A Hybrid and bio-inspired Architecture approach for self-configuring behaviours in Cognitive Agents

Student: Oscar Javier Romero López
Tutor: Angélica de Antonio
Languages, Information Systems and Software Engineering Department
Politécnica de Madrid University
Boadilla del Monte 28660
Madrid – Spain
OJRLOPEZ@HOTMAIL.COM

Abstract
In this work, an hybrid, self-configurable, multilayered and evolutionary subsumption architecture for cognitive agents is developed. Each layer of the multilayered architecture is modeled by one different Machine Learning System (MLS) based on bio-inspired techniques such as Extended Classifier Systems (XCS), Artificial Immune Systems (AIS), Neuro Connectionist Q-Learning (NQL) and Learning Classifier Systems (LCS) among others. In this research an evolutionary mechanism based on Gene Expression Programming (GEP) to self-configure the behaviour arbitration between layers is suggested. In addition, a co-evolutionary mechanism to evolve behaviours in an independent and parallel fashion is used. The proposed approach was tested in an animat environment using a multi-agent platform and it exhibited several learning capabilities and emergent properties for self-configuring internal agent's architecture.

Keywords: Bio-inspired Machine Learning, Gene Expression Programming, Hybrid Behaviour Co-evolution, Subsumption Architecture.

1 Introduction

In the last decades, Cognitive Architectures have been an area of study that collects disciplines as artificial intelligence, human cognition, psychology and more, to determine necessary, sufficient and optimal distribution of resources for the development of agents exhibiting emergent intelligence. One of the most referenced is the Subsumption Architecture proposed by Brooks [1].

According to Brooks [1], the Subsumption Architecture is built in layers. Each layer gives the system a set of pre-wired behaviours, where the higher levels build upon the lower levels to create more complex behaviours: The behaviour of the system as a whole is the result of many interacting simple behaviours. Another characteristic is its lack of a world model, which means that its responses are always and only reflexive as proposed by Brooks.

However, Subsumption Architecture results in a tight coupling of perception and action, producing high reactivity, poor adaptability and learning of new environments, no internal representation and the need of all patterns of behaviours are pre-wired.

Several extensions have attempted to add representation and behaviour arbitration to Subsumption like Behavior-Based Control Architecture [2] and Hormonal Activation Systems [3], but pre-wired behaviours and non-learning characteristics still remain becoming the architecture applicable and useful only for a specific pre-configured environments.

The present research focuses on developing an Hybrid Multilayered Architecture for Cognitive Agents based on Subsumption theory. Additionally this work proposes an Evolutionary Model which allows the Agent to self-configure and evolve its processing layers through the definition of inhibitory and suppressor processes, behaviours and number of layers. That means each agent

©2007 Oscar J. Romero

instead of having a pre-configured structure of layers and processes it will have an Artificial Evolutionary Process which is responsible for defining the multilayered structure. On the other hand, instead of using an Augmented Finite Machine System as Subsumption theory states [1] where no internal representation is done, in this paper we propose that each behaviour layer is driven by a different bio-inspired machine learning system (chosen from a repertoire where behaviour co-evolution occurs) which learns from the environment and generates an internal world-model by means of an unsupervised and reinforced learning.

The remainder of the paper is organized as follows. Section 2 presents a brief description of some fundamental concepts used in this work. The evolutionary approach for self-configurable cognitive agents is detailed in Section 3. Section 4 outlines and discusses the experimental results and emergent properties obtained. Finally concluding remarks are shown in Section 5.

2 Preliminaries

For the development of the proposed cognitive architecture, some fundamentals about evolutionary theory and some techniques of biologically inspired computational intelligence which are the basis of this work are summarized in this section. First, the multilayered processing theory is described, which the proposed architecture is based on, next several machine learning systems used in each layer of the hybrid architecture are introduced. Finally, a Gene Expression Programming mechanism for evolving and self-configuring the internal subsumption architecture of each agent is described.

2.1 Multilayered Processing

In 1986, R.A. Brooks [1] proposed a Multilayered structure for robotics that decomposes the problem into a set of asynchronous task achieving behaviours that he called Subsumption Architecture. According to Brooks [1], the task achieving behaviours locally and asynchronously operated are only loosely coupled to one another. In contrast to the conventional approach, each of the task achieving behaviours is typically in direct communication with the world (and each other) [4].

Brooks stated in [1] that in the subsumption architecture, various subsets of the task achieving behaviours typically exhibit some partial competence in solving a simpler version of the overall problem. Thus, the solution for more complex version of a problem can potentially be built up by incrementally adding new independent acting behaviours to existing ones. Potential conflicts among behavioral actions are resolved by a hierarchical arrangement of suppressor nodes.

On the other hand, Brooks [1] proposes that the layers of the Subsumption Architecture are composed of networks of augmented finite state machines (AFSM) with timers. Each AFSM has an input and output signal (Figure. 1). When the input of an AFSM exceeds a predetermined threshold, the behaviour of that AFSM is activated. The inputs of AFSMs come from sensors and outputs are sent to the agent's actuators. Each AFSM also accepts a suppression signal and an inhibition signal and these signals allow behaviours to override each other so that the system can produce coherent behaviour.

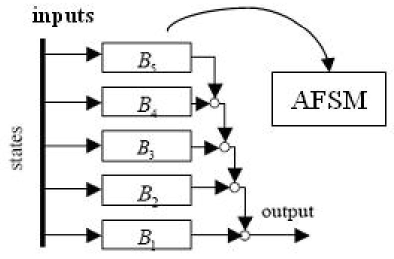

Figure 1. A typical parallel Subsumption Architecture

In this work, instead of using AFSM, an evolutionary Subsumption Architecture that uses machine learning and biologically inspired techniques in each layer to create internal representations of the world and learn from environment in an autonomous fashion, is developed.

