Computing and Informatics, Vol. 32, 2013, 595–627

INVESTIGATIONS INTO LAMARCKISM, BALDWINISM AND LOCAL SEARCH IN GRAMMATICAL EVOLUTION GUIDED BY REINFORCEMENT

Jack Mario Mingo

Computer Science Department
Universidad Autónoma de Madrid
Madrid, Spain
e-mail: mario.mingo@uam.es

Ricardo Aler

Computer Science Department
Universidad Carlos III de Madrid
Madrid, Spain
e-mail: aler@inf.uc3m.es

Darío Maravall

Artificial Intelligence Department
Universidad Politécnica de Madrid
Madrid, Spain
e-mail: dmaravall@fi.upm.es

Javier de Lope

Applied Intelligent Systems Department
Universidad Politécnica de Madrid
Madrid, Spain
e-mail: javier.delope@upm.es

Communicated by Gianfranco Rossi

Abstract. Grammatical Evolution Guided by Reinforcement is an extension of Grammatical Evolution that tries to improve the evolutionary process adding a learning process for all the individuals in the population. With this aim, each individual is given a chance to learn through a reinforcement learning mechanism during its lifetime. The learning process is completed with a Lamarckian mechanism in which an original genotype is replaced by the best learnt genotype for the individual. In a way, Grammatical Evolution Guided by Reinforcement shares an important feature with other hybrid algorithms, i.e. global search in the evolutionary process combined with local search in the learning process. In this paper the role of the Lamarck Hypothesis is reviewed and a solution inspired only in the Baldwin effect is included as well. Besides, different techniques about the trade-off between exploitation and exploration in the reinforcement learning step followed by Grammatical Evolution Guided by Reinforcement are studied. In order to evaluate the results, the system is applied on two different domains: a simple autonomous navigation problem in a simulated Kephera robot and a typical Boolean function problem.

Keywords: Hybrid algorithms, grammatical evolution, Lamarckism, Baldwinism

1 INTRODUCTION

Grammatical Evolution (GE) [1] is an Evolutionary Algorithm created with the aim of developing programs based on a grammar. In order to carry out this function GE employs a mapping process that transforms the codons which make the genome into useful information, with the purpose of helping to select the production rules in a BNF grammar. This mapping process is simple. Firstly, the genome is explored from left to right in order to generate an integer number sequence in a stage called *transcription*. Then, the integer string is used for getting a value that determines what production rule will be applied on each time along the creation program process. Usually, this second stage is known as *translation*.

Starting from this scheme and having as support the hypothesis that learning can guide the evolution [2], a Grammatical Evolution Guided by Reinforcement (GER) was proposed [3]. This system merges Grammatical Evolution and Reinforcement Learning [4] in an attempt to let individual learning during the lifetime of each individual. The main goal in the learning process is to generate new programs using the same grammar that is used for creating the original genetic program. Thanks to learning, new programs are built in an attempt to get a *learnt* program better than the original genetic program. According to the Baldwin effect [5], an organism that learns can evolve faster even if the result of learning is not copied back to the genotype. Giving a chance for the learned knowledge to be passed onto the offspring was proposed by Lamarck [6]. Nowadays, this proposal called *Lamarck Hypothesis* is not accepted from a biological viewpoint. However there is no reason for not using it from a computational viewpoint as GER does.

Hybrid or memetic algorithms (MA)[7] are a kind of algorithms created to solve complex problems that traditional evolutionary algorithms cannot solve easily. These algorithms enhance the global search capabilities associated with evolutionary algorithms with local search capabilities in order to explore the neighbourhood of an individual. Both features, local search and Lamarck hypothesis are used by GER although several differences exist between these systems as will be shown later.

The role of Lamarckian mechanism in GER was analyzed initially in [8] where two GER systems were compared: the first one included the Lamarck Hypothesis, i.e. it was a standard GER, and the second one did not include it. The results showed that the system without Lamarckian mechanism could not find solutions in the autonomous navigation domain used on tests. The present paper starts from this situation and studies the Baldwin effect, where the fitness after learning is kept (unlike Lamarck) but the learned program is not (like Lamarck). Another question this paper is concerned with refers to the exploration-exploitation trade-off that is inherent to Reinforcement Learning technique. Regarding different strategies [4, 9], this paper compares the initial e-greedy strategy of GER with probability based strategies.

The main aim of this paper is to compare different learning processes during a grammatical evolution guided by reinforcement (GER) under the same conditions. Of course, learning in evolution has been extensively studied but we are interested here in grammar based approaches only. Studying several alternatives in the learning process associated with GER is important to find out the most appropriate strategy in a specific domain. A second goal in the study is analysing how each learning alternative affect several parameters in the evolutionary process, such as the number of unique individuals in the population, the average fitness value or the number of individuals that really benefit from the learning. We apply different variants of GER in two different domains: firstly, in autonomous navigation of a Kephera robot wandering in an unknown environment, and secondly in a classical problem of finding a solution for an even 3 parity Boolean function. In the first problem, a simulated robot is used.

Next sections describe these points in detail. Firstly, in Section 2 we review learning and evolution from a general context and from a robotic perspective. Section 3 describes essential issues in grammatical evolution guided by reinforcement as a method of merging learning and evolution. Section 4 describes the problems to be considered. Section 5 explains all learning methods investigated in the experiments. Section 6 analyses results for each method and globally. Finally, Section 7 is dedicated to comment conclusions.

2 EVOLUTION, LAMARCK HYPOTHESIS AND BALDWIN EFFECT AS LEARNING MECHANISM

Relationships between evolution and learning are a constant topic of interest from natural and computational point of views. Computationally, evolution is a global

search while learning can be considered a local search in the neighbourhood of each individual. There are great advantages if both methods are combined because evolution lets adapt a population to slow changes in a time scale and learning lets adapt each individual to fast changes in the environment. Besides, evolution can operate in a parallel way and learning can smooth out the fitness landscape. From computational point of view two learning methods have been considered more extensively, namely Baldwinian and Lamarckian learning which correspond to biological theories established by the biologists Baldwin and Lamarck. According to Lamarck, the learning process followed by an individual is returned to its offspring via genotype, that is, Lamarck proposed a direct heritage of acquired features during the lifetime. According to Baldwin, the learning process followed by an individual is not returned directly to its offspring via genotype but the effects of learning drive evolution in a smoother way and these effects can appear finally in further generations. Current biologists did not confirm the Lamarck hypothesis and it is not used in that discipline because it is not plausible. However, from computational point of view there are no constraints to use it and some works have done it as we will show below.

Baldwin effect was computationally showed by Hinton and Nowlan [2] with a simple but elegant and compelling experiment. They used a genetic algorithm as evolutionary method and they used learning to find the only correct solution in an imaginative problem. When the population was driven only through evolution there was a single individual with a good fitness and the other ones had the same bad fitness. However, thanks to learning other individuals had a chance to improve their bad fitness as a function of the number of changes on its genotype to reach the solution. Therefore, learning could help evolution through a smoother search toward the only solution. Other works complement this initial proposal about interaction between learning and evolution, e.g., [10], where a study about interactions between adaptive processes on two levels was addressed. Individual level was mediated by learning while population level was driven by evolution. Ackley and Littman proposed an adaptation strategy known as *Evolutionary Reinforcement Learning (ERL)* which combined evolution with neural network learning and they studied how several populations behaved in a simulated world. Results showed that learning and evolution combined could produce better long-lasting populations than either alone. Belew et al. [11] studied interactions between learning and evolution in a strictly computational point of view and they argued that Lamarck mechanism is not possible when genotype and phenotype spaces are far from each other because *"it is computationally impossible to encode in the structural genotype the results of behavioural experiments"*. Anyway, as many others Belew et al. concluded that combined learning and evolution lead to efficient and reliable hybrid algorithm. Another early example comparing Lamarck and Baldwin effects is found in [12] where authors show how there are functions where simple genetic algorithm without learning as well as Lamarckian evolution converge to the same local optimum, while genetic search with Baldwin effect converges to the global optimum. Besides, they found out that an algorithm exploiting the Baldwin effect sometimes outperformed an algorithm exploiting Lamarckian evolution. However, Whitley et al. only considered

function optimization problems and their results showed as well that Baldwin search was slower than Lamarckian search. Gruau and Whitley compared these learning mechanism in [13] but they did not find important differences between results that the systems reached. However, Lamarck effect was more successful in an experiment developed by Ku and Mak [14], who compared the performance in a cellular genetic algorithm which combined evolution and both kinds of learning. Results show that the Lamarckian mechanism was able to support the cellular system when it was optimizing the weights of a recurrent neural network but the Baldwinian mechanism failed in this task. Convergence, in terms of the number of generations taken, was better with Lamarck than systems without learning and with Baldwin learning. We can finish this short general review with a work by Julstrom et al. [15], where these learning mechanisms were also compared in a different domain, specifically in a node partitioning problem with cycles of four minimum length. Again, Lamarck effect got better results than Baldwin effect. A consequence that we can extract of all these experiments is that performance of learning mechanism is very domain dependant.

From an evolutionary robotic domain point of view there is an obvious interest in merging evolution and learning as well. In a preliminary work about this topic, Nolfi et al. [16] studied an artificial neural network working in an environment with food elements. At the population level, the networks evolved to become fitter at one task on the basis of the number of food elements collected while at the individual level, each member tried to learn as to predict the consequences of its actions during the lifetime. Nolfi et al. did not use Lamarck mechanism in the experiments and the results showed a positive effect when evolution and learning were combined in spite of tasks being different. Authors explained this fact by considering that evolution tends to select individuals that are located in regions of the search space where both tasks are dynamically correlated. A different issue about evolution and learning was studied by Nolfi and Parisi [17]. In this case, a neural controller for a mobile robot was evolved with the aim to analyse if the controller was able to adapt to changes in the environment on each generation: colours of the walls switched from black to white and vice versa. Thanks to the complex interaction between evolution and learning, the evolved robots developed an ability to discriminate both environments because, as the authors mentioned, learning complements to evolution by its quicker adaptation to the environmental changes that are too fast. An example of Baldwin effect in robotic context is found in [18] where an analysis about benefits and costs of learning process was addressed. As Mayley wrote, *"it is the exploitation of the benefits and reduction of the costs that provide the selection pressure for firstly the adoption of a learned behaviour and subsequently its genetic assimilation"*. The genetic assimilation of acquired traits is a secondary but not less important consequence of the Baldwin effect.

