Modeling Adaptive Multi-Agent Systems Inspired by Developmental Biology

Samuel Landau and Sébastien Picault

LIP6, Université Pierre et Marie Curie
case 169, 4 place Jussieu, 75 252 Paris CEDEX 05, FRANCE
{Samuel.Landau,Sebastien.Picault}@lip6.fr
http://www-poleia.lip6.fr/~{landau,picault}

Abstract. This paper addresses the issue of adaptive multi-agent systems and their design based on living systems features such as phylogeny and ontogeny. We argue that the evolutionary design of agents behaviors implies several specific features that are missing in classical evolutionary approaches. Therefore we propose a new approach that would be more adequate to MAS, and present a model for building MAS as the result of evolving, interacting, self-organizing agents. We finally mention a use of such an approach for the embodiment of robots.

Keywords: adaptive multi-agent architecture, agent behavior evolution, development, body image

1 Introduction

Adaptability is a property of living organisms that not only relies on learning, but also on the ability to reorganize themselves, for instance to reshape parts of their body or of their cognitive structures, to give adequate responses to environmental changes. In addition, living organisms have a body image (a representation of their physical body) that is built by and during their development. This body image is the result of a coupling process between the growth of the physical body in an unpredictable and highly dynamical environment and the reorganization of cognitive structures.
We would like to investigate the problem of designing individuals that can exhibit similar properties: their development can indeed be modeled with an ecosystem (actually a MAS). This especially implies an evolutive approach, either to have such individuals gradually evolve (phylogeny) or to build them through an ontogenetic process (the whole system being organized by a selection of available agents behaviors).
In that context, the key problem is the ability to really have *agents behaviors* evolve, rather than just parameters in prior designed behaviors. To our point of view, existing evolutionary approaches, which aim at *solving problems*, are not completely adequate for that purpose. We will discuss in the next section the limitations of classical evolutionary approaches and present the requirements of a model allowing the evolution of agents behaviors. Section 3 will then describe

such a model, *ATNoSFERES*, with which we are currently working. Finally, we will deal with the issue of using this approach for embodiment in robots.

2 Towards *Ethogenetics*

The general principle of evolutionary algorithms involves the following steps:

1. "individuals" representing potential solutions for the problem are built *from a hereditary substratum* (the *genotype*)
2. these individuals belong to a population
3. their adequation to the problem is evaluated (either explicitly through a "fitness" function, or implicitly through survival criteria for instance)
4. some of them (mainly the most adapted) are selected to produce offspring (by mixing their genotypes) ; others are removed from the population
5. the process cycles with the new population.

We would like to adapt this general paradigm to the evolution of developing multi-agent systems in place of individuals, in order to get the adaptive properties mentioned in the introduction. The reorganization of the MAS relies on a collective behavior resulting from the behaviors exhibited by each agent in the system. In addition, it can be useful (for instance to bootstrap the system with prior knowledge, or to specify a behavioral constraint) to keep the possibility to explicitly design parts of the system. Thus we assume that a convenient evolutionary algorithm should have the following properties:

- There should be a quite good continuity between the genotype and the behaviors built from it: small variations in the genotype should induce in general only slight variations in the resulting behaviors, or else the evolutionary process gets closer to a pure random search.
- There should be a large expressive power in the behaviors built from the genotype. The behavior of an agent is actually not a succession of independent actions, but a consistent structured sequence of actions.
- The behavior design process should be incremental, i.e. adding a selective constraint on the system (e.g. a new collective goal to achieve) should not force the algorithm to restart from scratch, but reuse the existing behaviors and slightly modify them.

Unfortunately, the existing evolutionary computing paradigms are inadequate for this purpose:

- on the one hand, Genetic Algorithms [1–3] and Evolutionary Strategies [4, 5] have a very poor expressive power, since their purpose is the optimization of a set of parameters [6] in behaviors which *have to be given a priori*. However, they allow fine-grain encoding, so that small variations in the genotype generally induce small variations in the phenotype (in our case, the resulting behavior).

– on the other hand, the Genetic Programming paradigm [7], which is based on the evolution of *programs* (i.e. instruction trees), has a much higher expressive power. But in such a tree structure, genetic variations most of the time have a strong impact on behaviors (not only parameters, but also instructions are subject to modifications), all of the more since the impact of variations tightly depends on the location in the tree hierarchy. Additionally, the issue of incremental design of behaviors with instruction trees is still a difficulty [8].

