FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 15, No. 2, Aug. 2002, 265-279

Time Series Prediction Using a Recursive Algorithm of a Combination of Genetic Programming and Constant Optimization

Witthaya Panyaworayan and Georg Wuetschner

Abstract: In this paper we present a prediction process of Time Series using a combination of Genetic Programming and Constant Optimization. The Genetic Programming will be used to evolve the structure of the prediction function, whereas the Constant Optimization will determine the numerical parameters of the prediction function. The prediction process is applied recursively. In each recursion step, a sub-prediction function is evolved. At the end of the iteration all sub-prediction functions form the final prediction function. The avoiding of a major problem in the prediction called over-fitting is also described in this article.

Keywords: Time series prediction, genetic programming, sunspot numbers.

1 Introduction

Genetic Programming (GP) [1] is a branch of the Evolutionary Computation techniques [2] where a computer program with the desired behaviour is automatically produced. GP mimics the process of biological evolution by using a mechanism of natural selection and genetic variation, guides the searching by using various genetic operators and implements the principle of *survival of the fittest.*

One of its application areas is *Time Series Prediction.* In Time Series Prediction, a model considered as the best possible approximation of the dynamic system is estimated, based on observation data.

Manuscript received March 16, 2002.

The authors are with Institute for Informations Engineering at the Federal Armed Forces University Munich, D-85577 Neubiberg, Germany (e-mails: [e31bwitt, Georg.Wuetschner]@unibw-muenchen.de).

Some studies of GP used for Time Series Prediction have already been undertaken. Koza introduced the first studies on the discovery of programs for fitting time series data in [1]. His study was based on the short term prediction of the so called logistic map. The further studies are published in [3, 4, 5, 6]. To improve the efficiency of GP, we can combine GP with a Constant Optimization, such as Least Squares, Simulated Annealing or Genetic Algorithm. This idea can be found in [7, 8, 9, 10]. GP itself is used to evolve the structure of a neural network or of the polynomial of a complex function. This second idea can be found in [11, 12]. In [13] GP is used recursively to model time series. The main idea lies in a recursive regression scheme through which multiple basis prediction functions are derived. It is based on the assumption that the dynamics of a time series comprise both the stochastic part and the deterministic part.

The aim of this paper is to present the process of prediction of a Time Series using a recursive algorithm of a combination of GP and Constant Optimization. The paper is organized as follows: Section 2 defines the Time Series Prediction problem we address. Section 3 describes GP in general. Section 4 describes the prediction process with the combination of GP and the Constant Optimization. A experiment and its result are provided in Section 5. A conclusion is drawn in Section 6.

2 Time Series Prediction

In a Time Series Prediction, we are given a sequence of past values of a random variable and want to forecast the future values of the variable. As described in [14], there are two kinds of predictions:

Single-Step Prediction: The term *Single-Step Prediction* is used to predict one value $\hat{x}_{t+1}$ of the time series when all inputs x_t, x_{t-1}, x_{t-2}, ..., x_{t-m} are given (see eq. (1)).

$$\hat{x}_{t+1} = \hat{f}(x_t, x_{t-1}, x_{t-2}, ..., x_{t-m}) \; ; m < t, \tag{1}$$

where $\hat{f}$ is the prediction function for one prediction step.

Iterated Single-Step Prediction: The term *Iterated Single-Step Prediction* is used to predict further than one step into the future. In the *Iterated Single-Step Prediction*, the predicted output is fed back as input for the next prediction. All other input units are shifted back one unit. Hence, the input

consists at least partly of predicted values as opposed to observations of the original time series (see eq. (2)).

$$\begin{aligned}
\hat{x}_{t+1} &= \hat{f}(x_t, x_{t-1}, ..., x_{t-m}) \; ; m < t \\
\hat{x}_{t+2} &= \hat{f}(\hat{x}_{t+1}, x_t, ..., x_{t-m+1}) \; ; m < t \\
&\vdots \\
\hat{x}_{t+k} &= \hat{f}(\hat{x}_{t+k-1}, \hat{x}_{t+k-2}, ..., x_{t-m+k-1}) \; ; m < t \text{ and } k \geq 1
\end{aligned} \tag{2}$$

The predictive accuracy of the function $\hat{f}_k$ is evaluated by estimating the *normalized mean squared error (NMSE)*, defined as follows:

$$E_{\text{NMSE}_k} = \frac{1}{\sigma_x^2 \cdot N} \sum_{t=0}^{N-1} (x_{t+k} - \hat{x}_{t+k})^2, \tag{3}$$

where N is the number of the actual value that is used to evaluate the predictive accuracy, x_{t+k} is the actual value at time $t+k$, $\hat{x}_{t+k}$ is the predicted value at time $t+k$, k is the prediction step with $k \geq 1$ and σ_x is the standard deviation of the actual values $x_k, x_{1+k}, x_{2+k}, ..., x_{N-1+k}$.

