Learning schemes for Genetic Programming

Anna I. Esparcia-Alcázar & Ken Sharman
Dept. of Electronics + Electrical Engineering
The University of Glasgow
Glasgow G12 8LT
Scotland, UK
anna@elec.gla.ac.uk, k.sharman@elec.gla.ac.uk

ABSTRACT

A learning capability is introduced in the Genetic Programming (GP) paradigm. This is achieved by enhancing GP with Simulated Annealing (SA), where the latter adapts the parameter values (in the form of node gains) in the structures evolved by the former. A special feature of this approach is that, due to the particularities of the representation used, it allows engineering problems (in which numerical parameters are important) to be addressed, thus extending the applicability of the GP paradigm.

We study two different learning schemes, which we refer to as Darwinian and Lamarckian according to whether the learned node gains are inherited or not. We compare the results obtained by these two techniques to those obtained in the absence of learning (both with node gain representation and standard GP representation). The results show the great interest of both learning schemes.

The application presented is a classical Digital Signal Processing problem: the equalisation of a noisy communications channel.

1. Introduction

After our first contacts with GP it became clear that although the method is very powerful in discovering structures (of functions or computer programs) it lacks the ability of successfully identifying the parameters associated to them.

It is then obvious that, in order to apply GP in engineering problems some modification must be made to the standard algorithm that will help handle numerical values.

This is fully accomplished by the introduction of node gains. A node gain is a parameter that multiplies the output value of a node. These gains are then adapted by a Simulated Annealing (SA) algorithm.

In an analogy taken from Nature, GP provides the evolution of a population of individuals (representing particular solutions to the problem at hand) whereas SA is the learning process of each specific individual.

The hybrid GP/SA was a major breakthrough in the sense that it pioneered the use of GP in engineering problems. Till its introduction (Sharman & Esparcia-Alcázar, 1993), GP had been mainly applied to benchmark problems such as the prisoner's dilemma, travelling salesman, game strategies, artificial ant, various XORs and multiplexers etc. which aim at either evolving a behaviour pattern or a Boolean expression and therefore do not require numerical parameters.

The work related in this paper aims at three objectives:

- reassessing the advantages of the representation using a node gains + learning scheme.
- studying the subtleties of the learning process, its importance and effects.
- determining in which way the learning is going to be carried out.

The outline of this paper is as follows. Section 2 provides a background on our work and the two techniques used. Section 3 deals with different issues of the representation using node gains. Section 4 is focused on the importance of learning and the various aspects of it that have been addressed. In section 5 we present a selection of results, which are analysed in section 6. Section 7 adds further comments and outlines future areas of research and finally in section 8 we present some conclusions.

2. Background on Genetic Programming and Simulated Annealing

Genetic Programming (GP) is a subclass of the standard Genetic Algorithm (GA) which was introduced in the early 90s (Koza, 92). GP is a general purpose search algorithm whose aim is obtaining functions or executable computer programmes. This is in contrast to the standard GA whose aim is to evolve data (usually in the form of strings of bits or other symbols). GP emulates natural evolution and survival of the fittest in artificial populations. It employs a population of individuals, each representing a possible solution to the underlying task in the form of an expression tree. The performance of these trees is rated by applying a problem dependent evaluation function and some user-defined criterion, which assign a fitness value to each prospective solution. These fitness values guide the evolution of the population, which takes place by creating new individuals by means of genetic operators such as crossover and mutation.

Simulated Annealing (SA) is a stochastic searching strategy based on an analogy to the annealing process in statistical mechanics (i.e. the behaviour of systems with many degrees of freedom in thermal equilibrium at a finite temperature). SA is applied to the parameters of each tree, which have the form of gain values that are associated to each node (or element) of the tree.

GP is a global search strategy, whereas SA is a semi-global one. Both have the characteristics of being able to escape from local optima and of not requiring gradient information to guide the search.

The philosophy of GP is inspired by natural evolution, which is an inherently random process with no other restrictions than the ones given by the environment. This means that GP is controlled by a fairly simple algorithm. We consider that any arbitrary restrictions are undesirable, as they represent a form of "divine intervention". Therefore, it is an objective to try and keep the number of spurious parameters down to the minimum possible, and let evolution take care of everything.

On the other hand, having equated SA to learning leaves us with an open field of possibilities to explore. There are a number of circumstances that can affect the learning process of an individual and which can be translated by analogy into parameters that control the SA algorithm.

A more in depth analysis of GP can be found in Koza, (1992) and of SA in Szu & Hartley (1987). The hybridisation of GP and SA for application in Digital Signal Processing (DSP) is explained in Sharman & Esparcia-Alcázar (1993), Sharman, Esparcia-Alcázar & Li (1995), Esparcia-Alcázar & Sharman (1996a), Esparcia-Alcázar & Sharman (1996b) and Esparcia-Alcázar (1997).

3. Representation using node gains

3.1 Node vector and gain vector

The introduction of node gains is one the main differences between this system and standard GP.

