Time Series Prediction Using Grammar-directed Genetic Programming Methods

Santi Garcia Carbajal

***Abstract*— We use a modified Genetic Programming System to predict the values of the reduced set proposed as benchmark for the '07 Neural Forecasting Contest. Genetic Programming is a well known method used in symbolic regression, and classification, based in the evolution of arithmetic expressions according to a fitness function. We introduce here a grammar into the Genetic system, to let us use conditional expressions inside the syntactic trees representing the solutions to the problem. Additionally, we employ GA-P methods to automatically obtain constants inside the expressions. Our results proof the known power of Genetic Programming as a tool for solving Symbolic Regression problems, as the obtained expressions fit acceptably the proposed series. For the predicted values, some of them seem promising while others present too flat behaviors.**

I. Introduction. Genetic Programming

Genetic programming [1], [2] is a domain-independent problem-solving approach in which computer programs are evolved to solve a problem, or to find the best approximated solution. It is is based on the Darwinian *Theory of Evolution* -often paraphrased as "survival of the fittest"-.

The main idea consists of applying a Genetic Algorithm [3] on a population of computer programs of varying sizes and shapes. Although more complex approaches involving memory and have been proposed, in a typical Genetic Programming system the individuals are functions that depend on a certain number of variables, represented by their syntactic trees. An individual is a tree-like structure formed by two types of genes:

- Functions, and
- Terminals

Terminals are leaves in the tree (nodes without branches), and functions are nodes with a variable number of children. The function's children provide the arguments for the function.

After creation of an initial population of randomly generated trees, genetic operators are applied until the population converges to a solution (may be a suboptimal one) for the problem. Genetic operators are more complex in a Genetic Programming approach than in a canonical Genetic Algorithm, but are based on simple operations on the trees representing solutions, and will be further explained in this section.

In a Genetic Programming system, the steps towards the solution of a problem can be formulated in the following manner (see figure 1):

Santi Garcia is a Senior Lecturer at the Department of Computer Science, Oviedo University, Gijon, Spain (phone: 985-182-487; fax: 985-182-156; email: sgarcia@uniovi.es).

1) Generate an initial population of random individuals, composing the terminal (variables) and non terminal (functions) symbols. The quality of the solutions included in the initial population will be typically very poor.
2) Iteratively perform the following sub-steps until the maximum number of generation is reached, or the termination criterion has been satisfied:
   a) Execute each program in the population and assign it a numerical fitness value using the fitness function.
   b) Create a new population of computer programs by applying following genetic operators.
      - Reproduction: a randomly chosen individual is copied from the current generation to the next.
      - Crossover: operates on two programs included in the population, and produces two child programs. Two random nodes are selected from within each program and then the resultant "sub-trees" are swapped, generating two new programs. These new programs become part of the next generation of programs to be evaluated. figure 2 shows two randomly chosen individuals in the population, and the crossover points (randomly chosen, too), applying the operator. The offspring produced by the crossover operator is shown in the same figure.
      - Mutation: Create one new computer program from one existing program by mutating a randomly chosen node. See figure 3.
3) If the termination criterion is satisfied, or the maximum number of generations is reached, the current best individual in the population is proposed as the solution to the problem.

A. GA-P techniques.

GA-P technique [4] is an hybrid between genetic algorithms and genetic programming, which was first used in symbolic regression problems. Individuals in GA-P have two parts: a tree based representation and a set of numerical parameters. Different from canonical GP, the terminal nodes of the tree store not only numbers and variables, but linguistic identifiers that are pointers to a set of real numbers (see figure 4.) The behavior of the GA-P algorithm is mainly due to its crossover operator. Either or both parts of the individual may be selected and crossed. We have previously employed GA-P algorithms in the identification and control of complex dynamical processes, and in classification problems [5].