With this definition, the problem of Time Series Prediction is reduced to find the predictor $\hat{f}$ that minimizes its NMSE value E_{NMSE_k}. If $E_{\text{NMSE}_k} = 0$, the prediction is perfect, $E_{\text{NMSE}_k} = 1$ indicates that the performance is no better than a constant predictor $\hat{x}_{t+k} = \bar{x}$, where $\bar{x}$ is the average of the actual values.

3 Genetic Programming

GP was developed by J. R. Koza with the purpose to produce an executable computer program [1, 15]. The computer program in GP is usually called an *individual*. In standard GP, this individual is represented as an expression tree and can be used for structural optimization rather than numerical parameter optimization.

Fig. 1 summarizes the implemention of the GP paradigm. The termination criteria for a run of GP usually consists of either satisfying a problem-specific *success predicate* or completing a specified maximum number of generations to be run $G_{\max}$. The further important explanations are as follows:

Function Set: The function set $\mathcal{F}$ consists of the functions of the program. The functions may be standard mathematical functions, standard arithmetic operations, standard programming operations, logical functions, or domain-specific functions.

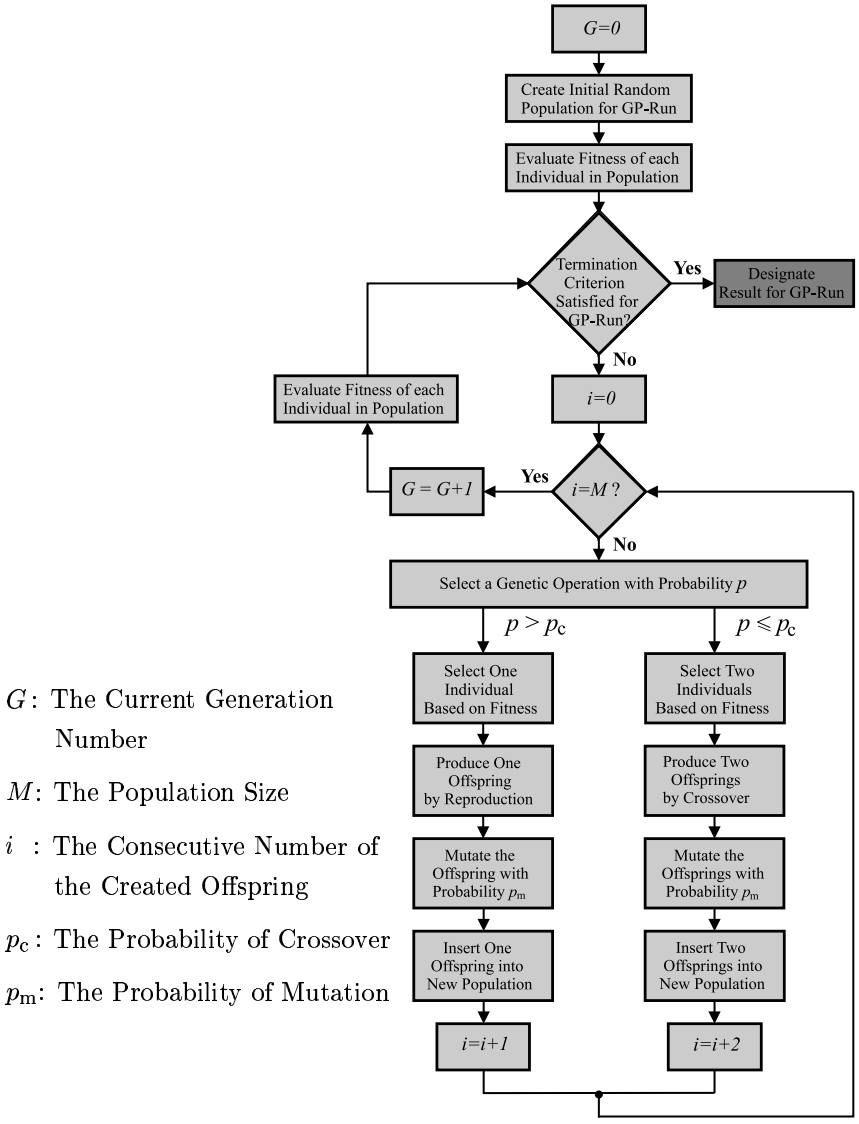

Fig. 1. Flowchart for Genetic Programming

Terminal Set: The terminal set $\mathcal{T}$ comprises the variables and constants that correspond to the inputs of the program.

Initializing Tree Structures: Tree Structures of GP individuals are built randomly from the terminal set $\mathcal{T}$ and function set $\mathcal{F}$.

Fitness: The fitness is the measure used by GP during a GP-run of how well an individual has learned to predict the outputs from the inputs, that is the features of the learning domain. The best program is the one with the smallest prediction error.

Selection: The fitness of each individual in a population is used as the basis for selecting members to be the parents of the next generation of potential solutions. In a *tournament selection*, a fixed number of individuals from the population, called the *tournament size*, is selected randomly, and a selective competition takes place. The better individuals in the tournament are chosen to be the parents.