Therefore, we propose a specific evolutionary approach, that we call *Ethogenetics*, the purpose of which is to provide general principles to the design of evolutive agent behaviors – the ability to *build* agent behaviors (with a large expressive power) from a meaningless genetic substratum. In addition to the properties required for this purpose, the Ethogenetics approach advocates the following features:

– *The structure describing the behaviors should be understandable.* It may be useful to provide the agents with *explicit understandable behaviors* : some control architectures such as artificial neural networks might be very efficient, but the resulting behavior cannot be clearly described. The ability to easily interpret the behaviors would allow on the one hand to understand *what* has been selected, and on the other hand, to explicitly specify some of the behaviors using this same structure, allowing to set *a priori* the behaviors of some agents.
– *Behaviors should be situated.* Since the system will have to operate in a given environment, it should be able to adapt itself, to reconfigure according to environmental constraints. Thus the semantic structure representing behaviors should avoid using explicit parameters : parameters are a kind of shortcut, they reflect prior knowledge about the environment. The building of *situated behaviors* has to be independent from any parameters, in order to keep more flexibility.

In the next section, we present ATNoSFERES[1], a model aimed at implementing those properties.

3 Description of the ATNoSFERES Model

3.1 General Principles

The ATNoSFERES model [9, 10] is a specific implementation of the Ethogenetics principles. It uses the SFERES framework [11] as a tool for modelling the agents classes, integrating those classes to the system, designing an environmental simulator and providing classical evolutionary techniques.

In particular, ATNoSFERES provides a general class, the `ATNAgent`, the behavior of which is produced from a bitstring (the genetic substratum) through the following steps:

[1] see http://miriad.lip6.fr/atnosferes

1. a translator produces *tokens* from the string,
2. an interpreter uses these tokens as instructions to *build a graph* (an ATN[2]),
3. finally, the graph is expressed to produce the behavior of the agent.

The translator and the interpreter themselves are agents; in the following lines, we will consider that their behavior is given, but it could evolve as well to provide the system with higher autonomy.

3.2 The ATN Graph and the `ATNAgent`

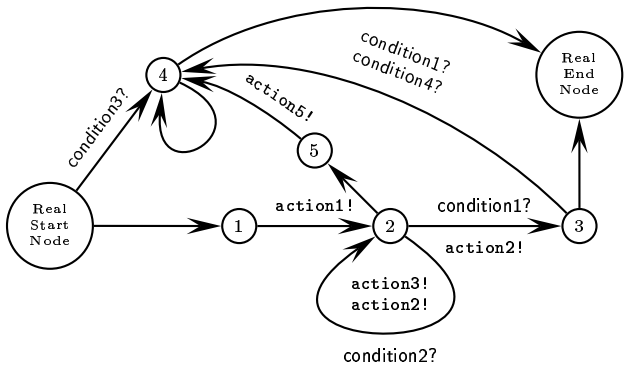

Fig. 1. An example of ATN.

The `ATNAgent` class is intended to behave according to an ATN graph [12]. ATN have previously been used by Z. Guessoum [13] for designing agent behaviors. Each subclass of `ATNAgent` is associated with two collections of tokens: condition ones and action ones. The actions are behavioral "primitives" that can be performed by the agent, the conditions are perceptions or stimuli that induce action selection. The edges of the graph are labeled with a set of conditions and a sequence of actions (see figure 1).
The ATN built by adding nodes and edges to a basic structure containing two nodes: a "Real Start Node" and a "Real End Node". At each time step, the agent (initially in the "Real Start Node" state) randomly chooses an edge among those having either *no condition* in their label, or *all conditions simultaneously true*. It performs the actions associated with this edge and jumps to the destination node. It stops working when its state is the "Real End Node".

3.3 The Interpreter

The purpose of the interpreter is to build an ATN from tokens. Some of these tokens will be action or condition ones that are used to label edges between nodes in the ATN. The other ones are interpreted as instructions, either to create nodes

[2] ATN stands for Augmented Transition Network

or connect them, or to manipulate the structure under construction.
As we mentioned above, the structure built by the tokens sequence has to be robust towards variations in the genotype. For instance, the replacement of one token by another, or its deletion, should have only a *local impact*, rather than transforming the whole graph. Therefore, we use a "top-level" programming language operating on a list (see table 1). If an instruction cannot execute

token	(initial list state) $\longrightarrow$	(resulting list)
condition?	$(x \ldots)$ $\longrightarrow$	(condition? x ...)
action!	$(x \ldots)$ $\longrightarrow$	(action! x ...)
node	$(x \ldots)$ $\longrightarrow$	$(N_i\ x \ldots)^a$
startNode	$(x \ldots)$ $\longrightarrow$	$(N_i\ x \ldots)^b$
endNode	$(x \ldots)$ $\longrightarrow$	$(N_i\ x \ldots)^c$
connect (c2? x N_i y c1? z a2! a1! t N_j u ...)	$\longrightarrow$	$(x\ N_i\ y\ z\ t\ N_j\ u \ldots)^d$
dup	$(x\ y \ldots)$ $\longrightarrow$	$(x\ x\ y \ldots)$
dupObject	$(x\ y\ N_i\ z \ldots)$ $\longrightarrow$	$(N_i\ x\ y\ N_i\ z \ldots)$
popRoll	$(x\ y \ldots z)$ $\longrightarrow$	$(y \ldots z\ x)$
pushRoll	$(x \ldots y\ z)$ $\longrightarrow$	$(z\ x \ldots y)$
swap	$(x\ y \ldots)$ $\longrightarrow$	$(y\ x \ldots)$
forget	$(x\ y \ldots) + [z \ldots]$ $\longrightarrow$	$(y \ldots) + [x\ z \ldots]^e$
recall	$(x \ldots) + [y\ z \ldots]$ $\longrightarrow$	$(y\ x \ldots) + [z \ldots]^e$

