Too Busy to Learn

Francisco B. Pereira[♠,♣], Penousal Machado[♠,♣], Ernesto Costa[♣], Amílcar Cardoso[♣], Alberto Ochoa-Rodriguez[♦], Roberto Santana[♦], Marta Soto[♦]

[♠]Instituto Superior de Engenharia de Coimbra, Quinta da Nora
3030 Coimbra, Portugal

[♣]Centro de Informática e Sistemas da Universidade de Coimbra, Polo II – Pinhal de Marrocos
3030 Coimbra, Portugal
{xico, machado, ernesto, amilcar}@dei.uc.pt

[♦]Center of Mathematics and Theoretical Physics, ICIMAF. CP 10400, La Habana, Cuba
{ ochoa, rsantana, mrosa}@cidet.icmf.inf.cu

Abstract- The goal of this research is to analyze how individual learning interacts with an evolutionary algorithm in its search for best candidates for the Busy Beaver problem. To study this interaction two learning models, implemented as local search procedures, are proposed. Experimental results show that, in highly irregular and prone to premature convergence search spaces, local search methods are not an effective help to evolution. In addition, one interesting effect related to learning is reported. When the mutation rate is too high, learning acts as a repair, reintroducing some useful information that was lost.

1 Introduction

Evolution and learning are the two major forces that promote the adaptation of individuals to the environment. Each one of these complementary forces takes place at different levels. Evolution, operating at the population level, includes all mechanisms of genetic changes that occur in organisms over successive generations. Learning occurs at a different time scale. It gives to each individual the ability to modify its phenotype during its life in order to increase its adaptation to the environment and, hence, its chance to survive and be selected for reproduction.

In our research we are interested in study how learning and evolution can be combined in computer simulations. More specifically, we would like to investigate how should these processes be related with each other in order to originate a greater adaptive advantage (for a good overview on the subject, see [Belew & Mitchell, 1996]).

In this paper we will use the Busy Beaver (BB) problem as the testbed to study the above mentioned interactions. In 1962, Tibor Rado proposed this problem in the context of the existence of non-computable functions [Rado, 1962]. It can be defined as follows: suppose a Turing Machine (TM) with a two way infinite tape and a tape alphabet = {blank,**1**}. The question Rado asked was: What is the maximum number of **1**'s that can be written by a N-state TM when started on a blank tape? This number, which is a function of the number of states, is denoted by $\Sigma(N)$. A TM that produces $\Sigma(N)$ non-blank cells is called a Busy Beaver. The BB is considered one of the most interesting theoretical problems and, since its proposal, it has attracted the attention of many researchers. Some values for $\Sigma(N)$ and the corresponding TMs are know today for small values of N. As the number of states increases the problem becomes harder, and for $N \geq 5$, there are several candidates that set lower bounds on the value of $\Sigma(N)$. To prove that a particular candidate is the N-state BB we must perform an exhaustive search over the space of all N-state TMs and verify that no other machine produces a higher number of ones. This is extremely complex due to the halting problem.

In the original setting, the problem was defined for 5-tuple TMs. This type of machines, given a current state and symbol, write a new symbol, enter a new state and move the read/write head left or right. One of the main variants consists in considering 4-tuple TMs. These machines, during the transition to a new state, either write a new symbol to the tape or move the head (the actions are not allowed simultaneously). In the next section we present a formal definition of the BB problem to both variants.

The search space of the BB problem possesses several characteristics, such as its dimension and its complexity, that make it extremely appealing to Evolutionary Computation (EC) techniques. The first attempt to apply EC techniques to the BB problem was reported by Terry Jones [Jones & Rawlins, 1993], who used a genetic algorithm to search for specific instances of the 5-tuple BB. In 1999, our research group obtained a remarkable success in our first attempt to apply EC algorithms to the 4-tuple variant of the problem [Pereira et al, 1999a]. Several new lower bounds were set, leading to a large increase in the productivity of the TMs (e.g., from 21 to 25 for 6-state TMs and from 37 to 196 for 7-state TMs). Among the contributions to the BB problem, we studied several alternative representation techniques

[Machado et al, 1999] and developed a new crossover operator that manipulates individuals in a way that is consistent with its representation [Pereira et al, 1999b].

Just like we mentioned, in this paper we focus our attention on the interactions between learning and evolution. We propose two different local search procedures, which will be used as learning models to help an evolutionary algorithm to search for good candidates for BB(6). Our goal is to determine what is, in this domain, the influence that learning methods have in the evolutionary process.