Crossover: In the crossover operation, two parents are sexually combined to form two new individuals or offsprings. The parents are chosen by a selection process. The creation of the offsprings with the crossover operation is accomplished by randomly deleting the crossover fragment of the first parent and then randomly inserting the crossover fragment of the second parent. The second offspring is produced in an analogous manner. For example consider the two parents in Fig. 2.

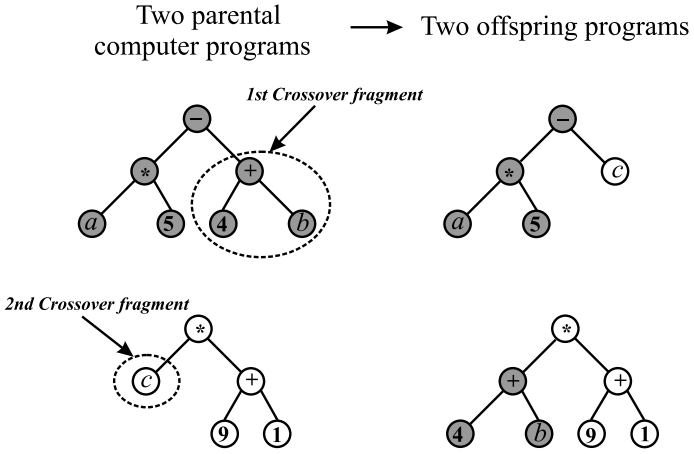

Fig. 2. Crossover in Genetic Programming

Mutation: In the mutation operation, a fragment of the offspring is chosen randomly and replaced by another randomly generated fragment.

4 Prediction Process

To easily understand the prediction process, we firstly present the Simple Prediction Process and afterwards the Recursion Prediction Process. Generally, both prediction processes are trained to evolve a prediction function for Single-Step Prediction of the times series. By applying the Iterated Single-Step Prediction to this evolved prediction function, we can predict times series further than one step into future.

4.1 Simple Prediction Process

The *Simple Prediction Process* can be splitted into two parts.

Evolution of the Prediction Function by GP: In this part, GP is trained with training data to evolve the best prediction function $\hat{f}_{\mathrm{GP}}$. The function set $\mathcal{F}$ can comprise the following mathematical functions:

$$\mathcal{F} = \{\sin, \cos, \log, \exp, +, -, \times, \div\} \tag{4}$$

Elements of the terminal set $\mathcal{T}$ are such as:

$$\mathcal{T} = \{x_t, x_{t-1}, x_{t-2}, ..., x_{t-m}, -10.0, -9.5, -9.0, ..., 9.0, 9.5, 10.0\}, \tag{5}$$

where $x_t, x_{t-1}, x_{t-2}, ..., x_{t-m}$ are the values of the time series at different previous time steps and $-10.0, -9.5, ..., 9.5, 10.0$ are simple constant values.

Constant Optimization: After the evolution of a prediction function by GP, we insert coefficients into the evolved prediction function $\hat{f}_{\mathrm{GP}}$. Afterwards, we optimize these coefficients.