An individual, which encodes a possible solution to the problem at hand, is represented by two vectors: a vector of nodes, $\vec{n}$ and a vector of gains, $\vec{g}$.

$$\vec{n} = \{n_0 \;\; n_1 \;\; \cdots \;\; n_{l-1}\}$$

$$\vec{g} = \{g_0 \;\; g_1 \;\; \cdots \;\; g_{l-1}\} \qquad (1)$$

where:

$l \in \aleph$ is the length of the tree,

$n_i : \Re^{\lambda(n)} \Rightarrow \Re, \quad n_i \in \mathbf{O}; \qquad i = 0 .. l\text{-}1$

$g_i \in \Re, \qquad i = 0 .. l\text{-}1$

$\lambda(n)$ is the number of arguments (or *arity*) of n_i

O is a set of allowed operations

O can be represented as the union of two sets:

$$\mathbf{O} = \mathbf{O}_F \cup \mathbf{O}_T \qquad (2)$$

which are usually referred to as function set and terminal set, respectively. The terminal set is characterised by

$$\lambda(n) = 0 \qquad \forall\, n \in \mathbf{O}_T \qquad (3)$$

and the function set by

$$\lambda(n) > 0 \qquad \forall\, n \in \mathbf{O}_F \qquad (4)$$

For DSP we use the following sets:

$\mathbf{O}_F = \{$`+,-,*,/,*2,/2,+1,-1,nlN,psh,Z`$\}$

$\mathbf{O}_T = \{$`1, cN, xN, yN, stkN`$\}$

where $N \in \aleph$ is an index number. These operations are described in Sharman, Esparcia-Alcázar & Li (1995) and Esparcia-Alcázar & Sharman (1996a).

A restriction on $\vec{n}$ is that it must be a syntactically correct tree. The two necessary and sufficient conditions for syntactical correctness are

$$\sum_{i=0}^{l-1} \lambda(i) = l - 1 \qquad (5)$$

$$\sum_{j=0}^{i-1} \lambda(j) \geq i \qquad \forall i \;\; 0 < i < l \qquad (6)$$

In standard GP it is usual to represent $\vec{n}$ as an expression in polish (prefix) notation. In the GP/SA system, this can be done in a compact way including $\vec{g}$ as well.

For instance, an individual represented by the vectors:

$$\vec{n} = \{\texttt{+, X0, Y1}\} \qquad (7)$$

$$\vec{g} = \{\texttt{0.5, 2.0, -1.4}\} \qquad (8)$$

would be written in polish notation as

```
( [0.5] + ( [2.0] X0  [-1.4] Y1 ) )
```

3.2 Crossover

Let T_1 and T_2 be two syntactically correct trees selected for crossover, with lengths l and m respectively, and whose expressions are:

$$T_1 \equiv \begin{cases} \vec{n}_1 = \{n_{1,0} & n_{1,1} & \cdots & n_{1,(l-2)} & n_{1,(l-1)}\} \\ \vec{g}_1 = \{g_{1,0} & g_{1,1} & \cdots & g_{1,(l-2)} & g_{1,(l-1)}\} \end{cases}$$

$$T_2 \equiv \begin{cases} \vec{n}_2 = \{n_{2,0} & n_{2,1} & \cdots & n_{2,(m-2)} & n_{2,(m-1)}\} \\ \vec{g}_2 = \{g_{2,0} & g_{2,1} & \cdots & g_{2,(m-2)} & g_{2,(m-1)}\} \end{cases}$$

Let us assume that T_1 is acting as the "mother" and T_2 as the "father"; this means T_1 provides the root and T_2 the branch to be inserted. Further assume that the crossover points are i, $0 \le i < l$, for T_1 and j, $0 \le j < m$, for T_2.

Let the subtrees starting at i and j comprehend all the nodes up to p and q in their respective trees.

The result of the crossover, T_{1x2}, is a tree of length $l - p + q$, whose expression is:

$T_{1x2} \equiv$

$$\begin{cases} \vec{n}_{1x2} = \{n_{1,0} & \cdots & n_{1,(i-1)} & n_{2,j} & \cdots & n_{2,(j+q-1)} & n_{1,(i+p)} \cdots & n_{1,(l-1)}\} \\ \vec{g}_{1x2} = \{g_{1,0} & \cdots & g_{1,(i-1)} & g_{2,j} & \cdots & g_{2,(j+q-1)} & g_{1,(i+p)} \cdots & g_{1,(l-1)}\} \end{cases}$$

The gain vector $\vec{g}_{1x2}$ has components inherited from T_1 and T_2. In an alternative scheme, the components would be set to 1 or initialised with random values. This distinction is the basis of the Lamarckian and Darwinian learning schemes, that will be dealt with later.

3.3 Importance of node gains

The interest of using node gains can be understood better using the tree in equation (7) as an example. This is depicted in Figure 1, where the gains are represented as weights associated to the links between nodes.

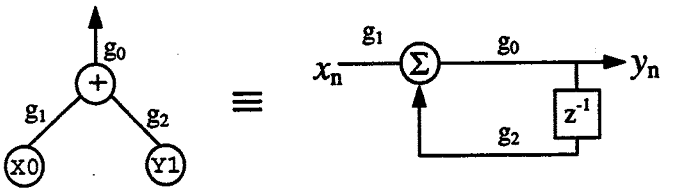


Figure 1: A tree and the system it represents using node gains.

