Ethogenetics: an Evolutionary Approach to Agents Organization

Sébastien Picault and Samuel Landau

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

Abstract

In this paper, we propose an evolutionary point of view on organization in multi-agent systems, in which the MAS is seen as an ecosystem. This indeed addresses the issue of the design of evolvable agent behaviors. We introduce therefore, through the concept of *Ethogenetics*, some important design principles that are not fulfilled by classical evolutionary approaches. Then, we describe a framework, ATNoSFERES, that implements this concept using a specific agent behavior model; we discuss its properties and finally propose further extensions.

Keywords

emergent organizations, agent behavior, complex systems, genetic encoding

1 Introduction

In some multi-agent systems (MAS), organization might be an explicit finality, for instance in the simulation of social behavior [15, 14, 5, 8]. But in other cases, the relevance of explicitly designing organizations in MAS is much more controversial. It is indeed very difficult (maybe utopian) to design generic organizational methodologies, since organizational roles in a system are tightly dependent on the function, the nature and the number of agents [3]. In addition to this, there is no indication that such a generic organization would produce adequate behaviors in a given situation.

To our point of view, in some cases, especially in real-world applications or massive systems, the explicit design of an organization is a strong bias between the collective behavior of the system and its adequation to a given environment. It is all the more the case since the system might have to reorganize itself in order to maintain its adaptation towards the collective task or an environment. In such a situation the organization cannot be addressed as a preexistent feature of the system, but rather as a permanent process [2]. Thus, we prefer to consider organization as an emergent property, that reflects *the equilibrium in an ecosystem* (coordination, cooperation between behaviors, competition for resources, ...).

According to that perspective, the natural selection paradigm might provide a convenient approach, provided that agent behaviors are able to *evolve*. 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.

The approach we propose consists in adapting this general paradigm to the evolution of multi-agent systems in place of individuals. A MAS exhibits indeed a collective behavior that is usually considered as the solution to a given problem.
Unfortunately, the existing evolutionary computing paradigms are inadequate for this purpose:

- on the one hand, Genetic Algorithms [9, 4, 6] and Evolutionary Strategies [18, 20] have a very poor expressive power, since their purpose is the optimization of a set of parameters [1] in behaviors which *have to be given a priori*. However, they allow fine-grain encoding, so that small variation in the *genotype* (the genetic substratum) generally induce small variations in the *phenotype* (the resulting behavior).

- on the other hand, the Genetic Programming paradigm [10], 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 modification), all of the more since the impact of variations tightly depends on the location in the tree hierarchy.

Therefore, we present in the next section a different approach, called "Ethogenetics". We will then describe in section 3 a first implementation, ATNoSFERES. Section 4 will discuss the properties of ATNoSFERES, and we finally conclude and outline some perspectives.

2 Ethogenetics

The purpose of "Ethogenetics" is to provide general principles for the design of evolutive agent behaviors – the ability to *build* agent behaviors (with a large expressive power) from a meaningless genetic substratum. Since Darwinian evolution is a blind process, its use to produce collective behaviors in MAS implies several properties, mainly consequences of two principles: continuity and expressive behavioral power.

2.1 Continuity

The environmental selection pressure acts on the whole system, on its ability to react, to perform a task, to reach collective goals and so on. Adapted individuals are selected to produce "offspring": other individuals, the genotype of which is a mixture of those of their "parents". The adaptation degree of the offspring systems should be close to their parents ones (if not, the adaptive effect of natural selection gets lost). Thus, the behavior building process has to be:

- *robust towards mutations*: small variations in the genotype should induce in most cases only small variations in the phenotype;

- *independent from the structure of the genetic substratum*: unlike Genetic Programming (where the hierarchical tree structure has heavy consequences), distant parts of the genotype should have few effects on each other. Thus it is useful to dissociate the semantic structure (that produces the behavior) from the "syntactic" one (the genetic substratum). When this requirement is not fulfilled, semantics tightly constrains syntax, so that syntactic manipulations (resulting from "blind" genetic operations) often destroys the semantic structure.

2.2 Expressive behavioral power

The second major requirement in order to produce *agent behaviors* is the ability to design *complex* behaviors. Thus the semantic structure used for that goal should have at least the expressive power of a program tree (a tree provides more interesting features as a semantic structure rather than as a syntactic structure). But these behaviors, even if complex, should meet some requirements:

- *Behaviors should be understandable*. It may be useful to provide the agents with *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 able to adapt*. 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, a framework aimed at implementing a concrete agent model featuring the properties required by Ethogenetics.