The coefficient insertion into mathematical functions and variables is such as:

$$\begin{array}{lcllcl} \sin(x) & \rightarrow & a \cdot \sin(x), & \cos(x) & \rightarrow & a \cdot \cos(x), \\ \log(x) & \rightarrow & a \cdot \log(x), & \exp(x) & \rightarrow & a \cdot \exp(x), \\ x_t & \rightarrow & a \cdot x_t, & x_{t-1} & \rightarrow & a \cdot x_{t-1}, \\ & \cdots & & x_{t-m} & \rightarrow & a \cdot x_{t-m}, \end{array}$$

where a is the inserted coefficient.

The optimization of the inserted coefficient can be done by several different methods. However, only *Genetic Algorithm (GA)* methods are used

in this paper. Like GP, GA, as described in [16], is a global optimization method based on natural evolution. GA makes heavy use of crossover. In GA, the individual in the population represents a legal solution of the problem and is composed of a string of genes. These genes can be represented by binary numbers $\{0, 1\}$ or real numbers. In our examples, the binary numbers is used.

Fig. 3 shows the problem representation as a binary string used by GA, when the coefficients a, b and c in function $y = ax^2 + bx + c$ are optimized, and an example of the crossover operator in GA.

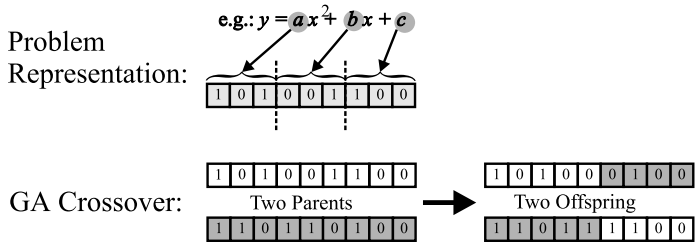

Fig. 3. *Top*: Problem representation as a binary string used by Genetic Algorithm *Bottom*: Crossover in Genetic Algorithm

4.2 Recursion Prediction Process

The *Recursion Prediction Process* consists of several steps, each of which represents a Simple Prediction Process. The training data of each step is the prediction error from previous step.

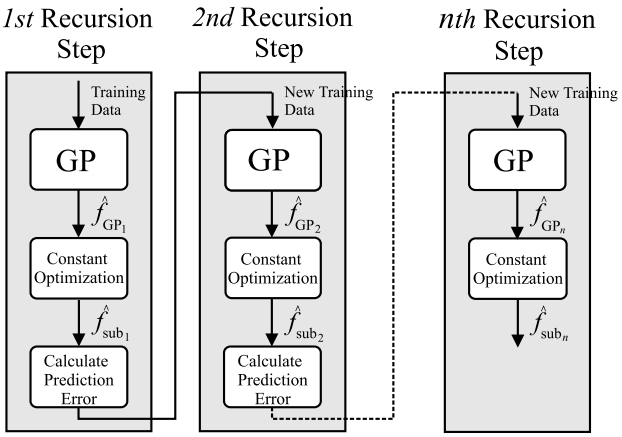

Fig. 4. Flowchart of the Recursion Prediction Process

Fig. 4 shows a flow chart of the Recursion Prediction Process. The Recursion Prediction Process works as follows:

***1st* Recursion Step:** Firstly, the prediction function $\hat{f}_{\mathrm{GP}_1}$ is evolved by GP. Afterwards, we insert coefficients in the prediction function $\hat{f}_{\mathrm{GP}_1}$ and optimize them with GA. We then obtain the sub-prediction function $\hat{f}_{\mathrm{sub}_1}$. At the end of the first recursion step, we determine the prediction error that occurs when the sub-prediction function $\hat{f}_{\mathrm{sub}_1}$ is used to predict the training data. This prediction error will now be used as the new training data for the second recursion step.

***2nd* Recursion Step:** The second recursion step firstly works also as a Simple Prediction Process, but as training data for this step, the prediction error of the first step is used. That way, a second sub-prediction function $\hat{f}_{\mathrm{sub}_2}$ is evolved. Referring to the original training data, this means that the collective prediction function corresponds to the sum of the first sub-prediction function $\hat{f}_{\mathrm{sub}_1}$ and the second sub-prediction function $\hat{f}_{\mathrm{sub}_2}$ at this point. To further improve this collective prediction function, the coefficients a_0, a_1 and a_2 are inserted, so that

$$\hat{f}_{\mathrm{sum}_2} = a_0 + a_1 \cdot \hat{f}_{\mathrm{sub}_1} + a_2 \cdot \hat{f}_{\mathrm{sub}_2} \tag{6}$$

Next, a_0 a_1 and a_2 are optimized by GA. The prediction function $\hat{f}_{\mathrm{sum}_2}$ in eq. (6) is now used to calculate the new prediction error representing the new training data for the third recursion step.

