Meteorological Time Series Modeling Using an Adaptive Gene Expression Programming

ALINA BĂRBULESCU and ELENA BĂUTU
Department of Mathematics and Computers Science
Ovidius University
124, Mamaia Blv., Constanța, 900527
ROMANIA
alinadumitriu@yahoo.com http://alina.ilinc.ro/ID_262/index.html
ebautu@univ-ovidius.ro http://csam.univ-ovidius.ro/~ebautu

Abstract: The precipitations are characterized by important spatial and temporal variation. Model determination for such series is of high importance for hydrological purposes (e.g. weather forecasting, agriculture, flood areas, administrative planning), even if discovering patterns in such series is a very difficult problem. The objective of the current study is to describe the use of an adaptive evolutionary technique that give promising results for the development of non-linear time series models.

Key-Words: time series modeling, gene expression programming, adaptive algorithm, precipitation

1 Introduction

Evolutionary techniques have been used successfully to solve various time series problems [1], [9]. The technique employed in this paper is an improved version of the Gene Expression Programming (GEP) algorithm.
The studied time series consists of the mean annual precipitation registered between January 1965 and December 2005, at the Medgidia meteorological station, situated in the South – East of Romania, on the Black Sea coast.

The complexity of the problem of modeling such meteorological time series derives from the diversity of phenomena that affect the climate, in general, and the precipitation, in particular (e.g. the greenhouse effect, human influences, solar influences, etc.). The problem is a topic of substantial interest in the literature [7], [8], [18]. Classical approaches, such as the linear model, rely on the assumption of a constant data generating process (whose characteristics do not vary with time). Often, they may fail to obtain adequate models due to the nonlinear dynamic behavior of time series, but also due to the lack of adaptation of the methods. This makes the problem very well suited for the use of heuristic methods, such as GEP.

Our article has the following structure. In the next section, some considerations regarding the time series modeling problem are provided. We proceed by a brief presentation of the basic ideas on the evolutionary technique used to derive the models. Then, we perform a statistical analysis of studied time series, followed by the presentation of the experiments and the results. The final section concludes the paper with a discussion of our results and possible directions of future research.

2 Problem Formulation

A time series model for the observed data (x_t) is a specification of the joint distributions of a sequence of random variables (X_t) of which (x_t) is postulated to be a realization.

In what follows we shall denote by *n* use the selection volume

The problem that arises is to find a model that fits the time series as good as possible. In order to do it, the first step is to decide how many previous data points are used – the "window size". One must also decide how the past data used by the model is sampled from the original time series. In this study, we denote the window size by *w*, and we sample the past data at a sampling lag $k = 1$. This means that, for example, if $w = 3$, the model will predict the value at a moment t, x_t, using the previous 3 values in the time series, namely $x_{t-1}, x_{t-2}, x_{t-3}$.

In a more formal manner, we are interested in finding a function *f* that predicts the values of a time series as accurately as possible:

$$\hat{x}_t = f\left(x_{t-1}, x_{t-2}, ..., x_{t-w}\right), t \le n .$$

The accuracy of a model is measured in terms of prediction error:

$$error = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \hat{x}_i\right)^2} .$$

Better models are those with smaller prediction error.

We also can report the ratio of prediction error over standard deviation as a measure of the prediction quality in a model.

Finding a function that fits the data is actually an inverse problem, since there may exist more than one

solution to it – making it a well-suitable candidate for a heuristic approach. We chose to tackle the problem with an enhanced GEP algorithm, described in the next section.

3 Gene Expression Programming

One of the most important goals of artificial intelligence in general is to endow computers with the ability to program themselves. John Koza proposed the most successful attempt of the problem of automatic programming in [14], where he described the Genetic Programming (GP) paradigm – a generalization of Holland's Genetic Algorithms (GA) [12].

GP belongs to the large family of evolutionary techniques, along side GA, Evolution Strategies, or Evolutionary Programming [3]. These techniques share mechanisms that come from Darwin's theory of evolution – natural selection. According to the survival of the fittest principle, the individual best adapted to its environment has the highest chance of survival and reproduction, therefore its traits get to live and perpetuate in next generations.

Since Koza's first book [7], many variants of GP have been proposed in the literature. The differences among them are triggered by the different representations used to encode the solutions. We focus on the Gene Expression Programming algorithm, proposed by Ferreira [10]. We will briefly describe it in the following.