3 Description of the ATNoSFERES model

3.1 General principles

The principles of Ethogenetics are currently implemented in a multi-agent framework, ATNoSFERES [16, 13, 12]. It is part of the SFERES framework [11] which provides tools for modelling the agents classes, integrating them to the system, designing an environmental simulator and providing classical evolutionary techniques.

The main feature of the ATNoSFERES model consists in using an ATN[1] [21]. ATN have already been used for designing agent behaviors [**?**, 7]. In an ATN, edges linking the states or nodes can be labelled with a set of *conditions* and a sequence of *actions*. Thus it is particularly adequate to describe the behavior of an agent.
ATNoSFERES provides a general class, the `ATNAgent`, which is intended to behave according to an ATN graph. 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. Those action and condition tokens are used to label the edges of the graph (see figure 1).

A structure such as an ATN graph would not fulfil the *continuity* requirement if it had to evolve through a blind, darwinian, process. Thus, it has to be *built* from a finer-grain substratum, e.g. a bitstring. Therefore, we use the following steps:

1. The population (initial population or offspring of adapted individuals), is composed of agents having their own bitstring (the genotype or hereditary substratum).
2. For each agent:
   (a) a *translator* produces *tokens* from the bitstring,
   (b) an *interpreter* uses these tokens as instructions and data to *build a graph* (the ATN),
   (c) finally, the graph is used as a state machine to produce the behavior of the agent.
3. The agents behave in their environment, according to their own ATN.
4. The agents are selected and reproduced according to their capabilities in surviving in their environment or performing a specified task. They produce offspring and the process cycles with the new population.

The translator and the interpreter themselves are agents; in the following lines, we will consider that their behavior is given and will not change in time, but it could evolve as well to provide the system with higher autonomy.
We will now detail the above steps, starting with the behavior of the agent and going back to the translation process.

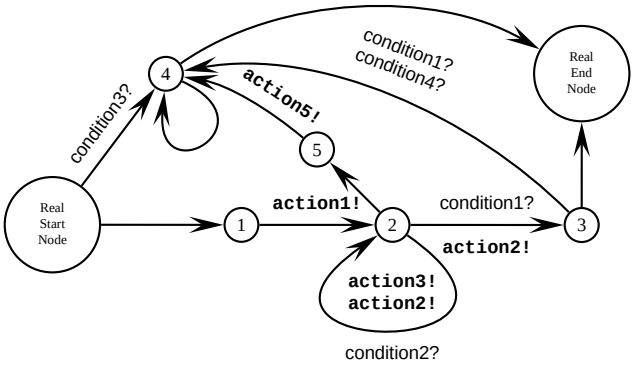

Figure 1: An example of ATN.

3.2 The ATN Graph and the `ATNAgent`

The ATN is built by an interpreter (see § 3.3) by adding nodes and edges to a basic structure containing two nodes: a "Real Start Node" and a "Real End Node". Once the ATN has been built, it can be used *as an automaton* to produce the behavior of the agent during its life cycle. 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. At the following time step, the process iterates from the new state. The agent stops working when its state is the "Real End Node".

We emphasize that this approach is different from classical evolutionary methods for producing adaptive behavior (Genetic Programming for instance) since in the latter, the structure used to determine the behavior is used *as a whole* at each timestep (for instance, the tree is executed again at each time step to select actions).

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 or connect them, or to manipulate the structure under construction.
As we mentioned in section 1, the structure built by the tokens sequence has to be robust towards mutations. 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).[2]

If an instruction cannot execute 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

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

[1] ATN stands for "Augmented Transition Network".
[2] More details on the very properties of this stack-based ATN building will be provided in [**?**].

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

[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. There are three categories of tokens: 1. conditions and actions (specific to an agent class), 2. node creation and connection, 3. stack manipulation (2 and 3 are understood by the interpreter as instructions, 1 as data).

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 bitstring they involve (neither on their location in the bitstring 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 bitstring. 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 [19]), 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.
In order to cope with these different levels of specification, we are now introducing a `CompositeAgent` in the framework, in order to allow encapsulation of agents at one level by other agents at a higher level.

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, the ATN structure used in ATNoSFERES provides a simplified tool, since only the conditions and actions of each agent class have to be specified.

The ATN structure for behavior production allows to directly describe the behavior of any agent: this is an interesting perspective for explaining *how* the behavior operates or *why* it has been selected, for bootstraping the system, for re-using parts of existing behaviors, for applying "high-level" operations such as learning techniques, etc.