The proceeding of the Recursion Prediction Process reiterates until either the maximal number of recursion step is reached or no further better sub-prediction function is found. At the end of this prediction process, we will obtain a prediction function $\hat{f}_{\mathrm{sum}_n}$, as shown in eq. (7).

$$\hat{f}_{\mathrm{sum}_n} = a_0 + a_1 \cdot \hat{f}_{\mathrm{sub}_1} + a_2 \cdot \hat{f}_{\mathrm{sub}_2} + ... + a_n \cdot \hat{f}_{\mathrm{sub}_n} \tag{7}$$

Note: the coefficients $a_0, a_1, a_2, ..., a_n$ are optimized in every single recursion step.

Additionally, if time permits, for each recursion step many GP-runs can be executed and only the best prediction function among these GP-runs will be chosen as the result of the recursion step.

4.3 Over-fitting Problem

A main problem in prediction with GP is that the evolved prediction function can well predict data inside the training data. However, it may be not suitable for a prediction outside the training data. This problem is called *over-fitting*. Especially, over-fitting occurs when training data is noised, since the prediction process attempts to fit the noise, too.

One of the most simple and most widely used means of avoiding over-fitting is *early stopping* as shown in [17]. This method is used in this paper when GP and Constant Optimization are applied. The simple early stopping method works as follows:

- We separate the training data into two subsets, a *training set* and a *validation set*. The validation set has to be independent from the training set. The separation of training data can be random or determined. The data outside the training data is named *test set*. The test set is used to assess the performance of the evolved prediction function when the training has finished. The prediction error inside the training set, validation set and test set are called *training error*, *validation error* and *test error* respectively.

- The prediction process trains only on the training set and simultaneously evaluates the validation error of the validation set in any training step. We stop training as soon as the validation error obtains its minimum value and begins to rise. The prediction function with the lowest validation error in the previous step is used as the result of the training.

However, reality is more complex. Real validation error curves almost always have more than one local minimum. Therefore, in the following experiment the training will be stopped if the maximum number of training step is reached.

5 Experiment

In this section the a Time Series of the monthly mean of sunspot numbers is used to assess the prediction process. The series used in this paper is gathered from [18] and was compiled by the US National Oceanic and Atmospheric Administration (NOAA).

In the experiment, the function set $\mathcal{F}$ and the terminal set $\mathcal{T}$ shown in eqs. (4) and (5) were used. Since the number of sunspots visible on the

sun waxes and wanes with an approximate 11-year cycle, the variable m in the terminal set $\mathcal{T}$ should be greater than eleven times twelve[1] and was arbitrarily set to 149.

Tab. 1 shows the most important parameters for controlling the GP-runs in the experiments.

Table 1. Parameter for controlling the GP-runs in the experiment.

Population size M	500
Maximum number of generation to be run $G_{\max}$	50
Probability p_{c} of crossover	0.9
Probability p_{r} of reproduction	0.1
Probability p_{m} of mutation	0.2
Selection method	Tournament
Tournament size	10

The NMSE as described in eq. (3) with a prediction step of $k = 1$ was used not only to evaluate the fitness of each individual in the population, but in the Constant Optimization as well. The fitness value of the best individual then corresponds to the training error. The validation error and test error are also evaluated using eq. (3) with a prediction step of one.

In this experiment, we used the Recursion Prediction Process with a maximum recursion step of five. For each recursion step 50 GP-runs were executed and only the best prediction function among these 50 runs was chosen as the result of the recursion step. Additionally, we evolved a prediction function when preprocessing the training data with a low-pass filter. The implemented low-pass filter is a second-order, digital Butterworth filter with a cutoff frequency of $f_{\mathrm{c}} = 1/6$ year^{-1}.

5.1 Result

The prediction function evolved with filtered training data was found very fast. Instead of five recursion steps, the training was stopped after the third recursion step, since no better sub-prediction could be evolved.

