Evolution of Recurrent Neural Networks to Control Autonomous Life Agents

Tony Abou-Assaleh
Dept. Computer Science
University of Waterloo
Waterloo, ON, Canada
N2L 3G1
taa@acm.org

Dr. Jianna Zhang
Dept. Computer Science
Brock University
St. Catharines, ON, Canada
L2S 3A1
jianna@cosc.brocku.ca

Dr. Nick Cercone
Dept. Computer Science
University of Waterloo
Waterloo, ON, Canada
N2L 3G1
ncercone@math.uwaterloo.ca

Abstract

Studies of artificial life (alife) attempt to simulate simple living beings. On the other hand, autonomous agents researchers are interested in building agents that are able to complete a particular task without supervision. In this research, these two areas of artificial intelligence are combined into what we call "Autonomous Life Agent" (ALA). ALA is an artificial agent that is sent to some environment in which to live without any supervision or any predefined behaviour rules. The primary goal of the agent is to learn how to survive in its artificial environment. We utilize a recurrent neural network (RNN) to determine the agent's actions. A novel ALA Training System is developed that evolves RNN topology and link weights simultaneously using genetic algorithms.

1 GENERAL DESCRIPTION

ALA has a number of internal variables such as energy and maintenance levels. Each variable is reduced as the agent acts in the environment by a user-controlled rate. The agent dies when one of the variables becomes negative. Therefore, to remain alive, the agent has to periodically increment the values of the internal variables by visiting specific cells in the world that correspond to each variable. This increment is analogous to acquiring energy from energy cells and receiving maintenance in home cells. There are three types of cells: empty cells that are passive, wall cells that are fatal, and active cells that increment internal variables. Active cells have a time delay before they can be active again after being used.

This task involves hidden system states. Some information about the world is not available to the agent as input. For instance, at any time, the agent can sense only the immediately surrounding cells and the current cell. Thus, the agent has to explore the world and encode its information in the agent's brain. Furthermore, some data is not available to the agent at all such as the time delay of the active cells. The agent must automatically discover this information by observing the visible system states.

2 APPROACH

The ALA Training System consists of five components: (1) agent, (2) genome, (3) population, (4) world, and (5) genetic engine. The internal structure of the agent is a three-layer RNN where each layer is fully connected with the next layer. Some of the nodes in the hidden layer have recurrent links. The genome completely defines an agent by specifying its topology definition and the link weights. The genome uses numerical encoding. The population uses tournament selection to select genomes for reproduction. The world evaluates agents by measuring how long they can survive in the environment. If an agent can live up to the maximum age then it is considered a solution and the evolution terminates.

3 EXPERIMENTAL RESULTS

We are able to evolve agents that can live a 10x10 world with 3 active cells. The agents start their life span from a random cell and are evolved on two distinct but similar environments. The agents are also able to live in unseen environment that is similar to the training environment. These agents have few hidden neurons (1-3) and some of them do not have any recurrent links.

4 CONCLUSION

The training system is robust, flexible, fully configurable, and efficient. The contribution of this research extends beyond building training and simulation environment. It leads to discovering that few nodes in the hidden layer of the RNN are sufficient to control ALAs in non-trivial world definitions. Furthermore, this research showed through empirical experiments that recurrent links do not enhance the capabilities of neural networks in controlling autonomous agents in this problem definition.

An immunological algorithm for discovering small-disjunct rules in data mining

Deborah R. Carvalho [1,2]
[2] Universidade Tuiuti do Paraná (UTP)
Computer Science Dept.
Av. Comendador Franco, 1860. Curitiba-PR
80215-090 Brazil
deborah@ipnet.com.br

Alex A. Freitas [1]
[1] Pontifícia Universidade Católica do Paraná (PUCPR)
Postgraduate Program in Computer Science
R. Imaculada Conceição, 1155. Curitiba – PR
80215-901. Brazil
alex@ppgia.pucpr.br
http://www.ppgia.pucpr.br/~alex

1 INTRODUCTION

In this work we propose a hybrid decision tree/immunological algorithm method for rule discovery that copes with the problem of small disjuncts. The basic idea is that examples belonging to large disjuncts are classified by rules produced by a decision-tree algorithm, while examples belonging to small disjuncts are classified by rules produced by a new immunological algorithm, specifically designed for discovering small-disjunct rules.

