2006 IEEE International Conference on
Systems, Man, and Cybernetics
October 8-11, 2006, Taipei, Taiwan

A New Approach to Genetic Programming based on Evolution Strategies

Eduardo Oliveira Costa and Aurora Pozo

***Abstract*— This paper proposes a new approach to induction of programs by Genetic Programming (GP) using the ideas of Evolutionary Strategies (ES). The goal of this work is to develop a variety of Genetic Programming algorithm by doing some modifications on the classical GP algorithm and adding some concepts of Evolutionary Strategies. The new approach was evaluated using two instances of the Symbolic Regression problem – the *Binomial–3 problem* (a tunably difficult problem), proposed in [5] and the *Time Series problem* (an application of symbolic regression) – and a problem of a different domain, the *Santa Fe Artificial Ant problem*. The results discovered were compared with the classical GP algorithm. The Symbolic Regression problems obtained excellent results and an improvement was detected, but this does not happened with the Artificial Ant problem.**

I. INTRODUCTION

The Genetic Programming is a Machine Learning technique which shown be an efficient paradigm to solve problems in many knowledge areas such as digital circuits, data mining, molecular biology, optimization tasks and others [13], [1], [12]. Its stochastic search allows GP to explore practically and robustly, if not necessarily efficiently, huge search spaces [13]. GP can be used "to learn" mathematical functions or to find patterns in a set of data which provide to GP a huge applicability field.

Recent studies have been made with the intention to detect and fix some problems in GP [3], [11], [5], [18], [19]. The analysis of the problems which are difficult to be solved by GP [5] and the factors that could increase the likelihood of success in GP [3] are examples of researches that have been done in this field.

Also, the code growth (or bloat), that is an excess of code growth caused by the genetic operators in search of better solutions, without a corresponding improvement in fitness have been studied. This problem often leads to the stagnation of the evolutionary process [18].

Considering this facts, this work presents a preliminary study of a new approach to Genetic Programming algorithm based on researches presented in [5], [3], [19], together with some concepts of the theory of Evolutionary Strategies. The principal goal of this work is to propose a new GP algorithm that is more based in the Evolutionary Strategies than in the theory of Genetic Algorithms, proposed by John Holland [9].

This work was sponsored by CNPq (National Council for Scientific and Technological Development, Brazil)

E. O. Costa belongs to Computer Science Department, Federal University of Parana (UFPR), PO Box 19081, 81531-970, Curitiba, Brazil, costa@inf.ufpr.br

A. Pozo belongs to Computer Science Department, Federal University of Parana (UFPR), PO Box 19081, 81531-970, Curitiba, Brazil, aurora@inf.ufpr.br

This paper is organized as follow: The Section II and III present an overview o Genetic Programming and Evolutionary Strategies respectively. The Section IV presents the proposed approach to GP. In Section V the methodology of the Experiments is described and in section VI the Results are shown. Finally, Section VII presents the Conclusion and the Future Works.

II. AN OVERVIEW OF GP

Genetic Programming is a new paradigm to the resolution of problems based on mechanisms observed on nature and formalized on Darwin's Natural Selection Theory. In nature, the individuals that better adapt to the environment that surrounds them have a greater chance to survive. They pass their genetic characteristics to their descendents and consequently, after several generations, this process tends to select naturally the best individuals [6].

A. Elements of Genetic Programming

The main elements of Genetic Programming are:

- Program Structure: A tree is the most used structure for represent programs in GP. Each node can be a function or a terminal. A function has to be evaluated considering its parameters while a terminal has its own value. The user needs to provide functions and terminals sets according to the problem.

- Fitness: In the GP the entity that reflects the degree of adaptation is the fitness function. The programs that better solve the problem at hand will receive a better fitness value, and will consequently have a better chance of being selected. An adequate choosing according to the domain of the problem is essential to provide good results.