Tab. 2 shows the prediction error of the best prediction functions for original and filtered training data. The test error is clearly greater than the training and validation error in both prediction functions.

Fig. 5 contains the predicted values of the monthly mean of sunspot numbers by Single-Step Prediction compared to the actual values.

[1] Result of 11 years times the number of months per year

Table 2. The error of the best prediction function trained with original training data and filtered training data.

Training data	Training error (10.1956-09.1986)	Validation error (10.1986-09.1996)	Test error (10.1996-02.2001)
Original	$4.77 \cdot 10^{-2}$	$7.54 \cdot 10^{-2}$	$23.24 \cdot 10^{-2}$
Filtered	$7.94 \cdot 10^{-2}$	$9.20 \cdot 10^{-2}$	$16.10 \cdot 10^{-2}$

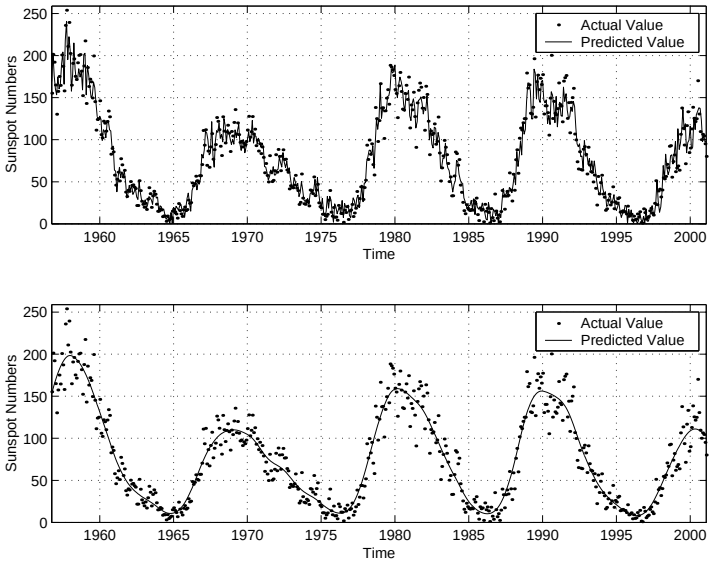

Fig. 5. Comparison between the predicted values of the monthly mean of sunspot numbers by Single-Step Prediction and the actual values. *Top*: usage of original training data, *Bottom:* usage of low-pass filtered training data

The prediction error of an Iterated Single-Step Prediction is shown in Fig. 6. The prediction function trained with filtered training data is clearly suitable for the Iterated Single-Step Prediction. Sunspots are a natural phenomenon. However, the sunspot number is influenced by the precision of the observation due to noise. Therefore the Iterated Single-Step Prediction of time series near the training data is better than the prediction of time series far from it.

Finally, Fig. 7 shows prediction of the monthly mean of sunspot numbers till 08.2008 by the prediction function trained with filtered training data using various start values as compared to the prediction of NOAA. Both predicted curves in Fig. 7 show a similar development and are only little different, predicting the next minimum to occur in 2007.

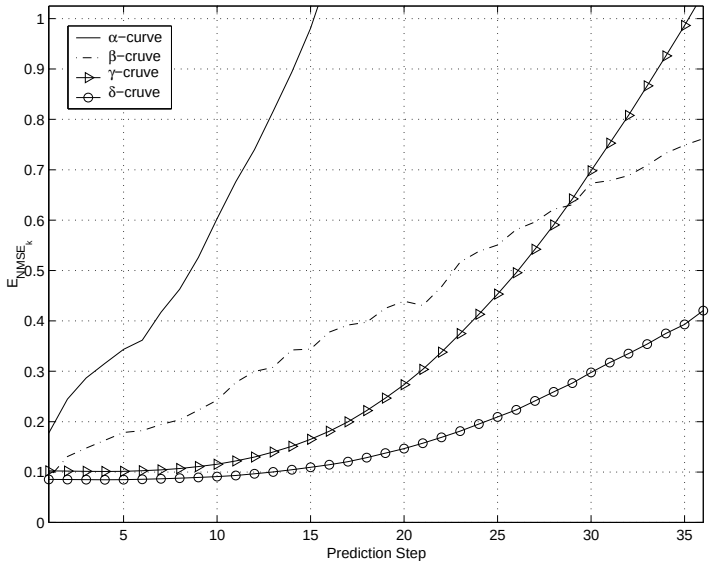

Fig. 6. Prediction error E_{NMSE_k} of an Iterated Single-Step Prediction of the monthly mean of sunspot numbers, where the α-curve is the E_{NMSE_k} of the prediction function of the entire data (from 01.1749 to 02.2001), the β-curve that of the data near training data (from 09.1918 to 02.2001) and the γ- and δ-curve the same as the α resp. β-curve, but of a prediction function trained with filtered training data.