2.2 Extended Classifier Systems XCS

According to Wilson [5], a classifier system is a machine learning system that seeks to gain reinforcement from its environment based on an evolving set of condition-action rules called classifiers. Via a Darwinian process, classifiers useful in gaining reinforcement are selected and propagate over those less useful, leading to increasing system performance. The classifier system idea is due to Holland [6], who laid out a framework that included generalization of classifier conditions, internal message-passing and reinforcement, and computational completeness. However, despite considerable research, the performance of the traditional system has been mixed, and there have been few advances on the initial theory.

In [5] Wilson et. al. proposed an Extended Classifier System (XCS). XCS is a recently developed learning classifier system (LCS) that differs in several ways from traditional LCSs. Wilson stated [5] that in XCS, classifier fitness is based on the accuracy of the classifier's payoff prediction instead of prediction itself. Second, the Genetic Algorithm (GA) takes place in the action sets instead of the population as a whole. Finally, unlike the traditional LCS, XCS has no message list and so is only suitable for learning in Markov environments (XCS extensions using an internal-state register have shown promise in non-Markov environments).

In keeping with the typical LCS model, Wilson [5] proposes that the environment provides as a input to the system a series of sensory situations σ (t) $\in \{1, 0\}^L$, where L is the number of bits in each situation. In response, the system executes actions α (t) $\in$ {a1, …, an} upon the environment. Each action result in a scalar reward ρ (t) (possibly zero). The reinforcement program determines the reward according to the current environmental input and the action that was executed. Figure 2 illustrates the interaction of the environment and the reinforcement program with XCS.

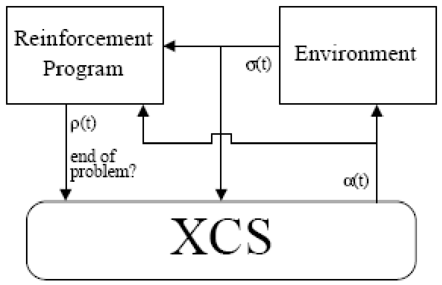

Figure 2. XCS interacts with an Environment and a Reinforcement Program [5]

2.3 Artificial Immune Systems AIS

Inspired by immunological theory, Artificial Immune Systems (AIS) [7] are adaptive systems based on models and principles which emulate mechanisms of defense observed in nature. Researches are interesting in several immunological properties from which have been generated a significant number of theoretical and computational models to solve real-world problems: self-identity, diversity, robustness, fault tolerance, pattern recognition and self- learning.

In 1974, N.K. Jerne stated in [8] that a biological immune system is a regulated network of cells and molecules that possess a dynamical behavior, even in absence of any external stimulation (antigens). Thus, artificial immune networks are models that emulate Jerne´s architecture. A set of antigens that correspond to an input data set in most artificial network applications, will stimulate an immune network to goes through a dynamic process, until it reaches some type of stability. AiNet [9] is a model based on the immune principles and implements a discrete immune network that was developed for data compression and clustering and later for optimization.

This paper references the work of D. Romero [9] who proposed an Artificial Cognitive Immune System (ACIS). Based on the advantages of AiNet, an ACIS algorithm that combines the structure of AiNet and a reinforcement machine learning mechanism is proposed by Romero [9]. Particularly, AiNet capabilities to perform data clustering, learning and optimization were exploited. Next, the proposed immune algorithm (ACIS) was used in the object recognition and optimization tasks. In Figure 3 the algorithm of ACIS is depicted.

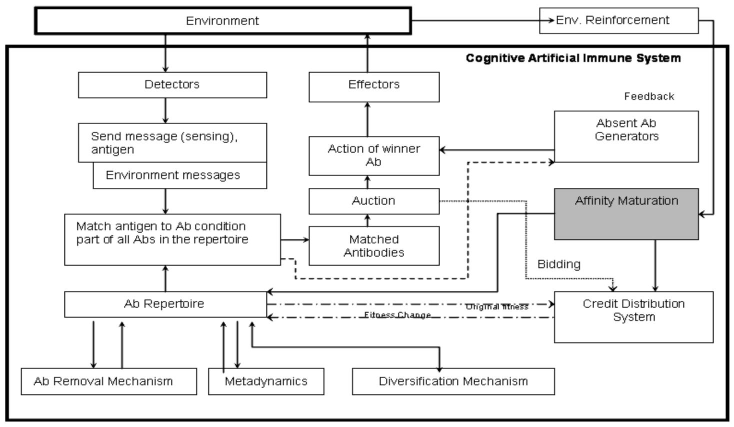

Figure 3. Flow diagram of the proposed cognitive artificial immune system (ACIS) [9]

2.4 Neuro Connectionist Q-Learning System NQL

The Q-Learning algorithm suggested by Watkins in 1989 [10] belongs to a group of reinforcement learning algorithms. In accordance with Watkins [10] the main feature of that technique is that in the process of learning the system is not shown how to act in a specific situation. Instead, learning develops by trial and error using reward and penalty signals. As a result of the Q-learning algorithm, a function of state-action pair evaluation appears that has a tabular representation. When the state-action space is large, it might be difficult to meet one of Q-learning algorithms convergence conditions (multiple approbation of all possible state-action pairs) and more resources are required to store the table of evaluation. According to Kuzmin [11], to solve those problems, additionally it must be introduced some generalization means. Using a Neural Net of type Multilayer Perceptron (MLP) as a Q-learning table approximator is one of the possible generalization means proposed by Kuzmin [11]. The joint use of MLP and the Q-learning algorithm is what Kuzmin [11] called connectionist Q-learning.

Kuzmin stated [11] that the use of neural network for Q-value approximation has the following advantages:

- Effective scaling for the space of large dimension inputs;
- Generalization for large and continuous state spaces;
- Possibility of implementation on parallel hardware.

Kuzmin [11] makes use of the methodology of working with a neural network that consists in applying a separate neural network for each action, see Figure. 4. During each iteration of the algorithm, the current state of the system is forwarded to the inputs of each neural network, but the weights are only updated for the network whose action was selected.