- Genetic Operators: Once the individuals are selected it is time to apply one of the three basic genetic operators. They are: Reproduction - an individual is replicated to the next generation, Crossover - two programs are recombined to generate two offspring; and Mutation - a new sub-tree replaces a randomly selected part of a program.

- Parameters: The behavior of the algorithm is determined by a set of parameters that, among other things limit and control how the search is performed. Some of them are: genetic operator's rate (reproduction, crossover, and mutation), population size, maximum number of generations, selection rate, maximum depth of the individual, etc.

B. The GP Algorithm

In the Genetic Programming, the genetic algorithm works in a population of computational programs which differ by

form and size [13]. This population will be evolved until generate a new population composed by best individuals, using genetic operators (reproduction, crossover and mutation). The process is guided by an adaptation function, called fitness.

The first step of the algorithm is creating an initial population of computer programs randomly (Generation 0). After that there are two major tasks processed in a loop:

- Evaluation of each program by the use of a fitness function (this is defined according to the problem).

- Creation of a new population by the selection of individuals, based on their fitness values and by applying genetic operators such as: reproduction, crossover and mutation.

Each run of this loop represents a new generation of computer programs that substitute the previous one. This process is repeated until a solution is found or until a maximum number of generations are reached.

III. AN OVERVIEW OF ES

Evolutionary Strategies is a technique that makes part of the set of techniques of the Evolutionary Computation. This technique can be defined as an algorithm where individuals (potential solutions) are codified by a real-value variable set, the "genome" [14].

According to [7], the ES were initially developed with the purpose of parameter optimization.

The first ES algorithm proposed, used a mutation-selection schema, known as two-membered ES or $(1+1)-ES$ algorithm [7]. According to [2], the $(1+1)-ES$ works with an individual, which creates an offspring through mutation. The best of these both individuals is deterministically selected to integrate the next generation.

This process posteriorly was improved, adding the idea of population [7]. Rechenberg proposed the multimembered ES, $(\mu+1)-ES$, where μ parents, $(\mu > 1)$, participate in the generation of one descendant. In this method all parents have the same likelihood of matching.

After, the $(\mu+\lambda)-ES$ specifies that μ parents produce λ descendants, where $(\lambda > \mu)$. The descendants compete with their parents in the selection of the best μ individuals to the creation of the next generation. This procedure presents local optimals. To solve it, the $(\mu,\lambda)-ES$ was proposed, where the lifetime of an individual is just during a generation. Recent evidences points that the last algorithm is as good as the first one in practical applications [7].

The notation of an ES algorithm is the following:

$$(\mu+\lambda)-ES \quad or \quad (\mu,\lambda)-ES$$

Where:

μ: is the size of the population.

$+/,$: The "+" operator indicates that μ and λ will compete to the next generation. The μ best individuals will be selected. The "," operator indicates that the μ best individuals will be selected just only between the descendants.

λ: is the number of descendants created at each generation.

This work proposes the use of the concepts here presented to creation of a new GP approach algorithm, as is shown in the next section.

IV. THE $(\mu+\lambda)-GP$ ALGORITHM

One of the motivations to propose this new approach to GP is the research presented in [3], where Daida performed experiments trying to identify an initial metric in populations that could increase the likelihood of success in Genetic Programming problems.

Also, the ideas of multicrossover to prevent the occurrence of code growth, presented in [19], gave some suggestions to formulate the considered algorithm. The hypothesis behind this implementation is that many crossovers between the best individuals could avoid the loss of good genetic material and also, the likelihood to generate good individuals could be extended.

Furthermore, in this work we propose a preliminary study doing some modifications at mutation operator, in attempt to prevent some needless code generated. The original mutation operator proposed by [13] just selects randomly a sub tree and change the sub tree selected for another sub tree also chosen randomly. Considering that in some situations the trees contain needless code, an elimination of a sub tree could be good to the evolutionary process.

Considering these facts, we propose the selection of a little percentage of the best individuals created at each generation to be applied the genetic operators. In this step we used the elitist selection, choosing the $n\%$ best individuals.