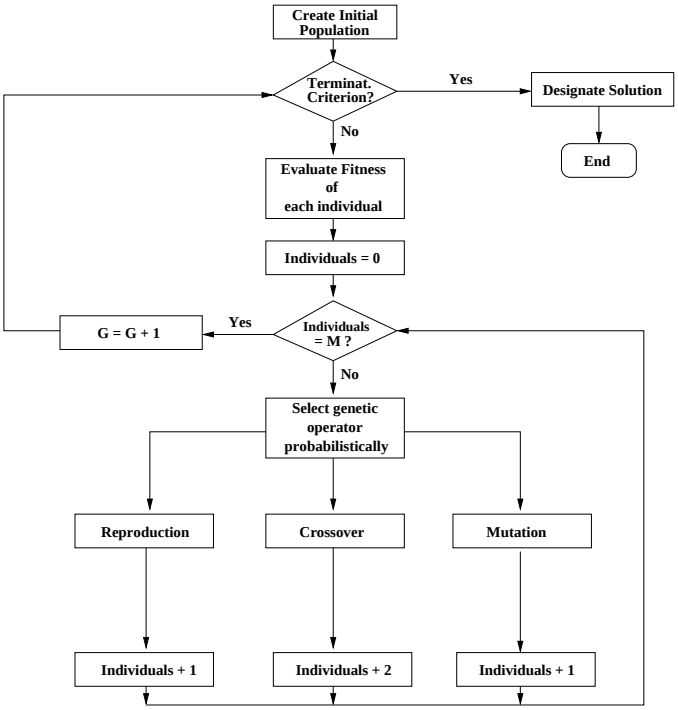

Fig. 1. Genetic Programming flowchart

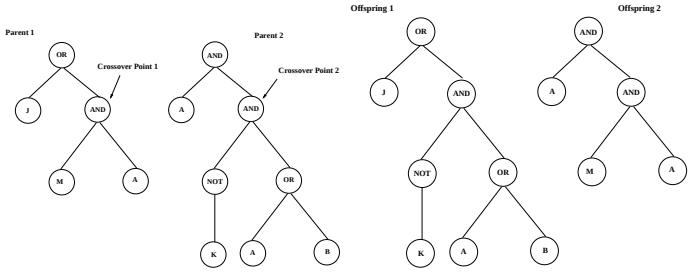

Fig. 2. Progenitors for the crossover operator (left). Offspring produced by the operator (right)

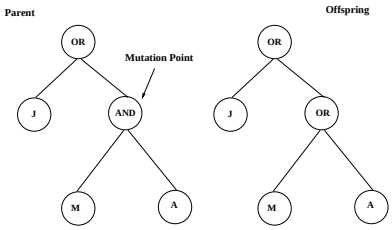

Fig. 3. Effect of the mutation operator

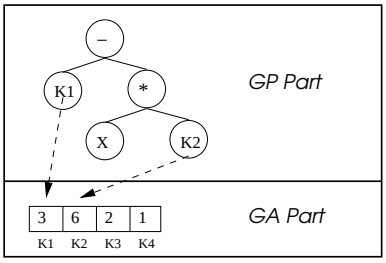

Fig. 4. Representation of a generic individual in GA-P techniques. GA-P individuals consist of two parts: a tree and a set of numerical constants. Genetic operators can take place in the tree (GP part), or in the set of constants (GA part)

in the next section.

TABLE I
PARAMETER SET

Objective	Time series prediction
Function Set	+-,*,/,sin,cos, cond
Terminal Set	$\{X_0..X_9\}$
Fitness Cases (F_c)	1 for each series
Fitness Function	$F(\tau) = \sum_{t=0}^{t=K} \|P(t) - S(t)\|$
Population Size	M=10000
Generations	G=200
Mutation Rate	0.05
Maximum Initial Depth	10
Termination Criterion	g=G

B. Experimental Setup

In any Genetic Programming method, the following parameters of the algorithm must be defined:

1) the terminal set, T.
2) the function set, F.
3) the goal, and a fitness function capable of evaluate the performance of any valid individual.
4) the set of parameters of the algorithm.
5) the method for designating a solution and the criterion for terminating a run.