Examples like the previous ones are only a small sample of relationship between evolution and different classes of learning. While it seems clear that hybrid systems get better performance than standalone systems, it is not clear that Baldwin effect outperforms Lamarck mechanism or vice versa. Although, there are more studies in other domains than in the robotic context, most of the conclusions have been

obtained for neural networks and they can be applied to this context because this type of structure is the most used to build robot controllers.

3 GRAMMATICAL EVOLUTION GUIDED BY REINFORCEMENT

GER tries to merge evolution and learning in a simple way: it allows individuals of the population to rewrite its own program several times, trying different choices by means of the same BNF grammar. This process is easy: initially each programmer-individual is evaluated by executing the program created starting from its original chromosome. Details about the mapping genotype-phenotype can be examined in [1]. Then, the individual can create new programs using other grammatical rules. If some learnt program is better than the original genetic program, a Lamarckian mechanism is carried out. This mechanism substitutes the original genotype by the learnt genotype. Basically the process consists of three stages:

1. transcription
2. translation
3. learning.

Figure 1 summarizes the process driven by GER.

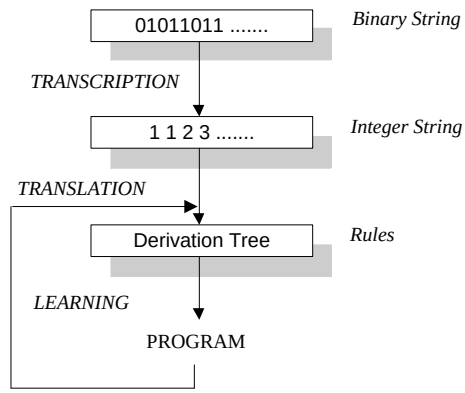

Fig. 1. Stages in Grammatical Evolution Guided by Reinforcement

Transcription is the process that transforms the original binary string into an integer number string, while *Translation* uses the integer number string for getting a value that represents the rule to apply for the current non terminal. The next formula shows how to get this value:

$$Rule = IntegerValue \,\%\, NumberOfRules \tag{1}$$

where *IntegerValue* is the current number in the integer string, % represents the module operator and *NumberOfRules* means the maximum number of rules corresponding to the non terminal that will be expanded each time. Finally, the *Learning* stage uses a Reinforcement Learning mechanism for generating new programs.

There are four essential items in GER, in addition to the actual evolution: *Reinforcement Learning, Q Tree, Lamarck Hypothesis* and *Exploration-Exploitation trade-off*. Evolution in GER performs just as in a standard GE or similar evolutionary algorithms. Next sections briefly describe all these issues. We will finish this review of GER with a discussion about the local optimization that GER addresses.

3.1 Reinforcement Learning

In Reinforcement Learning, an agent in a specific state can choose an action among a group of possible actions. We will show here how GER applies reinforcement learning but a good review about this topic can be found in [4]. Applying the reinforcement learning scheme to GER the following items are identified:

- An individual-programmer that represents the agent.
- A group of states that represent the derivation steps used up to a point in the building of the program.
- A group of production rules to apply in each of these states. Production rules stand for actions to apply.

The aim of the individual-programmer is to calculate a policy for the actions appropriate for any state. In GER, an action is simply a production rule to be applied. A state is the partial derivation steps sequence applied up to a particular instant. This way, state 0 would correspond to the start symbol of the grammar; state 1, to the result of substituting the start symbol by one of the production rules appropriate to it; state 2 would result from substituting the left most non terminal symbol by a production rule valid for it and so on. This process can be concluded because of one of two reasons:

1. The individual generates an analysis tree, that is, a tree whose leaves only contain terminal symbols. In this case, the learnt program is evaluated and its fitness is obtained.
2. The individual does not generate an analysis tree after a determined number of rule selections.

In this case, the individual obtains a very poor fitness. The reinforcement learning process is repeated as many times as it is established by means of a system parameter, and the consequence will be the creation of a group of learnt programs. Some of these programs will be grammatically wrong and they will get a very poor fitness while other ones will be grammatically right and they will get a fitness based on their capacity to solve the problem.

3.2 Q-Tree

GER uses a reinforcement learning based on Q-Learning [19]. The purpose of Q-learning is to learn which production rules should be used. Initially, a non-terminal symbol can be rewritten by any of the rules associated to that symbol, but the reinforcement process learns to give more weight to the most appropriate rules. GER employs a tree structure instead of a table because the state space might be very big for some domains. This tree is known as Q-Tree. Q-Tree stores the production rules more suitable on each state. With this aim Q-Tree keeps the following information on each node (state):

- A numeric value that represents the production rules used to reach the state. Root node is an exception because it is labelled with the start symbol in the grammar.
- A group of numeric values called Q values, which represent the different production rules that can be applied in the state.

An example will help to clarify the concepts and will show how the Q-Tree is filled. Next grammar is used for solving the navigation task although details about this problem will be described later.

```
N = {<code>, <line>, <if-statement>, <op>}
T = {move_left(), move_right(), move_forward(),
     stop(), wall_left(), wall_right(),
     wall_front(), if, else, (, ), {, }}
S = <code>

<code> ::= <line>                                        (0)
         | <code> <line>                                 (1)
<line> ::= <if_statement>                                (0)
         | <op>                                          (1)
<if-stm>::= if (wall_left()) {<line>} else {<line>}     (0)
          | if (wall_right()) {<line>} else {<line>}     (1)
          | if (wall_front()) {<line>} else {<line>}     (2)
<op> ::= move_left();                                    (0)
       | move_right();                                   (1)
       | move_forward();                                 (2)
       | stop();                                         (3)
```

where N means the non terminal symbols set, T means the terminal symbol set and S is the start symbol of the grammar as usual. Table 1 shows a hypothetical output of a transcription process applied to the initial binary string.

Figure 2 shows a possible mapping process with a wrapping value [1] (maximum number of times that a chromosome can be used to build a program starting from the grammar) equal to 5. A wrapping value is needed in order to avoid recursive productions rules producing an infinite loop during the mapping process.

1	1	3	0

Table 1. Chromosome of an individual-programmer

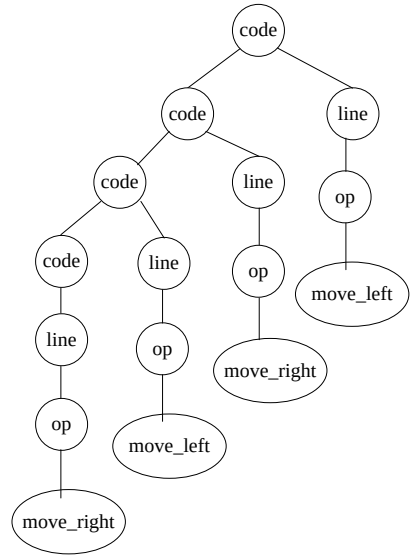

Fig. 2. Derivation Tree for the individual chromosome

Internal nodes in Q-Tree represent non terminal symbols and leaf nodes are terminal symbols. In this example the chromosome represented in Table 1 has been used 3 times to generate the tree.

While the derivation tree is being built a specific process builds the Q-Tree as well. Initially each Q value is configured with a default value of -1. There are as many slots as the maximum number of production rules on each node (in the example there are four Q values because there are four rules associated with the non terminal $\langle op \rangle$). When the original genotype program is evaluated, the genotype which is represented by means of integer values is covered and the integer values are saved in the Q-Tree according to the production rule that they select. In this way, in the example, the first node in the Q-Tree would have as label the start symbol of the grammar $\langle code \rangle$. The second node would have as symbol the value 1 (first integer on the string), the third one would save the symbol 1 (second integer on the string), the fourth one is the symbol 3 (third integer on the string) and so on (see the right branch in the Q-Tree represented in Figure 3 for details). The nodes of Q-Tree are linked through pointers from a higher node to a lower node.

Once the original genotype has been evaluated, a learning process is applied. To this purpose, the individual can look for new production rules during a number of learning steps which are defined by a parameter in the system, so the tree grows

according to learning process. In a learning step a new program is generated either right or wrong (from a grammatically point of view). In Figure 3, we can see the earliest stages in the Q-Tree as the result of a specific execution.