[a] creates a node N_i
[b] creates a node N_i and connects "RealStartNode" to it
[c] creates a node N_i and connects it to "RealEndNode"
[d] creates an edge between N_j and N_i, with (c1?& c2?) as condition label and the list {a1!,a2!} as action label
[e] with an auxiliary stack

Table 1. The ATN-building language.

successfully, it is simply ignored, except instructions operating on nodes (i.e. *connect* and *dupObject*) which are "pushed" in the list until new nodes are produced; then they try to execute again with the new data. Finally, when the interpreter does not receive tokens any more, it terminates the ATN: actions and conditions tokens still present between nodes are treated as *implicit connections* (so that new edges are created) and the consistency of the ATN is checked ("Real Start Node" is linked to nodes having no incoming edges, except from themselves; in the same way, nodes having no outgoing edges are linked to "Real End Node").

3.4 The Translator

The translator has a very simple behavior. It reads the genotype (a string of bits) and decodes it into a sequence of tokens. It uses a *genetic code*, i.e. a function

$$\mathcal{G} : \{0,1\}^n \longrightarrow \mathcal{T} \qquad (|\mathcal{T}| \leq 2^n)$$

where $\mathcal{T}$ is a set of tokens, which includes both action and condition ones (specific to the agent to build) and those understood by the interpreter (see table 1). Depending on the number of tokens available, the genetic code might be more or less redundant. If necessary, it can be designed in order to resist mutations, but we will not discuss this issue in this paper.

4 Features of the ATNoSFERES Model

4.1 Evolutionary Computation Considerations

As an evolutive approach, the ATNoSFERES model provides three main features. First, it separates the genetic information structure (plain bit string, the lexical structure) from its interpretation (ATN, the semantic structure). Thus, thanks to the interpreter language, the semantic structure that is built is *always correct*. The behavior described by the ATN always has a meaning – even if it is not adequate.
The second main evolutive feature is related to the genetic operators. The level of influence of the classical genetic operators – mutation and crossover – does not depend on the parts of the bit string they involve (neither on their location in the bit string nor on their size). This is also a main advantage over many evolutive approaches. As a matter of fact, mutations only have a local impact in the expression of the genetic information, and crossovers involve bit substrings which carry locally functional genetic code. We might also consider more exotic genetic operators, such as deletions/insertions in the bit string. These operators in particular permit to smoothly manage string resizing, since they only have a local impact in the ATNoSFERES model.
The third feature is that the model does not use any parameter to build behaviors. The behaviors execution only depends on environmental conditions, thus hard-coded genetic parameters are not even needed. Apart from behaviors design, parameters encoding is a problem in many evolutive approaches, (see for example discussion on epistasis in [14]), but as long as building behaviors is concerned, we think it should be considered not to rely on fixed parameters in order to produce situated, adaptive behaviors.

4.2 MAS Design Considerations

As a model for designing multi-agent systems, the ATNoSFERES model does not set any restriction neither on the agent level specification nor on the choice of the agents. The granularity of the system modelisation is free ; furthermore, agents can be introduced later on at a lower organization level (for instance inside an agent), keeping the latter structure, if a finer-grain agent specification is needed. If the designer has prior knowledge about the system structure, he can specify and fix some agents behaviors, and use them as a constraint to drive the evolution of the system organization. On the other hand, the only specification that must be given for the evolving agents of the system is their sets of actions and

perceptions, and consequently the micro-environment in which they operate. Not only can this micro-environment be a part of the system environment, but it can also for instance be the inside of an upper-level agent.

4.3 Agents Behaviors Design Considerations

As a model for automatic behavior design (as part of ATNoSFERES), ATNs use a simplified framework, where only the conditions and actions of the agents have to be specified. With an ATN as the structure for behavior description, it is possible to directly describe the behavior of any agent : this is an interesting perspective for behavior explanation.

4.4 The ATNoSFERES Model with regard to Ethogenetics

The ATNoSFERES model fulfills the Ethogenetics requirements expressed in section 2. Preliminary experiments [10] have validated the use of ATNoSFERES regarding the following aspects:

- the ability to evolve *adequate* agents behaviors in a simple situation, from *random graphs*;
- the consistency of the ATN-building evolutionary language.