4.4 The ATNoSFERES model with regard to Ethogenetics

The ATNoSFERES model fulfills the Ethogenetics requirements expressed in section 2. Preliminary experiments [13, 17] 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 [13], and genetic related issues are discussed in [17]).

5 Conclusions and perspectives

We have presented Ethogenetics, an approach for the design evolutive agents behaviors, 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).

This model has been tested on simple agents behaviors [13, 17]; we are currently experimenting collective strategies (especially predator-prey simulations and real-robots applications[3]).

We assume that a multi-agent system, in which agents behaviors have been evolved that way, exhibit organizational features as a consequence of the selection pressure that shaped individual behaviors.

[3] As part of the MICRobES Project: see http://miriad.lip6.fr/microbes

In further works, we plan to use resources so as to regulate the agents behaviors. These *metabolites* will be associated with the actions within the ATN, and will act as environmental constraints, allowing or disabling some edge transitions at a given time.

References

[1] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[2] Alain Cardon. *Conscience artificielle & systèmes adaptatifs*. Eyrolles, Paris, 1999.

[3] Alain Cardon. Étude de la conception et du comportement d'une organisation d'agents massive (in prep.). 2001.

[4] Kenneth 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.

[5] Alexis Drogoul and Jacques Ferber. Multi-Agent Simulation as a Tool for Modeling Societies: Application to Social Differentiation in Ant Colonies. *Proceedings of the MAAMAW Workshop*, 1992.

[6] David Edward Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Pub. Co., 1989.

[7] Zahia Guessoum. *Un environnement opérationnel de conception et de réalisation de systèmes multi-agents*. PhD thesis, LIP6 – Universit´e Paris VI, may 1996.

[8] P. Hogeweg and B. Hesper. The Ontogeny of the Interaction Structure in Bumble Bee Colonies: a MIRROR Model. *Behavioural Ecology and Sociobiology*, (12):271–283, 1983.

[9] John Henry 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.

[10] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, Massachusetts, 1992.

[11] Samuel Landau, St´ephane Doncieux, Alexis Drogoul, and Jean-Arcady Meyer. SFERES, a Framework for Designing Adaptive Multi-Agent Systems. Technical report, LIP6, Paris, 2001.

[12] Samuel Landau and S´ebastien Picault. Modeling Adaptive Multi-Agent Systems Inspired by Developmental Biology. In *Proc. of the Workshop on Adaptability and Embodiment using Multi-Agent Systems (AEMAS 2001)*, pages 238–246, june 2001.

[13] Samuel Landau, S´ebastien Picault, and Alexis Drogoul. ATNoSFERES: a Model for Evolutive Agent Behaviors. In *Proc. of AISB'01 symposium on Adaptive Agents and Multi-agent systems*, pages 95–99, 2001.

[14] S´ebastien Picault. A Multi-Agent Simulation of Primate Social Concepts. *Proceedings of ECAI'98 (13th European Conference on Artificial Intelligence)*, 1998.

[15] S´ebastien Picault and Anne Collinot. Designing Social Cognition Models for Multi-Agent Systems through Simulating Primate Societies. *Proceedings of ICMAS'98 (3rd International Conference on Multi-Agent Systems)*, 1998.

[16] S´ebastien Picault and Samuel Landau. Ethogenetics - A Darwinian Approach towards Individual and Collective Agents Behavior. Technical report (in press), LIP6, Paris, 2001.

[17] S´ebastien Picault and Samuel Landau. Ethogenetics and the Evolutionary Design of Agent Behaviors. In Nagib Callaos, Susanna Esquivel, and Jamika Burge, editors, *Proc. of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2001)*, volume 3, pages 528–533, 2001.

[18] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzi pien der biologischen Evolution*. Frommann–Holzboog, Stuttgart, 1973.

[19] Ralf Salomon. Increasing Adaptivity through Evolution Strategies. In Pattie Maes, Maja J. Mataric, Jean-Arcady Meyer, Jordan Pollack, and Stewart W. Wilson, editors, *From animals to animats 4, Proc. of the fourth international conference on simulation of adaptive behaviour*, pages 411–420. MIT Press, 1996.

[20] Hans-Paul Schwefel. *Evolutionsstrategie und numerische Optimierung*. Dr.-Ing. Thesis, Technical University of Berlin, Department of Process Engineering, 1975.

[21] William A. Woods. Transition networks grammars for natural language analysis. *Communications of the Association for the Computational Machinery*, 13(10):591–606, 1970.