The structure of the paper is the following: in the next section we describe the Busy Beaver problem. In section 3 we present the developmental process of an individual and describe the learning models used. Section 4 comprises some experimental details about the simulation. In section 5 we present the results of the experiments performed, whilst, in section 6, we analyze them and suggest possible explanations for the results achieved. Finally, in section 7, we present some conclusions and suggest directions for future work.

2 The Busy Beaver Problem

A deterministic TM can be specified by a sextuple $(Q,\Pi,\Gamma,\delta,s,f)$, where [Wood, 1987]:

- Q is a finite set of states
- Π is an alphabet of input symbols
- Γ is an alphabet of tape symbols
- δ is the transition function
- s in Q is the start state
- f in Q is the final state.

The transition function can assume several forms. The most usual one is:

$$\delta: Q\times\Gamma \rightarrow Q\times\Gamma\times\{L,R\}$$

where L denotes move the head left and R move right. Machines with a transition function with this format are called 5-tuple TMs. A common variation consists in considering a transition function of the form:

$$\delta: Q\times\Gamma \rightarrow Q\times\{\Gamma\cup\{L,R\}\}$$

Machines of this type are known as 4-tuple TMs. When performing a transition, a 5-tuple TM will write a symbol on the tape, move the head left or right and enter a new state. A 4-tuple TM either writes a new symbol on the tape or moves its head, before entering the new state.

The original definition of the BB [Rado, 1962] considered deterministic 5-tuple TMs with N+1 states (N states and an anonymous halt state). The tape alphabet has two symbols, Γ={blank, **1**}, and the input alphabet has one, Π={**1**}. The productivity of a TM is defined as the number of **1**'s present, on the initially blank tape, when the machine halts. Machines that do not halt have productivity zero. $\Sigma(N)$ is defined as the maximum productivity that can be achieved by a N-state TM. This TM is called a Busy Beaver.

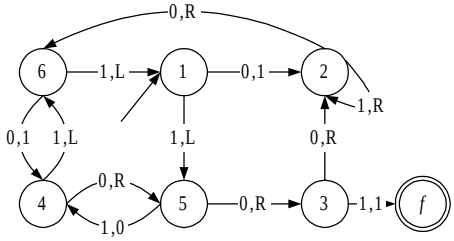
In the 4-tuple variant productivity is usually defined as the length of the sequence of **1**'s produced by the TM when started on a blank tape, and halting when scanning the leftmost one of the sequence, with the rest of the tape blank. Machines that do not halt, or, that halt on another configuration, have productivity zero [Boolos & Jeffrey, 1995]. Thus, the machine must halt when reading a **1**, this **1** must be the leftmost of a string of **1**'s and, with the exception of this string the tape must be blank.

3 Development of an Individual

Development is the process by which a genotype is transformed into a final phenotype. Usually these changes are divided in two categories, to distinguish whether they are due to genetic causes or to interactions with the environment [Hart et al, 1993]:

- Maturation: process of transforming a genotype into a phenotype. In the end of this phase we have a newborn individual.
- Learning: includes all modifications produced in the phenotype of the individual during its life. Changes induced by learning should promote an increase in the adaptation of the individual to the environment where it lives.

In this research we are essentially concerned with the second mechanism. Nevertheless, it is important to describe how maturation is performed because learning will be applied to the individual resulting from this first phase of development.



δ	By blank		By one	
Q	New State	Action	New State	Action
1	2	**1**	5	L
2	6	R	2	R
3	2	R	*f*	**1**
4	5	R	6	L
5	3	R	4	0
6	4	**1**	1	L

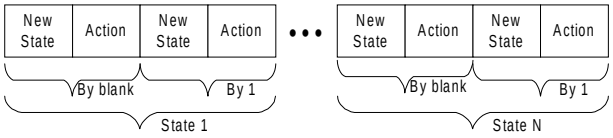
Figure 1: A six state 4-tuple TM and its corresponding transition table. The blank symbol is represented by 0.

3.1 Maturation

In the experiments described in this paper we are searching for the BB(6). Without loss of generality we consider

Q={1,2,3,4,5,6, *f*}, set 1 as the initial state and *f* as the final state. Since Γ={blank, **1**} the essential information needed to represent a potential solution is reduced to the transition function: δ: Q×{blank, **1**} → Q×{L, R, blank, **1**}.

