whose event and condition parts are true.

2 Architecture

What we propose, based on the previous observations, is an "automatic" way of rule firing upon an event. For the moment there is no condition part in out architecture, although we plan to add a condition part in the future. When an event has occurred the system can take several actions. For each of the events, the system holds an ordered set of possible actions that can be taken when the event occurs. The first action is always selected, but a genetic algorithm running in parallel can dynamically change the order of the actions. Obviously this approach requires a measure of the performance of the agent, which must be available at run time, to be given to the genetic algorithm. So one can say that the reactive behaviour of the agent is controlled by the genetic algorithm. The rational part of the agent can then control several parameters of the GA, restart it when needed, or schedule it to be run when the load is low.

Our architecture can also be embedded in more complex systems. When event and actions languages are necessary for the building of an agent type system, our method can be used for a subset of the events and the actions of the system. It simplifies the design and reduces testing and maintenance times when compared to a complicated program with many conditions and checks.

3 Application

Genetic algorithms have been used before in the design on agent systems ([8],[10]). We use genetic algorithms in order to evolve the rule base of agents whose task is to optimise self adaptive networks.

A self adaptive network can be defined as a network that can automatically adapt to changes in its environment without human intervention being necessary. The network behaves like a black box. While load conditions change and nodes and links may fail, the network continues to operate near the optimum state requiring little or no assistance from its operators. In other words, the network must be autonomous, intelligent and have distributed control. There should be no global knowledge for the network. On the contrary all information must be kept as local as possible.

Our network model is a simple but yet powerful one. The network is composed of a set of nodes and a set of connections between them. Each node can exchange messages only with the ones it is connected with. There is no global knowledge for the topology of the network stored in any node. There is a set of services provided by the network and each node can provide some or all of the services. The task for every node is to provide the services requested from it with the minimum cost.

In each node of the network there is an agent running which is built using the proposed architecture. There is a set of events and a set of actions for each event. When a service is requested by a node there are two actions that can be triggered: the service can be provided either remotely or locally. When a request for a service is to be sent to another node a separate event is generated for each service. The actions corresponding to this event are all the nodes that this node is connected with.

A simple genetic algorithm ([7],[3],[6]) is used to try out

[1] Department of Computer Science, Kings College London,UK. Email: vagelis@dcs.kcl.ac.uk

© 1998 E. Nonas
Submitted to ECAI 98
April 27, 1998

several permutations of the rule set and finally find the best rule set for each node. We define a correspondence between all possible permutations of the actions for an event and an integer between 0 and n!-1, where n is the number of actions. The binary representation of this number is placed in the chromosome to encode the ordered set of actions for an event. The whole chromosme is composed of K (where K is the number of events) numbers placed in it, in their binary representation, one after the other. In this way one chromosome can encode the rule set with which each agent works. Each agent has a chromosome pool which is initially randomly instantiated. Then the chromosomes in the chromosome pool are evolved by the genetic algorithm to better solutions. We use a constant population size, selection proportional to fitness and full replacement of parents by their children. Miltiple point crossover is used for breeding. Crossover points are set at the end of each event in the chromosome.

The fitness of each rule is calculated as follows: When a node provides a service to another node, it also sends to it the cost of the service. This cost, is a function of the number of the number of intermediate nodes and links the service is using as well as their load and free capacity respectively. Obviously, when the service is provided locally, the cost is minimum. Each rule in the chromosome pool is used for service provision for some time and the costs of the services provided using this rule are averaged. The fitness then for this rule is inverlsy proportional to this average cost. So, the bigger the cost, the smallest the fitness for the rule and vice-versa.

4 Conclusions and future work

The advantages of our approach are apparent: distributed solution, load balancing and sharing, self adaptation to varying load conditions and fault situations, and near optimum performance.

One could argue that the genetic algorithm can find a local optimum and then stop. This is always a danger with a genetic algorithm, but again it depends on the search space. In a network where service distribution across nodes is done in such way that neighbouring nodes have some services in common there are many good solutions and the genetic algorithm will find one of them. In extreme cases where there is only one good solution the genetic algorithm may fail, but again it can be restarted by the agent with many chances to find a better solution. Finally, in such cases the advantages of adaptation, autonomy and distributed operation are more important than the discovery of the best solution, especially in a dynamic and continually changing environment where keeping track of global information would be difficult if not impossible.

One could also argue that this architecture is not powerful enough since it does not work based on an event and action language. However there is nothing to prevent this architecture from being a subset of a rich and powerful event and action language. In such a case it can be used to pick the rule to be fired when there is no other criteria for rule selection. In other cases it may be better to let the genetic algorithm pick the rule to be fired, instead of having many conditions which will complicate the program and consequently increase design, test and maintenance times.

Future work could include the use of a different type of a genetic algorithm for optimising the rule set, or even another method for finding the optimum rule set (neural network, or other heuristic search method like simulated annealing). Finally an event and action language for our network simulator is also included in our future research directions.

REFERENCES

[1] J. Bailey, M. Georgeff, D. B. Kemp, and D. Kinny, 'Active databases and agent systems — A comparison', *Lecture Notes in Computer Science*, **985**, 342–356, (1995).

[2] Michael Bratman, *Intention, plans, and practical reason*, Harvard University Press, 1987.

[3] Lawrence Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[4] K. R. Dittrich, S. Gatziu, and A. Geppert, 'The active database management system manifesto: A rulebase of ADBMS features', *Lecture Notes in Computer Science*, **985**, 3–17, (1995).

[5] Klaus Fischer, Jörg P. Müller, and Markus Pischel, 'A pragmatic BDI architecture', in *Proceedings on the IJCAI Workshop on Intelligent Agents II : Agent Theories, Architectures, and Languages*, volume 1037 of *LNAI*, pp. 203–218, Berlin, (19–20 August 1996). Springer Verlag.

[6] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass., 1989.

[7] David Goldberg, *Genetic Algorithms*, Addison Wesley, Reading, 1989.

[8] Thomas Haynes and Sandip Sen, 'Evolving behavioral strategies in predators and prey', in *IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*, pp. 32–37, (1995).

[9] David Kinny, Michael Georgeff, and Anand Rao, 'A methodology and modelling technique for systems of BDI agents', in *Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, volume 1038 of *LNAI*, pp. 56–71, Berlin, (22–25 January 1996). Springer Verlag.

[10] Mauro Manela and J. A. Campbell, 'Designing good pursuit problems as testbeds for distributed AI: A novel application of genetic algorithms', in *Proceedings of the 5th European Workshop on Modelling Autonomous Agents in a Multi-Agend World (MAAMAW'93)*, volume 957 of *LNAI*, pp. 231–252, Berlin, GER, (August 1995). Springer.

[11] Anand S. Rao and Michael P. Georgeff, 'BDI agents: from theory to practice', in *Proceedings of the First International Conference on Multi–Agent Systems*, pp. 312–319, San Francisco, CA, (1995). MIT Press.

[12] Johan van den Akker and Arno Siebes, 'Enriching active databases with agent technology', in *Proceedings ot the First International Workshop on Cooperative Information Agents*, volume 1202 of *LNAI*, pp. 116–125, Berlin, (February26–28 1997). Springer.