After that in the set of $n\%$ best individuals, using the Roulette Wheel method with fitness proportional selection [8], the genetic operators (crossover or reproduction) are applied until generate λ individuals.

The next step is the application of the mutation operator. The mutation is applied with some modifications. The original mutation is preserved, being added one more option that is the removal of a sub tree (chosen randomly). This both options of operators are chosen randomly. In a similar way to the previous step, the candidates to the mutation operator were selected using the Roulette Wheel method.

The descendants will compete with their parents in the selection of μ best individuals to the next generation [7]. This approach was based on the $(\mu+\lambda)-ES$ algorithm. The Algorithm 1 presents the proposed $(\mu+\lambda)-GP$ algorithm.

V. EXPERIMENTS

Three problems were chosen to verify the behavior of the proposed approach. A tunably difficult problem: the Binomial–3 problem, an application of symbolic regression: the Time Series problem, and the Santa Fe Artificial Ant problem. In this section the methodology used and the experiments conducted in this work are described.

A. The Binomial–3 problem

The first experiment conducted was with the Binomial–3 problem. The Binomial–3 was proposed as a tunably difficult problem by Daida, et. al. in [5]. The Binomial–3 problem is an instance of symbolic regression problems and the involved function for the solution is

$$f(x) = 1+3x+3x^2+x^3 \quad (1)$$

Algorithm 1 Proposed $(\mu+\lambda)-GP$ algorithm

1: Initialization:
   (a) Generate the initial population.
   (b) Calculate fitness.
   (c) Select the μ best individuals.
2: **Repeat until** the solution was not found or the maximum number of generations was not reached
3: Using the Roulette Wheel method, select parents and apply genetic operators until generate λ descendants.
4: Apply mutation on λ descendants:
   (a) Replace sub trees (original mutation).
   (b) Choose a sub tree and replace for a terminal.
5: Calculate fitness.
6: Select the μ best parents between $(\mu+\lambda)$.
7: **End repeat until**

The term "binomial" refers to the sequence of the coefficients in this polynomial and the "3" to the order of this polynomial [5].

This problem also, shares many properties that are common with other problems in GP such as: affords GP to choose from multiple approaches to solve the same problem, allows several types of introns, some which involve the structure $(-X,X)$. According to [4], the total number of ways to solve the Binomial–3 problem is on the order of a few thousand. This problem was also used in [3] to study the effect of tuning attributes of a population to increase the likelihood of GP success.

The α is defined as the tuning parameter. It is a real number that controls the problem difficulty. Adjusting the range over which the random constants occur, this problem can be tuned from a relatively easy problem to a difficult one. According to [3], in general, values of α that are farther from 1, result in settings that increase the difficulty for GP to solve this problem.

The Binomial–3 problem was defined according to [5]: Fitness cases are 50 equidistant points generated from the equation $f(x) = 1+3x+3x^2+x^3$, over the interval [-1, 0). The raw fitness is the sum of the absolute error. A hit is defined being within 0.01 in ordinate of a fitness case for a total of 50 hits. The stop criterion is when an individual in a population first scores 50 hits.

The function set defined is $\{+,-,\div,\times\}$, which correspond to the basic arithmetic operators (with protected division). The terminal set is defined as $\{x,\beta\}$, where x is the symbolic variable and β is the set of ephemeral random constants (ERCs).

The ephemeral random constants are uniformly distributed over a specified interval of the form $[-\alpha,\alpha]$, where α is a real number that defines the range for ERCs. Each ERCs was generated once at population initialization and is not changed in value during the course of a GP run.

Seven values for α were used, they are: {0.1, 1, 2, 3, 10, 100, 1000}. An option with no ERCs was run also. Eight datasets were collected according to Table I. Each dataset was trained with 600 runs for a total of 4800 runs. Experiments were accomplished with the classical GP method and with the new GP approach.