2 A HYBRID DECISION -TREE / IMMUNOLOGICAL ALGORITHM

Our hybrid algorithm consists of two phases. In the first phase we run the C4.5 decision tree induction algorithm. The induced, pruned tree is transformed into a set of rules. Hence, a decision tree with *d* leaves is transformed into a rule set with *d* rules (or disjuncts). Each of these rules is considered either as a small disjunct or as a "large" (non-small) disjunct, depending on whether or not its coverage (the number of examples covered by the rule) is smaller than a given threshold.

The second phase consists of using an immunological algorithm to discover rules covering the examples belonging to small disjuncts. We have developed a new immunological algorithm (IA) for this phase.

The recognition and response to antigens (small-disjunct examples) is performed by antibodies, which in our system are represented by IF-THEN rules.

Each antibody represents a conjunction of conditions composing a given rule antecedent. Each condition is an attribute-value pair. Each gene represents a rule condition of the form A_i Op_i V_{ij}, where the subscript i identifies the rule condition, i = 1,..., n (where n is the number of attributes); A_i is the i-th attribute; V_{ij} is the j-th value of the domain of A_i; and Op is a logical/relational operator compatible with attribute A_i – see Figure 2. B_i represents an active bit, which takes on the value 1 or 0 to indicate whether or not, respectively, the i-th condition is present in the rule antecedent.

A_1 Op_1\{V_{1j}..\}	B_1	. . .	A_i Op_i\{V_{ij}..\}	B_i	. . .	A_n Op_n\{V_{nj}..\}	B_n

Figure 1: Structure of an antibody (rule antecedent).

In our experiments we have used a commonplace definition of small disjunct, based on a fixed threshold of the number of examples covered by the disjunct. The general definition is: "A decision-tree leaf is considered a small disjunct if and only if the number of examples belonging to that leaf is smaller than or equal to a fixed size *S*." We did experiments with four different values for the parameter *S*: *S* = 3, *S* = 5, *S* = 10 and *S* = 15.

The examples of the test set were classified as follows: the examples belonging to large disjuncts are classified by rules produced by a decision-tree algorithm (C4.5) and examples belonging to small disjuncts are classified by rules produced by our immunological algorithm.

Table 1: Accuracy rate (on test set) comparing our hybrid C4.5/IA algorithm with C4.5

		Hybrid C4.5 / IA			
data set	C4.5	S = 3	S = 5	S = 10	S = 15
Adult	0,7860	0,7818	0,7782	**0,7927**	**0,7869**
Wave	0,7554	0,7239	0,6982	0,6135	0,5860
Connect	0,7862	0,7795	0,7773	0,7681	0,7625
Splice	0,4598	**0,4618**	**0,4638**	**0,4680**	**0,4803**
Hepatitis	0,8364	0,8254	0,8058	**0,8621**	**0,8534**
Segmentation	0,9767	0,9462	0,9344	0,9297	0,9326
Voting	0,9463	0,9144	0,8990	0,8800	0,8869
CRX	0,8453	0,7918	0,7707	**0,8521**	**0,8683**

In Table 1 the entries where the hybrid C4.5/IA outperforms C4.5 alone are shown in bold. (We used public-domain data sets from the UCI repository – www.ics.uci.edu/~mlearn/MLRepository.html.) The results of Table 1 show that, when *S*=10 or *S*=15, the hybrid C4.5/IA is competitive with C4.5 alone.

3 CONCLUSION

The above computational results show that, as long as we are careful to define what constitutes a small disjunct, the hybrid C4.5/IA is competitive with C4.5 alone. A reasonable definition of small disjuncts is a decision tree leaf node having ≤ 10 or ≤ 15 examples (*S*=10 and *S*=15). Smaller values of *S* would mean there are two few examples for reliable generalization, and larger values of *S* would go against the notion of "small" disjuncts.