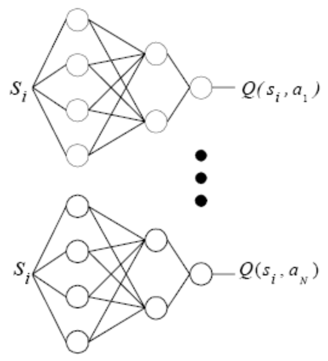

Figure 4. Q-function approximation by a set of Neural Networks [11]

2.5 Gene Expression Programming GEP

Gene expression programming proposed by Ferreira [12] is, like Genetic Algorithms and Genetic Programming, a genetic algorithm since it uses populations of individuals, selects them according to fitness, and introduces genetic variation using some genetic operators.

In accordance with Ferreira [12], the fundamental difference is the nature of three algorithms. In GEP the individuals are encoded as linear strings of fixed length (the genome or chromosomes) which are afterwards expressed as nonlinear entities of different sizes and shapes (e.g., simple diagram representations or expression trees).

The interaction of chromosomes (replicators) and expression trees (phenotype) in GEP implies an unequivocal translation system for converting the language of chromosomes into the language of expression trees (ETs). The structural organization of GEP chromosomes allows a truly functional genotype/ phenotype relationship, as any modification made in the genome always results in syntactically correct ETs or programs. Indeed, the varied set of genetic operators developed by Ferreira [12] to introduce genetic diversity in GEP populations always produces valid ETs. Thus, GEP is an artificial life system, well established beyond the replicator threshold, capable of adaptation and evolution.

Ferreira stated [12] that the advantages of a system like GEP are clear from nature, but the most important should be emphasized. First, the chromosomes are simple entities: linear, compact, relatively small, easy to manipulate genetically (replicate, mutate, recombine, transpose, etc.). Second, the ETs are exclusively the expression of their respective chromosomes; they are the entities upon which selection acts and, according to fitness, they are selected to reproduce with modification. During reproduction it is the chromosomes of the individuals, not the ETs, which are reproduced with modification and transmitted to the next generation.

A GEP Algorithm is proposed in this work to evolve individuals (agents) and their multilayered structures, learning both applicability predicates for behaviour activation and the conflict resolution hierarchy for behaviour arbitration. As a simple example [4], suppose that there are three task achieving behaviors with strictly decreasing priority. The applicability predicates and the suppressor nodes of these three behaviors are equivalent to the following composition of ordinary IF conditional functions:

```
(IF AP1 BEHAVIOR1
    (IF AP2 BEHAVIOR2
            (IF AP3 BEHAVIOR3)
```

In particular, if the first applicability predicate (AP1) is satisfied, then BEHAVIOR1 is executed. Otherwise, if AP2 is satisfied, BEHAVIOR2 is executed. Otherwise, the lowest priority behavior (e.g. BEHAVIOR3) is executed. In our approach, the GEP mechanism will be in charge of defining the applicability predicates and conditional rules from which behaviour layers are built for each agent.

3 Proposed hybrid, self-configurable and evolutionary approach for Cognitive Agents

In order to design an hybrid, self-configuring, self-organizing, scalable, adaptable, and evolutionary architecture for cognitive systems which exhibits emergent behaviours and learning capabilities, the proposed work is explained as follows.

Consider a virtual environment where there exists several agents interacting with objects, food, each others, etc., it arises some mayor questions and constraints:

- Environmental conditions change, e.g. about objects: quantity, type of object, location, size, etc., about other agents: intentions and desires, goals, etc.
- There is a variable number of desired behaviours: avoiding-obstacles, wandering, feeding, hunting, escaping, etc.
- How many behaviours can be integrated into a single agent? And how can agents arbitrate behaviours?
- When does a single agent know if it has to inhibit or suppress a behaviour if an applicability predicate is not preestablished?
- How can a behaviour that drives one of the layers in a single multilayered agent, generate a model of the world, couple with the environment via the agent's sensors and actuators, learn from its own interaction with the environment and receive a reinforcement of its actions, so the internal state of the behaviour evolve?

These questions address the following proposed approach of an hybrid, self-configurable and bio-inspired architecture for cognitive agents, depicted in Figure. 5:

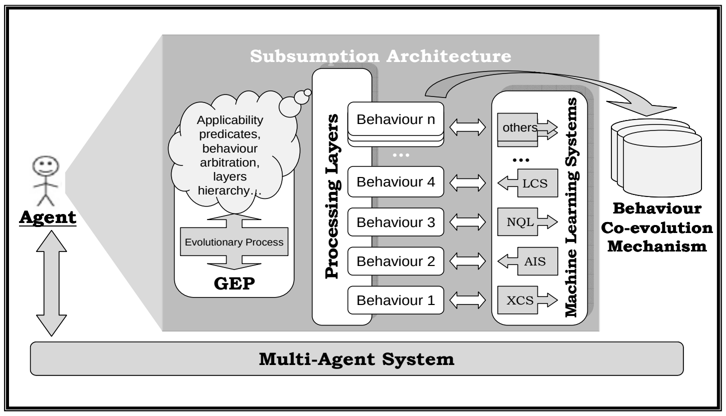

Figure 5. Hybrid and Evolutionary Architecture for Cognitive Agents

The Figure 5 shows an hybrid architecture from which all the questions mentioned above can be solved. An internal architecture based on subsumption principles but with few variations can be observed in every agent:

- Each processing layer is connected randomly with a different learning machine system (Extended Classifier System XCS, Artificial Immune System AIS, Neuro Connectionist Q-Learnig System NQL, Learning Classifier System LCS and scalable to others) which replaces the typical AFSMs proposed by Brook's architecture [1].
- After being trained in the agent, every behaviour is sent to a behaviour repertoire according to its type, where a co-evolutionary mechanism is applied so that every behaviour not only will learn in a local way inside of each agent but also will evolve in a global way, to be selected afterwards by another agent in the next generation.
- There is an evolutionary process driven by a Gene Expression Programming Algorithm GEP, which is in charge of self-configuring the agent: defining the number of layers, the

behaviours that the agent will use, the connections and hierarchies between them (inhibit, suppress, aggregate, etc.): behaviour arbitration, the applicability predicates where it is determined which behaviour is activated at a certain situation and an activation time controlled by a timer.