Figure 1 shows a 4-tuple TM with 6 states (plus the halting state *f*) and its transition table. To codify the information contained in the transition table we use an integer string with 24 genes (4 genes per state) with the following format:



To simulate a TM we need to interpret the chromosome. There are several alternative ways to build a TM with the encoded information. The most evident is to directly decode the chromosome into a TM. Another option, which was analyzed in a previous study [Machado et al, 1999], deals with the existence of sets of TMs with equivalent behavior.

In this paper we will use the direct interpretation of the information contained in the chromosome with one exception. It is clear that best solutions to the BB problem are TMs that use all their states and all their transitions. According to this assumption, during simulation it is possible to perform a straightforward modification in the structure of the TMs, which is described in the next paragraph.

In a given step S of the simulation, consider that the set DT includes all transitions of the TM that were already used and that the set DS includes all states that were already visited (they have at least one transition in DT). If the following three conditions hold:

- the transition T to be used in step S does not belong to DT (i.e., it was not used before);
- T is the last transition of the states belonging to DS left to be defined;
- T leads to a state belonging to DS;

then the state to where T leads is randomly changed to a new state not belonging to DS. This modification prevents the simulation of the TM to be locked inside a subset of the states without ever halting.

3.2 Learning Models

An individual's ability to learn is directly related to its phenotypic plasticity, i.e., the capacity to change its behavior in response to environment stimulus.

In EC optimization, learning has been essentially implemented as local search algorithms (see, e.g., [Sasaki & Tokoro, 1999], [Whitley et al, 1994]). These methods iteratively perform some exploratory searches in the close neighborhood of the individual, testing several alternatives, in order to discover better solutions. In the end of this process the quality of an individual will be, not only the measure of its initial fitness, but also of its ability to improve, which leads to a better understanding of the fitness landscape. In this research we will test two different local search procedures.

3.2.1 Model 1: EDA Local Search

Evolutionary estimation distribution algorithms (EDAs) use a probabilistic model of promising solutions to guide further exploration of the search space [Pelikan et al, 1999], [Muhlenbein & Paass, 1996]. Generally, in an EDA, the estimation of the probability distribution of the best individuals of one generation is used to generate the individuals of the next population. There are no standard genetic operators, such as crossover or mutation. EDAs can be classified according to the complexity of the model used to capture the interdependencies between variables that constitute a solution.

When building a probabilistic model two opposing forces must be measured and an acceptable balance has to be found. In one side, the expressiveness of the model depends on the interactions that are considered. Whilst simple models consider an inferior number of dependencies, complex models try to cope with a higher of dependencies. On the other side, the computational cost of building such a model must also be taken into account. Computational costs of EDA implementations are determined, to a large extent, to the time spent to update and sample the probabilistic model.

A first approach to the BB problem with an EDA bivariate model is reported in [Santana et al, 2000]. Results obtained in searching for BB(6) prove that this approach is competitive with other evolutionary approaches to this problem. Here, we will use the probabilistic model proposed in [Santana et al, 2000] to perform local search.

In the BB problem, it is clear that all $4*N^1$ variables that represent a TM can interact during simulation. Nevertheless we performed a simplification and, in the model proposed, we only consider the probabilistic information of the 2*N variables that codify the actions performed by the TM.

The goal of the probabilistic model is to identify the dependencies that exist between the selected variables. These variables can be grouped in different ways. Our model joins pairs of variables representing two actions performed by transitions leaving the same state of the TM. There are 16 possible pairs of actions. To build the model, for a given selected set of individuals, we need to count the frequency of occurrence of these pairs in the TMs belonging to the set.

Despite its simplicity this model proved to be very efficient when searching for BB(6). If we observe the frequency of the four possible actions in good BB

[1] N is the number of states of the TM. We consider as variables the information that is contained in the transition table.

candidates it is obvious that there is an unequal distribution of these values. An empirical analysis would certainly say that a good candidate should have more transitions writing a **1** than transitions writing a blank. The probabilistic model can keep these differences in the frequency of single transitions. In addition, it can capture the dependencies between actions associated with the same state of the TM [Santana et al, 2000].

In each generation the probabilistic model is built using the information gathered from the individuals selected for reproduction (the selected set). When an individual is selected for learning the steps of the following algorithm are executed:

1. Select one transition used in the simulation of the TM.
2. Modify the selected transition according to the probabilistic model.
3. Simulate the resulting TM.
4. If the modification leads to an increase in the fitness of the individual or if the maximum number of learning steps has been equaled then stop learning. Otherwise, discard the change and go to step 1.