Modeling of Evolution of Signaling Networks in Living Cells by Evolutionary Computation

Alexander Kosorukoff
Dept. of General Engineering
University of Illinois
at Urbana-Champaign
Urbana, IL 61801
alex3@illigal.ge.uiuc.edu

Jay Mittenthal
Dept. of Cell & Structural Biology
University of Illinois
at Urbana-Champaign
Urbana, IL 61801
mitten@life.uiuc.edu

David Goldberg
Dept. of General Engineering
University of Illinois
at Urbana-Champaign
Urbana, IL 61801
deg@illigal.ge.uiuc.edu

Abstract

This work models the emergence of signaling pathways in living cells using techniques borrowed from genetic and evolutionary computation. It uses five-level GA: five cooperating GAs working in parallel on different levels of abstraction.

The need to coordinate the behavior of individual cells for the benefit of the whole organism has led to the emergence of elaborate communication mechanisms known as signaling networks. Signaling networks relay information from the outside world to the nucleus of the cell so that it can make decisions adequate for the organism's survival.

The multi-level representation of our GA reflects the complexity of living cells. Such representation suggests the multi-layered structure of the algorithm. There are five levels of abstraction in this evolutionary procedure: (1) domain, (2) protein, (3) cell, (4) population, and (5) ensemble. An upper level embeds a set of objects of the next lower level. On each level we have some evolutionary operators defined: mutation, crossover, and selection, although not all of them should necessarily be present at each level. In terms of one particular level, the representation can be regarded as linear, so the usual GA procedures can be used. The genotype of each upper level is a set of genotypes of its adjacent lower level. This forms a hierarchical genotype structure that reflects the hierarchical nature of the object of modeling.

Finding appropriate criteria for selection was an important task in this research, because the corresponding criteria in nature are not known.

Results for biology and GAs

Software for modeling of signaling net evolution was developed. Experiments show how signaling nets emerge as a result of evolution, as well as showing the structure of signaling nets under different criteria of selection and different initial conditions.

In the presence of many layers of evolutionary process two effects can be observed: compensation and relativity of genetic operators. The lack of an operator on a particular layer is compensated by peer processes on other layers. The relativity of genetic operators is another property of evolutionary process. The observed result of the same process appears to be different for observers at different layers: an effect of crossover in one layer appears as a mutation to an observer in the adjacent upper layer.

The full version of this paper is available as IlliGAL Technical Report #2001006 at www-illigal.ge.uiuc.edu.

Acknowledgments

The authors would like to thank Chi-Sheng Ho for contributing valuable ideas, Jie Chen for discussions about signaling pathways. This research was supported by a fellowship from the Research Board of the University of Illinois. Professor Goldberg's contribution to this work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grants F49620-97-1-0050 and F49620-00-0163, by the National Science Foundation under grant DMI-9908252, by the U. S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0003.

Equilibrium Selection in a Sequential Multi-issue Bargaining Model Using Evolutionary Algorithms: A Preliminary Study

Norberto Eiji Nawa †‡[1]
† ATR International - ISD
2-2-2 Hikari-dai, Seika-cho
Soraku-gun, Kyoto 619-0288
Japan

Katsunori Shimohara †‡

Osamu Katai ‡
‡ Grad. School of Informatics
Kyoto University
Yoshida-honmachi, Sakyo-ku
Kyoto 606-8501, Japan

Abstract

Computational experiments were performed with a multi-issue alternating-offers bargaining model in which several issues are negotiated in sequence. The strategies that determine the agents' behaviors are generated by individual evolution strategies. As opposed to the conventional setting where the issues are disputed simultaneously in bundles, agents negotiate each issue individually. Conformity with game theoretic predictions was experimentally verified.

Recently, there has been a growing intensification in the relations between economics and computer science, more specifically with the area of artificial intelligence. Such cross-fertilization is doubtless beneficial for both sides. For instance, economics can loosen up from the restrictions of perfect rationality, that have been molding in a great extent, large part of their models. On the other hand, as the number of market institutions in the Internet grows, there is an imminent possibility that computational agents will increasingly play an important role in the determination of the market dynamics. Since the field of economics has dedicated much of its efforts to the study of coordination of decentralized systems populated with self-interested individuals, from the computer science point of view such body of work represents a valuable source of ideas for the design of innovative approaches to deal with distributed systems.