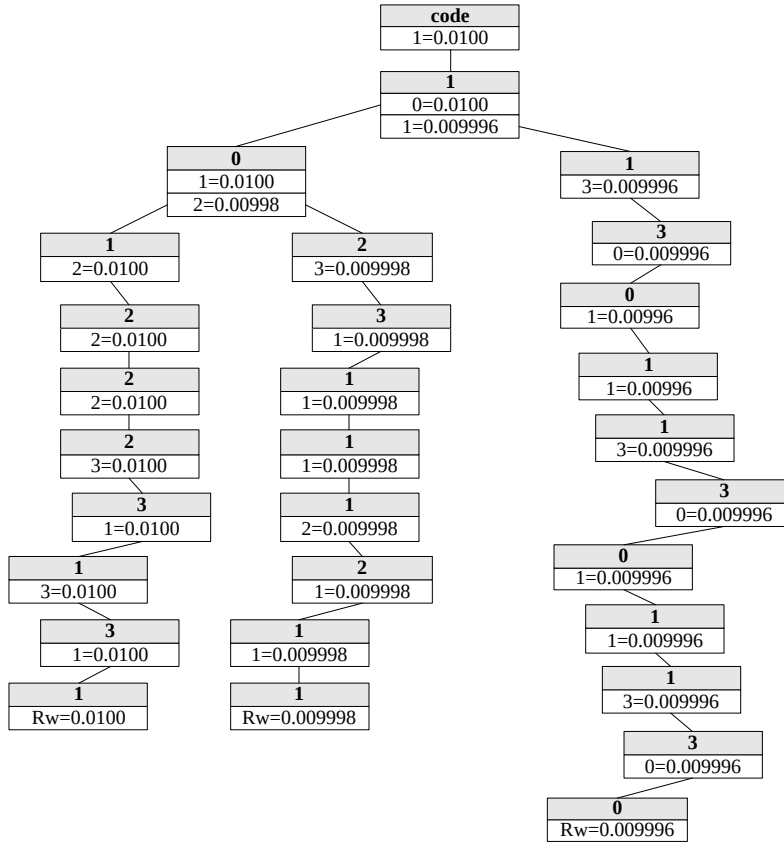

Fig. 3. Q-Tree for the individual chromosome

The symbol *Rw* stands for the *reward* [3] associated to the individual. On each node of the Q-Tree we have represented only keys (rules) that correspond to derivation steps tested by the individuals. The right branch of the Q-Tree shows the sequence of derivations associated to the original genotype (see Table 1 for details). Other two sequences of derivations associated with learnt genotypes are shown in the left and middle branches. Reward is propagated from the last derivation (leaf node) to the first one (level one) and it is computed by means of the following expression:

$$Reward = 1/FitnessValue \tag{2}$$

where *fitnessValue* depends on the specific problem we are going solve. As we can see in the Q-Tree shown in Figure 3 the reward is the same for all nodes in a derivation step. This is a consequence of using a discount factor of 1 when the updating rule for Q-learning [19] is applied:

$$Q(s_j, a_j) = r_j + \gamma \max_{a'} Q(s_{j+1}, a') \tag{3}$$

where, r_j is the reward in the level j of the Q-Tree, s_j stands for the state in the level j, s_{j+1} is the state in the level $j+1$, a_j is the production rule applied in the node s_j and a' is the rule with the higher value Q for the node s_j. Finally, γ stands for the discount factor and it is fixed to 1 in GER as we commented above.

To finish this section we notice that the Q-Tree plays global knowledge role in the system and it is maintained between population members and generations. It keeps different derivation steps which have been evaluated in the environment by all the individuals. It guides the search when the individuals have the chance to learn because it provides resources to try new steps starting from the old ones. To some extent we also can consider the Q-Tree as a type of cultural learning which might bias the results but we think that cultural learning can be included or considered as another source of knowledge that the individual can find during its life time and we can talk about learning in general terms. This way, we would not need to separate cultural and individual learning processes. We will adopt this view in the experiments but we are aware of this supposition would be analysed with more detail in other works.

3.3 Lamarck Effect

Once the learning process has finished, several programs have been created. Some of them are syntactically correct and some are not. Correct programs are evaluated by means of their execution and they will have a fitness according to their skill in the task. The learnt fitness is compared with the fitness reached by the genetically created program for the individual. If a learnt fitness is better than the initial fitness, the Lamarck Hypothesis can be applied and learning can be copied over the original genotype. This way, we can replace the original genotype of the individual by the best of the learnt genotype.

The use of the Lamarck Hypothesis in GER was initially analysed in [8] where a study about its exclusion from the system was tested in order to check its influence on the results. In [8], it was supposed that the learning process included in GER through the Q-Tree would be powerful enough to drive the population toward advantageous fitness landscapes. However, results showed that correct solutions were not possible without the use of the Lamarck Hypothesis. Nevertheless, in that work the Lamarck Hypothesis was excluded completely, i.e., neither the original genotype was replaced by the best learnt genotype nor the original genetic fitness was replaced by the best learnt fitness. This way, the system tested in that work trusted only in the knowledge the Q-Tree kept. An approach where the best learnt fitness replaces

the original fitness is essentially a Baldwin effect and we will study this approach in this work.

3.4 Balance Between Exploration and Exploitation

A common problem in reinforcement learning is the trade-off between exploring new actions or exploiting previously learnt actions. GER uses an e-greedy strategy to select actions (production rules). Random selection is a problem with this strategy and GER tries to solve it using a non constant factor that varies following the expression:

$$EGreedyFactor = 1 - (learnStep/learnStepsNum) \tag{4}$$

where *learnStep* means the current learning step and *learnStepsNum* represents a system parameter which specifies the total number of learning steps.

The *EGreedyFactor* value is used when an action must be selected during the learning process. In this case the action means to select a production rule and GER selects a rule depending on the e-greedy factor. If the factor is less than a threshold (a system parameter), GER selects a known rule. Otherwise, GER selects a new rule. In terms of values, when *learnStep* increases then *EGreedyFactor* decreases and it is more probable that *EGredyFactor* does not go beyond the threshold, so GER will select a known rule. By means of this strategy it is possible to contribute to the exploration in the first learning steps and the exploitation in the last learning steps. This strategy tries to use a number of learning steps that ensures an almost-full evaluation in the earliest stages in the process. This way, the last learning steps can take advantage of a Q-Tree with more information about the production rules tested.

3.5 Local Optimization in GER

GER is a hybrid algorithm with two main components: an evolutionary process which addresses a global search and a learning process which drives a local search. Reinforcement learning used in GER is local because once GER generates and evaluates the original genotype, it starts to build new programs but it creates programs that are close to the original genotype. To do it, all the learnt derivation trees start applying the same initial rule number than the original derivation tree. Besides, as a consequence of the balance between exploration and exploitation after some initial exploration steps the individual will search already exploited nodes.

Local optimization is a distinctive feature of memetic algorithms (MA) but there are several differences between GER and MA. Firstly, in MA local optimization is usually implemented by iteratively applying a mutation operator to the agent. However, GER applies a method based on reinforcement learning to find new solutions for each individual. Secondly, local optimization in MA can be applied in different stages [20], i.e. after other genetic operators, applying it only in the final step, etc. GER applies reinforcement learning after other operators have been applied.

Generally, MA tries to include domain knowledge through heuristics while GER does not add this kind of knowledge because reinforcement learning is driven by the goal of getting the maximum reward and it chooses rules depending on its utility to find that reward. Nevertheless, GER applies Lamarck mechanism like most MA do. We can argue that both GER and MA try to improve the performance with local optimization but they do it in different ways.

4 PROBLEMS AND DOMAINS

In order to analyse the learning process in GER we have applied this system in two different domains and problems. The first domain is related to robotics and the second one is a classical problem in evolutionary computation. A robotic domain is very useful if we want to analyse the process in dynamic environments because the sensors and actuators of the robot can introduce noise and this fact allows us to reproduce a non-deterministic problem. On the other hand, if we want to analyse the process in static problems we can test it in a second domain. In this case, we have chosen a Boolean function because this kind of problems are deterministic, i.e. input and output values are known in advance and the algorithm only tries to find a correct solution. We think this combination of static and dynamic domains can support the study more extensively.

4.1 Autonomous Navigation

Autonomous navigation is a very studied task. In this work, we only consider a simple task: a robot wanders around an unknown environment. The robot must navigate without hitting against the walls or other obstacles. We use a simulation of the Kephera robot which was developed by J. C. Gallager and S. Perreta. This version can be downloaded freely from [21].

To evaluate the robot's movements, the environment is divided into 16 zones. Zones are used for evaluating the fitness function together with a measure about the collisions. The fitness function is calculated by means of the formula:

$$Fitness = (Colls/Steps) + ((Steps((Visits/Steps)) \tag{5}$$

where *Colls* stands for the number of collisions against the walls incurred by the robot, *Steps* refers to time units that an individual spends while executing the program and *Visits* represents the number of zones visited during navigation. It is important to take into account the number of visited zones with the aim of limiting the effect in the evolution of some individuals, who do not hit any wall but simply turn around itself. An individual is successful if it visits 7 or more zones and does not hit anything. Figure 4 shows the environment and its division in 16 zones. These zones are not visible during the simulation but they are shown for clarity.

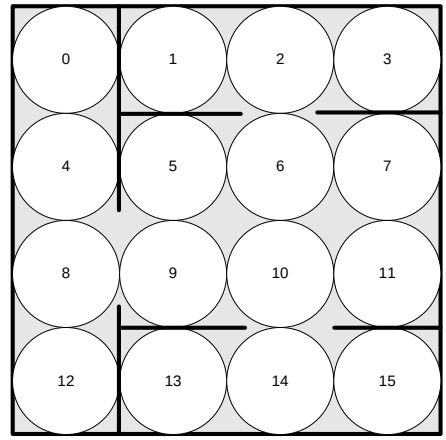

Fig. 4. Environment distributed into 16 zones

The grammar we used for solving this task was mentioned in Section 3.2 when the Q-Tree was reviewed. Table 2 shows the most significant parameters in the system.

goal	Find a program that allows to simulate a Kephera robot wandering around an environment without collisions
Fitness Cases	2 cases. One for each starting point
Case's Raw fitness	$(collisions/steps) + (steps - (visits/steps))$
Standardized fitness	*Case's Raw fitness/number of fitness cases*
Termination criterion	Maximum number of generations or the first individual who gets an standardized fitness less than or to than 99.93 (0 collisions and 7 visited zones)
Codons by individual	10
Wrapping events	5
Population size	50
Maximum generations	20
Learning steps	10
Discount factor	1.0
Crossover probability	0.9
Mutation probability	0.03
Duplication probability	0.05

Table 2. General parameters in the autonomous navigation problem

Learning steps and *discount factor* parameters where introduced in [3]. The remaining parameters were defined previously in [1]. As we will show, these parameters seem to be sufficient to solve the problem.