The experimental results have also confirmed that the generation of behaviors do not rely on a precise structure in the genotype: various adequate solutions have been found, based either on different graph-building strategies, or on the use of properties of the graphs (more details can be found in [10], and genetic related issues have been discussed in [15]).

5 ATNoSFERES and Embodiment for Robots

ATNoSFERES should allow to incrementally build multi-agent systems able to reorganize and develop, by evolving agent behaviors that can be previously partly specified. The system and the agents have to face internal and external constraints that induce selective pressure. Throughout evolution, the selective process shapes the organization of the system and its collective behavior, including the capacity to reorganize.
The reorganization and adaptation take place during the system lifetime and while it is acting. These processes reflects the movement of the system towards adequation to the changing environment – be it inner environment, that is the environment of the agents inside the system, or outer environment, that is the "real-world" environment in which the system is immersed. When they take place in a multi-agent system that is put into a robot that the system must control and that shapes the system, we think that these reorganization and adaptive processes are a distributed representation of the robot body. This is a weaker distributed conception of embodiment for robots than the one that can be found of others works involving, in particular, a body consciousness [16].

6 Conclusions and Perspectives

We have presented Ethogenetics, an approach for evolving adaptive multi-agent systems, and discussed its specific features. To summarize, interesting agents behaviors can be built through an evolutionary approach that is able to ensure *continuity* between the genetic substratum and the phenotypic behavior, and a *high expressive power* in the behavior produced. We propose therefore a two-step building that leads to graph-based behaviors (the ATNoSFERES model).
We assume that a multi-agent system, whose agents behaviors have been evolved that way, exhibit organizational features as a consequence of the selection pressure that shaped individual behaviors. When implemented on a robot, the organization of the system is able to constitute a representation of the robotic body. In order to more precisely investigate the organizational features of the ATNoSFERES model, further experiments regarding collective strategies (especially predator-prey simulations) are currently ongoing. The embodiment issues will be studied through real-robots applications, as part of the MICRobES Project[3] [17,18] will be led in the following months.

References

1. J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* Ann Arbor: University of Michigan Press, 1975.
2. K. A. De Jong, *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, Dept. of Computer and Communication Sciences, University of Michigan, 1975.
3. D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley Pub. Co., 1989.
4. I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Stuttgart: Frommann–Holzboog, 1973.
5. H.-P. Schwefel, *Evolutionsstrategie und numerische Optimierung.* Dr.-Ing. Thesis, Technical University of Berlin, Department of Process Engineering, 1975.
6. T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.
7. J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, Massachusetts: MIT Press, 1992.
8. S. Perkins, *Incremental Acquisition of Complex Visual Behaviour using Genetic Programming and Shaping.* PhD thesis, University of Edinburgh, 1998.
9. S. Picault and S. Landau, "Ethogenetics - A Darwinian Approach towards Individual and Collective Agents Behavior," technical report (in press), LIP6, Paris, 2001.
10. S. Landau, S. Picault, and A. Drogoul, "ATNoSFERES : a Model for Evolutive Agent Behaviors," in *Proc. of AISB'01 symposium on Adaptive Agents and Multi-agent systems*, 2001.
11. S. Landau, S. Doncieux, A. Drogoul, and J.-A. Meyer, "SFERES, a Framework for Designing Adaptive Multi-Agent Systems," technical report, LIP6, Paris, 2001.

[3] see http://miriad.lip6.fr/microbes

12. W. A. Woods, "Transition networks grammars for natural language analysis," *Communications of the Association for the Computational Machinery*, vol. 13, no. 10, pp. 591–606, 1970.
13. Z. Guessoum, *Un environnement opérationnel de conception et de réalisation de systèmes multi-agents.* PhD thesis, LIP6 – Université Paris VI, may 1996.
14. R. Salomon, "Increasing Adaptivity through Evolution Strategies," in *From animals to animats 4, Proc. of ther fourth international conference on simulation of adaptive behaviour* (P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, eds.), pp. 411–420, MIT Press, 1996.
15. S. Picault and S. Landau, "Ethogenetics and the Evolutionary Design of Agent Behaviors," in *Proc. of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2001)*, 2001.
16. A. Cardon, "The approaches of the Concept of Embodiment for an Autonomous Robot," in *Proc. of the Adaptive and Embodiment using Multi-Agent Systems workshop (AEMAS) of ACAI'01*, 2001.
17. S. Picault and A. Drogoul, "The MICRobES Project, an Experimental Approach towards "Open Collective Robotics"," *Proceedings of DARS'2000*, 2000.
18. S. Picault and A. Drogoul, "Robots as a Social Species: the MICRobES Project," *Proceedings of SIA'2000, AAAI Fall Symposium Series*, 2000.