In this double-folded spirit, computational experiments were performed with a sequential multi-issue alternating-offers bargaining model, based on the framework presented in [1, 2]. A bargaining situation consists of two or more players trying to engage in a mutually beneficial agreement over a set of issues. The players, or agents, have a common interest to cooperate; the question that remains open is which one of the possibly several compromise settings will be chosen by the players. In an alternating-offers bargaining model, the path towards an agreement involves the direct interaction of the players, who will make offers and counteroffers to each other through several stages, until a compromise that is accepted by all the parties is achieved.

[1]Web address: `http://www.isd.atr.co.jp/~eiji`

In the conventional setting, the issues are disputed simultaneously. Our interest in the sequential setting of bargaining processes lays on the fact that often the negotiated issues have time-varying, inter-dependent complementarities. That is, from the point of view of the players, the valuation of certain issue may change according to the utilities that were obtained with other issues in the agenda; if the negotiation is made in bundles, the players have to consider these inter-relationships in advance in order to calculate the utilities of the possible outcomes and settle an agreement that provides a good trade-off between all the issues. By making the negotiation of the issues sequential, we expect to facilitate the evolutionary process to deal with such inter-issue complementarities.

The players have their behaviors determined by strategies which are developed by means of evolution strategies (ES). Time pressure over the agents is represented by a fixed discount on the players' utilities of the prize at stake; the longer the negotiation process takes, the lower the utility of the original unity. The task of the ES is to build strategies that lead to high payoffs.

Preliminary results showed that the evolved strategies qualitatively reproduce the game theoretic predictions, despite the fact that no assumptions concerning the rationality of the agents or their behaviors were made.

References

[1] E. H. Gerding, D. D. B. van Bragt, and J. A. L. Poutré. Multi-issue negotiation processes by evolutionary simulation: Validation and social extensions. Technical Report SEN R0024, CWI, Centre for Mathematics and Computer Science, 2000.

[2] D. D. B. van Bragt, E. H. Gerding, and J. A. L. Poutré. Equilibrium selection in alternating-offers bargaining models: The evolutionary computing approach. In 6^{th} *Int. Conf. of the Society for Computational Economics on Computing in Economics and Finance (CEF'2000)*, July 2000.

Stability of Arbitrary Genes: a New Approach to Cooperation

Maurizio Patrignani
Dipartimento di Informatica e Automazione
Università di Roma Tre
e-mail: patrigna@dia.uniroma3.it

Abstract

We describe a model to explain how an arbitrary gene may persist in a population of self-interested agents even when it is detrimental to the individual. Since the arbitrary gene may code for a cooperative behavior, disadvantageous to the owner but advantageous to the whole population, the model may be used to explain the persistence of cooperative behaviors in societies of selfish agents.

THE MODEL

Explaining the persistence of cooperative behaviors is particularly difficult when cooperation is not directly beneficial to the individuals fitness and when it involves more than a handful of agents, weakening any mechanism based on reciprocity. We consider an iterated cooperation opportunity whose advantage is shared between all individuals of the population, whether they are cooperators or not. Also, in our simple model the cooperative behavior is the same for all cooperating agents. In such a case, the gene that codes for cooperation tends to disappear, since defectors share the advantages of cooperation without sharing the costs. A widespread gene coding for a retaliation against defectors may support cooperation. Unfortunately, since the punishment of the defectors is a cooperative effort itself, the vengeful gene tends to disappear in its turn.

In our model each agent can attack any other, determining symmetric penalties. The gene that codes for the vengeful behavior, though, does not code for a direct attack against defectors. It triggers, instead, an attack against each agent that behaved differently from the owner agent in the previous step. To determine the targets of its attacks, each agent maintains a succinct representation of the events that took place at the previous step consisting for each other agent of the simple information "did the other agent behave the same as me?"

In the experiments we considered a population of 100 individuals. At each time step the fitness of all the agents is increased by $4n$ points, where n is the number of cooperators. Further, the fitness of each agent is decreased by 1 point for each attack involving the agent and is decreased by 3 points if the agent is cooperating. After 100 time intervals the population reproduces with binary tournament selection. The results of the experiments are shown in Figure 1. The starting point of each arrow corresponds to the initial frequencies of the two genes, while the ending point corresponds to the frequencies of the next generation. Each arrow is obtained averaging ten experiments with the same initial gene frequencies. Our experiments show that if the frequency of the vengeful gene is high enough, the population is pushed towards uniformity, and any gene that is widespread into the population (the vengeful gene itself, a cooperative gene, or an arbitrary gene whatsoever) gains stability.