4.2 Even 3 Parity Boolean Function

Boolean function is a classical problem in order to compare different evolutionary strategies. As we only intend to analyse the learning process in GER we have chosen the even 3 parity case because it is not a very hard problem but it is large enough to explore its evolution. In order to solve the even 3 parity problem, the following grammar is used:

```
N = {<expr>, <op>, <var>}
T = {and, or, nand, nor, (, )}
S = {<expr>}

<expr> ::= <op> ( <expr> , <expr> )   (0)
         | <var>                      (1)
<op> ::= and                          (0)
       | or                           (1)
       | nand                         (2)
       | nor                          (3)
<var> ::= d0                          (0)
        | d1                          (1)
        | d2                          (2)
```

The group of terminals has been taken from [22]. Table 3 shows the common configuration for the problem in this case. We think these parameters can solve the problem because some previous work has shown that a population of 200 suffices to obtain good results in this kind of domains [8]. This reference also shows that beyond 30–50 generations, the system begins to stagnate.

5 ANALYSIS OF THE LEARNING IN GER

This section is concerned with the analysis of different variants about the learning process in GER. We consider as learning in this study the Lamarckian mechanism, the Baldwin effect and the exploration-exploitation trade-off usual in reinforcement learning. The last term is part of the local search implemented by GER. Next sections describe the different systems which implement all cases.

5.1 Grammatical Evolution Guided by Reinforcement (GER)

This system refers to a standard GER as it was defined in [3] with an e-greedy mechanism of balancing between exploration and exploitation. It embodies the Lamarckian hypothesis. This system is proposed as baseline.

goal	Finding a logical function which solves the even 3 parity Boolean problem, that is, the one that returns true when there is an even number of bits set to 1 arguments and false when not
Fitness Cases	The eight possible logical combinations for three arguments
Case's Raw fitness	The number of success in the eight fitness cases
Standardized fitness	$8 - \textit{Raw fitness}$
Termination criterion	When the standardized fitness is 0
Codons by individual	10
Wrapping events	4
Population size	400
Maximum generations	50
Learning steps	30
Discount factor	1.0
Crossover probability	0.9
Mutation probability	0.01
Duplication probability	0.01

Table 3. General parameters in the even 3 parity Boolean function problem

5.2 Grammatical Evolution Guided by Reinforcement with Probabilistic Exploration Strategy (GER-Kprob)

This system refers to a standard GER with a typical exploration strategy in reinforcement learning and it is based on probabilities according to the next expression [23]:

$$P(a) = k^{ER(a)} / \Sigma_{a' \epsilon A} k^{ER(a')} \tag{6}$$

where a represents the rule (action), $ER(a)$ measures the probability to choose the action and k is a constant value that was fixed in 0.5 after several tests. GER-Kprob and GER only differ in the rule they use for exploring and exploiting. Equation (6) is used when an action must be selected during the learning process. In this case the action means to select a production rule and GER-KProb selects a rule depending on this expression.

5.3 Grammatical Evolution Guided by Reinforcement with Boltzmann Exploration Strategy (GER-Eprob)

This system implements a standard GER with another common strategy in reinforcement learning as the Boltzmann exploration strategy. In this case, the expression is described as [9]:

$$P(a) = k^{ER(a)/T} / \Sigma_{a' \epsilon A} k^{ER(a')/T} \tag{7}$$

where a means the rule (action), $ER(a)$ measures the probability to choose the action, k is a constant value and T is a parameter usually called *temperature*. This parameter decays with the time and as consequence exploration decays too. Temperature parameter decays according to the learning step as expression:

$$Temperature = 1 - (lerningStep/lerningStepsNum). \tag{8}$$

Equation (7) is used when an action must be selected during the learning process. In this case the action means to select a production rule and GER-EProb selects a rule depending on this expression.

GER, GER-Kprob and GER-Eprob are systems created with the aim to test the best exploration/exploitation policy while the individual is building its derivation tree. All the systems are standard GER and they apply Lamarck hypothesis but they apply different expressions during the learning process. GER uses Equation (4), GER-KProb uses Equation (6) and GER-EProb uses Equation (7).

5.4 Grammatical Evolution Guided by Reinforcement with Baldwin Effect (GER-Baldwin)

A GER-Baldwin system is implemented like a standard GER without applying the Lamarck hypothesis. Therefore it does not replace the original genotype by the best learnt genotype if any exists. Nevertheless, if a learnt program is better than the original genetic program the learnt fitness replaces the original fitness. This way, the system implements GER with Baldwin Effect instead of GER with Lamarckism.

5.5 Grammatical Evolution Guided by Reinforcement without Applying a Lamarckian Mechanism and without Replacing the Original Fitness (GER-QTree)

The last system consists of a GER system without the Lamarck hypothesis like the previous one. However, in this case the original fitness is not replaced if a best learnt program is found during the process learning. This system was implemented previously in [8] and it does not include either Lamarck hypothesis or Baldwin Effect. Nevertheless, this is an interesting system to test if the Q-Tree is enough for solving the task because Q-Tree is an essential element in GER as learning mechanism. It keeps a memory of alternatives (derivation steps) used during the process of learning and evolution and it seems possible that Q-Tree can implement a Baldwin Effect implicitly.

6 RESULTS

6.1 Autonomous Navigation

In order to compare results in the case of the navigation problem we tested 40 executions for each system described in the previous section (cf. Table 2 for general parameters). The following criteria were used in the comparative analysis:

- Success probability.
- Average standardized fitness by execution.
- Average individual learning by execution.
- Average genetically correct individual by execution.
- Average unique individuals by execution.

Next sections describe these criteria and the results that each system attained. Average values are computed taking into account all the completed generations on each test.

6.1.1 Success Probability

Success probability is a widely used measure in evolutionary computation and it allows to test whether a system can find a solution to a problem. It measures the successful number of tests in relation to the total number of tests. Figure 5 shows results for each system.

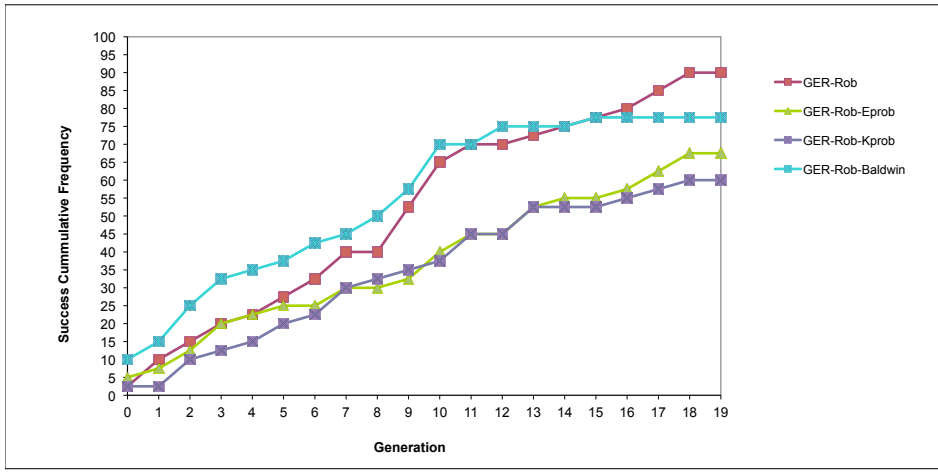

Fig. 5. Navigation Problem: Success cumulative frequency

Figure 5 only shows four systems because GER-QTree system did not find a solution to the navigation problem after 40 executions. As Figure 5 shows, the standard

GER system (with e-greedy selection) reached the best results with 90 % success probability, while the GER-Baldwin system that refers to GER implementing Baldwin Effect reached 78 % success probability. However, differences between both systems are not statistically significant according to a Chi-Square test (*Chi-Square* = 2.296, *p-Value* = 0.12970). On the other hand, systems with probabilistic action selection reached 68 % (GER-Eprob) and 60 % (GER-Kprob). In this case, differences between GER and GER-Eprob are statistically significant (*Chi-Square* = 6.05, *p-Value* = 0.01390) and differences between GER and GER-Kprob are also significant (*Chi-Square* = 9.6, *p-Value* = 0.00194). Finally, differences between probabilistic systems (GER-EProb and GER-KProb) are not statistically significant according to a Chi-Square test (*Chi-Square* = 0.487, *p-Value* = 0.48526).

6.1.2 Average Standardized Fitness related to Best Individuals

This measure represents average standardized fitness on each of 40 executions. This value is computed by dividing the sum of fitness values between the number of completed generations in the execution. The fitness considered corresponds to the best individual in each generation. This way, this average fitness value is related only to the best individuals and it is not related to the whole population. We consider the best individuals because we want to analyse how the best individuals evolve during the process taking into account all the tests. Analysing average values for the best individuals we can check how a system is learning or evolving because if the average value is near the optimum value, the system is learning or evolving correctly, but if the average value is far from this optimum value the system neither learns nor evolves properly.

In the navigation problem, an execution is considered to be correct if an individual gets a fitness equal to or higher than 99.93 because this value means that 7 zones are visited and there are no collisions against any wall. Figure 6 shows results grouped by the system.