The system equation is :

$$y_n = g_0 \cdot (g_1 \cdot x_n + g_2 \cdot y_{n-1}) \qquad (9)$$

The tree depicted in Figure 1 represents the equation of a first order system, where X0 is the current input and Y1 is the output in the previous instant (see references). When no gains are used, i.e. $g_i = 1$, the system is unstable, as it has a pole on the unit circle. On the other hand, with a proper set of gains, such as the ones given by equation (8), the system is stable.

It can be argued that the same can be achieved using constant nodes, as shown in Figure 2.

The system equation for this tree is:

$$y_n = x_n + C0 \cdot y_{n-1} \qquad (10)$$

With an appropriate value of C0, e.g. C0 = -0.7, the system shown in Figure 2 is stable and equivalent to the one in Figure 1, but the first thing that can be noticed is that the size of the tree has increased. Also, by adding one constant we have only achieved to stabilise the system. If we were interested in the other parameters present in the previous equation, further constants (and multiplication nodes) should be introduced.

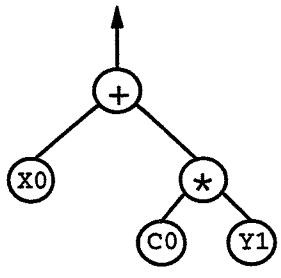


Figure 2: A similar system, without node gains

This leads us to an important characteristic of the GP/SA method: the use of node gains means that the same system can be represented by shorter, more compact trees. This "compacting effect" is going to affect the performance of the method in two ways:

- shorter subtrees are less susceptible of disruption by crossover
- handling shorter trees is less computer intensive.

4. Learning

4.1 Characteristics of learning

Two characteristics of learning in natural systems are going to be of interest for us (Anderson, 1995). The first characteristic is that learning slows down the reproduction (crossover) rate, because time devoted to learning cannot be spent in reproduction. In a population that learns, the production of a given number of individuals can take between tens to hundreds of times more than when no learning is performed. This can be a drawback in problems where the value of the parameters doesn't have much influence in the output.

Learning can also diminish the differences in fitness between individuals, which helps preserve genetic diversity in the population. This means that the whole population won't converge to a particular structure that happens to be good under a certain environment. This characteristic is of great interest in environments that are likely to experiment changes.

4.2 Need for learning

The tree in Figure 1 has more degrees of freedom than the one in Figure 2 because there are three gain values that need to be adjusted, instead of only one constant. This implies that there will be many combinations of gains that represent good solutions to the problem at hand, but also that there will be many that will represent unstable systems or simply bad solutions. This is the reason why learning is needed and here is where Simulated Annealing comes in. As stated before, what we refer to as learning is the adaptation of the node gains by SA.

Taking into account the characteristics mentioned above, we can give two general rules as to when to apply a learning scheme:

- when the performance of a system is sensitive to parameter variations; for instance, when a wrong selection of parameters can make the system unstable (as seen in Section 3).
- when environmental changes are expected; for instance in equalisation in mobile systems, where the unknown channel is constantly varying.

4.3 Learning by Simulated Annealing

The reasons why we chose SA as a learning algorithm are twofold. On the computational front, due to the simplicity of implementation. On the philosophical front, due to the similarity to the learning process in nature, in two ways. First, as an individual undergoes learning, the probability of a big gain jump decreases. This is in accordance with what we observe in nature: the amount that an individual can learn decreases in time. Second, big decreases in fitness are mainly restricted to the early stages of the learning process. Pursuing the natural analogy, when the individual is young it can still accept a decrease in "status" but that is less likely as it grows older.

The SA algorithm works as follows:

1. Perturb $\vec{g}(i)$ to get $\vec{g}(i)'$
2. Evaluate the fitness, $f(i)'$ using the perturbed gain vector $\vec{g}'(i)$
3. If $(f(i)' \geq f(i))$ then accept the perturbation: $\vec{g}(i+1) = \vec{g}(i)'$ and continue
   Else accept the perturbation with probability $\left(1+e^{\frac{f(i)-f'(i)}{T}}\right)^{-1}$ and continue
4. Reduce the temperature T according to an annealing schedule and go back to step 1

4.4 Darwinian and Lamarckian learning schemes

In standard GP evolution follows Darwinian rules. Darwinian evolution is a two-step process, taking place as follows,

1. random genetic variations take place, caused by recombination and mutation only
2. "artificial selection" favours the survival of the fittest among these variants.

The former implies that individual learning does not affect the genetic material and therefore cannot be inherited.

On the other hand, another classical theory of evolution, Lamarckism, is essentially a theory of directed variation. In the face of an environmental change, an organism would react by incorporating preferentially favourable genetic information, which would then be transmitted to offspring. The latter, also known as "inheritance of acquired characters", has taken over the meaning of the word Lamarckism, and it is by this definition that we will use it.

Although Lamarckian evolution (in any of its meanings) has not been observed in biological history[1], it can be said that the evolution of human culture (or learning in higher mammals) is Lamarckian in character: knowledge is transmitted from one generation to another.

We implement Lamarckian evolution in the GP/SA system by allowing the annealed node gains to be inherited by the offspring, as seen in section 3.2. In Darwinian evolution, the gains are set to 1 or initialised with random values before the annealing takes place.