GEP is a flavor of GP that uses a novel representation that takes advantage of some features of the classical GA. In GP, candidate solutions to the problem at hand encoded by the individuals are computer programs expressed as complex hierarchical structures. In the context of our problem, a candidate solution is a composition of mathematical functions, variables, and constants and therefore can be very well represented as its parse tree. GP individuals are obliged to no constraints with respect to their shape or size, other than the physical limitations of the system. In most cases individuals are subject to a constraint regarding the maximum allowed depth, or the maximum allowed number of symbols (functions, variables, constants).

On the other hand, GEP individuals are fixed size strings of symbols. Nonetheless, they encode non-linear expressions, in our case compositions of mathematical functions with functions and variables. In GEP, individuals are composed of one or more genes of equal length; the number of genes (chromosome) is constant throughout the population over all generations and is given as a parameter of the algorithms, as is the gene size. A gene is a linear string of symbols.

By symbols we understand mathematical functions (e.g. arithmetic operators like $+, -, *, /$, trigonometric functions, exponential, logarithmic, etc), constants or variables. The set of symbols at the algorithm's disposal is a parameter of the algorithm.

Every gene encodes a mathematical expression expressed as an expression tree. Ferreira proposed a special syntax for GEP genes that ensures the validity of the de-codification process. A GEP gene is structured in two parts, named "head" and "tail". The tail is constrained to contain only constants or variables, whereas the head may contain any symbol. Also, if we denote the head's size by h and the tail's size by t, the relation:

$$t = h(n-1)+1,$$

must hold, where n represents the maximum arity of the functional symbols used by algorithm. This rule is a guarantee that each GEP gene decodes into a correct expression tree, i.e. a correct mathematical function. Fig. 1 presents a possible GEP individual for the time series modeling problem.

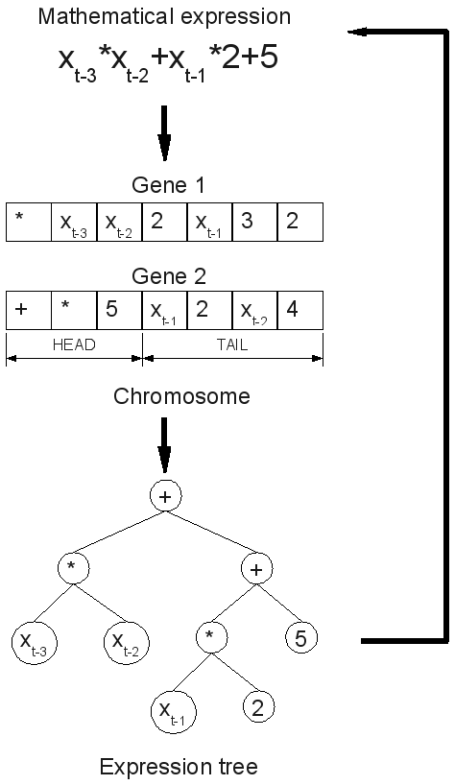

Fig. 1. GEP individual

In the process of decoding a GEP individual, the expressions encoded by the genes are linked together by means of a linking function. The linking function is also a parameter of the algorithm. It depends very much on the type of problem, but usually it is the addition. In the case of Boolean problems, the most used is logical AND.

The linear structure of GEP chromosomes allowed the operators in GEP to perform structural changes similar to those performed by classical GA operators. All operators are implemented so as to respect the rule that enforces only terminal symbols in the tail of each gene. The unary operators defined in GEP include mutation and transposition. The mutation operator changes a randomly chosen symbol in the chromosome. There exist three transposition operators defined in standard GEP. They work by duplicating sequences of genetic code in a chromosome; the differences among them come from the selection way of the code that is to be duplicated (called *transposon*). The transposon may be any subsequence of the chromosome (IS transposition); it may be selected to begin with a function (RIS transposition), or may consists of an entire gene (gene transposition). As for binary operators, GEP has three kinds of crossover operators, all inspired from their GA counterparts: one-point crossover, two-point crossover and gene crossover. Each of them acts by randomly picking two parents from the population. Then, one point crossover randomly picks a position and swaps the genetic material downstream between the two parents. Gene crossover swaps entire genes among the parents, while two-point crossover swaps the genetic material between two randomly chosen positions. For details on the inner workings of standard GEP operators, see [11]. The criterion used by the algorithm to evaluate the candidate solutions uses the prediction error.

3.1 Improved GEP

The number of genes in a chromosome is one of the most important parameters in GEP. It has the same value for all the individuals in a population and is constant throughout a run of the algorithm. This is a rather hard constraint, since it controls the shape and the size of the solutions evolved. Determining the optimum number of genes is an empirical process very much based on the intuition and the experience of the person performing the experiments.