As can be seen in the figure, the worst system is the same as for success probability (GER-QTree) because it maintains fitness practically constant and very above 99.93 (approximately near 99.99 which is a great difference if we look at the scale in the Y-axis). We remember that the higher the fitness the worse the individual is, so we can consider this result as reasonable because GER-QTree did not find any solution. Other systems exhibit similar tendencies according to this criterion. A simple explanation is needed here to explain why the worst system maintains populations with bad individuals and poor fitness. Most individuals in GER-QTree are bad, both in the first generations and the last ones, because the system can not drive the population toward optimal positions. On the other hand, successful systems support good individuals, maybe not initially but when evolution and learning progress these systems drive toward the right landscape and better fitness is attained practically for all the best individuals in each generation. Results in this section show an important consequence: GER-QTree seems not to evolve and not to learn properly in dynamic domains. Therefore, it should be concluded that it

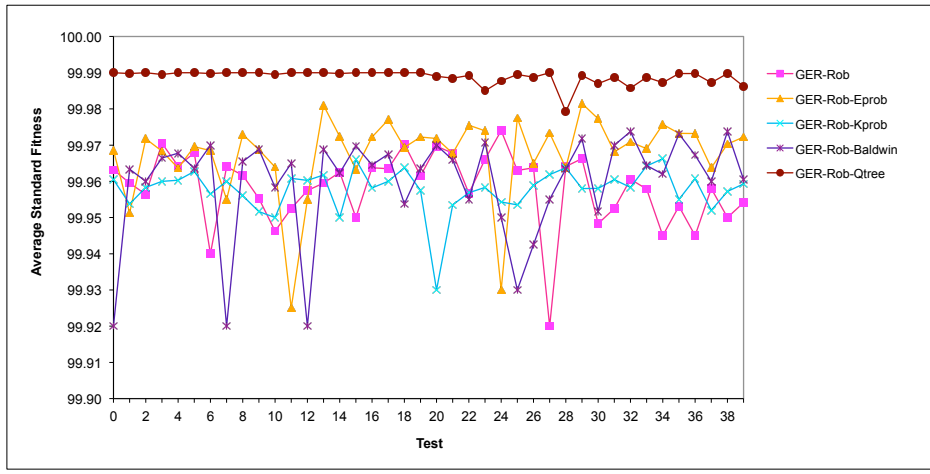

Fig. 6. Navigation Problem: Average standardized fitness

are important that the results of learning are reflected in the individual, either by altering the improved fitness (Baldwin) or by altering the individual (Lamarck).

6.1.3 Average Individual Learning

Average individual learning by execution is useful to analyse the average of learning individual on each of 40 executions. This value is computed by dividing the sum of effective learnings between the number of completed generations in the execution. In this work an individual is considered to have learned when, at least, it is able to find a program fitter than the original during its learning process. This value allows the systems to test if individuals take advantage of the learning process. Contrary to the fitness average value, this average value is related to the whole population. Again, Figure 7 shows results group by system.

Results show that values are similar in all systems except in the case of GER-Baldwin. Values, in general, are above 30, which indicates that on each execution nearly 30 individuals would take advantage of the learning process against 20 individuals that would not. A curious difference is shown in the figure between GER-QTree and GER-Baldwin systems. None of them implements the Lamarck hypothesis but the first one does not change the fitness and the second one does. Average individual learning is higher in GER-Baldwin. A possible explanation might be in the change of fitness when an individual learns although its genotype remains unchanged. The original genotype could be very poor and consequently during the learning process it is easy to find better programs. However, as the fitness is changed to be better, the individual has the chance to be selected in genetic operations. So, the individual can pass to the next generation where it would learn again because its original genotype was bad. By contrast, in GER-QTree neither genotype nor fitness are

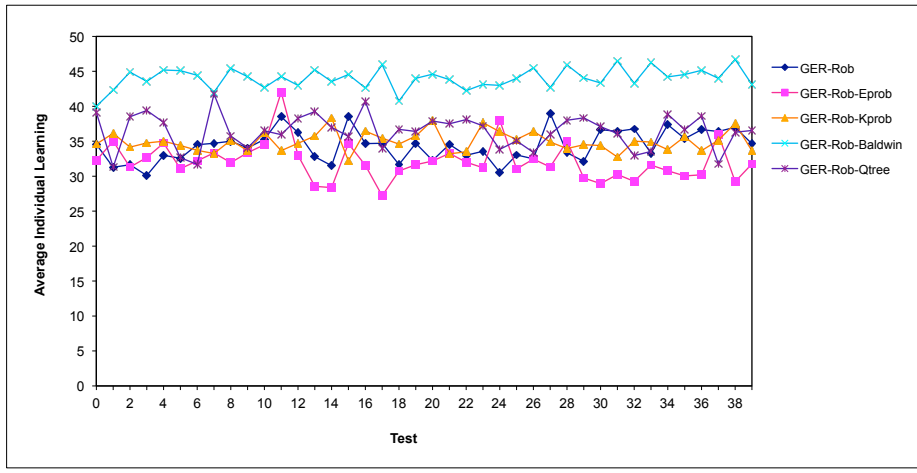

Fig. 7. Navigation Problem: Average individual learning

changed, so it is more difficult to pass to the next generation because its fitness remains poor. It seems that GER-Baldwin maintains a group of individuals with bad original genotype but good fitness reached during its lifetime through a learning process while the remainder systems discard bad individuals and maintain a number of individuals more or less stable that always learn.

6.1.4 Average Genetically Correct Individuals

This measure counts how many individuals have an original genotype that generates a correct program from a syntactical point of view. Actually, we use the sentence "syntactical point of view" to express that an individual is syntactically incorrect when it is not possible to translate its genotype into a correct phenotype. In other words, a genetically correct genotype allows building a correct program after the translation stage in GER (see Figure 1 for details). Mapping process in grammatical evolution basically depends on the genotype length and the *wrapping event* parameter, i.e. in order to avoid infinite loops the algorithm tries to apply rules for a while but if a tree with only terminals as leaves can not be built we consider the individual as *syntactically incorrect* although a term like *bad individual* could be more appropriate.

The results grouped by systems are shown in Figure 8. This value is computed by dividing the sum of genetically correct individuals between the number of completed generations in the execution and it is a value related to the whole population. The number of genetically correct individuals is important in terms of evolution.

As we can see in the figure the systems again obtain a similar behaviour except in GER-Baldwin. Average genetically correct individual is close to 40 and this value seems to indicate that learning is returned into genotype in the systems that

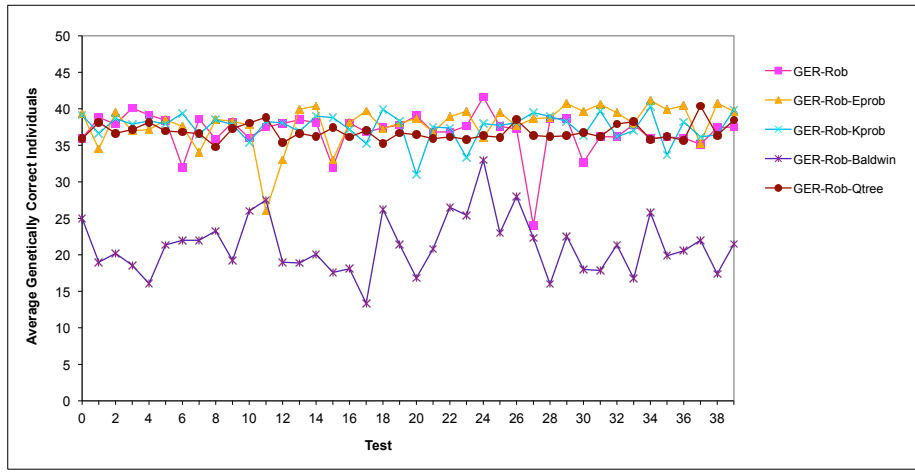

Fig. 8. Navigation Problem: Average genetically correct individuals

implement this feature. This way successive generations can take advantage of this circumstance and they can maintain individuals with an initially correct code. The hypothesis would be convenient in cases like GER, GER-Eprob and GER-Kprob because they are systems that implement the Lamarckian mechanism. However, GE-QTree system does not return the results of learning on the genotype and fitness. In this system the number of correct genetic individuals only depends on the beneficial genetic operations. A high average value can indicate that system lead population toward individuals with a syntactically correct code, even though, as it was shown in Figure 5 when we analysed the success probability, these individuals can not solve the problem. Finally, GER-Baldwin system did not reach a result as good as in other systems; but this system could be damaged because if the fitness is changed for an individual but its genotype remains unchanged, the individual might be selected and pass to the next generation, but then, if its original genetic was grammatically incorrect, it will be equally incorrect in the next generation. This could be an aspect to consider for explaining a smaller number of genetically correct individuals in GER-Baldwin.

6.1.5 Average Unique Individuals

The measure of average unique individuals provides the level of diversity inside of population. Information about this factor in the experiments is shown in Figure 9. Like the learning and genetic averages, this value is also related to the whole population.