5. Results

5.1 The four methods

We will be comparing four methods: three of them use node gains and the fourth uses no gains (i.e. the gains are equal to 1). The latter we refer to as NGNL (no gains - no learning).

In the first three methods the gains are initialised at random in the first population. In the Darwinian learning scheme the gains are also initialised at random for every individual that is born and then subjected to annealing. In Lamarckian learning the annealed gains of the parents are inherited by the offspring, which then undergo their own annealing process. Finally, in the last method we are testing the gains are part of the structure. They remain fixed (i.e. there is no learning) except in case of a random mutation, and thus are inherited by the offspring. We refer to this as RGNL (random gains - no learning).

A further difference between the methods is the number of constant nodes that can be employed. All methods will employ a given number *n* (here, *n* =7) of predefined constants. In NGNL there will also be a further N-*n* constants that will be initialised with random values[2]. This is to compensate for the absence of node gains.

The comparison is based on a DSP application: the channel equalisation problem. This is fully explained in the references (Sharman, Esparcia-Alcázar & Li, 1995 and Esparcia-Alcázar & Sharman, 1996b)

[1] Although genetic changes can be due to exposure to radiation or chemical agents, these changes are random, not directed.

[2] This leaves another open issue, namely determining the *range* of the random constants (and also their number, N)

5.2 Fixed environments

5.2.1 Overview

Strictly speaking a fixed environment would be one in which the population has reached equilibrium, i.e. the average fitness remains approximately constant. When tackling DSP problems, however, we are not usually concerned about what happens after an optimum has been reached, but rather about reaching it.

In the particular case of the experiments related here we are interested in both the solutions and the evolutionary process itself. We will then define as a fixed environment a GP run of the channel equalisation problem in which the unknown channel does not change for the duration of the run. The run proceeds for up to a given number of node evaluations (as explained below) regardless of whether or not a suitable solution has been found or equilibrium has been reached.

Two cases are studied: a linear channel (LC1) and a nonlinear one (NLC).

5.2.2 Study of the performance

We can divide the study of the performance into two aspects. The first one is related to how well the solutions obtained perform with unseen data, i.e. the generalisation ability.

It is a matter of discussion among the GP community whether or not the existence of local learning schemes decreases the generalisation ability (and therefore the quality) of the potential solutions. Some argue that learning will cause overfitting of the training data and therefore, when the data is changed, the performance will be poor because the solution was biased towards the training data.

We are willing to see whether or not this is the case with our learning schemes. To do so, we run each experiment a number of times using 70 samples (of which the first 20 are rejected for the fitness calculation as transient). The termination criterion for evolution is that the number of node evaluations equals or exceeds a given limit (1e8 for LC1, 3e8 for NLC). The solutions are tested with a further 10100 samples (of which the first 100 are rejected) to obtain a fitness value and a bit-error-rate. These two values themselves provide a measure of "how good" the solutions are.

We also compare the fitness in the test with the one obtained during evolution and measure the discrepancy as follows:

$$d = \frac{1}{No.OfRuns} \sum_{i=1}^{No.OfRuns} (f_{test_i} - f_{evol_i})^2 \qquad (11)$$

This gives an idea of how well the fitness during evolution can predict the subsequent behaviour of the solutions obtained by each method, therefore being a measure of reliability.

Table 1 and Table 2 give the averages of these values.

These results show that, in average, Darwinian learning outperforms the other methods, the differences being more noticeable in the case of NLC, which is a more difficult problem. On the other hand, the discrepancy in LC1 is lower for Lamarckian learning, whereas in NLC the lowest value corresponds to Darwinian learning; both results show that learning methods generalise better than non learning ones.

Table 1: Comparison of results (50 runs for LC1)

	fitness (in test)	**BER** (in test)	**fitness** (after evolution)	***d***
Darwinian	0.9765	0	0.9825	0.0018
Lamarckian	0.9679	0.00806	0.9704	0.0002
RGNL	0.9731	0.00006	0.9890	0.0037
NGNL	0.8231	0.03045	0.8933	0.0487

Table 2: Comparison of results (24 runs for NLC)

	fitness (in test)	**BER** (in test)	**fitness** (after evolution)	***d***
Darwinian	0.9620	0.00564	0.9816	0.0013
Lamarckian	0.9261	0.01071	0.9620	0.0052
RGNL	0.7418	0.02920	0.9655	0.2090
NGNL	-	-	-	-

The second aspect in measuring the performance is the *success rate*. This is given by the ratio of successful runs over the total number of runs. We define a successful run as one in which the fitness of the best solution measured during the evolution is greater than 0.9. The values are shown in Table 3.

Table 3: Success rates for LC1 and NLC

	LC1 (50 runs)	**NLC** (24 runs)
Darwinian	1	0.92
Lamarckian	0.96	0.83
RGNL	1	0.92
NGNL	0.78	-

Darwinian learning maintains the advantage because it has both high success rates and low discrepancies. RGNL has higher success rates than Lamarckian learning, but on the other hand, as shown in the previous tables, it also has a greater discrepancy, which makes its solutions less reliable.