In step 2 of the algorithm, the modification of the selected transition is done using the table with the bivariate marginal distributions. Assume that transition T1 was selected for modification. T1 has associated one action with value A1[2]. The other transition, T2, from the same state has associated one action with the value A2. There are three new possible actions NA1 to associate with T1. The new value of NA1 depends on the probability of the pairs (NA1, A2) in the probabilistic model. We illustrate this situation with a concrete example. Suppose that T1 is the transition by blank of state 4 from the TM of Figure 1. In this situation, A1 is move Right and A2 is move Left. The actual value of NA1 will be probabilistically selected in accordance to the frequencies of the pairs (blank, L), (**1**, L), (L, L) in the probabilistic model.

Assume now that a partial view of the model reveals the following values:

Pairs	..	(blank, L)	…	(1, L)	…	(L, L)	…
Prob.		0.04		0.12		0.02	

In this situation the probability NA1=blank is approximately 0.22 (obtained with the division 0.04/(0.04+0.12+0.02)). Probabilities for the remaining values are obtained in a similar way. If all pairs have probability 0 the new action is chosen randomly.

3.2.2 Model 2: Random Local Search

The only difference with respect to the local search procedure described in the previous section concerns step 2 of the algorithm. Using random local search the choice of the new action for the selected transition is done randomly.

[2] Possible values for one action: {blank, 1, L, R}.

By comparing both methods we expect to gain understanding of what might be the importance of the probabilistic information when performing local search.

3.3 Learning Strategies

There are two different ways to combine evolution and learning in artificial systems, inspired in two biological theories:

- Lamarckian theory of evolution claims that phenotypic characteristics acquired by individuals during their lifetime are somehow encoded in their genes and directly inherited by their descendants. This process requires the existence of an inverse of the maturation function[3].
- Baldwin proposed a non-Lamarckian view of evolution, where acquired characteristics could be indirectly inherited. This process, known as the Baldwin effect, occurs in two steps [Turney et al, 1997]: first, phenotypic plasticity allows an individual to adapt to successful mutations. These mutations lead to an increase in the fitness of the individual and so it will tend to proliferate in the population. Given sufficient time, a behavior that was once learned may become innate.

We will test both learning strategies with each one of the models proposed.

4 Experiments

4.1 Simulation and Evaluation

The evaluation phase comprises the interpretation of chromosomes and the simulation of the resulting TM. Due to the halting problem we must establish a limit for the maximum number of transitions (MaxT). Machines that don't halt before this limit are considered non-halting TMs. To assign fitness we consider the following factors in decreasing order of importance [Pereira et al, 1999a]:

- Halting before reaching the predefined limit for the number of transitions;
- Accordance to the 4-tuple rules;
- Productivity;
- Number of used transitions;
- Number of steps made before halting.

We consider all these factors to assign fitness because we intend to explore differences between "bad" individuals. With this fitness function a TM that never leaves state 1 is considered worse than another one that goes through 3 or 4 states, even if both are non-halting machines and have the same productivity. This approach proved to be more

[3] Even though Lamarckian theory proved to be wrong in biological systems, from the viewpoint of adaptive artificial systems it should be considered.

effective than using productivity alone as fitness [Pereira et al, 1999].

4.2 Experimental Settings

The experiments presented concern the search for the 4-tuple BB(6). Current best candidate has productivity of 25 and performs 256 transitions [Pereira et al, 1999b]. According to these values we set MaxT to 500. All experiments were performed using a modified version of GALLOPS 3.2 [Goodman, 1996]. The settings of the evolutionary algorithm are the following:

- Number of evaluations: 60000000;
- Population Size: 500;
- Generation Gap: 1;
- Elitist Strategy;
- Tournament Selection with tourney size 5;
- Single point Mutation;
- Mutation rate: {0.01, 0.025, 0.05};
- Graph Based Crossover
- Maximum graph crossover size: 3
- Crossover rate: 0.7

Graph based crossover was presented in [Pereira et al, 1999]. It was designed to work with individuals with a graph-like structure. The main idea of this operator is the exchange of sub-graphs between individuals. Maximum graph crossover size defines the number of states belonging to each sub-graph. Results presented confirmed that, in this domain, it clearly outperforms two-point crossover.

In experiments with learning there are two extra parameters to consider:

- Learning Rate (LR): probability of an individual being subject to learning. We will perform experiments with 3 different values: {0.1, 0.5, 1}.
- Learning Length (LL): maximum number of learning steps. This parameter is set to 10.