It can be seen in Figure 9 that there is an evident difference between systems which maintain an explicit learning (GER, GER-Eprob, GER-Kprob and GER-Baldwin) and the system with implicit learning (GER-QTree). The last system got

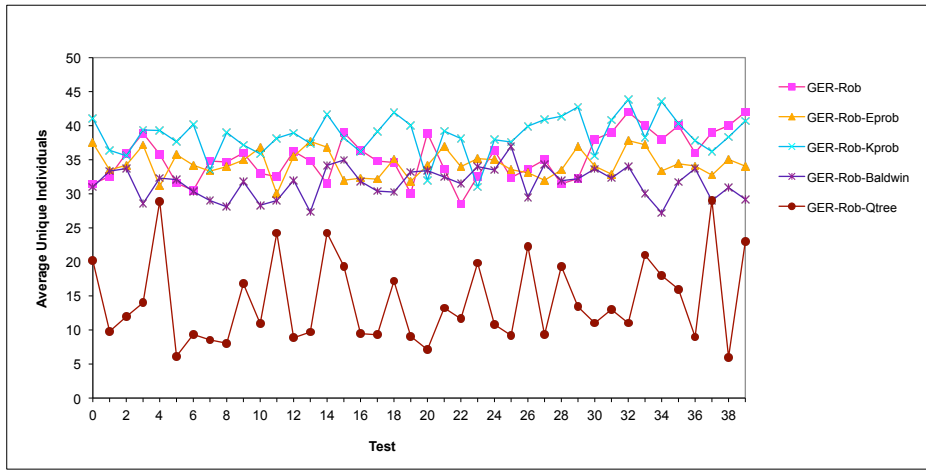

Fig. 9. Navigation Problem: Average unique individuals

lower results on this measure because it had fewer unique individuals. It seems that diversity of population is smaller when learning does not revert into population again through individuals both via fitness and via genotype. Probably, the general parameters (Table 2) are not enough to check more paths in the Q-Tree and the GER-QTree can not test the branches extensively. This system only trusts in the Q-Tree as the single mechanism to learn and evolve but the Q-Tree can only be valuable if it contains a lot of information. In other case, the Q-Tree can repeat bad branches continuously and the number of individuals to be built is reduced. Therefore, the diversity of population is smaller. A higher learning steps parameter (Table 2) could improve this behaviour but finding a solution could be very computationally expensive.

6.2 Even 3 Parity Boolean Function

As the even 3 parity Boolean function problem is more computationally expensive than the navigation problem we only tested 10 executions for each system in this case (see Table 3 for general parameters). We analyse the same criteria here and the next sections discuss the results. Average values are computed taking into account all the completed generations on each test.

6.2.1 Success Probability

Figure 10 shows results for each system related to the success probability.

We can see again that Figure 10 only shows four systems because GER-QTree system did not find a solution to the Boolean function problem either. As Figure 10 shows, the GER system (with e-greedy selection) reached the best results with 80 %

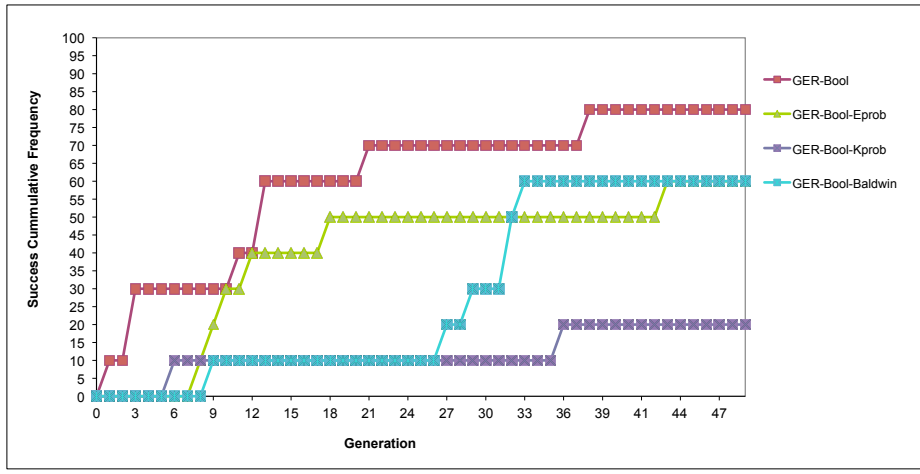

Fig. 10. Boolean Function Problem: Success cumulative frequency

success probability, while the GER-Baldwin and GER-EProb systems reached 60 % success probability. Differences are not statistically different according to a Chi-Square test ($Chi\text{-}Square = 0.952$, $p\text{-}Value = 0.32921$). The other probabilistic system (GER-KProb) performed badly in this static problem. Except for this system, these results are similar to the navigation problem, i.e. a standard GER performs better than other systems but GER implementing Baldwin also performs well. The only difference between GER-EProb and GER-KProb is the *temperature* parameter (T) in the expression to compute the probability value (see Equations (6) and (7) for details). This parameter depends on the learning step and the maximum number of learning steps and we use the same formulae to compute both in standard GER and GER-EProb. Temperature parameter is a way of balance between exploration and exploitation as we mentioned earlier and we think this is the answer to poor performance in GER-KProb. This system does not use the temperature parameter and it cannot take advantage of the exploration-exploitation trade-off as the other systems do. From a statistical point of view, differences between GER and GER-Eprob are not significant ($Chi\text{-}Square = 0.952$, $p\text{-}Value = 0.32921$) but differences between GER and GER-Kprob are significant ($Chi\text{-}Square = 7.2$, $p\text{-}Value = 0.00729$). Finally, differences between probabilistic systems (GER-EProb and GER-KProb) are not statistically significant according to a Chi-Square test ($Chi\text{-}Square = 3.333$, $p\text{-}Value = 0.06790$)

6.2.2 Average Standardized Fitness Related to Best Individuals

Regarding the average standardized fitness taking into account only the best individuals, Figure 11 shows results grouped by system.

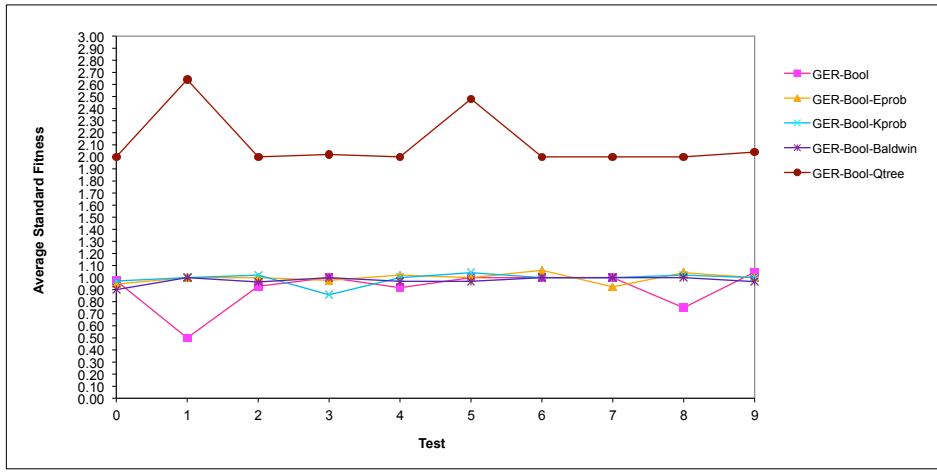

Fig. 11. Boolean Function Problem: Average standardized fitness

As can be seen in the figure, the worst system is also GER-QTree because it maintains a fitness far from the 0 value which is the optimum value (approximately around 2). Other systems show similar tendencies according to this criterion. If we compare this figure with Figure 6 we can see that average standardized fitness is more homogeneous for all the systems (excluding GER-QTRee) in a static problem than it was in a dynamic one. We think that this different behaviour can be explained in terms of noise and non-determinism. In a static problem the input and output values are not changed during the execution and the fitness evaluation process is always determined. This way, the same individual gets always the same fitness value each time it is evaluated. However, in a dynamic problem, the fitness evaluation process cannot be completely determined because the sensors and actuators can measure values with some error between different evaluations. This way, the same individual can get a slightly different fitness value each time it is evaluated. In graphical terms, deterministic problems show homogeneous values while non-deterministic problems show more irregularity.

6.2.3 Average Individual Learning

Figure 12 shows results group by system related to the individual learning criteria.

Results about learning reveal a similar behaviour both in the Boolean function problem and the navigation problem. Average individual learning is higher in GER-Baldwin as it was in the other case and we can argue the same opinion, i.e. Baldwin effect takes into account the fitness value but the genotype is kept as it originally was. This way, if the original genotype was not good the individual must learn again. Systems like GER, GER-EProb and GER-KProb perform basically equally. Around half of population learnt on each test (200–250 individuals) while in the

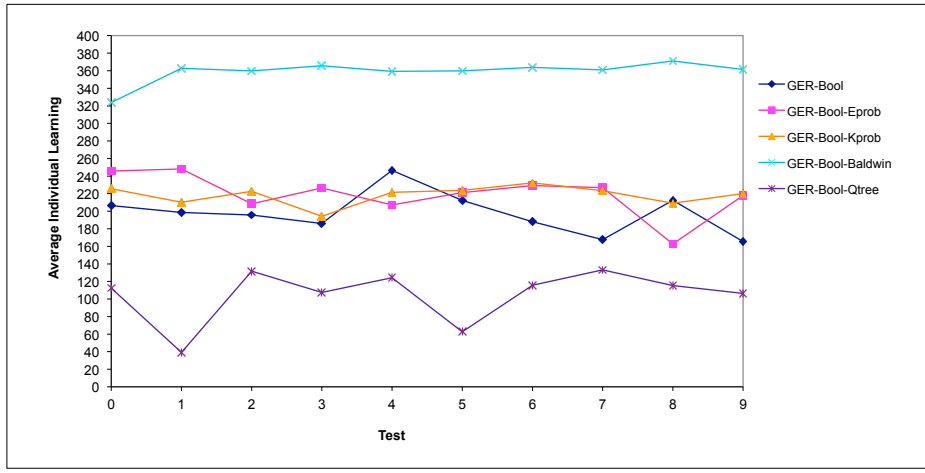

Fig. 12. Boolean Function Problem: Average individual learning

navigation problem this value was slightly higher (around 30–35 individuals learnt). Learning process is affected by the number of unique individuals in the population and by the quality of these individuals. Figure 9, related to the unique individuals in the navigation problem, showed that most individuals were unique during each execution. However, Figure 7, related to the learning process in the navigation problem, showed that most individuals had to learn during the execution. Both results reveal that there are a lot of unique individuals in the population but these individuals do not seem genetically good and they have to learn during their lifetime. In the Boolean function problem, there are fewer individuals who must learn but as we will see below, there also are fewer unique individuals in the population.