5.3 Variable environments

5.3.1 Overview

A variable environment is one in which the unknown channel is modified during the run.

The implementation of this is as follows. Evolution proceeds as explained in the previous section for LC1, up to 1e8 node evaluations approximately. Then a new set of data is generated for the modified channel LC2. LC2 has the same structure as LC1 but one of its coefficients is slightly different in absolute value and has opposite sign.

The population is then re-evaluated and re-annealed and evolution continues for approximately up to 3e8 node evaluations.

5.3.2 Study of the performance

The solutions are tested as explained before with 10100 samples generated for the channel LC2. A comparison of the solution performances is given in Table 4.

Table 4: Comparison of results in a variable environment. Averages of 51 runs for LC1 → LC2

	fitness (in test)	BER (in test)	fitness (after evolution)	*d*
Darwinian	0.9078	0.002047	0.9592	0.0030
Lamarckian	0.8930	0.001986	0.9606	0.0208
RGNL	0.8296	0.038525	0.9146	0.0313
NGNL	0.6062	0.195249	0.7719	0.1242

Darwinian learning maintains the advantage, both in fitness and reliability, clearly over RGNL and NGNL and slightly over Lamarckian learning.

If we now study the success rates, given in Table 5, it turns out that the Darwinian learning scheme has a slightly lower probability of success than Lamarckian learning. Nevertheless, the performance of the solutions obtained by this method is higher due to the low discrepancy.

Table 5: Success rates (51 runs for LC1 → LC2)

Darwinian	0.9608
Lamarckian	0.9804
RGNL	0.8431
NGNL	0.5882

6. Analysis

6.1 Statistical comparison: Mann-Whitney test for two independent samples

We are not satisfied with the conclusions drawn in the previous section because the fitness distributions are unknown. Therefore, we can't be confident that the comparison of average values we have performed is sufficient. In this section we analyse the results statistically in order to attain confidence in these conclusions.

For this purpose we employ a Mann-Whitney test to compare the methods two by two. The details of the test are not given here (Conover, 1980); it is enough to know that the test aims at proving the hypothesis (H_0) that two given samples come from the same population (or the alternative hypothesis that they don't).

For a certain level of significance, α, H_0 is rejected if the test statistic T lays outside the critical interval $[w_{\alpha/2}, w_{1-\alpha/2}]$, where w_p is the p^{th} quantile of T. If $T \in [w_{\alpha/2}, w_{1-\alpha/2}]$ then H_0 is accepted.

The quantiles may be approximated by

$$w_p = n \cdot \frac{(N+1)}{2} + x_p \sqrt{n \cdot m \cdot \frac{(N+1)}{12}}$$

$$w_{1-p} = n \cdot (N+1) - w_p$$

where x_p is the p^{th} quantile of a standard normal random variable.

Accepting H_0 implies that the two methods compared don't have a significant difference in performance. Rejecting H_0 means that the method with higher T has a better performance.

6.1.1 *Mann-Whitney test for* LC1

We choose $\alpha = 0.05$ (i.e. a confidence of 95%) and looking up in a standard normal table we get $x_{0.025} = -1.96$. Since 50 runs were performed for all methods, $n = m = 50$, the quantiles are:

$$w_{0.025} = 50 \cdot \frac{101}{2} - 1.96 \cdot \sqrt{50^2 \cdot \frac{101}{12}} = 2240.68$$

$$w_{1-0.025} = 50 \cdot 101 - w_{0.025} = 2809.31$$

Therefore the critical interval is [2240.68 , 2809.31].
The results for this test are shown in Table 6
The conclusion that can be drawn from this test is that the Darwinian, Lamarckian and RGNL schemes are indistinguishable, and all of them outperform NGNL Therefore we can conclude that, in this example, using node gains is better than not using them, but nothing can be concluded about learning.

Table 6: Mann-Whitney test for LC1. Critical interval: [2240.68, 2809.31]

Comparison (1 / 2)	T1	T2	Result
Darwin / Lamarck	2505.5	2544.5	Accept H_0
Darwinian / RGNL	2247	2803	Accept H_0
Lamarckian / RGNL	2267	2783	Accept H_0
Darwinian / NGNL	3307	1743	Reject H_0
Lamarckian / NGNL	3214	1836	Reject H_0
RGNL / NGNL	3309	1741	Reject H_0

6.1.2 Mann-Whitney test for NLC

We'll consider 24 runs for all cases, so $n = m = 24$ and the critical interval is [492.95 , 683.05]. The results of the test are shown in Table 7.

Table 7: Mann-Whitney test for NLC. Critical interval [492.95, 683.05]

Comparison (1 / 2)	T1	T2	Result
Darwin / Lamarck	658	518	Accept H_0
Darwinian / RGNL	724	452	Reject H_0
Lamarckian / RGNL	673	503	Accept H_0
Darwinian / NGNL	763.5	412.5	Reject H_0
Lamarckian / NGNL	717	459	Reject H_0
RGNL / NGNL	633	543	Accept H_0

These results present us with a paradox. The Darwinian and Lamarckian schemes are indistinguishable, Darwinian outperforms RGNL but Lamarckian is indistinguishable from RGNL. We attribute this to the fact that the formula for calculation of the quantiles is not exact, but rather an approximation. However, the numerical values of the statistic T in the Lamarckian/RGNL comparison seem to indicate that Lamarckian learning performs *slightly* better than RGNL. This conclusion would be supported by the fact that the fourth method, NGNL, is outperformed by both the Darwinian and Lamarckian learning schemes, being at the same time indistinguishable from RGNL.