All these parameters are listed in table I. All the experiments were performed with a population of ten thousand individuals, finishing when the maximum number of generations is reached (current=g=200). The initial maximum depth of the syntactic trees is set to a value of ten. The obtained arithmetic expressions include addition, subtraction, multiplication, and division, plus the C-style conditional operator. The fitness function, that allows us to determine how good is an individual in the prediction work is explained

C. Fitness Function. Forecasting method

Let $S(t)$ be the value of the series to be predicted at time t. We calculate the predicted value for that point, $P(t)$, as the result of the evaluation of the best individual in the population (syntactic tree), mapping the previous ten values of S to the variables appearing in the tree. See figure 5.

Let $P(t)$ be the predicted value for the same time step ($P(t)$ is actually a function of the previous ten values of $S(t)$). We call to *O*bservation Window the set of values of S that we use to predict the next value, at any point inside the interval of study. The Fitness Function of a syntactic tree, τ whose terminal symbols will represent some of the values of the *O*bservation Window, is calculated as follows:

$$F(\tau) = \sum_{t=0}^{t=K} |P(t) - S(t)|, \quad (1)$$

being K the maximum length of the interval. The goal for the Genetic System is to minimize the value of F, searching for

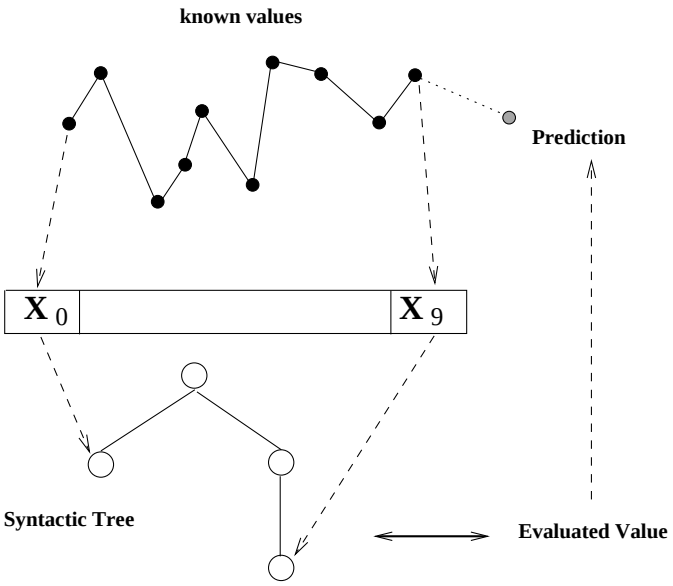

Fig. 5. Forecasting. Fitness function. The previous values of the series are mapped into the terminal nodes of the syntactic tree, and a prediction value is computed.

arithmetic expressions that fit the proposed series as much as possible.

At any step of the prediction, we use the values of the previous ten values of the series. Consequently, we can make no prediction for these ten initial values. After this point is passed, the window is displaced at each step. Figure 5 shows how the known values of the series are used as input for the evaluation of the syntactic trees, and the prediction is calculated.

The observation window is displaced over all the known range of the series. At each point, the difference between the predicted and the actual value of the series is calculated, and *F*itness Function. Best individual in the population will be the one with lower error. See figure 6.

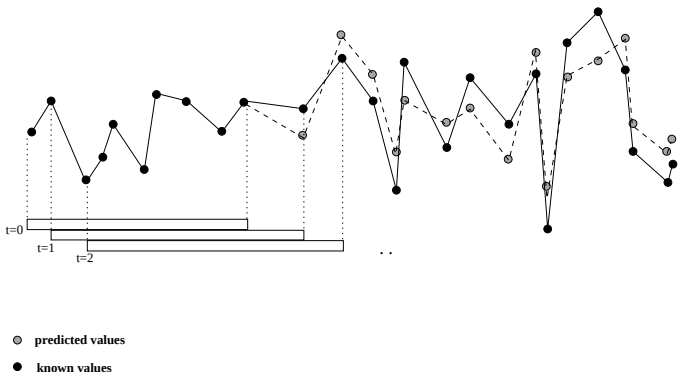

Fig. 6. Forecasting. Displacement of the observation window. The first ten values are used to calculate the first prediction. The difference is computed and the window is then displaced.