TABLE I
DATASETS OF ERCS

Dataset	*ERC*	*Dataset*	*ERC*
N	–	3	[-3, 3]
0	[-0.1, 0.1]	4	[-10, 10]
1	[-1, 1]	5	[-100, 100]
2	[-2, 2]	6	[-1000, 1000]

B. The Time Series problem

A time series can be defined as a set of observation ordering in a period of time [16]. Some examples of time series can be: sales of a kind of product during the Christmas, the consume of electrical energy in a house during a month, the quantity of fails of a software, etc.

The forecast of time series is a very useful tool that permits to describe the behavior of a time series, doing estimations and getting the factors that influenced the behavior of a time series. There are many techniques to forecast of time series. The majority of this models are based on the past observations of the series [21].

Some examples of the traditional methods of forecasting of time series are: Linear Trend, Random Walk, Moving Average (MA), Simple Exponential Smoothing, Auto Regressive (AR), Auto Regressive and Moving Average (ARMA), Box and Jenkins [16], being the two last ones, the most complex models.

The forecast of these models presents some difficulty and complexity to be done. To adjust a model is necessary to provide the format of the equation to be obtained and normally many attempts are needed until obtain the best equation.

Considering this scenario, the GP appears as an alternative method to reduce this difficulty on forecast of time series. Many works confirm that GP is a suitable approach to this task in different applications, such as forecast of financial time series, based on these works [17], [12], [15], [10].

The set of functions used for this problem is:

$$\{+,-,\div,\times,exp^x,sqrt,log,sin,cos\}$$

The division and the logarithm operators were created as protected operators. The set of terminals is composed by $\{Z_{t-1},Z_{t-2},Z_{t-3},Z_{t-4},\theta\}$ where Z_{t-n} is the n previous values of the series and θ is the ephemeral random constant, between $[0,1)$.

The fitness function used in this problem is the Root Mean Square Error (RMSE). The RMSE is one of the most measures used to scale the accuracy of a forecast method. The RMSE is defined according to the formula below. In this experiment, individuals with RMSE equal to 0 or near to 0 are the best.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(|x_i-\hat{x}_i|)^2}{n}} \quad (2)$$

Where x_i is the real value, $\hat{x}_i$ is the estimated value and n is the size of the series.

For this problem, a set of 50 time series was generated using the statistical software R [20]. Time series of the following models were generated: AR, MA and ARMA varying randomly the constants and parameters of these models. For each time series 90 runs, with different seeds where performed for each GP approach.

C. The Artificial Ant problem

The third experiment conducted was with the classical Santa Fe Artificial Ant problem. The methodology and the description of the experiment are described as follow.

This problem was implemented according to [22]. The Artificial Ant problem was popularized by Koza [13], but was originally developed for the field of Artificial Life. The Ant problem consists of finding the best strategy for picking up food pellets along a trail in a grid. The solution to the problem is an algorithm for collecting food.

The Santa Fe trail is often used for the Ant problem. The Santa Fe trail consists of a grid of 32x32 with 89 food pellets. The 'ant' starts in the North-west corner, facing East.

The functions and terminals are executed to move the ant on the grid during evaluation. The terminals provide the 'ant' with locomotion and change of direction. The fitness for this problem is measured as the number of pellets consumed of the total on the trail.

The time-steps were defined in 600 which fitness function is measured by the number of food consumed by the ant. For this problem, 600 runs for each GP method were performed, using different seeds.

D. Configurations

The software Lilg-gp [22] was used to perform the experiments. Some modifications were made in the kernel of the system to integrate the new selection and mutation operators.

The parameter settings used in the tests was practically the same as used in [5] and founded in Chapter 7 of [13] and they are described in the Table II.

The experiments were performed in a cluster using mono–processed Athlon 64 machines, with 2GB of memory each.