During learning, each simulation of a TM counts as one evaluation. The initial population is randomly generated and for every set of parameters we performed 30 runs with the same initial conditions and different random seeds.

5 Experimental Results

Table 1 presents the average number of ones written by the best individual of the final population, for all considered configurations. A brief perusal of the results suggests that learning does not cause any significant improvement in the search process.

From the three values considered, a mutation rate of 2.5% seems to be the best choice. Focusing our attention in the experiments performed with this value it is obvious that no learning strategy and no learning method was able to induce an increase in the productivity obtained by evolution alone.

Another interesting result is that, with lower mutation rates (1% and 2.5%), there is no significant difference between the results obtained with different learning strategies and/or different learning models. This suggests that the evolutionary algorithm was unable to use the potential offered by learning.

We performed some previous experiments using a probabilistic evolutionary approach to this problem [Santana et al, 2000]. A bivariate model, similar to the one proposed here was used as the evolutionary algorithm and the results proved that, despite its simplicity, this model can be very effective when searching for the BB(6). In the same research we performed some additional experiments with a probabilistic evolutionary algorithm where the selection of the new action was done randomly. In this last situation results were clearly worse, if compared to those obtained when information from the bivariate model was used.

In the experiments presented in this paper, learning performed with the EDA model was unable to take advantage of the probabilistic information (at least this information was not useful to help to guide the search process). If we average the productivity of all experiments performed we obtain the following results: evolution-only: 17.2; experiments with the EDA model: 17.4; experiments with the random model: 17.3.

Concerning the two learning strategies the conclusions are similar: the average productivity of the experiments with the Baldwin effect is 17.3, whilst the average productivity of the experiments with Lamarckian learning is 17.4.

		Baldwin						**Lamarck**						**Evolution**
		EDA			Random			EDA			Random			-
	LR	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	-
Mutation	0.01	18.8	18.4	18.2	18.7	18.5	18.6	18.8	17.9	18.0	18.6	18.2	18.2	18.7
	0.025	19.7	19.7	20.1	20.0	19.5	19.8	19.4	19.2	19.1	19.9	19.2	19.5	20.0
	0.05	12.9	13.2	13.9	12.7	14.2	13.7	13.1	15.7	17.1	13.2	12.9	15.5	12.8

Table 1: Productivity achieved by the best individual of the final population. Each experiment was repeated 30 times. Results are the averages. Column labeled Evolution presents results from experiments without learning. Columns labeled Baldwin and Lamarck present results from experiments with both learning strategies.

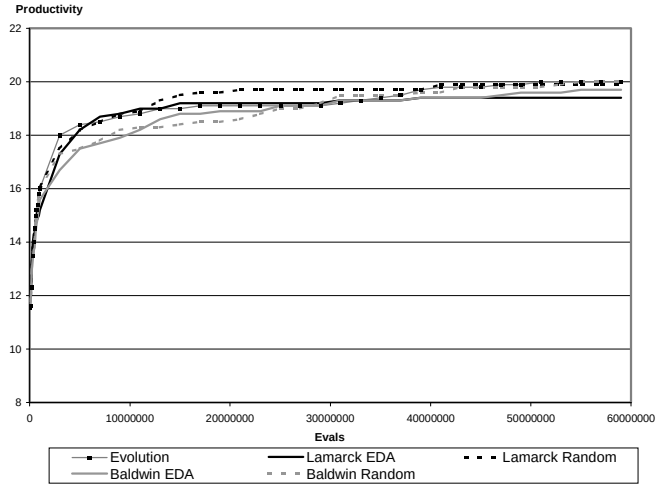
Given these values, the main conclusion is that learning, implemented as a local search algorithm, was ineffective to help evolution:

- There is no difference between a probabilistic and a random model of learning.
- There is no difference between a Lamarckian strategy and the Baldwin effect.
- There is no relevant difference between experiments with and without learning.