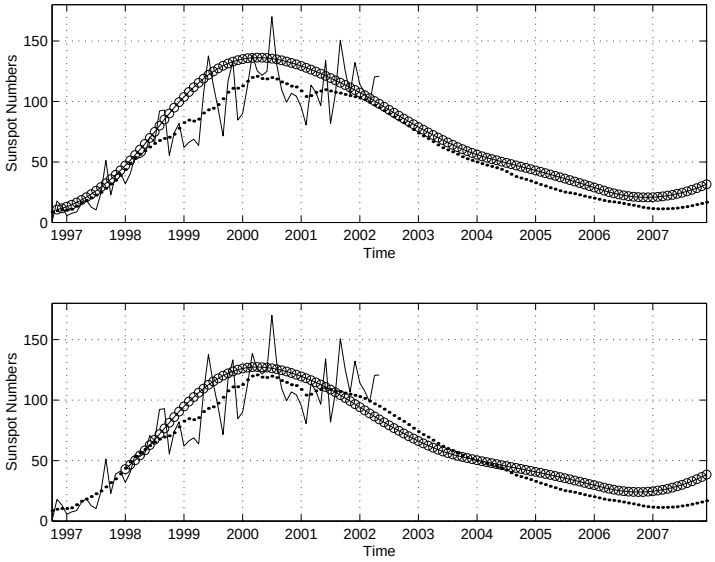

Fig. 7. Prediction of the monthly mean of sunspot numbers till 08.2008 using various start values. *Solid line:* original curve, *dot line:* predicted curve by NOAA, *solid line with circles:* predicted curve with filtered training data

5.2 Discussion

In other prediction processes [19, 20], one firstly has to choose a suitable structure for the prediction function before starting a prediction. The achieved results then depend to a high degree on this structure. GP, on the other hand, will take over this task by itself. GP evolves a structure of the prediction function from the terminal set $\mathcal{T}$ and the function set $\mathcal{F}$, which are set by analysis of the time series or by empirical knowledge. Structural resp. mathematical knowledge about the function to be found is unnecessary. Additionally, the evolved prediction function by GP may contain information of how the time series depends on its inputs.

The Constant Optimization evolves the best numerical parameters for the prediction function. The prediction function will be then recognized more effectively. Since the evolved function is nonlinear, an optimization not sticking to local optima is desired. GA fulfils this condition. To improve the prediction process, each individual in the population must be optimized during the fitness evaluation instead of optimizing only the best function at the end of a GP-run. Of course, this implies that the computational effort will increase.

By using the Recursion Prediction Process, we evolve a sub-prediction function in each recursion step which becomes a part of the final prediction function. That way, none of the best already evolved prediction functions will be lost during a GP-run. At the end of the process all sub-prediction functions together form the prediction function $\hat{f}_{\mathrm{sum}_n}$.

The results from the experiments show that the validation error appears to be greater than the training error, and the test error appears to be greater than both. Without a method to avoid over-fitting, the evolved prediction function will not be suitable to predict the time series outside the training data, especially if the training is strongly noised.

It should be noted that our prediction functions were evolved based on Single-Step Prediction, so we don't have any guarantee that they can be used for an Iterated Single-Step Prediction. To improve a long term prediction by Iterated Single-Step Prediction, we should reduce the noise in the training data of the time series, as done by low-pass filtering. Additionally, the prediction function trained with filtered training data will be faster evolved. However, the low-pass filtering can not always be applied, since most Time Series contain high frequencies which will be lost after low-pass filtering.

6 Conclusions

In this paper, by using GP, we demonstrated the prediction of some Time Series without the requirement of complicated structural knowledge. GP is a method to automatically produce the sought-after solution to a problem by mimicking the process of biological evolution. In this article, GP combined with a Constant Optimization was used. The prediction process was applied recursively. In each recursion step, a sub-prediction function was evolved. At the end of the process all sub-prediction functions were put together to form the prediction function. To improve the capability of a long term prediction by Iterated Single-Step Prediction, the noise in the training data was reduced by low-pass filtering.