When the *O*bservation Window reach the first point outside the provided data, the predicted values are used themselves, and here is where the obtained expressions behave worse, apparently.

II. Main Results

Figures 7,8 and 9 show the results we have obtained for each one of the eleven series included in the reduced dataset. Each plot shows the actual values of the series, and the predicted values inside and outside the known interval. Due to the lack of knowledge about the actual values of the series outside the training interval, we can only say that results for series 1,2, and 4 seem more promising than the others, being apparently bad for series 6, 7, and 8. Inside the training intervals, the behavior of the obtained expressions is very similar for all the examples.

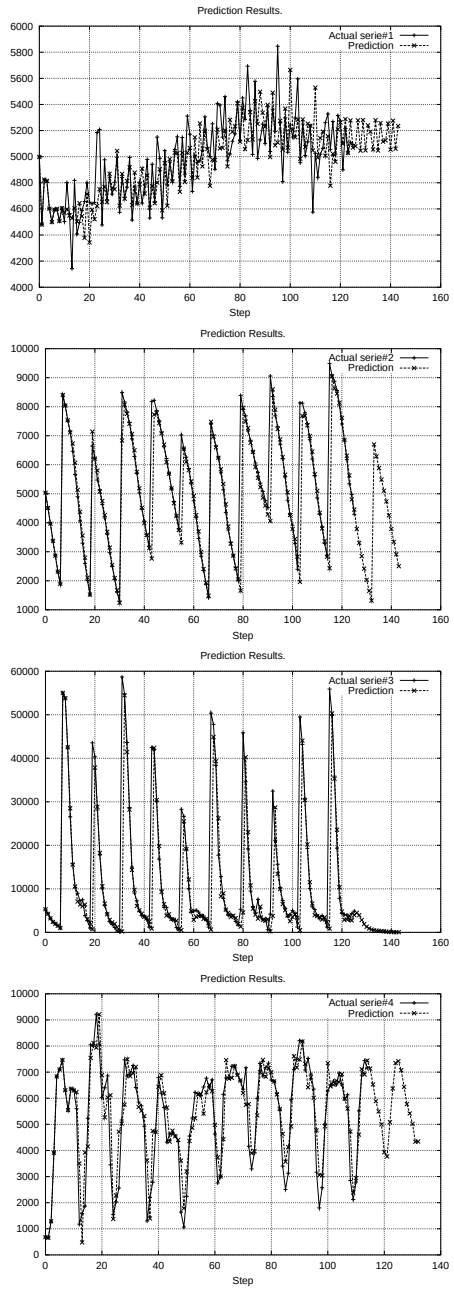

Fig. 7. Preliminary results

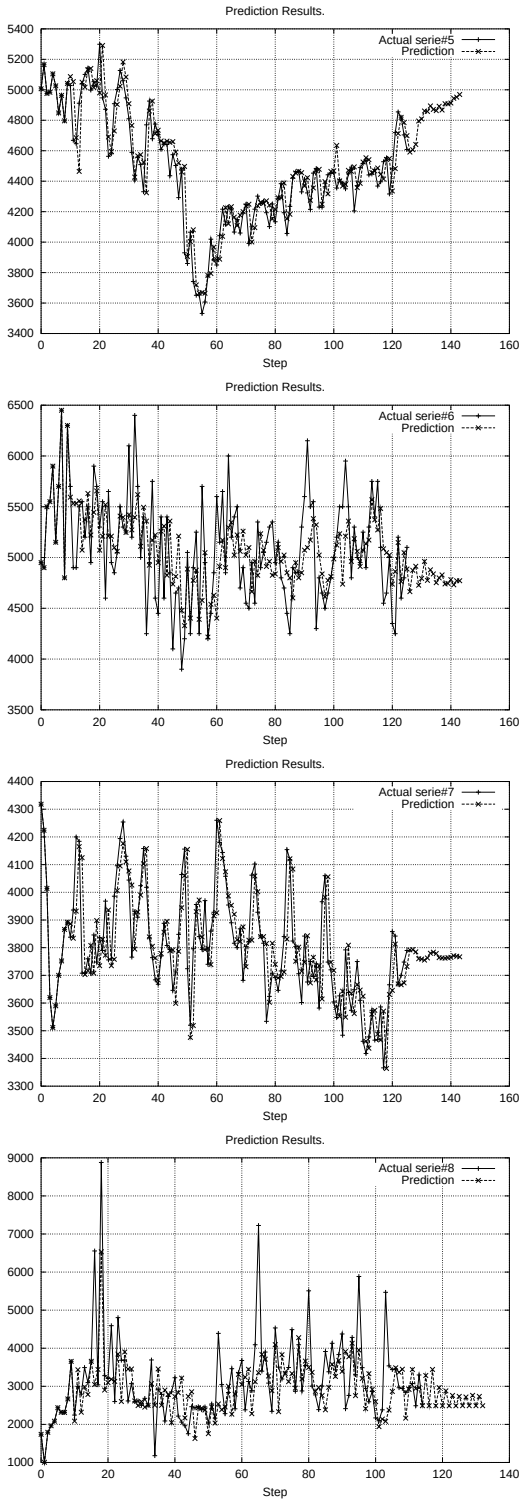