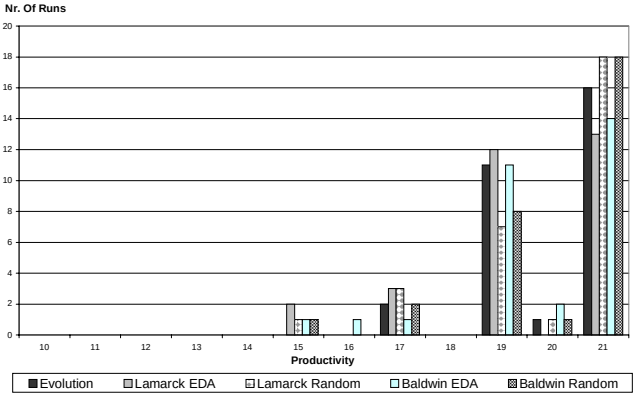
Other measures of the behavior of the evolutionary algorithms help to support this idea. As an example, the rates of convergence for experiments without learning, with the Baldwin effect and with Lamarckian learning are similar. This suggests that premature convergence did not affect learning experiments.

In graph 1 we present the evolution on the number of **1**'s written by the best individual for the different experiments performed with a mutation rate of 0.025. Only experiments with LR=0.1 are shown, even though the conclusions are similar for other values of this parameter. It is visible that all lines are almost coincident. Concerning the same experiments, in graph 2 we present the productivity of the best individual of the final generation, for each one of the 30 runs. The distribution of productivities is similar in all experiments, which corroborates and reinforces our conclusion.

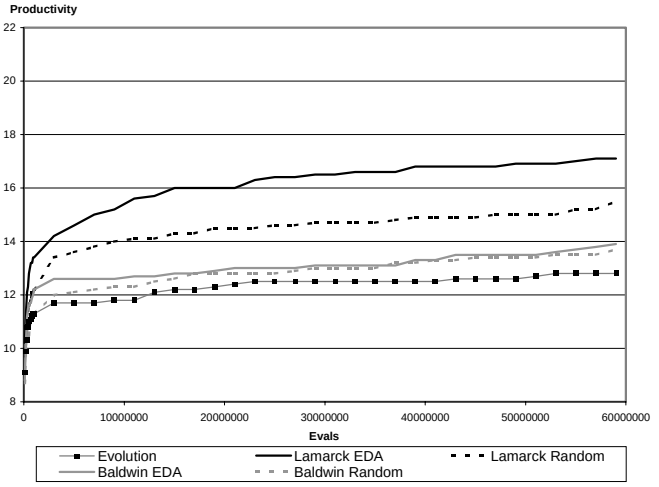
If we look to the results from another point of view we can see that, even though learning was unable to improve performance, its effect is nevertheless visible in one particular situation. A closer look at the results obtained by the experiments performed with a mutation rate of 5% leads us to formulate some interesting hypothesis.



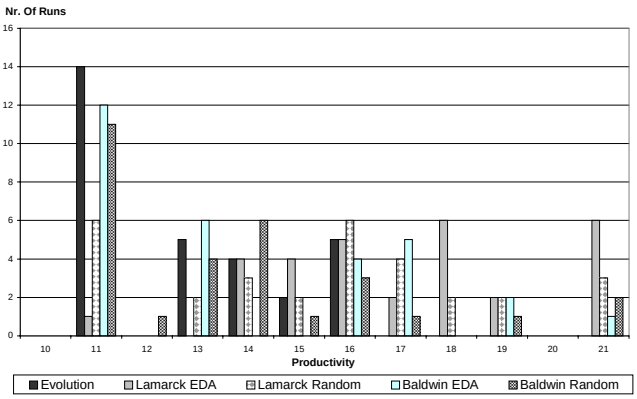
Graph 1: Evolution on the number of ones written by the best individual on the different experiments performed with a mutation rate of 0.025. In the experiments with learning, value for LR is 0.1. All values are averages of series of 30 runs.



Graph 2: Productivity of the best individual of the final generation for each one of the 30 runs. All experiments were performed with a mutation rate of 0.025. In the experiments with learning, LR has value 0.1.



Graph 3: Evolution on the number of ones written by the best individual on the different experiments performed with a mutation rate of 0.05. In the experiments with learning, value for LR is 1. All values are averages of series of 30 runs.



Graph 4: Productivity of the best individual of the final generation for each one of the 30 runs. All experiments were performed with a mutation rate of 0.05. In the experiments with learning, LR has value 1.

Experiments with higher mutation rates are the ones that obtained worst results. It is clear that a 5% mutation rate is too high to enable an efficient sample of the search space. The evolutionary process is unable to combine the building blocks in an efficient way and to converge to promising areas, because mutation constantly changes too many values to allow the process to converge. Nevertheless it is in these experiments that the effect of learning is visible. In these hard conditions, learning, probably, acts as a gene repair, fixing some values that were lost by mutation. In accordance, it is not surprising that the best results are obtained by Lamarckian learning applied to all individuals in each generation (LR = 1). Repairing is needed to all individuals and must be done directly. Repairing performed by the Baldwin effect is lost to the next generations. Graphs 3 and 4 present results concerning experiments performed with 5% mutation rate (learning experiments have LR = 1). In both graphs the repair effect is clearly visible.