We use AdaGEP [3] – an algorithm that overcomes this issue by identifying the appropriate number of genes automatically. AdaGEP uses an adaptive gene deactivation mechanism inspired by genetic algorithms. Each AdaGEP chromosome is enhanced with a bit string, called "genemap". The genemap size is equal to the number of genes in a GEP individual. Each bit in the genemap corresponds to a gene of the chromosome and it controls whether that gene is used during chromosome decoding. If a genemap bit is set, the decoding process interprets the corresponding gene as in the classical GEP. If its value is 0, the decoding process ignores the corresponding gene (i.e. the gene is "deactivated"). For example, if the genemap is 011, the first gene is deactivated and the AdaGEP chromosome

```
012345678 012345678 012345678
*+x*xx1xx *-x*x3x2x /x+3x1xx5
```

decodes into $2x^2 + \frac{x}{3+x}$. The classical GEP decoding process would have resulted in the expression $4x^2 + \frac{x}{3+x}$.

The genemaps evolve similarly to the population of a classic genetic algorithm. On every GEP iteration, a GA iteration is performed on the genemaps: we apply mutation and crossover on the population of genemaps as they are defined in classical GA. A genemap survives the selection process only if its corresponding chromosome survives. Thus, AdaGEP allows each chromosome to filter out genes that are not useful.

3.2 Parameter Settings

We performed experiments using the AdaGEP extension implemented for the `gep` package of ECJ[1]. In all experiments, we performed 50 independent runs for each setup. The number of genes in a chromosome was set to 5 for the GEP runs. AdaGEP used a number of genes of 10 per chromosome. The head size of a gene was set to 5, the population size was set to 200, and the maximum number of generations the algorithm is allowed to run was 1000. The operator rates used the default values provided by the ECJ, which are set as recommended in the literature [11]. The function set used by the algorithm consisted of $\{+, -, *, /, \sin\}$, where division is implemented as a Koza style protected operator [14].

Finding the optimum window size is an optimization problem by itself and there exists no precise algorithm to compute it. Since this is not the main purpose of our article, we do not employ a special algorithm to decide on a specific window size. Instead, we take on a brute-force approach: we perform experiments for all window sizes in the interval $w \in \{1, 2, 3, 4, 5, 6\}$ and the lag $k = 1$ and report the best model over all. Moreover, the nature of the search process employed by GEP allows it to identify automatically the variables that are most useful to estimate future values among the past n input variables. For the window size $w = 5$ and the lag $k = 1$, it means that the model is built using the most recent five past values $x_{t-1}, x_{t-2}, x_{t-3}, x_{t-4}, x_{t-5}$. It is possible that the function identified by GEP as the model that best fits the data does not depend on x_{t-3} and x_{t-4}, or any other combination.

[1] ECJ is an open-source evolutionary computation research system developed in Java at George Mason University's Evolutionary Computation Laboratory and available at http://cs.gmu.edu/~eclab/projects/ecj/

4 Data analysis and models

The studied time series is presented in Fig. 2. The mean is 449.92 (mm) and the standard deviation 109. 24.

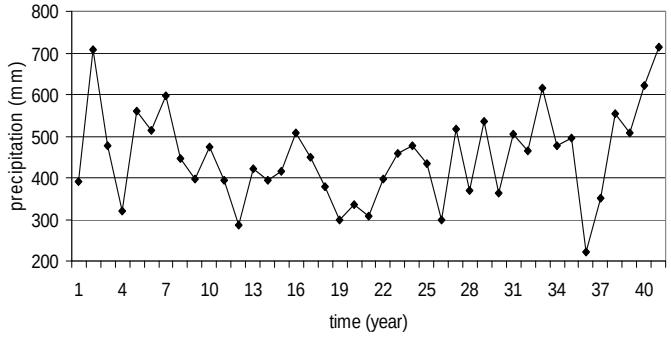

Fig. 2. The data series

4.1 Data analysis

In our study the following procedures and statistical tests were used:

1. Kolmogorov – Smirnov, Jarque - Bera tests or Q-Q plot – to test the normality hypothesis [16];
2. Rank correlation test [16] – to verify the hypothesis that the series is random;
3. The autocorrelation function [5] – to test the hypothesis that the series is uncorrelated;
4. Bartlett test [2] for homoscedasticity;
5. Buishard [6] and Pettitt [15] tests and Hubert's segmentation procedure [13].