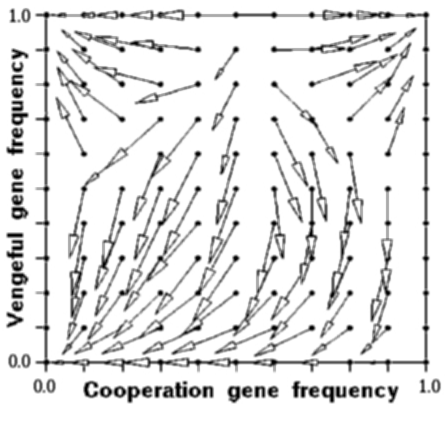

Figure 1: The results of the experiments.

A Game Based Approach to Co-evolutionary Optimization

Robert D. Rhyne, II
Tactical Electronic Warfare
Naval Research Laboratory
Washington, D.C., 20375-5000

Dr. James F. Smith, III
Tactical Electronic Warfare
Naval Research Laboratory
Washington, D.C., 20375-5000

A fuzzy logic based expert system, referred to as the resource manager (RM) has been developed that automatically allocates electronic attack (EA) resources in real-time over many dissimilar platforms. The initial version of the algorithm was optimized using a genetic algorithm (GA) employing fitness functions constructed based on expertise. To produce a RM with a more robust response, a new approach is being explored that involves embedding the RM in an electronic game environment. The game allows a human expert to play against the resource manager in a simulated battlespace. Each of the defending blue platforms is exclusively directed by the fuzzy RM and the attacking red platforms are controlled by a human expert (HVC mode) or operate autonomously under their own logic (CVC mode). The game automatically creates a database reflecting a domain expert's knowledge that is used with a GA for co-evolutionary re-optimization of the RM. The game has its own criteria as to when to re-optimize and when the process is complete. In biology co-evolution refers to the fact that a biological system never evolves alone, both system and environment simultaneously evolve. Within the game co-evolution refers to the simultaneous evolution of red and blue agents. CVC and HVC mode experiments have been conducted. HVC mode exhibits faster convergence than CVC mode. Preliminary results point to CVC mode as potentially providing a better optimization procedure. This is due to the tendency of human experts to fixate on particular strategies: a characteristic not shared by computerized opponents, due to the GA's ability to escape local maxima.

The game environment will allow a human EA expert to play competitively against the RM and allows the RM to adapt to the strategies employed by the human player, effectively allowing the RM to learn from its opponents. The human player will control at least 1 platform in competition with a group of individual blue agents each running their own copy of the RM. The number of degrees of freedom available to the player determines the level of control a human player exercises over a platform. More complicated optimization requires that the human player have access to additional degrees of freedom.

For a human expert to play against the RM in HVC mode it was necessary to write a significant amount of software allowing the expert to control a red platform, launch missiles and receive sensor input. A digital simulation of a radar plan position indicator (PPI) display similar to those used in real radar systems was created. The simulated PPI display is designed to imitate the properties of a real PPI display. In this way decisions made by the EA expert playing the game and subsequently automatically stored in the database for later data mining will reflect truth.

GA based co-evolutionary optimization can be used to produce a RM effective not only against the most recent generation of enemy agents; but also, all past generations. This is done by using a symbolically recursive fitness function. The first step in producing a symbolically recursive fitness function involves multiplying the fitness function used in the previous co-evolutionary generation for blue optimization by a product of Heaviside step functions. The argument of each Heaviside step function is the difference between the offending root concept membership function evaluated appropriately at each time step and a threshold. The resulting product fitness function is referred to as a symbolically recursive fitness function. The idea is that unless the GA optimization returns a root concept membership function which for this set of input data, exceeds the threshold, the symbolically recursive fitness function will return a value of zero. Using the symbolically recursive fitness function the GA can be used to ensure that the membership function value for a particular root concept will be above a certain threshold. This triggers an appropriate action by the RM the next time red exhibits the behavior that led to blue's loss and subsequent re-optimization.