3.1 Hybrid Learning Architecture: Behaviours driven by different Machine Learning Systems

Every behaviour layer in the multilayered architecture will be associated to a Machine Learning System MLS, that allows the architecture being hybrid and not only reactive since each behaviour will be able to exert deliberative processes using the acquired knowledge. Besides, this mechanism gives plasticity to the architecture because every behaviour "learns" in an unsupervised, independent and parallel way, through its interaction with the environment, generating internal representations, rules and both specific and generalized knowledge. This mechanism is favored by the MLSs characteristics: robustness, fault tolerance, use of bio-inspired techniques, adaptability and it does not require a previous definition of knowledge (unsupervised learning).

There are two principles formulated by Stone [13] that have motivated the proposed layered learning approach:

- "Layered learning is designed for domains that are too complex for learning a mapping directly from an agent's sensory inputs to its actuator outputs. Instead the layered learning approach consists of breaking a problem down into several behavioral layers and using MLSs at each level. Layered learning uses a bottom up incremental approach to hierarchical task decomposition."
- "MLS is used as a central part of layered learning to exploit data in order to train and or adapt the overall system. MLS is useful for training behaviors that are difficult to fine-tune manually."

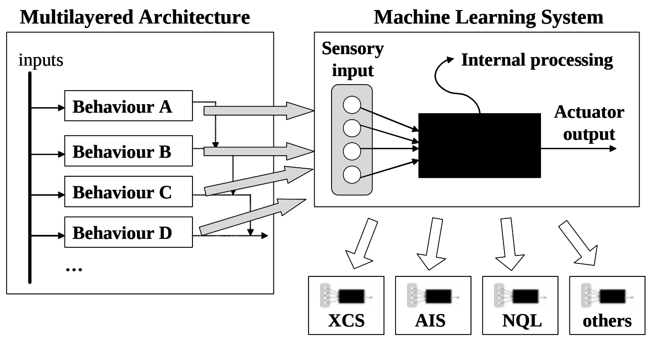

Figure 6. Multilayered Architecture connecting with MLS interface

The sensory inputs of each MLS read the distance and type of object sensed around the agent whereas the actuator outputs indicate actions that the agent must to execute, e.g. turn-to-the-left, turn-to-the-right, move-backward, move-forward, jump, stop, etc.

Accordingly, a common interface for all MLSs (XCS, AIS, NQL, LCS, etc.) is proposed so, although each MLS has a different internal process, they all have a similar structure that it lets the system to be scalable introducing new MLSs if is required and connecting them in an easy way with each behaviour layer in the agent's multilayered architecture, as depicted in Figure 6.

3.2 Hybrid Behaviour Co-evolution: evolving globally

A co-evolutionary mechanism is proposed to evolve each type of behavior separately in its own genetic pool. Most evolutionary approaches use a single population where evolution is performed; instead, the behaviours are discriminated in categories and make them evolve in separate behaviour pools without any interaction [14].

First, each agent defines a specific set of behaviours that builds its own multilayered structure. For each required agent's behaviour, a behaviour instance is chosen from the pool (this instance is connected with one MLS). Subsequently each agent will interact with the environment and each agent's behaviour will learn a set of rules and generate an own knowledge base.

After certain period time a co-evolutionary mechanism is activated. For each behaviour pool is applied a probabilistic selection method of behaviours where those behaviours that had the best performance (fitness) will have more probability to reproduce. Then, a crossover genetic operator is applied between each pair of selected behaviours: a portion of knowledge acquired by each agent's behaviour (through its MLS) is selected and interchanged with the other one: heritage of Knowledge and Experience.

Finally, new random rules are generated until complete the maximum size of rules that behaviours can have in their own knowledge base, so a new pair of behaviors is created and left in the corresponding behaviour pool to be selected by an agent in the next generation.

3.3 Self-configurable Architecture: Behaviour Arbitration

If each agent has an arbitrary behaviour set, how to determine: the interaction between them, the hierarchy levels, the Subsumption process (inhibition and suppression) and the necessary layers to do an adequate processing? These questions are solved next.

The internal multilayered structure of each agent is decomposed in atomic components which can be estimated and used to find the optimal organization of behaviors during the agent's lifetime [14]. The main goal is that the agent in an automatic way self-configures its own behaviours structure. The model proposed by Ferreira [12] called Gene Expression Programming GEP is used to evolve internal structures of each agent and generate a valid arbitration of behaviours.

GEP uses two sets: a function set and a terminal set. The proposed function set is: AND, OR, NOT, IFMATCH, IFOBJECT, INHIBIT, SUPRESS. The AND, OR and NOT functions are logic operators used to group and exclude subsets of objects, behaviours, etc. The conditional function IFMATCH is a typical applicability predicate that matches with a specific problem situation. This function has four arguments; the first three arguments belong to the rule's antecedent: the first indicates what object is sensed, second one is the activated sensor ant the third argument is the current behaviour running on the agent. If the first three arguments are applicable then the fourth argument, the rule's consequent, is executed. The fourth argument should be a INHIBIT or SUPPRESS function, or maybe and AND/OR function if more elements are necessary. The INHIBIT and SUPPRESS functions have two arguments (behaviourA, behaviourB) and indicate that behaviourA inhibits/suppresses behaviourB.

On the other hand, the terminal set is composed by the behaviour set, the environmental element set (objects, agents, food, etc.) and an agent's sensor set. Additionally "don't care" elements are included so whichever sensor, behaviour or object can be referenced.