Acknowledgements

The evolved prediction functions are very long and complex, so they can not be shown in this paper. However, they can be found on our web pages at: *http://speedy.et.unibw-muenchen.de/persons/panya/Ergebnisse/Ergebnisse.html.*

References

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

[2] D. B. Fogel: *What is evolutionary computation?.* In: *IEEE Spectrum*, Vol. 37, No. 2, 2000, 26-32.

[3] H. Iba, T. Sato and H. de Garis: *System Identification Approach to Genetic Programming.* In: Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Orlando, FL, 1994, 401-406.

[4] H. Oakley: *Two Scientific Applications of Genetic Programming: Stack Filters and Non-Linear Equation Fitting to Chaotic Data.* In: Advances in Genetic Programming, Kinnear, K. E. (ed.), MIT Press, Cambridge, 1994, 369-389.

[5] B. S. Mulloy, R. L. Riolo and R. S. Savit: *Dynamics of Genetic Programming and Chaotic Time Series Prediction.* In: Genetic Programming: Proceedings of the First Annual Conference, Stanford, CA, 1996, 166-174.

[6] R. S. Maust and R. L. Klein: *Nonlinear MISO Modelling Using Genetic Programming.* In: Proceedings of the Thirtieth Southeastern: Symposium on System Theory, 1998, 42-46.

[7] G. J. Gray, T. Weinbrenner, D. J. Murray-Smith, Y. Li and K. C. Sharman: *Issues in Nonlinear Model Structure Identification Using Genetic Programming.*

In: Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, Glasgow, 1997, 308-313.

[8] K. Rodríguez-Vázquez and P. J. Fleming: *Genetic Programming for Dynamic Chaotic Systems Modelling.* In: Proceeding of the 1999 Congress on Evolutionary Computation, Washington DC, 1999, 22-28.

[9] I. Yoshihara, M. Numata, K. Sugawara, S. Yamada and K. Abe: *Time Series Prediction Model Building with BP-like Parameter Optimization.* In: Proceedings of the 1999 Congress on Evolutionary Computation, Washington DC, 1999, 295-301.

[10] H. Cao, L. Kang, T. Gao, Y. Chen, and H. de Garis: *A Two-Level Hybrid Evolutionary Algorithm for Modelling One-Dimensional Dynamic Systems by Higher-Order ODE Models.* In: IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 30, No. 2, 2000, 351-357.

[11] P. J. Angeline: *Evolving Predictors for Chaotic Time Series.* In: Proceedings of SPIE (Volume 3390): Application and Science of Computational Intelligence, Bellingham, 1998, 170-180.

[12] N. Nikolaev and H. Iba: *Inductive Genetic Programming of Polynomial Learning Networks.* In: IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks, San Antonio, TX, 2000, 158-167.

[13] G. Lee: *Genetic Recursive Regression for Modelling and Forecasting Real-World Chaotic Time Series.* In: Advances in Genetic Programming, 3, Spector, L.; Langdon, W. B.; O'Reilly, U.; Angeline P. J. (eds.), MIT Press, Cambridge, 1999, 401-423.

[14] A. S. Weigend and D. E. Rumelhart: *The Effective Dimension of the Space of Hidden Units.* In: IEEE International Joint Conference on Neural Networks, Vol.3, 1991, 2069-2074.

[15] W. Banzhaf, P. Nordin, R. E. Keller and F. D. Francone: *Genetic Programming- An Introduction: On the Automatic Evolution of Computer Programs and its Applications.* Morgan Kaufmann Publishers, Inc., San Francisco, California, 1998.

[16] J. Holland: *Adaptation in natural and artificial systems.* MIT Press, Cambridge, 1992.

[17] L. Prechelt: *Automatic Early Stopping Using Cross Validation: Quantifying the Criteria.* In: Neural Networks, Vol.11(4), 1998, 761-767.

[18] http://www.ngdc.noaa.gov/stp/SOLAR/SSN/ssn.html

[19] A. S. Weigend and N. A. Gershenfeld: *Time Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1994.

[20] D.I. H. Abarbanel: *Obtaining Order in a World of Chaos.* In: IEEE Signal Processing Magazine, May 1998, 49-65.