In CVC mode, red is re-optimized using a symbolically recursive fitness function in a manner similar to blue. If red has sufficient *a priori* knowledge of blue's fuzzy membership functions and their related thresholds, then red can use this knowledge to determine trajectories that do not alert blue to his intention. This can be built into red's recursive fitness function, potentially making red an extremely formidable enemy who can make near optimal decisions at each time step accelerating the development of a more robust blue RM. The only limit on this process is red's knowledge of blue.

Evolving Cases for Case-Based Reasoning Multiagent Negotiations

Leen-Kiat Soh
Information and Telecommunication Technology Center
University of Kansas
Lawrence, KS 66044
lksoh@ittc.ukans.edu

Costas Tsatsoulis
Dept. of Electrical Eng. and Computer Science
University of Kansas
Lawrence, KS 66045
tsatsoul@ittc.ukans.edu

Mara Elizabeth Jones
ScriptPro
5828 Reeds Road
Mission, KS 66202
mjones@scriptpro.com

Arvin Agah
Dept. of Electrical Eng. and Computer Science
University of Kansas
Lawrence, KS 66045
agah@ukans.edu

Extended Abstract

The work reported in this paper applies genetic algorithms (GAs) to the automatic generation of cases for a case-based reasoning (CBR) system employed for multiagent negotiations. Our problem domain is in resource allocation and constraint satisfaction. In particular, our multiagent system has a set of agents working cooperatively to track multiple moving targets as accurately as possible. Each agent maintains its local information base and has a limited sensor capability (each sensor can only cover a small area of the environment and accurate target tracking requires at least three sensors). Thus, the agents controlling the sensors have to be communicative—to inform their neighbors of incoming targets and to ask them to perform certain tasks. Moreover, the agents may share the same CPU platform and thus need to re-allocate their usage of the limited CPU resources according to their tasks and their perceived environments. This also motivates the agents to share local constraints and cooperate.

We use a case-based argumentative negotiation model to facilitate cooperative behavior. An agent argues to persuade its neighbor to perform a requested task, and via that process, the agents know more about each other by exchanging locally maintained information. This distributed, bottom-up characteristic is the key of our problem domain. An agent uses case-based reasoning (CBR) to obtain a negotiation strategy for each negotiation that arises. Using the most similar retrieved case, an agent can quickly adapt its past experience to the current situation to derive a suitable negotiation strategy. Our CBR approach is designed to deal with the highly dynamic and time-critical agent environment. A case is defined as a semantically rich description of a negotiation previously experienced. Cases are considered to be either initiating or responding. Each agent has its own set of cases for both case types in its local case base. The reason for this distinction of case types is that agents react differently when they need to initiate a negotiation rather than when they need to respond to a negotiation.

One of the major issues with the described approach is the need for generating and populating the initial sets of case bases that each agent is equipped with prior to the negotiations. We need a set of initial cases to bootstrap our agent and CBR design including learning, negotiations, and tracking. Note also that since we are dealing with a dynamic and real-time environment, we do not expect all representative cases to be part of the initial case bases. Thus, we also equip our CBR module with a learning capability to learn more refined and useful cases as each agent progresses in its lifetime. This relaxes the requirement on the correctness of the initial cases. In order to generate these initial cases, genetic algorithms (GAs) are used.

This paper reports on the use of GAs to evolve the sets of cases for the CBR agent negotiations. A representation scheme is developed to properly represent each case as a chromosomal string to be used by the GA operations, such as crossover, mutation, etc. A fitness function is defined to evaluate each evolved case to be used by the GA. The outcome, i.e., the applicability of GAs to multiagent negotiation, is evaluated through the analysis and evaluation of the generated cases.

Each generated case maps a situation, or a collection of similar situations, to a negotiation strategy, i.e., solution. Therefore, it is very important for the generated cases to cover a wide range of possible situations. In addition, a good case base will cover each possible situation with a wide range of possible situations. The GA results were evaluated using clustering algorithms in order to determine (1) whether the evolved cases provide a good coverage of the clusters of possible situations, and (2) whether the evolved cases, for each situation cluster, provide a good coverage of possible solutions.

We have found that, based on experimental results, the application of genetic algorithms is an effective technique for automatic generation of cases in our problem domain. Another work in progress is the use of GAs for continuous updating of the cases in the CBR module of the agents and for assisting in case-based learning as each agent adapts itself to its dynamic environment.