This is a slightly different view of learning, which might be subject of future research. It helps evolution not to find promising areas, but to keep important discovered information that could otherwise by lost due to the effect of mutation. A similar effect was also proposed in [Harvey, 1997].

6 Discussion

The repair effect suggested in the end of the previous section implies that learning, even though in an unusual way, is present in the evolutionary process. In this section we will try to advance some explanations to why was learning unable to improve the performance when the mutation rate is lower.

One hypothesis is that the evolutionary algorithm is so effective in this problem that leaves no room for learning to act. Based on the experiments conducted, our believing is that this is partially true. It is clear that the current evolutionary algorithm is remarkably adapted to the BB problem and, at least when searching for 6-state candidates, there is not much room for improvement.

Nevertheless, it is important to notice that, in the experiments performed in this research, the current best know solution for BB(6) was never found, which suggests that there were some potential improvements that could have been induced by learning.

We performed an empirical study on the search landscape to gain some knowledge about the topology of the space and to try to understand why is local search learning so ineffective. We verified that, in different areas of the search space, there are several TMs that write 21 **1**'s. These TMs are surrounded by low fitness solutions, which makes the space very irregular and prone to premature convergence. Evolution by itself is usually able to quickly identify the areas where the TMs that produce 21 **1**'s are. After finding one of these local maxima, there is a high evolutionary pressure towards this point and it is almost impossible for the evolutionary process to escape premature convergence. Learning methods employed in this study search for better points in the close neighborhood of the solutions. In the described situation, the efforts of local search are helpless, since the exploration of the neighborhood will, most likely, lead to the already discovered local maxima.

Some preliminary experiments performed with 4-tuple BB(7) confirm that, in this domain, local search methods are not effective. The problem of premature convergence is even magnified and learning is unable to help evolution to escape such local maxima. In most of the experiments that we performed, with both learning models and different learning strategies, the evolutionary algorithm converges to areas in the neighborhood of TMs with productivity ranging from 37 to 49. Several 7-state TMs with productivities above 100 are known and were infrequently obtained by our research group.

7 Conclusions

In this paper we analyzed several interactions that occur between learning and evolution in artificial adaptive systems. The application domain was the BB problem, more specifically the search for BB(6) candidates.

To help evolution, we proposed two different learning procedures, designed to act as local search methods. Experimental results show that, in this domain, the structure of the fitness landscape prevents local search methods from being an effective assistance to evolution.

We also reported one interesting effect that happens when the mutation rate is too high. In this situation, learning fixes some genes that were lost, acting as a repair of the extreme perturbation introduced by this operator. To accomplish this effect, Lamarckian learning is more effective because it directly repairs undesirable mutations and passes it to descendants.

In spite of the inefficiency of local search methods, we believe that there is room for learning in this kind of problems. The space is hard to search and learning might help to provide a better understanding of the fitness landscape. Our future research efforts will be devoted to the development and testing of alternative ways of considering learning in this domain. Assuming that learning includes any kind of modifications occurring during the lifetime of an individual as a result of its interactions with the environment, it is possible to devise several learning algorithms alternative to the traditional view of local search. Several ideas that will be subject of future research include:

- We will conduct several experiments where individuals learn to perform tasks not directly related to the main goal of the optimization. Several studies [Nolfi et al,

1994],[Harvey & Stone, 1995] refer that learning an ability which is different from the ability for which individuals are selected, might provide a beneficial effect on evolution. Concerning our problem, there are alternative ways to assess the complexity of a TM. Candidate TMs can, for example, learn to perform simple arithmetic operations with a unary alphabet.

- Another possibility is to let other individuals from the current population help the learning process of one of them. The idea is to develop some mechanism of cooperative learning, where a group of individuals tries to find the solution to a problem. If each individual has partial relevant information then the group might symbiotically find the solution that otherwise would be unobtainable. A simple model of symbiotic relationships occurring in an adaptive system is proposed in [Watson & Pollack, 1999].

Acknowledgments

This work was partially funded by the Portuguese Ministry of Science and Technology, under Program PRAXIS XXI.