TABLE II
PARAMETER SETTINGS

Parameter	*Binomial-3*	*Time Series*	*Artificial Ant*
Number of runs	600	90	600
Maximum generations	200	200	2000
Population size	500	500	200
Selection method	best	best	best
Best individuals selected	20%	20%	20%
Initialization method	half_and_half	full	full
Initialization depths	2-6	2-10	2-10
Maximum depth	26	10	10
Maximum nodes	100	50	50
Crossover rate	80%	80%	80%
Reproduction rate	10%	10%	10%
Mutation rate	20%	20%	20%

VI. RESULTS

In this section the results obtained with the experiments are shown and discussed.

For each experiment a comparison between the proposed method and the classical GP was performed using paired two-sided t tests, at a 95% confidence level. The statistically significant difference ($p\text{-}value < 0.05$) is in bold.

A. The Binomial–3 problem

Tables III and IV present the results obtained with the experiments.

TABLE III
BINOMIAL-3 – CLASSICAL GP RESULTS

Dataset	*Fitness Average*	*Hits Average ± Deviation*	*Success %*
N	0.191	46.45 ± 10.44	92.90
0	1.339	22.16 ± 14.26	44.32
1	0.357	40.37 ± 10.96	80.73
2	0.418	37.86 ± 12.12	75.73
3	0.481	36.44 ± 12.42	72.89
4	0.914	27.38 ± 14.55	54.86
5	1.603	15.93 ± 12.49	31.86
6	1.927	14.47 ± 12.22	28.94

TABLE IV
BINOMIAL-3 – NEW GP RESULTS

Dataset	*Fitness Average*	*Hits Average ± Deviation*	*Success %*
N	0.164	46.85 ± 8.67	93.70
0	0.765	30.13 ± 14.29	60.26
1	0.248	44.66 ± 8.11	89.32
2	0.285	43.42 ± 9.15	86.85
3	0.294	45.30 ± 10.33	90.60
4	0.512	36.84 ± 12.22	73.70
5	0.692	33.23 ± 16.16	66.66
6	1.081	25.00 ± 15.12	49.89

According to the results presented, the probability of success in the experiments with the new approach of GP is bigger than with the classical GP. In the dataset 6 (which is the most difficult degree), the difference between the methods is around of 40%.

In Table V are shown the results of the comparison of the proposed method with the classical GP. Considering the results, just in the dataset N (which is the easier difficult degree) the *p-value* is > 0.05, what permit us consider that there is not statistically significant difference between the two sets compared. In the others datasets, the statistically significant difference is confirmed, that is, the proposed method is statistically better than the classical GP.

TABLE V
BINOMIAL-3 – P-VALUES RESULTING FROM PAIRED TWO-SIDED T TESTS COMPARING THE METHODS

Dataset	*p-values*	*Dataset*	*p-values*
N	0.536	3	**< 0.001**
0	**< 0.001**	4	**< 0.001**
1	**< 0.001**	5	**< 0.001**
2	**< 0.001**	6	**< 0.001**

In Figure 1, a graphic with the evolution of the solution's growth is presented. The dataset 6, that presents the more

difficult step, was chosen to be compared between the two approaches.

The values presented are the average of the fitness and the number of hits of 50 runs during the evolutionary process. Analyzing the graph, we can observe that the evolution of the mean fitness using the new approach of GP is greater than the classical GP. Considering the number of hits, this difference is still greater using the proposed algorithm.

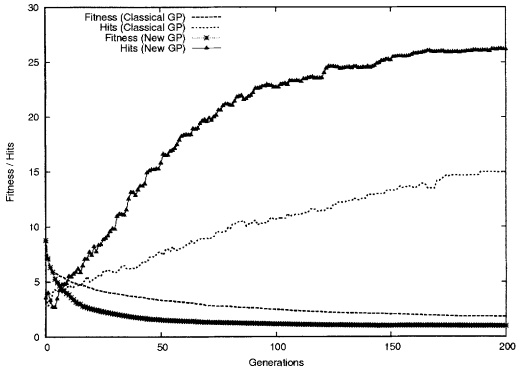

Fig. 1. Binomial-3 – Evolution of the Mean Fitness in Dataset 6

B. The Time Series problem

By the analysis of the results, we can see that in the majority of the time series the fitness average in the experiments with the new approach are lower than with the classical GP.