Fig. 8. Preliminary results

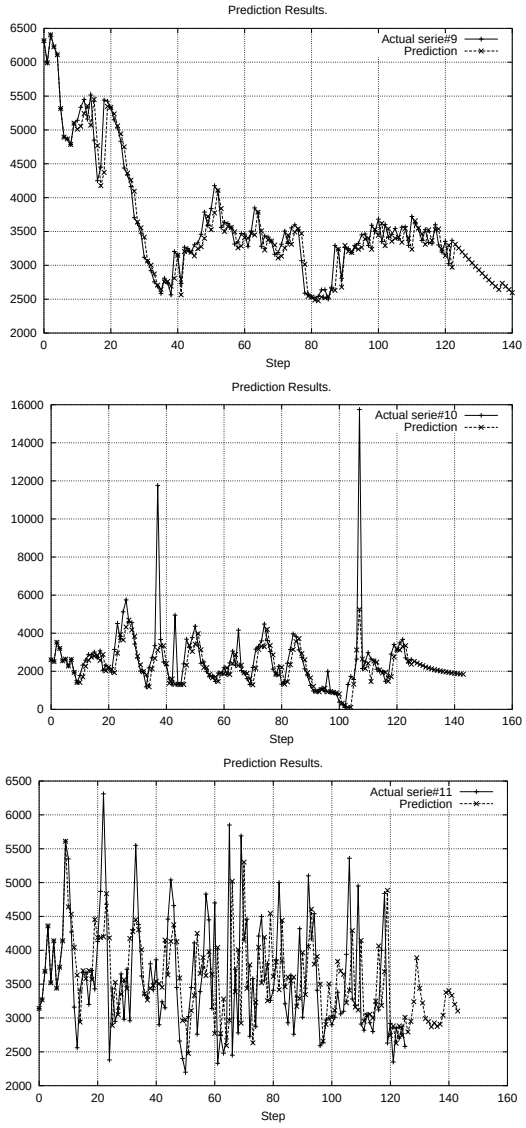

Fig. 9. Preliminary results

III. Conclusions

References

[1] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA, USA: MIT Press, 1992.

[2] ——, "Genetic programming: On the programming of computers by means of natural selection," *Statistics and Computing*, vol. 4, no. 2, June 1994.

[3] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[4] L. M. Howard and D. J. D'Angelo, "The GA–P: A genetic algorithm and genetic programming hybrid," *IEEE Expert*, vol. 10, no. 3, pp. 11–15, June 1995.

[5] S. Garca, F. Gonzlez, and L. Snchez, "Evolving fuzzy rule based classifi ers with GAP: A grammatical approach," in *Genetic Programming, Proceedings of EuroGP'99*, ser. LNCS, R. Poli, P. Nordin, W. B. Langdon, and T. C. Fogarty, Eds., vol. 1598. Goteborg, Sweden: Springer-Verlag, 26-27 May 1999, pp. 203–210.