Each agent has a chromosome with information about its self structure, e.g. the agent A can have a chromosome as: [{IFMATCH}, {wall}, {looking-for-food}, {sensor1}, {INHIBIT}, {avoiding-obstacle}, {AND}, {wandering}, {looking-for-food}], and this chromosome is a valid rule because both the antecedent and the consequent of IFMATCH function match to each required argument type: [{IFMATCH}, {object}, {behaviour}, {sensor}, {INHIBIT / SUPRESS}, {behaviour}, {behaviour}]. The above chromosome traduces in the following rule:

```
IFMATCH:
    There is a wall
    Is Activated looking-for-food behaviour
    Reading by sensor1
THEN:
```

Avoiding-obstacle INHIBIT wandering AND
looking-for-food

Analyzing this rule we can infer that the agent has three behaviour layers: avoiding-obstacle, wandering and looking-for-food, and the two last ones are inhibited by the first one when sensor1 identifies a wall in front of the agent. However, these chromosomes (applicability predicates) don't have always a valid syntax, so the GEP mechanism is used to evolve the chromosome until it becomes in a valid syntactic rule.

Each individual (agent) has a multigenic chromosome, that means, each chromosome has a gene set where each gene is an applicability predicate like the example, so the agent has multiples rules (genes) as part of its genotype and each one is applied according to the situation that matching the rule antecedent. Each gene is become to a tree representation and then a genetic operator set is applied between genes of the same agent and genes of other agents [12]: selection, mutation, root transposition, gene transposition, two-point recombination and gene recombination.

After certain number of evolutionary generations, valid and better adapted agent's configurations are generated. A roulette-wheel method is used to select individuals with most selection probability derived from its own fitness. Fitness represents how good interaction with environment during agent's lifetime was.

3.4 Emergent Properties of the Architecture

Brooks postulates in his paper [3] the possibility that intelligence can emerge out of a set of simple, loosely coupled behaviours, and emergent properties arise (if at all) due to the complex dynamics of interactions among the simple behaviours and that this emergence is to a large extent accidental.

The proposed architecture articulates a behaviour set that learns about environmental conditions in an independent and parallel fashion, and on the other hand evolve inside a categorized pool.

Each simple behavior can be applied to a subset of specific situations but not to the whole problem space, however the individual level interaction between behaviours (inside each agent) allows covering multiple subsets of problem states and some characteristics are generated: robustness, redundancy in acquired knowledge: fault tolerance and a big plasticity level, so emergent properties in the individual and inside of the society (Multi-agent systems) appear.

Then, the emergent properties arise from three points of view in a bottom-up approach:

- Atomic: in each behaviour of the multilayered architecture, when the associated MLS learns from the environment how to associate sensory inputs and actuator outputs, in an automate way.
- Individual: when the agent self-configures its internal structure (chromosome), hierarchy and arbitration of behaviours through an evolutionary process driven by GEP.
- Social: when an hybrid behaviour co-evolution mechanism is applied to all agent's behaviours, so behaviours learn not only themselves via the MLS associated but also cooperating with other agents and communicating the acquired knowledge between them.

It is important to notice that emergence in different levels, from atomic to social point of view, provokes an overall emergence of the system, where some kind of intelligence we hope to arise. The experimentation focused on discovering some characteristics of identity in the animats, e.g. we expected to see some animat agents behaving like depredators and others behaving like preys. Depredators should include some behaviours like avoiding-obstacles, looking-for-water, persecuting-preys, rounding-up, hunting-preys, etc. and Preys should include some behaviours like avoiding-obstacles, looking-for-food, looking-for-water, hiding, escaping, etc.

Nevertheless, expected emergent properties can vary according to the environment and the pre-configured behaviour set.

4 Experimentation

In order to evaluate the proposed architecture, following aspects were considered in each level:

About Machine Learning Systems:

- Learning convergence rate of each proposed systems: XCS, AIS, LCS and NQL.
- Generalization and Robustness (reactions to environmental changes)

About hybrid behaviour co-evolution:

- Learning and evolution convergence rate of each behaviour pool.
- Knowledge diversity in each behaviour pool

About GEP Algorithm to self-configuring Subsumption conditions and behaviour arbitration:

- Variation of success rate vs. number of genes
- Progression of fitness increment of the population
- Syntactically well-formed gene convergence rate

About overall System:

- Subsumption architectures obtained on individuals after n iterations and emergent properties.

An artificial life environment called Animat (animal + robot) [6] is proposed to test the experiments. The environment simulates virtual agents competing for getting food and water, avoiding obstacles, hunting, escaping from depredators, etc. This animat environment was selected because is more friendly to see emergent behaviours but it is not the only applicable environment. Each animat driven by an agent in the environment disposes a set of 14 proximity sensors (see Figure. 7) simulating a limited sight sense. 12 sensors read a safe zone and 2 sensors read a danger zone (to avoid collisions), as proposed by D. Romero [9].

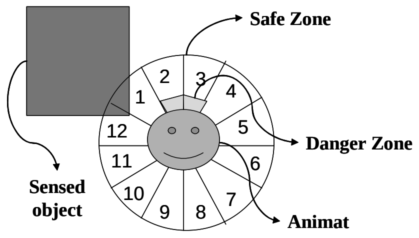

Figure 7. Animat Sensor distribution

Additionally, a simulated environment with objects, food, water deposits, animats, obstacles, traps, etc. is depicted in Figure 8.

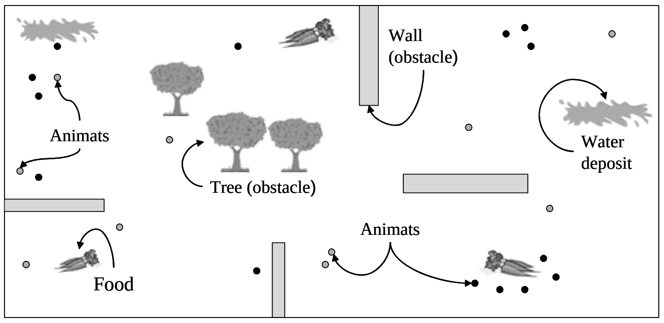

Figure 8. Simulated Animat Environment

Thus, some experiments designed to evaluate the performance aspects mentioned above are described next.