Finally, a slight difference is found in the GER-QTree system between the Boolean function problem and the navigation problem. Figure 12 shows a poor learning process for this system. Again, this fact confirms that GER-QTree does not learn in similar conditions to other systems. GER-QTree learning was similar to learning in other systems when we tested the navigation problem (see Figure 7 for details) but we think that the complexity of the problem could be the answer.

6.2.4 Average Genetically Correct Individuals

Figure 13 shows results grouped by system related to the number of genetically correct individuals.

As we can see in the figure the systems again obtain a similar behaviour except in GER-Baldwin and GER-QTree. The average correct genetically individual is close to 70 % and this value seems to indicate that learning is returned into genotype in the systems that implement this feature (GER, GER-EProb and GER-KProb). This way successive generations can take advantage of this circumstance and they can

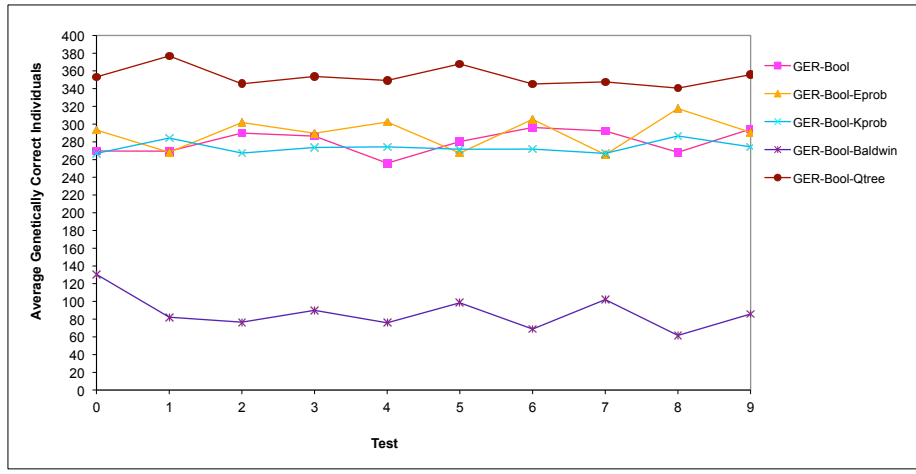

Fig. 13. Boolean Function Problem: Average genetically correct individuals

maintain individuals with an initially correct code. Navigation problem showed the same behaviour for this feature as well. GER-Baldwin system did not reach a result as good as in other systems because if the fitness is changed for an individual but its genotype remains unchanged, the individual might be selected and pass to the next generation. If the original genotype was grammatically incorrect, it will be equally incorrect in the next generation. Similar behaviour was also observable in the navigation problem. Curiously, GER-QTree exhibits more genetically correct individuals in this case than other systems and this value is also higher than the equivalent value for the navigation problem. A possible explanation could be that the Q-Tree in the navigation problem is simpler than it is in the Boolean problem. Besides, the general parameter seems to be unsatisfactory in this more complex problem in order to expand the Q-Tree. This way, Q-Tree in the Boolean problem manages less useful information with only a reduced number of analysed branches. As the Q-Tree is the force driving the process, in the Boolean problem most of the time is devoted to repeating the evaluation of similar individuals. This reduced set of individuals can be genetically correct as Figure 13 shows but they are far from the optimal solution as the other figures reveal.

6.2.5 Average Unique Individuals

We finish the Boolean function problem analysis with results for the unique number of individuals.

It can be seen in Figure 14 that the Boolean problem is equivalent to the navigation problem in the number of unique individuals. Again, a clear difference between systems that maintain an explicit learning (GER, GER-Eprob, GER-Kprob and GER-Baldwin) and the system with implicit learning (GER-QTree) is reflected.

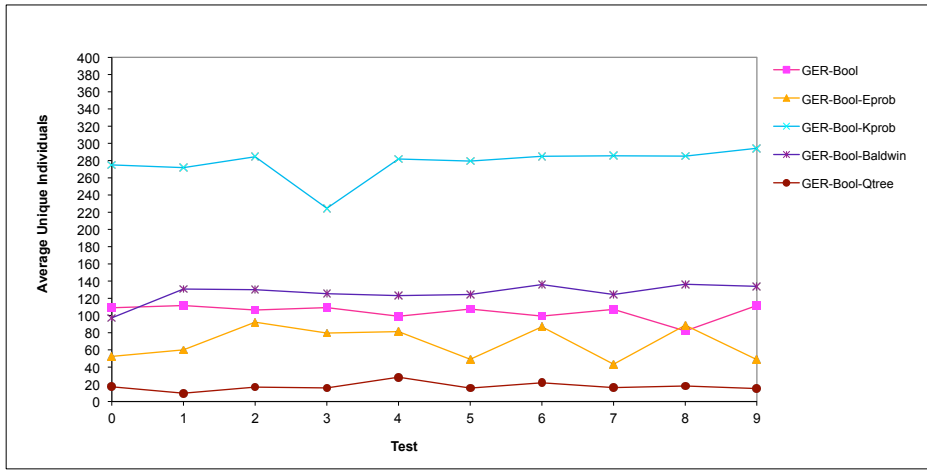

Fig. 14. Boolean Function Problem: Average unique individuals

The last system got lower results on this measure because it had fewer unique individuals. As we have mentioned before the general parameters (Table 3) might not be enough to check more paths in the Q-Tree and the GER-QTree can not test the branches extensively. In any case, the poor number of unique individuals in GER-QTree confirms what we supposed when we analysed the number of genetically correct individuals: in GER-QTree there are few unique individuals so the diversity of population is smaller. Besides, the individuals are not good and the Q-Tree can not guide the process toward a solution.

With regard to systems with explicit learning Figure 14 shows an evident difference between GER-KProb and other systems. This fact was also visible in the navigation problem (see Figure 9 to compare) but here the difference is greater. GER systems based on probabilities only differ in the way they compute the next production rule to apply during the mapping genotype-phenotype. GER-EProb uses a *temperature* parameter while GER-KProb does not use this parameter. A *temperature* parameter allows the system to control the exploration-exploitation balance. In the implemented systems this parameter depends on the number of learning steps as we described in Equations (4) and (8). Both GER and GER-EProb compute rules by means of this parameter and they usually try new rules around 50 % times and exploit known rules around 50 % times. However, in GER-KProb this proportion can vary. According to results that Figure 14 shows, in a deterministic or static problem, the number of unique individuals in the population is higher when using this type of probability than when using a probability based on *temperature*. Nevertheless, this fact does not imply a better performance in terms of success probability because, as Figure 10 shows, GER-KProb got worse results than GER and GER-EProb systems.

7 CONCLUSIONS

GER tries to enhance evolutionary systems with a learning process. It was developed in order to improve the task solving process by means of evolutionary search and learning. The learning process is a very important component in the system and it deserves to be analysed in depth. The aim of this work is to be a first approach to the subject and several systems have been developed to test different choices related to learning. We focus the study in two different domains: a navigation task for simulated robots which represent a non-deterministic problem and a Boolean function problem which represent a deterministic one. In order to analyse the learning process we define the learning scope according to three issues:

1. the Lamarckian mechanism to replace the original genotype,
2. the Baldwin Effect
3. the exploration-exploitation trade-off at the moment of choosing a production rule.

These aspects represent different ways the learning can be included in an evolutionary algorithm and they are studied in the context of a grammatical evolution. Besides, we are interested in analysing how each method can affect some parameters in the system such as the number of unique individuals, the average fitness value, the number of individuals genetically correct or the number of individuals that benefit from the learning process. In short, a detailed study about learning is important because it can help to select the most appropriate strategy to solve a problem in a specific domain by means of evolutionary techniques.

From a success probability viewpoint, a system based on standard GER reached the best result in both problems (90 % in the navigation problem and 80 % in the Boolean problem). This system implements the Lamarck hypothesis and uses an e-greedy exploration-exploitation mechanism. Another GER system without Lamarck but with possibility to change fitness (GER-Baldwin) also reached good results (78 % and 60 %, respectively) and this fact is according to the Baldwin effect as it was firstly proposed by Hinton and Nowlan in [2] and subsequently confirmed in other studies. In the Baldwin effect we can notice how the learnt behaviour can be transformed in a genetic behaviour in subsequent generations. Difference in success probability between GER and GER-Baldwin indicates a slightly better performance with Lamarck in both contexts (static and dynamic environment). However, these differences are not statistically significant according to a Chi-Square test. This is consistent with some studies where a Lamarck strategy outperforms a Baldwin strategy while in other studies the opposite was the case. Nevertheless, it is clear that the influence of Lamarck seems to be less important than it was supposed in [8] because a system with Baldwin gets competitive results as well. Regarding strategies for exploring and exploiting, the results show that e-greedy technique is more effective in both cases than probabilistic methods. On the other hand, it seem more useful to include a specific parameter to control the exploration-exploitation trade-

off (as the *temperature* parameter does) because GER-EProb performed better than GER-KProb in the deterministic problem. A higher performance for the probabilistic method vs. e-greedy method has not been theoretically proved yet [4] and it might be related with distance between best actions and the rest ones as it was proposed in [9].

In this work some of the most significant parameters that can impact system performance were analysed. Standardized fitness did not offer a lot of information because results were very similar for all systems that solve the problem (GER, GER-EProb, GER-KProb and GER-Baldwin). Anyway, it was useful for testing that systems that do not solve the problem presented average fitness values far from the solution (GER-QTRee).