6.1.3 *Mann-Whitney test for* LC1→ LC2

In this case, for all methods $n = m = 51$ and the critical interval is [2333.64 , 2919.36]. The results for this test are shown in Table 8.

Table 8: Mann-Whitney test for LC1→LC2. Critical interval: [2333.64, 2919.39]

Comparison (1 / 2)	T1	T2	Result
Darwin / Lamarck	2601	2652	Accept H_0
Darwinian / RGNL	3035	2218	Reject H_0
Lamarckian / RGNL	3029	2224	Reject H_0
Darwinian / NGNL	3570	1683	Reject H_0
Lamarckian / NGNL	3559	1694	Reject H_0
RGNL / NGNL	3233	1953	Reject H_0

The conclusion that can be drawn from this test is that the Darwinian and Lamarckian schemes are indistinguishable, both of them outperform RGNL and NGNL, and RGNL outperforms NGNL.

6.2 Darwinian vs. Lamarckian learning

The analysis in the previous section didn't allow us to determine whether one type of learning is preferable to the other. Here we try to outline the differences between the two.

We define as *a point in the gain space* a particular vector of gain values associated to a tree. Given an initial point we define as a *learning path* the set of points obtained during the annealing process concluding in a certain final point. Because in Lamarckian learning the gain values are inherited, the annealing process tends to exhaust a particular path until an optimum point is reached. The process is in many cases irreversible and can end in a "blind alley" from which no better points can be reached. This behaviour results in optima that can be reached very quickly but that are useless if they happen to be local optima instead of global ones.

The Lamarckian learning scheme explores the gain space intensively: when a "right path" is found, Lamarckian-evolved solutions will give a higher fitness. Otherwise, they will get stuck at a point which, in spite of being good, is not the best.

On the other hand, the Darwinian learning scheme explores the gain space in an extensive way. As evolution proceeds the population might converge structurally but there remains parametrical diversity due to the random initialisation of the starting points for annealing. This means that the gain space can be explored more exhaustively; if a particular structure takes over the population is because it has a high fitness with many different gain vectors.

Intuitively: Lamarckian learning can be faster but Darwinian tends to be more robust.

6.3 Influence of learning on evolution.

To trace the influence of learning on evolution we introduce a new variable: the fitness at birth or ***fab***. We will compare the statistics of the ***fab*** and the fitness after learning, or ***fit***. With no learning, ***fit*** = ***fab***. We are interested in the variations in the distributions of ***fit*** and ***fab*** as the run proceeds.

For this study 24 runs are performed for Darwinian, and Lamarckian learning and RGNL with the same set up as in section 5.2. In this case we are not interested in the solution but in the evolutionary process itself and therefore the experiments run for a longer time. (In practise the termination criterion for evolution is a number of node evaluations greater than or equal to 3e8).

The distributions for LC1 are shown in Figure 3 and Figure 4. (The distributions for NLC showed similar results and are not displayed to avoid repetition).

It can be seen that the ***fit*** distribution moves to the right as the run proceeds. This displacement is slowest in Darwinian learning and fastest in RGNL. In more difficult problems we have observed that the ***fit*** distribution in RGNL doesn't reach the higher values of the scale. Instead, it gets "stuck" at suboptimal values.

The ***fab*** distribution shows an increasing peak at zero in Darwinian learning. This means that individuals that can learn are selectively preferred to those that are naturally fit. The opposite occurs in Lamarckian learning: the ***fab*** distributions are displaced towards the right, as would be expected.

In both cases the displacement of the ***fab*** distributions is much slower than that of the ***fit*** for RGNL (remember that for RGNL ***fit*** = ***fab***). This shows how the presence of learning slows down the evolution of the genotype.

6.3.1 The Baldwin effect

The Baldwin effect is a mechanism by means of which learned behaviour and characteristics at the level of *individuals* can significantly affect evolution at the level of the *species* (French & Messinger, Hinton & Nowlan 1987). It can be considered as Nature's way of achieving something similar to Lamarckian evolution but without its potential drawbacks.

Any discussion about the Baldwin effect is only meaningful in the case of Darwinian learning. In the presence of Baldwin effect we would expect that the evolution of the ***fab*** distribution would follow the ***fit*** one. This is not observed in the experiments presented here; thus nothing would seem to indicate the existence of this effect.