The tests results are:

1. The data are normally distributed, since in Q-Q plot diagram (Fig. 3) the observed values are distributed along the straight line that represents the theoretical normal distribution.

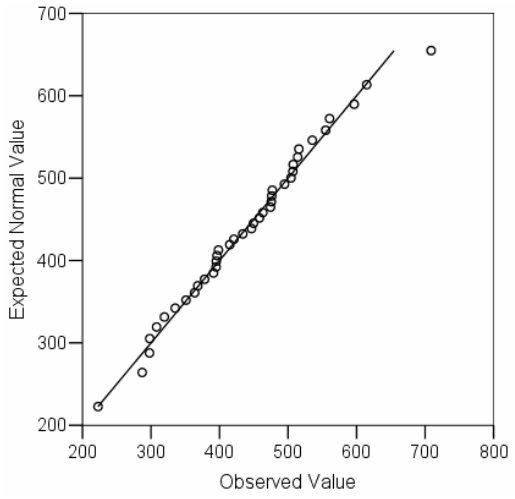

Fig.3. Q-Q plot

2. The series is random.
3. The series is not correlated, since the values of autocorrelation function and partial autocorrelation functions (denoted respectively by ACF and Partial ACF) are inside the confidence limits at a confidence level of 95% (Figs. 4 and 5).

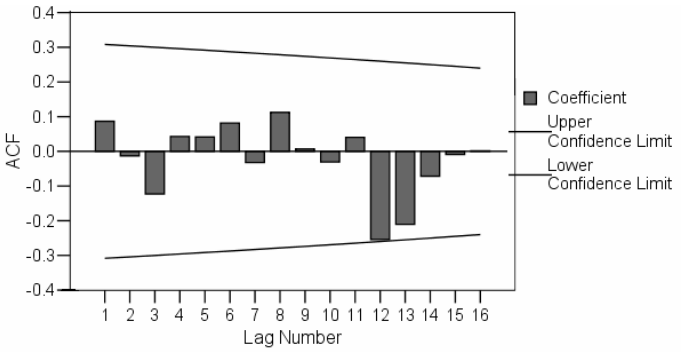

Fig.4. ACF

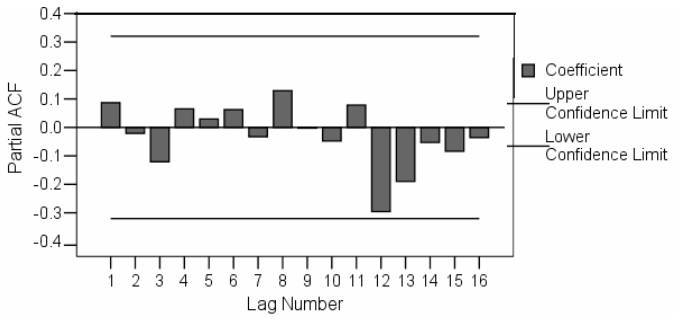

Fig.5. Partial ACF

4. Dividing the data in two parts (the first 20 and the last 21 values), and calculating the value of the statistic X^2, in Bartlett test, we obtain:

$$X^2 = 0.7298 < \chi^2_{0.95}(2-1) = 3.84,$$

where $\chi^2_{0.95}(2-1)$ is the values of χ^2 function, with $2-1$ degrees of freedom, at a significance level 95%.

Thus, the hypothesis that the time series is homoscedastic is accepted.

5. After the application of Buishard and Pettitt tests, the hypothesis that there is no break in the series is accepted at the confidence level of 95%. Hubert's segmentation procedure detects a break in 2003.

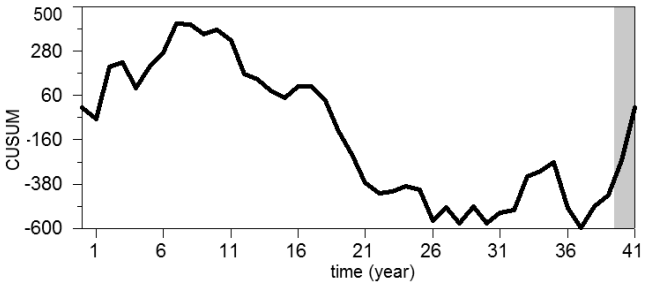

Fig. 6. CUSUM diagram

Also, the CUSUM procedure [17] gives a change point in 2003 (Fig. 6), but since we have only two data after this year, we can not confirm the last hypothesis.