Measures about average learning individuals offered similar results among systems, except in the system with Baldwin effect (GER-Baldwin) and the system with learning only via Q-Tree (GER-QTree). GER-Baldwin system reached higher values as to the average if it is compared with the other systems. As a possible explanation we can argue that in this system the learning process is only reverted to the individual via fitness while in a standard GER with Lamarck hypothesis the learning process is also reverted via genotype. In a Baldwinian system if the genotype is not modified to directly include the learnt behaviour it is more probable that the individual will have to learn again in the next generation and it is more probable that the individual will be bad genetically. Of course, this reasoning is applied specially to the selected individuals, i.e., the elitist individuals because other individuals will be affected by the evolutionary operators in any case. Regarding GER-QTree, the results in the Boolean function problem showed that this system can not learn appropriately although this system showed a good learning process in the navigation problem. We think the navigation problem is easier to solve and the Q-Tree could be more expanded than it was in the Boolean problem. It is worth to mention that all system were executed with the same general parameters. This fact is important because we want to compare systems performing under equal circumstances. In this sense, maybe GER-QTree could get better results if it was executed with higher values in the parameters. However, with a low configuration other systems perform better.

With regard to average correct genetically individuals, results are also homogeneous in systems with Lamarckian mechanism such as GER, GER-EProb and GER-KProb and they show differences in GER-Baldwin and GER-QTree. A system based on Baldwin maintains fewer genetically correct individuals than a system that replaces the original genotypes with the best learnt genotypes. According to Baldwin, the genotype is not replaced and the learning is reflected in the population in a long-term approach. We think this is the reason why in a reduced number of generations than we set in the tests, the Baldwin effect could be less notorious in the whole population. Results in both problems show how GER-Baldwin can maintain the most promising individuals between generations because they are able to learn during its lifetime even though its original genotype is not good. This way, GER-Baldwin performs reasonably well in deterministic and non-deterministic con-

texts. Curiously, GER-QTree shows a good level of genetically correct individuals although, as we commented before, this group of individuals is not good.

A measure about population diversity is very similar for systems with success as well (with the exception of GER-KProb in the deterministic case as mentioned). Systems that failed to find a solution (GER-QTree) have a poor genetic diversity and if learning is not possible or it is hard, the system could not drive the population toward a good solution. On the other hand, systems which find a solution do not need a great diversity in the population because either the learning or the evolution or both can guide the population to the solution.

To summarize, this work shows the learning process as an important item for improving evolution. Experimental results show a slightly better performance with Lamarck hypothesis than with Baldwin effect in two domains: deterministic and non-deterministic. Nevertheless, in the case of the navigation problem, the simplicity of the task can overcome some problems associated with Lamarck, i.e. the diversity of the population can fall down due to the fact that the same chromosome is frequently replicated because of its skill and the Lamarckian mechanism. In any case, the results show that GER can be a valuable alternative to solve some kind of specific problems. Future work may establish if this result escalates to much more complex problems, but this would need much more computing power, given the number of experiments to be carried out.

REFERENCES

[1] Collins, J. J.—Ryan, C.—O'Neill, M.: Grammatical Evolution: Evolving Programs for an Arbitrary Language. Lecture Notes in Computer Science 1391, Proceedings of the First European Workshop on Genetic Programming, Springer-Verlag 1998, pp. 83–95.

[2] Hinton, G. E.—Nowlan, S. J.: How Learning can Guide Evolution. Complex Systems, Vol. 1, 1987, pp. 495–502.

[3] Mingo, J. M.—Aler, R.: Grammatical Evolution Guided by Reinforcement. IEEE Congress on Evolutionary Computation 2007, pp. 1475–1482.

[4] Sutton, R. S.—Barto, A. G.: Reinforcement Learning: An Introduction. MIT Press 1998.

[5] Baldwin, M. J.: A New Factor in Evolution. The American Naturalist, Vol. 30, 1896, pp. 441–451.

[6] Lamarck, J. B.: On the Influence of the Environment on the Activities and Habits of These Living Bodies in Modifying Their Organization and Structure. Zoological Philosophy, MacMillan, London 1914, pp. 106–127.

[7] Moscato, P.—Cotta, C.: A Gentle Introduction to Memetic Algorithms. Handbook of Metaheuristics. Kluwer Academic Publishers 2003, pp. 105–144.

[8] Mingo, J. M.—Aler, R.: The Role of the Lamarck Hypothesis in the Grammatical Evolution Guided by Reinforcement. 6th International Workshop on Practical Applications of Agents and Multiagent Systems, Salamanca 2007, pp. 201–207.

[9] Kaelbling, L. P.—Littman, M. L.: Reinforcement Learning: A survey. Journal of Artificial Intelligence Research, Vol. 4, 1986, pp. 237–285.

[10] Ackley, D.—Littman, M.: Interactions Between Learning and Evolution. In C. G. Langton, C. Taylor, J. D. Farmer and S. Rasmussen (Eds.): Artificial Life II, Studies in the Sciences of Complexity, Vol X, Addison Wesley 1991, pp. 487–509.

[11] Belew, R. K.—McInerney, J.—Schraudolph, N. N.: Evolving Networks: Using the Genetic Algorithm with Connectionist Learning. Proceedings of the Second Artificial Life Conference, Addison-Wesley 1990, pp. 511–547.

[12] Whitley, D.—Gordon, V. S.—Mathias, K.: Lamarckian Evolution, the Baldwin Effect and Function Optimization. Lecture Notes in Computer Science, Vol. 866, Springer-Verlag 1994, pp. 6–15.

[13] Gruau, F.—Whitley, D.: Adding Learning to the Cellular Development of Neural Networks: Evolution and the Baldwin Effect. Evolutionary Computation, Vol. 1, 1993, pp. 213–233.

[14] Ku, K. W. C.—Mak, M. W.: Exploring the Effects of Lamarckian and Baldwinian Learning in Evolving Recurrent Neural Networks. In Proceedings of 1997 IEEE International Conference on Evolutionary Computation 1997, pp. 159–163.

[15] Julstrom, B. A.: Comparing Darwinian, Baldwinian and Lamarckian Search in a Genetic Algorithm for the 4-Cycle Problem. Congress on Evolutionary Computation, Late Breaking Paper in Genetic and Evolutionary Computation Conference, Orlando (USA) 1999, pp. 134–138.

[16] Nolfi, S.—Elman, J. L.—Parisi, D.: Learning and Evolution in Neural Networks. Adaptive Behaviour, Vol. 1, 1994, pp. 5–28.

[17] Nolfi, S.—Parisi, D.: Learning to Adapt to Changing Environments in Evolving Neural Networks. Adaptive Behaviour, Vol. 1, 1997, pp. 75–98.

[18] Mayley, G.: The Evolutionary Cost of Learning. From Animals to Animats. In Maes, P., Mataric, M. J., Meyer, J. A., Pollack, J. and Wilson, S. W. (Eds.), Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior, MIT Press 1996, pp. 458–467.

[19] Watkins, C.—Dayan, P.: Q-Learning. Machine Learning, Vol. 8, 1992, pp. 279–292.

[20] Moscato, P.—Cotta, C.: An Introduction to Memetic Algorithms. Revista Iberoamericana de Inteligencia Artificial, 2003, No. 19, pp. 131–148.

[21] Gallager, J. C.—Perreta, S.: WSU Kephera Robot Simulator. Availaible on: `http://carl.cs.wright.edu/reg/ksim/downloads/downloads.html`.

[22] Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press 1992.

[23] Callan, R.: Artificial Intelligence. Palgrave MacMillan 2003.

Jack Mario Mingo received his B. Sc. in informatics engineering in 1993 and his M. Sc. in informatics engineering from Universidad Politécnica de Madrid in 2002. Currently, he is working towards his Ph. D. at the same university. Since 1991 he has been working as programmer, analyst, project manager and databases specialist in several projects related to financial, logistics and telecom services. In parallel with these activities he is a Part-Time Professor at Technical School, Universidad Autónoma de Madrid. His current research interests include grammatical evolution, evolutionary computation and autonomous robots.

Ricardo Aler received his M. Sc. in informatics engineering from Universidad Politécnica de Madrid in 1992 and his Ph. D. degree from the same university in 1999. He received his M. Sc. in decision support systems from the University of Sunderland (UK) in 1993. His current research interests include genetic programming, evolutionary computation, machine learning and brain-computer interface. He has published extensively on these subjects and he has participated in both national and international projects. Currently, he is Associate Professor at Technical School, Universidad Carlos III de Madrid.

Darío Maravall received his M. Sc. in telecommunication engineering from Universidad Politécnica de Madrid in 1978 and his Ph. D. degree from the same university in 1980. Between 1980 and 1988 he was Associate Professor at School of Telecommunication Engineering, Universidad Politécnica de Madrid. In 1988 he was promoted to Full Professor at Faculty of Computer Science, Universidad Politécnica de Madrid. Between 2000 and 2004 he was the Director of the Department of Artificial Intelligence of the Faculty of Computer Science at Universidad Politécnica de Madrid. His current research interests include computer vision, autonomous robots and computational intelligence. He has published extensively on these subjects and has directed more than 20 funded projects, including a five-year R & D project for automated inspection of wooden pallets using computer vision techniques and robotic mechanisms, with several operating plants in a number of European countries (Spain, France, Italy and United Kingdom) and in USA. As a result of this project he holds a patent issued by the European Patent Office at The Hague, The Netherlands

Javier de Lope Asiaín received his M. Sc. in computer science from Universidad Politécnica de Madrid in 1994 and his Ph. D. degree from the same university in 1998. Currently, he is Associate Professor in the Department of Applied Intelligent Systems at Universidad Politécnica de Madrid. His current research interest is focused on study, design and construction of modular robots and multi-robot systems, and on development of control systems based on soft computing techniques. He is currently leading a three-year R & D project for developing industrial robotics mechanisms which follow the guidelines of multi-robot systems and reconfigurable robotics. In the past he also worked on projects related to computer-aided automatic driving by means of external cameras and range sensors and design and control of humanoid and flying robots.