4.1 Learning convergence and Generalization level of each MLS

In this experiment we chose an environment where the animat has to interact with using one different MLS on a time. This scenario consists of a coarse rectangular grid. Inside the grid, there will be the animat (represented by a rabbit) in a maze which will have to avoid obstacles and follow the food path iteratively until it has learned the pattern. Each 200 iterations changes are applied to the environment and the animat will have to learn the new pattern and make knowledge generalizations in an adaptive way. Figure 9 shows the variations of the environment.

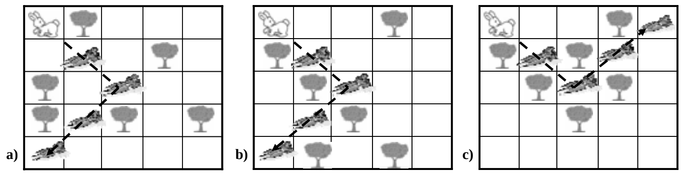

Figure 9. a) initial environment; b) soft variation of initial environment; c) hard variation of initial environment

Figure 10 shows a chart of the learning curve of the following MLSs: XCS, AIS, LCS, simple NQL and multilayered NQL. The learning convergence experiment was carried out with the environments on Figure. 9 and the following parameters:

- Number of epochs per run = 1000
- Number of runs = 20

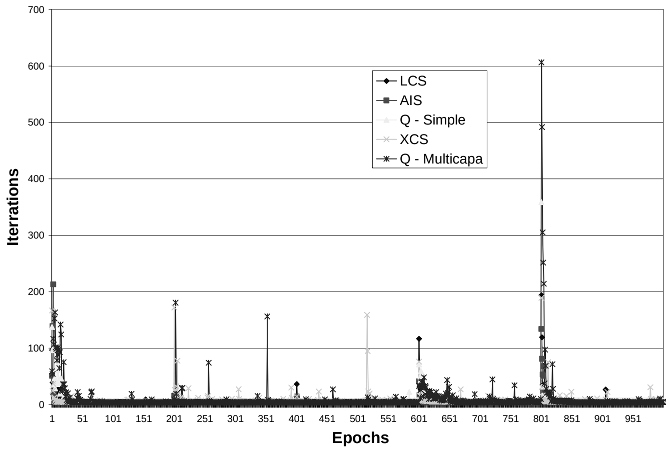

Figure 10. Learning Curve of each MLS

Figure 10 shows the different MLSs converging quickly after 20th epoch in average for the first pattern, however when any change in the environment is executed (each 200 iterations), each MLS registers a peak and then it converges again after 5-7 epochs. This demonstrates the capacity of MLSs to generalize the previous acquire learning and apply it to new situations, in this case, the learning of a new environmental pattern.

Besides, Figure. 10 illustrates that MLSs of type AIS and NQL are more adaptive and robust than the others converging more quickly when changes in the learned environmental pattern are introduced.

4.2 Learning and evolution convergence of each behaviour pool.

The goal of this experiment is to examine if the fitness of every separate behaviour pool increments gradually until reaches a convergence point whereas evolution takes place.

In this experiment 3 behaviours pools were studied: “looking-for-food”, “avoiding-obstacles” and “escaping-from- depredators” and measurements of average fitness in each behaviour pool were done.

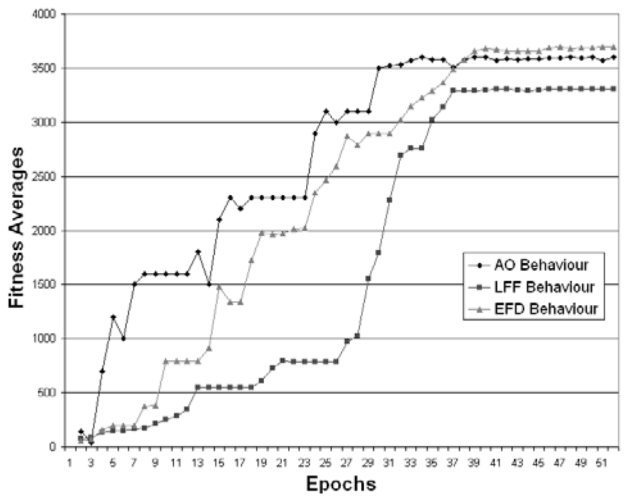

Figure 11. Evolution convergence rate in 3 behaviour pools

The learning curve of parallel behaviour evolution is depicted on Figure 11. The learning convergence experiment was carried out with the following parameters:

• Number of epochs per run = 50

• Number of runs = 50

In each epoch, 50 executions calculating the fitness average in the pool were done. Initially, in Figure 11 the looking-for-food behaviour has a learning curve slower in contrast with avoiding-obstacles behaviour, and on the other hand the avoiding-obstacles behaviour has the learning with more peaks in comparison with the other two behaviours which have softer curves due to differences in environmental conditions of each behaviour pool (number of positive rewards per time unity, impact of negative rewards, obstacles dispersion vs. food dispersion, etc.), however the 3 pools tried to converge and reach certain stability in the same number of epochs approximately (after 30 epochs), that means the evolution was effective and each behaviour pool has established a coherent knowledge base getting a consensus between its own behaviour instances about what the “behaviour” should do.

4.3 Variation of success with the number of genes in GEP

In order to define the number of genes that each chromosome should have, several experimental tests gradually incrementing the number of genes were done. Table 1 shows the used parameters in the experiment and Figure 12 illustrates the curve of variation of success rate vs. number of genes.