Bibliography

Belew, R. and Mitchell, M. (1996). *Adaptive Individuals in Evolving Populations: Models and Algorithms*, Santa Fe Institute in the Sciences of Complexity, Vol. XXVI, Reading, MA: Addison-Wesley.

Boolos, G., and Jeffrey, R. (1995). *Computability and Logic*, Cambridge University Press.

Goodman, E. (1996). GALOPPS, The Genetic Algorithm Optimized for Portability and Parallelism System, *Technical Report #96-07-01*, Michigan State University.

Hart, W. E., Kammeyer, T. E. and Belew, R. K. (1995). The Role of Development in Genetic Algorithms. In Whitley, D. and Vose, M. D. (Eds.), *Foundations of Genetic Algorithms 3*, California: Morgan Kaufmann.

Harvey, I., Stone, J. (1995). Unicycling Helps your French: Spontaneous Recovery of Association by Learning Unrelated Tasks. *Cognitive Science Research Paper CSRP 379*, School of Cognitive and Computing Sciences, University of Sussex, England, UK.

Harvey, I. (1997). Is There Another New Factor in Evolution?, *Evolutionary Computation*, Vol. 4(3), pp. 313-329.

Jones, T. and Rawlins, G. (1993). Reverse Hillclimbing, Genetic Algorithms, and the Busy Beaver Problem. In Forrest, S. (Ed.). *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA-93)*, pp.70-75, San Mateo, CA, Morgan Kaufmann.

Machado, P., Pereira, F. B, Cardoso, A., Costa, E. (1999). Busy Beaver – The Influence of Representation, *Proceedings of the Second European Workshop in Genetic Programming*, Göteborg, Sweden.

Muhlenbein, H. and Paass, G. (1996). From Recombination of Genes to the Estimation of Distributions I. Binary Parameters. In Voigt, H-M, Ebeling, W., Rechenberg, I. and Schwefel, H.-P. (Eds.*), Parallel Problem Solving from Nature*, pp. 178-187, Springer:Berlin.

Nolfi, S. Elman, J, and Parisi, D. (1994). Learning and Evolution in Neural Networks. Adaptive Behavior, Vol. 3(1), p. 5-28.

Pelikan, M., Goldberg, D., and Lobo, F. (1999). A Survey of Optimization by Building and Using Probabilistic Models. *IlliGAL Report No. 99018*, University of Illinois.

Pereira, F. B., Machado, P., Costa, E. and Cardoso, A. (1999). Busy Beaver: An Evolutionary Approach. *Proceedings of the 2nd Symposium on Artificial Intelligence*, pp.212.216. Havana, Cuba.

Pereira, F.B., Machado, P., Costa, E. and Cardoso A. (1999). Graph Based Crossover - A Case Study with the Busy Beaver Problem. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (Eds.). *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1149-1155, Orlando, Florida, USA. Morgan Kaufmann.

Rado, T. (1962) On non-computable functions, *The Bell System Technical Journal*, vol. 41, no. 3, pp.877-884.

Santana, R., Rodriguez, A. O., Soto, M., Pereira, F. B., Machado, P., Costa, E. and Cardoso, A. (2000). Probabilistic Evolution and the Busy Beaver Problem. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000), (to appear).*

Sasaki, T. and Tokoro, M. (1999). Adaptation under Changing Environments with Various Rates of Inheritance of Acquired Characters: Comparison Between Darwinian and Lamarckian Evolution. In McKay, B., Yao, X., Newton, C. S., Kim, J. H. and Furuhashi, T. (Eds.), *Proceedings of 2nd Asia-Pacific Conference on Simulated Evolution and Learning (SEAL-98)*, Canberra, Australia.

Turney, P., Whitley, D., and Anderson, R. (1997). Evolution, Learning and Instinct: 100 Years of the Baldwin Effect, *Evolutionary Computation*, Vol. 4(3), iv-viii.

Watson, R., and Pollack, J. (1999). How Symbiosis Can Guide Evolution. In Nicoud, J.-D. and Mondada, F. (Eds.) *Proceedings of the 5th European Conference on Artificial Life*, Springer-Verlag.

Whitley, D., Gordon, S. and Mathias, K.(1994). Lamarckian Evolution, the Baldwin Effect and Function Optimization. In Davidor, Y., Schwefel, H. P. and Manner, R. (Eds.), *Parallel Problem Solving from Nature - PPSN III*, 6-15. Berlin: Springer-Verlag.

Wood, D. (1987). *Theory of Computation*, Harper & Row, Publishers.