Considering the results of the t test, just two *p–values* are greater than 0.05. In Table VI the results obtained with the experiments are shown. The average of fitness ± the standard deviation for each time series and the *p–values* are presented.

The results obtained confirm that the GP is a powerful technique to forecast of time series. With the new approach we can verify an improvement in the fitness, and consequently a reduction of the RMSE between the real and the predicted values.

C. The Artificial Ant problem

The results obtained with the Artificial Ant points to the classical approach as the best method to this problem, considering the results of average of consumed food and the average time. The difference is statistically significant considering the *p–value* , which comproves that the classical approach had a better behavior than the proposed approach. In Table VII the results obtained with the experiments are shown.

In Figure 2, the progress of the evolutionary process is presented. The values in the graph are the average of the consumed food of 100 runs during the evolutionary process. The evolution of the first 250 generations is in the graph.

Analyzing the results we can verify that in generation 100, the classical GP obtains the best results and happens a stagnation of the evolutionary process. With the new approach, this does not happen. The evolutionary process makes progress nearly the generation 200. However, this progress is not enough to find a good result, as the results reached by the classical GP.

TABLE VI

TIME SERIES RESULTS

Model	*Time Series*	*Classical GP*	*New GP*	*p–value*
AR	AR1_100	0.963±0.01	0.953±0.04	0.083
	AR1_101	0.992±0.01	0.957±0.04	< **0.001**
	AR1_102	0.910±0.01	0.877±0.04	< **0.001**
	AR1_103	0.976±0.01	0.953±0.04	< **0.001**
	AR1_104	0.981±0.01	0.954±0.05	< **0.001**
	AR1_105	1.034±0.01	0.980±0.06	< **0.001**
	AR1_106	1.047±0.01	1.023±0.05	< **0.003**
	AR1_108	0.944±0.01	0.927±0.02	< **0.001**
	AR1_109	1.014±0.01	0.959±0.03	< **0.001**
	AR1_110	1.009±0.01	0.980±0.07	< **0.006**
	AR1_11	1.079±0.01	1.046±0.05	< **0.001**
	AR2_351	0.932±0.01	0.897±0.04	< **0.001**
	AR2_352	0.907±0.01	0.853±0.03	< **0.001**
	AR2_353	1.024±0.01	0.997±0.03	< **0.001**
	AR2_354	0.963±0.01	0.916±0.03	< **0.001**
	AR2_355	0.920±0.01	0.869±0.03	< **0.001**
	AR2_356	0.957±0.01	0.908±0.03	< **0.001**
	AR2_357	0.942±0.02	0.891±0.03	< **0.001**
	AR2_358	1.045±0.01	0.982±0.04	< **0.001**
	AR2_359	1.065±0.01	0.987±0.04	< **0.001**
	AR2_360	0.969±0.01	0.919±0.03	< **0.001**
	AR2_362	1.027±0.01	0.976±0.03	< **0.001**
	AR2_363	0.971±0.01	0.911±0.04	< **0.001**
	AR2_364	0.967±0.01	0.893±0.03	< **0.001**
	AR2_36	0.982±0.01	0.925±0.04	< **0.001**
ARMA	ARMA_100	1.004±0.01	0.931±0.04	< **0.001**
	ARMA_101	1.058±0.01	0.963±0.03	< **0.001**
	ARMA_102	1.077±0.02	0.973±0.05	< **0.001**
	ARMA_10	1.127±0.01	1.014±0.05	< **0.001**
	ARMA_1	1.060±0.02	0.977±0.05	< **0.001**
	ARMA_2	0.922±0.01	0.854±0.02	< **0.001**
	ARMA_230	1.024±0.01	0.931±0.03	< **0.001**
	ARMA_231	1.043±0.01	0.959±0.04	< **0.001**
	ARMA_69	1.146±0.01	1.104±0.07	< **0.001**
	ARMA_70	1.332±0.01	1.293±0.07	< **0.001**
	ARMA_71	1.263±0.01	1.200±0.07	< **0.001**
	ARMA_75	1.289±0.02	1.222±0.06	< **0.001**
	ARMA_77	1.020±0.01	0.980±0.06	< **0.001**
MA	MA1_113	0.926±0.01	0.900±0.04	< **0.001**
	MA1_114	0.978±0.01	0.945±0.04	< **0.001**
	MA1_115	1.035±0.01	1.003±0.04	< **0.001**
	MA1_116	0.962±0.01	0.937±0.05	< **0.001**
	MA1_117	1.018±0.01	0.972±0.05	< **0.001**
	MA1_118	1.059±0.01	1.002±0.04	< **0.001**
	MA1_119	0.946±0.01	0.933±0.05	0.060
	MA1_120	1.045±0.01	1.003±0.05	< **0.001**
	MA1_121	1.078±0.00	1.031±0.06	< **0.001**
	MA1_122	1.021±0.01	1.004±0.03	< **0.001**
	MA1_123	0.924±0.01	0.901±0.04	< **0.001**
	MA1_12	0.983±0.01	0.943±0.04	< **0.001**