GEP Parameter	Value
One-point Mutation rate	0.45
One-point Recombination rate	0.15
Two-point Recombination rate	0.15
Gene Recombination rate	0.15
Root Transposition rate	0.10
Trasposition of IS elements rate	0.10
Gene transposition rate	0.10

Table 1. GEP Parameters

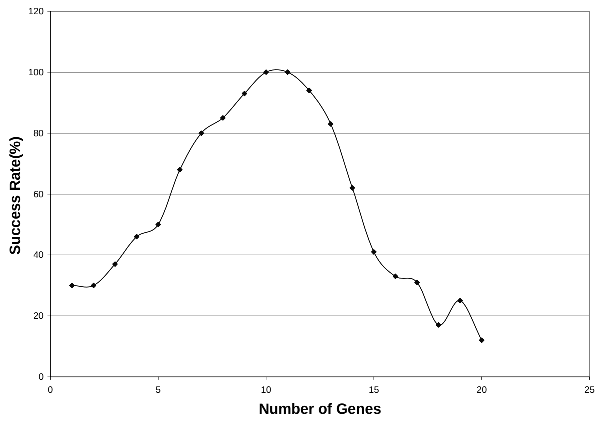

Figure 12. Variations of success rate vs. Number of Genes

Figure 12 above depicted that the success rate has the higher percentage when individuals have a chromosome with 10 or 11 genes. Therefore a multigenic chromosome with 10 genes using the configuration described in table 1 is proposed in the architecture.

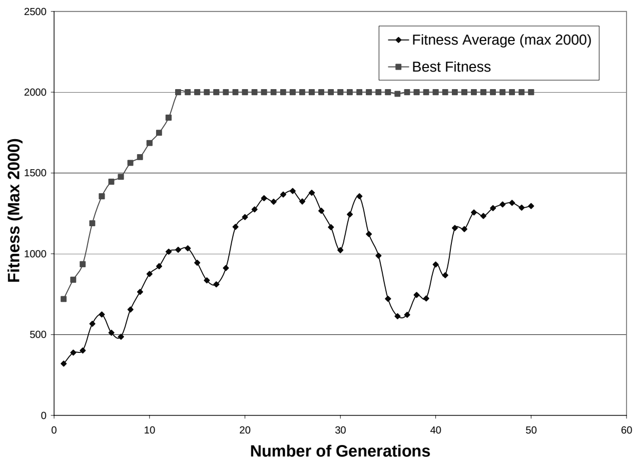

Figure 13. Progression of average fitness of the population vs. number of Generations

Additionally, we analyzed the progression of average fitness of the population whereas the number of generations was incremented. Using the GEP parameters in table 1, results of progression in 50 runs are shown in Figure 13. In this experiment a perfect solution was found in generation 12.

4.4 Syntactically well-formed gene convergence

In this experiment, the progression of the number of syntactically well-formed structure (multigenic chromosomes) of each individual was analyzed. Figure 14 shows how the number of valid chromosomes increments whereas generations evolve through the time. The experiment was executed with a population of 300 individuals.

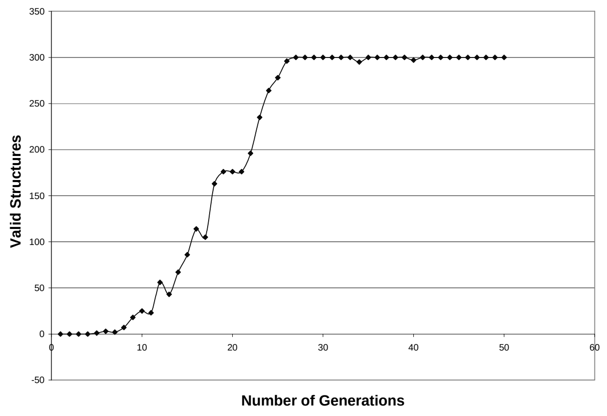

Figure 14. Valid Structures (chromosomes) through Generations

Figure 14 shows that a point of convergence (that means all chromosomes in population are valid) is given in the generation 27 approximately. Then, the system will need between 25 and 30 generations to evolve all individuals in the population.

4.5 Analysis of evolved architectures

Finally, after the whole system has evolved during a specific number of generations, we have analyzed the final structures of the best adapted agents where emergent properties arose.

Figure 15 shows the genotype (Expression Trees ETs) and phenotype respectively of an initial architecture of a random agent without any evolutionary phase; in contrast, Figure 16 shows the genotype and phenotype respectively of the evolved architecture of the same agent.

In Figure 15.b the chromosome represents four behaviours: looking-for-water, looking-for-food, avoiding-obstacles and hiding, where l-f-w inhibits l-f-f and hiding and l-f-w suppresses a-o, but there is a contradictory process when l-f-f tries to suppress l-f-w and l-f-f has been inhibited by l-f-w already. This is solved with the evolved architecture in Figure. 16.b, which proposes a new structure adding escaping-from-depredators behaviour and excluding hiding behaviour.