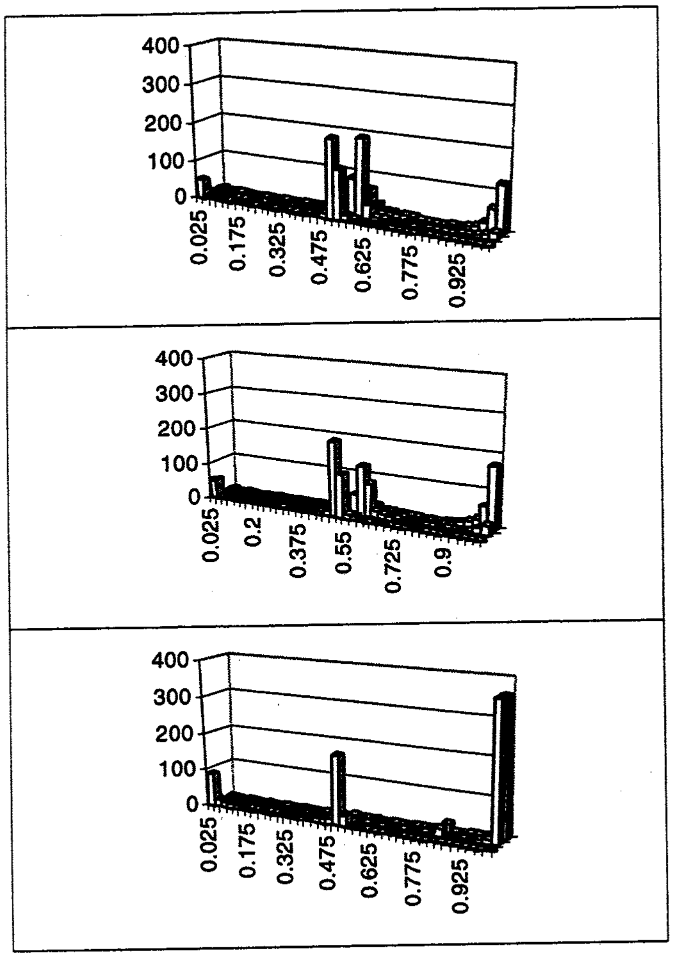


Figure 3: Evolution of the *fit* for Darwinian (top), Lamarckian (middle) and RGNL (bottom). Averaged histograms for initial, middle and final stages of the run.

One reason for this is that there is no penalty associated to learning, as would be in a real world example. In our runs all individuals learn "for free" therefore there isn't any selective pressure favouring a high fitness at birth. In fact what happens is quite the contrary: good learners are preferred to natural born fit individuals.

The other reason is that the *fab* does not really measure the fitness of the structure, which is what is affected by evolution (and therefore by the Baldwin effect). The gains are also involved and by initialising them at random we are introducing a discontinuity in the fitness (which is then smoothed by the learning). This indicates that the *fab* is not a proper measure to track the Baldwin effect and that alternative measures should be investigated.

7. Further comments: Extending the analogy

The performance of the Darwinian and Lamarckian learning schemes is influenced by the characteristics of the experiments that have been related here, in particular:

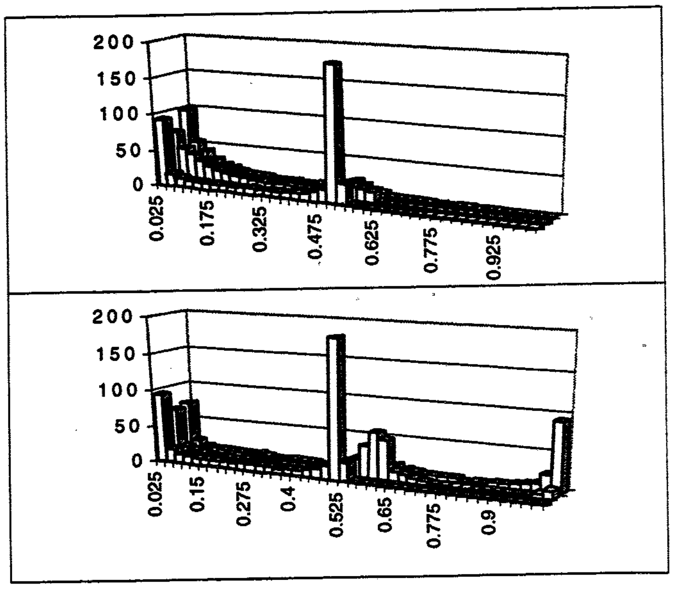


Figure 4: Evolution of the *fab* for Darwinian (top) and Lamarckian (bottom) learning. Averaged histograms for initial, middle and final stages of the run.

- all individuals have the same probability of learning (equal to 1)
- the maximum number of annealing iterations is fixed
- in the experiment with varying environments, the unknown channel is modified only once.

In a "real world" situation, these characteristics wouldn't apply (which also accounts for the absence of Baldwin effect).

Furthermore, the conclusions have been drawn from a small set of problems. A wider range of experiments should be necessary to increase the confidence in these conclusions.

Further work will aim at resolving these issues.

8. Conclusions

We have addressed here two ways of implementing an SA learning scheme in GP and their characteristics. We have shown that, in the examples presented here, using node gains provides better results than using random constant nodes only. Also, the use of learning improved the performance in the more complex problem addressed.

We have introduced a measure of the generalisation ability which allows us to show that overfitting is not a problem in GP/SA; on the contrary, the two learning schemes generalise better than the non learning ones. This is due to the fact that learning efficiently explores the gain space and as a result the structures that dominate a population are the ones with a high fitness over a number of different gain vectors. This is indeed a paradox: learning means the actual values of the gains have little influence on the performance, which accounts for greater robustness when dealing with noisy environments. No learning implies that the gain values are going to be 1 (or random), and that is a bias like any other that learning would introduce, with the difference that it cannot be overcome.