TABLE VII

ARTIFICIAL ANT RESULTS

Method	*Average of Consumed Food ± Deviation*	*Success %*	*Average Time*	*p–value*
Classical GP	87.61 ± 4.64	98.44	562.84	< **0.001**
New GP	70.87 ± 12.32	79.63	599.80	

VII. CONCLUSIONS AND FUTURE WORKS

This work presented a proposal of a new approach of Genetic Programming algorithm, based on the concepts of Evolutionary Strategies and some related studies about the problems and improvement of performance of the GP technique [5], [3], [4], [18], [19].

Preliminary experiments were conducted using problems

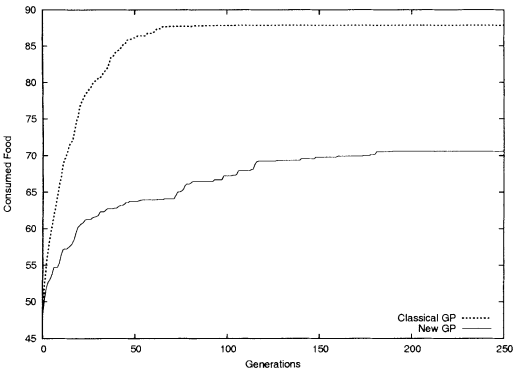

Fig. 2. Artificial Ant – Progress of the evolutionary Process

of different domains. Two instances of the symbolic regression problem: the Binomial–3 problem (a tunably difficult problem) and the Time Series problem (an application of symbolic regression), and the classical Santa Fe Artificial Ant problem.

In the first experiment, with the Binomial–3 problem, we obtained good results with the new approach. The improvement comparing the results with the classical GP, it was about 50%, in the most difficult level of the Binomial–3 problem. The evolution of the mean fitness presented and the paired t test results confirmed that the new approach in this problem is more efficient than the classical GP.

In the second experiment using Time Series we obtained better fitness results with the new approach also. The improvement was not so bigger than in the first experiment, but it shows us that the GP is an efficient technique to forecast Time Series.

The last experiment, with the Santa Fe Artificial Ant problem, shown better results with the classical GP. The Santa Fe Artificial Ant problem is quite different of the Symbolic Regression domain. The set of terminals and functions work in a different way from the sets of the Symbolic Regression problem. These facts suggest us that to this specific domain of problem, a new approach must to be implemented to obtain good results.

Considering the results of the experiments, we can conclude that the concepts of the Evolutionary Strategies techniques can be aggregated to the classical GP, and the algorithm constructed can perform in general in a better or equivalent way.

Future works include the incorporation of others concepts of Evolutionary Strategies in the algorithm, the creation of new mutation operators, a more detailed test using Time Series from different statistical models to confirm the results of this approach, a better study about the factors and reasons that the Santa Fe Artificial Ant problem did not get good results using the proposed algorithm and a creation of a new specialization of the proposed method considering the different domains presented.