As consequence, we eliminate the last two values and we find the model for the period 1965 – 2003.

4.2 Models

In this section we present the best models obtained using GEP and AdaGEP.

The overall quality of the solutions was best in the runs that used the window size of 5. Therefore, we resume at the presentation of best solutions depicted over all runs by GEP, respectively AdaGEP.

The first model, obtained using GEP for the mean annual precipitation evolution is presented in Fig.7.

The number of symbols in this original GEP solution is 43, while the average number of symbols the GEP solutions over the 50 runs is 34.

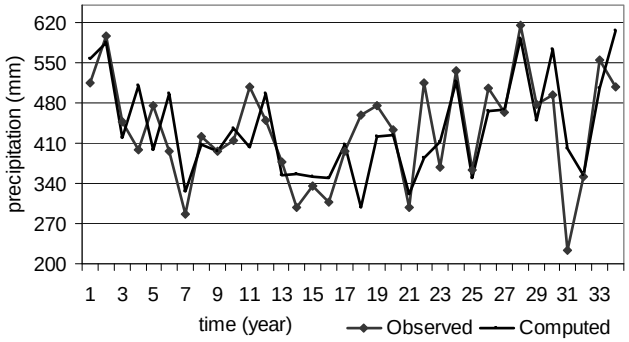

Fig. 7. The first model (GEP)

The residual is normally distributed (Fig. 8) and uncorrelated before the lag 12 (Fig. 9), since the probabilities to reject the correlation hypothesis are bigger than 0.8. They are also homoscedastic.

The prediction error was 68.26, and the ratio between the prediction error over the standard deviation was 0.74.

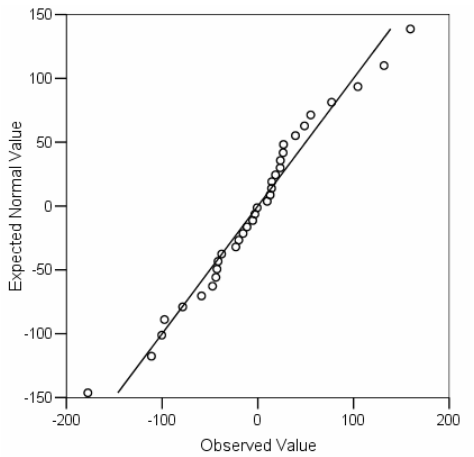

Fig. 8. Residuals' Q - Q plot

The second model was determined in similar conditions, but using AdaGEP algorithm (Fig. 10).

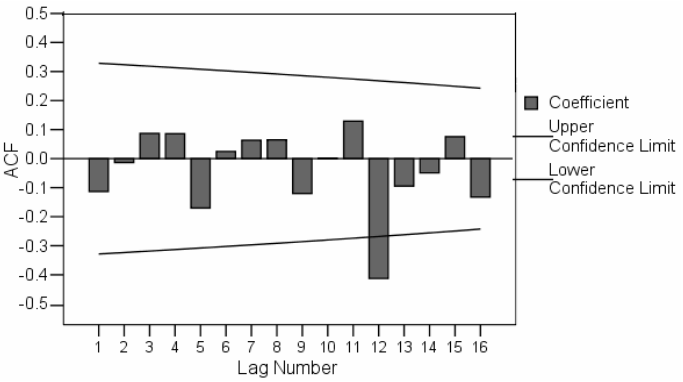

Fig. 9. Residuals' ACF

The algorithm evolved towards this solution that uses only 4 out of the 10 genes in the chromosome. It may be noted that AdaGEP chromosomes are similar to real DNA which contains large sequences of unused code. The number of symbols in this solution was 35, while the average number of symbols over all 50 AdaGEP solutions was 20.

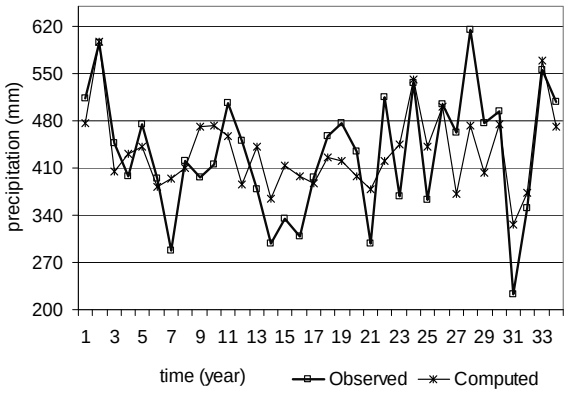

Fig. 10. The second model (AdaGEP)

The residual has the same properties as in the previous case. The prediction error was 64.17, and the ratio between the prediction error and the standard deviation was 0.69.