The differences in the behaviour of the two learning schemes used (Darwinian and Lamarckian) have been pointed out and, although no conclusion has been reached as to which performs better, the Darwinian scheme looks intuitively more indicated for a variety of problems, as the more robust of the two.

Bibliography

R.W. Anderson, "Learning and Evolution: A Quantitative Genetics Approach", *Journal of Theoretical Biology, no. 175,* 1995.

W.J. Conover, "Practical Nonparametric Statistics", 2nd ed. John Wiley & sons, 1980.

A.I. Esparcia-Alcázar & K.C. Sharman, "Evolving Recurrent Neural Network Architectures by Genetic Programming", presented at the First International Conference on Genetic Programming, GP'96, Stanford University, USA, July 1996.

A.I. Esparcia-Alcázar & K.C. Sharman, "Some Applications of Genetic Programming in Digital Signal Processing", *Late Breaking Papers at the GP'96 conference*, Stanford University, USA, July 1996.

A.I. Esparcia-Alcázar "An investigation into a Genetic Programming technique for adaptive Signal Processing", *Second Year Report*, Department of Electronics and Electrical Engineering, Glasgow University, Jan. 1997.

R.M. French & A. Messinger, "Genes, Phenes and the Baldwin Effect: Learning and Evolution in a Simulated Population", *Artificial Life IV.*

S.J. Gould, "The panda's thumb", W.W. Norton & company, 1980

G.E. Hinton & S.J. Nowlan, "How learning can guide evolution", *Complex Systems, vol. 1,* 1987

J. Koza, "Genetic Programming: On the programming of computers by means of natural selection". The MIT Press, 1992

B. Mulgrew, "Applying Radial Basis Functions" *IEEE Signal Processing Magazine*, Vol. 13 No. 2, pp. 50-65, March 1996.

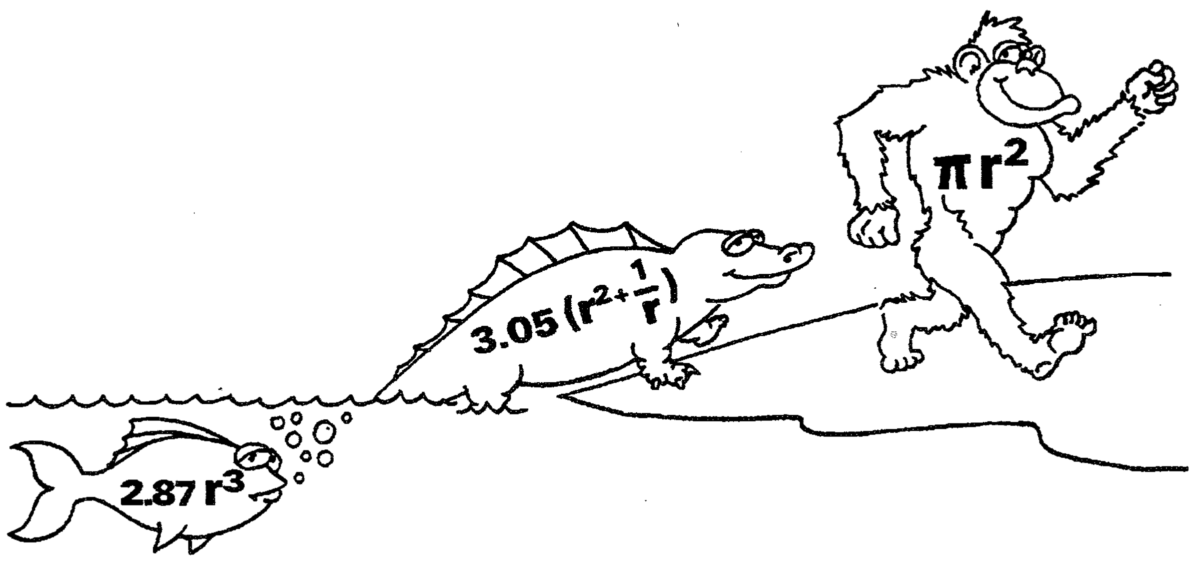
K.C. Sharman & A.I. Esparcia-Alcázar, "Genetic Evolution of Symbolic Signal Models". *Procs. of the 2nd IEE/IEEE Workshop on Natural Algorithms in Signal Processing,* 1993.

K.C. Sharman & A.I. Esparcia-Alcázar, "Evolving Signal Processing Algorithms by Genetic Programming", *Procs. of IEE/IEEE Genetic Algorithms in Engineering Applications, GALESIA, 1995.*

H. Szu & R. Hartley, "Fast Simulated Annealing", Physics Letters A, Vol. 122, No. 3,4, June 1987.

[illegible]

Late Breaking Papers at the Genetic Programming 1997 Conference Stanford University July 13 –16, 1997



Edited by
John R. Koza
Computer Science Department
Stanford University

Stanford Bookstore
Stanford University
Stanford, California 94305-3079 USA
415-329-1217 or 800-533-2670

ISBN

0-18-206995-8