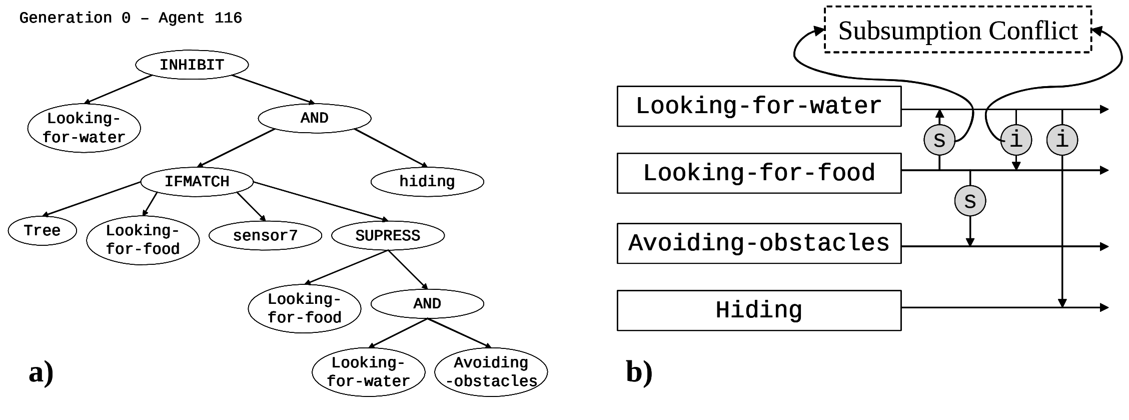

Figure 15. Genotype and Phenotype of an initial Agent's Architecture

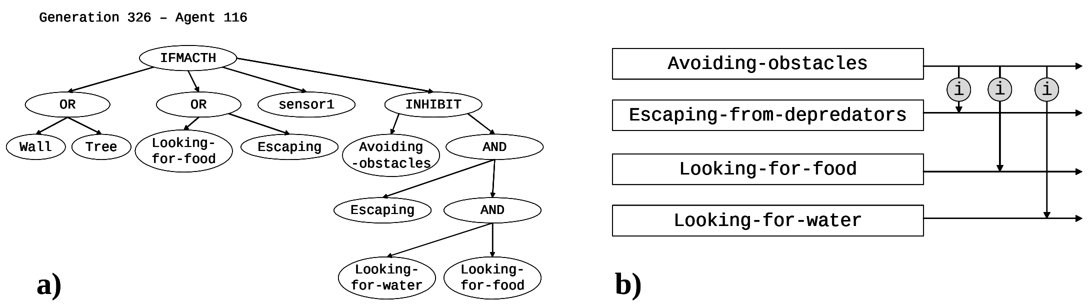

Figure 16 Genotype and Phenotype of Agent's Architecture after 326 evolutionary generations

As depicted in Figure. 16.b, the initial contradictory inhibitory/suppressor processes in the agent's architecture (see Figure. 15.b) are solved, and only hierarchical inhibitory processes are proposed by the evolved architecture. Furthermore, we can deduce too that evolved architecture has collected a set of specific behaviours becoming the agent to an animat with prey identity.

It is important to notice in evolved architecture that escaping-from-depredators behaviour inhibits looking-for-food and looking-for-water behaviours but if the animat is escaping and its sensor7 reads a "wall" or a "tree", then escaping-from-depredators behaviour is inhibited by avoiding-obstacles behaviour until the obstacle is not in front of the animat anymore, and after that the animat continues its getaway, so we can say that emergent behaviour arises.

Finally, the experimentation demonstrate that specific parameter configurations in MLSs, GEP and Co-evolutionary mechanism are required to reach certain robustness, learning and adaptation capacities in the overall system. Nevertheless, emergent properties didn't arise always or in a quick way, in several experiments animats died quickly and they couldn't learn to survive.

5 Conclusions

The integration of multiple Machine Learning Systems in controlling the behaviours layers of an hybrid Subsumption Architecture approach, instead of using the typical Augmented Finite State Machines, have demonstrated important advantages in learning about the world of the agent, making internal knowledge representations and adapting to environmental changes.

The evolutionary mechanisms used in this work, provided a plasticity feature allowing the agent to self-configure its own multilayered behaviour-based architecture; thus it can avoid creating exhaustive and extensive knowledge bases, pre-wired behaviour-based multilayered structures and pre-constrained environments. Instead, a cognitive agent using our architecture only needs to interact with an arbitrary environment to adapt to it and take decisions in a reactive and deliberative fashion.

Some problems were faced when we tried to define the applicability predicates which evolve through the GEP algorithm because several validations about function set must be done, and maybe this work will have to be done each time the architecture will be applied to other contexts.

In the experimentation, the emergent properties were difficult to discover because it takes a lot of time to evolve the overall system despite of using a multi-agent platform in a distributed configuration. Maybe, it can be similar to the natural evolution where adaptation occurs slowly and sometimes produces poor adapted creatures.

In our future work we expect to continue working on designing more adaptive and self-configurable architectures, using fuzzy techniques in the MLSs to improve the sensors readings. In the future, one concrete application of this research will be the development of a Cognitive Module for Emotive Pedagogical Agents where the agent will be able to self-learn about its own perspectives, believes, desires, intentions, emotions, skills and perceptions.

Acknowledgements

Supported by the Programme Alban, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E05D056455CO".

Diego Romero of Mechanical Engineering Department, Luis F. Niño of Computer and Systems Engineering at National University of Colombia; and Angélica de Antonio of Software Engineering Department at Politécnica de Madrid University, who made numerous contributions in the development of this research.

References

[1] R.A. Brooks, A Robust Layered Control System For A Mobile Robot, *IEEE Journal Of Robotics And Automation*, RA-2, 1986, 14-23.

[2] M.J. Mataric, Behavior-based control: Main properties and implications, *Proceedings of the IEEE International Conference on Robotics and Autonomation*, Nice, Francia, 1992, 2-8.

[3] R.A. Brooks, How to build complete creatures rather than isolated cognitive simulators, *Architectures for Intelligence*, 1991, 225-239.

[4] J. R. Koza, Evolution of subsumption using genetic programming, *Proceedings of the First European Conference on Artificial Life*, Paris, 1992, 110-119.

[5] S.W. Wilson, State of {XCS} Classifier System Research, *Lecture Notes in Computer Science*, 1813, 2000, 63-81.

[6] J.H. Holland, "Induction, Processes of Inference, Learning and Discovery", Mich:Addison-Wesley, 1953.

[7] L. N. de Castro, J. Timmis, "Artificial Immune Systems: A New Computational Intelligence Approach", Ed. Springer, 2002.

[8] N.K. Jerne, The Immune System, *Scientific American 229*, No. 1, 52-60, 1973.

[9] D. Romero, L. Niño, An Immune-based Multilayered Cognitive Model for Autonomous Navigation, *IEEE Congress on Evolutionary Computation*, Vancouver, 2006, 1115-1122.

[10] C. Watkins, Q-learning, Machine Learning 8, Boston, 1992 – pp. 279-292.

[11] V. Kuzmin, Connectionist Q-learning in Robot Control Task, *Proceedings of Riga Technical University*, 2002, 112-121.

[12] C. Ferreira, Gene Expression Programming: A new adaptive algorithm for solving problems, *Proceedings on Complex Systems*, forthcoming, 2001.

[13] P.Stone, Layered Learning in Multiagent Systems, (Doctor Thesis CMU-CS-98-187, 1998)

[14] A. Farahmand, Hybrid Behavior Co-evolution and Structure Learning in Behavior-based Systems, *IEEE Congress on Evolutionary Computation,* Vancouver, 2006, 979-986.