References

[1] F. Banzhaf, W. Nordin, P. Keller, and F. D. Francone. *Genetic Programming: An Introduction*. Morgan Kaufmann, 1998.

[2] T. Bäck, G. Rudolph, and H. P. Schewell. Evolutionary programming and evolution strategies: similarities and differeces. In *Annual Conference of Evolutionary Programming*, pages 11–22. San Diego, USA, 1993.

[3] J. M. Daida. Towards identifying populations that increase the likelihood of success in genetic programming. In *GECCO-2005 - Genetic and Evolutionary Computation Conference*, pages 1627–1634. Association for Computing Machinery, Washington, D.C., USA, June 2005.

[4] J. M. Daida, R. R. Bertram, J. A. Polito 2, and S. A. Stanhope. Analysis of single-node (building) blocks in genetic programming. In *Advances in Genetic Programming 3*. The MIT Press, Cambridge, 1999.

[5] J. M. Daida, R. R. Bertram, S. A. Stanhope, J. C. Khoo, S. A. Chaudhary, O. A. Chaudhri, and J. A. Polito 2. What makes a problem gp-hard? analysis of a tunably difficult problem in genetic programming. *Genetic Programming and Evolvable Machines*, 2:165–191, June 2001.

[6] C. Darwin. *On the origin of species by means of natural selection or the preservation of favored races in the struggle for life*. Murray, London, UK, 1859.

[7] M. Dianati, I. Song, and M. Treiber. An introduction to genetic algorithms and evolution strategies. Technical report, University of Waterloo, Ontario, Canada, 2002.

[8] D. E. Goldberg. *Genetic Algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.

[9] J. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.

[10] A. Hui. Using programming to perform time series forecasting of stock prices. In *Genetic Algorithms and Genetic Programming*, pages 83–90. John Koza, Stanford, California, USA, 2003.

[11] D. Jackson. Dormant program nodes and the efficiency of genetic programming. In *GECCO-2005 - Genetic and Evolutionary Computation Conference*, pages 1745–1751. Association for Computing Machinery, Washington, D.C., USA, June 2005.

[12] M. A. Kaboudan. Genetic programming prediction on stock prices. *Journal Computational Economics*, 16:207–236, 2000.

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

[14] A. Kusiak. Evolutionary computation and data mining. In *SPIE Conference on Intelligent Systems and Advanced Manufaturing*, pages 1–10. Boston, MA, USA, November 2000.

[15] B. S. D. Minglei. Time series predictability. Master's thesis, Marquette University, Milwaukee, USA, 2002.

[16] P. A. Morettin and C. M. C. Toloi. *Análise de Séries Temporais*. Edgar Blucher, Brazil, 2004. (In Portuguese).

[17] R. J. Povinelli. *Time Series Data Mining: Identifying Temporal Patterns for Characterization and Prediction of Time Series Events*. PhD thesis, Marquette University, Milwaukee, USA, 1999.

[18] S. Silva and E. Costa. Resource-limited genetic programming: The dynamic approach. In *GECCO-2005 - Genetic and Evolutionary Computation Conference*, pages 1673–1680. Association for Computing Machinery, Washington, D.C., USA, June 2005.

[19] J. Stevens, R. B. Heckendorn, and T. Soule. Exploiting disruption aversion to control code bloat. In *GECCO-2005 - Genetic and Evolutionary Computation Conference*, pages 1605–1612. Association for Computing Machinery, Washington, D.C., USA, June 2005.

[20] W. N. Venables, D. M. Smith, and the R Development Team. *An introduction to R*. R Development Core Team, October 2005.

[21] S. C. Wheelwright and S. Makridakis. *Forecasting Methods for Management*. John Wiley & Sons Inc, New York, USA, 1985.

[22] D. Zongker and B. Punch. *Lil-gp 1.0 User's Manual*. Michigan State University, July 1995.