We presented here only the best solution over 50 independent runs of each algorithm, for a window size of 5. It is interesting to note that on average, AdaGEP solutions used fewer symbols, leading to fewer function evaluations, and therefore a reduction in the algorithm's running times. The mean number of genes in the best-of-run solutions was 3.5, the mean number of symbols 20 and the standard deviation of the number of symbols 6. The number of symbols of GEP best-of-run solutions had a mean of 34, with a standard deviation of 7.

5 Conclusion

This study shows the suitability of GEP and AdaGEP to the time series modelling problem. Although the improvements in the quality of solution are not so

impressive, the advantage of using the adaptive version of GEP over the classical version resides in the potential shown for obtaining solutions in a less complex shape and also a significant reduction in the number of fitness evaluations, and consequently in the running time.

AdaGEP was used also to model the long series of mean monthly precipitations January 1965 – December 2005 and the results were comparable with those obtained by GEP. It seems that the algorithm works better if the series are shorter. Further studies will be done in this direction.

Further research includes evolving teams of individuals to be used as an ensemble model of time series, in order to improve the robustness of the models. AdaGEP will be extended to adaptively find appropriate operator rates for GEP and for the embedded GA.

References:

[1] A. Agapitos, M. Dyson, J. Kovalchuk, S.M. Lucas, Fast segmentation algorithms for long hydrometeorological time series, *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation* (Atlanta, GA, USA, July 12 - 16, 2008), M. Keijzer, Ed. GECCO '08. ACM, New York, NY, 2008, pp. 1163-1170

[2] A. Bărbulescu, *Time series with applications*, Junimea, Iasi, 2002 (in Romanian)

[3] E. Băutu, A. Băutu, H. Luchian, AdaGEP – An Adaptive Gene Expression Programming Algorithm, *Proceedings of the Ninth international Symposium on Symbolic and Numeric Algorithms For Scientific Computing (September 26 - 29, 2007)*. SYNASC. IEEE Computer Society, Washington, DC, 2007, pp. 403-406.

[4] T. Bäck, D. Fogel, Z. Michalewicz, *Handbook of Evolutionary Computation*, Oxford University Press, 1997

[5] P. Brockwell, R. Davies, *Introduction to time series*, Springer, New York, 2002

[6] T. A. Buishard, Tests for detecting a shift in the mean of hydrological time series, *Journal of Hydrology*, Vol.73, 1984, pp. 51-69

[7] A. Busuioc, H. von Storch, Conditional stochastic model for generating daily precipitation time series, *Climate Research*, Vol. 24, 2003, pp. 181–195

[8] S. P. Charles, B. C. Bates, I. N. Smith, James P. Hughes, Statistical Downscaling of observed and modeled atmospheric fields, *Hydrological Processes*, Vol. 18 (8), pp. 1373-1394, 2004

[9] I. De Falco, A. Della Cioppa, E. Tarantino, A Genetic Programming System for Time Series Prediction and Its Application to El Niño Forecast, *Advances in Soft Computing*, Vol. 32, 2005, pp. 151-162

[10] C. Ferreira, Gene Expression Programming: A New Adaptive Algorithm for Solving Problems, *Complex Systems*, Vol. 13, No.2, 2001, pp. 87-129

[11] C. Ferreira, *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*, Springer - Verlag, 2006

[12] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975

[13] P. Hubert, J. P. Carbonnel, Segmentation des séries annuelles de débits de grands fleuves Africains, *Bulletin de liaison du CIEH*, Vol. 92, 1993, pp. 3-10

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

[15] A. N. Pettitt, A non-parametric approach to the change-point problem, *Applied Statistics*, Vol. 28, No. 2, 1979, pp. 126 - 135.

[16] D.J. Seskin, *Handbook of parametric and nonparametric statistical procedures*, Chapmann & Hall/CRC, Boca Raton, 2007

[17] W. Taylor, Change – Point Analyse Entreprises, Libertyville, Illinois, http://www.variation.com/cpa, 2000

[18] D. S. Wilks, R. L. Wilby, The weather generation game: a review of stochastic weather models, *Progress in Physical Geography*, Vol. 23, 1999, pp 329-357

Acknowledgements: This article was supported by grants PNII ID_262 and NatCOMP (PNCDI2 2120/2007).