Genet Program Evolvable Mach (2011) 12:429–456
DOI 10.1007/s10710-011-9140-7

Long memory time series forecasting by using genetic programming

Emiliano Carreño Jara

Received: 11 September 2010 / Revised: 9 March 2011 / Accepted: 29 April 2011 /
Published online: 12 June 2011

Abstract Real-world time series have certain properties, such as stationarity, seasonality, linearity, among others, which determine their underlying behaviour. There is a particular class of time series called long-memory processes, characterized by a persistent temporal dependence between distant observations, that is, the time series values depend not only on recent past values but also on observations of much prior time periods. The main purpose of this research is the development, application, and evaluation of a computational intelligence method specifically tailored for long memory time series forecasting, with emphasis on many-step-ahead prediction. The method proposed here is a hybrid combining genetic programming and the fractionally integrated (long-memory) component of autoregressive fractionally integrated moving average (ARFIMA) models. Another objective of this study is the discovery of useful comprehensible novel knowledge, represented as time series predictive models. In this respect, a new evolutionary multi-objective search method is proposed to limit complexity of evolved solutions and to improve predictive quality. Using these methods allows for obtaining lower complexity (and possibly more comprehensible) models with high predictive quality, keeping run time and memory requirements low, and avoiding bloat and over-fitting. The methods are assessed on five real-world long memory time series and their performance is compared to that of statistical models reported in the literature. Experimental results show the proposed methods' advantages in long memory time series forecasting.

Electronic supplementary material The online version of this article (doi:10.1007/s10710-011-9140-7) contains supplementary material, which is available to authorized users.

E. Carreño Jara (✉)
Departamento de Informática, Universidad Nacional de San Luis, Ejército de los Andes 950, San Luis D5700HHW, Argentina
e-mail: ecarreno@unsl.edu.ar

Keywords Long memory · Time series forecasting · Genetic programming · Multi-objective search · ARFIMA models

1 Introduction

The aim of time series forecasting (TSF) problems is to predict future values of a certain variable by analyzing a set of its current and past values and, optionally, those of other related variables. For example, meteorological agencies try to predict the future value of temperature based on its past values and past values of other related variables such as relative humidity, wind velocity, wind direction, etc. There are two general kinds of methods to solve TSF problems. Those emerging from the statistical and mathematical fields, known as classical or statistical-mathematical methods; and those proposed within the framework of computational intelligence (CI), known as modern heuristic or simply CI methods.

Real-world time series have certain properties, such as stationarity, seasonality, linearity, among others, which determine their underlying behaviour. There is a particular class of time series called long-memory processes, characterized by a persistent temporal dependence between distant observations, that is, the time series values depend not only on recent past values but also on observations of much prior time periods. Stationary long memory time series have autocorrelations which decrease at a very low rate (hyperbolically) converging to zero. In real-world time series the long memory phenomenon entails that unexpected shocks or innovations to a time series do not have a permanent nor short-run transitory effect, but that they have long lasting effects.

The main purpose of this research is the development, application, and evaluation of a computational intelligence method specifically tailored to forecast (univariate) long memory time series, with emphasis on many-step-ahead prediction. An approach from the statistical-mathematical field to forecast long memory time series consists in using an autoregressive (AR) model of high order (e.g., $AR(50)$), as an attempt to approximate the long memory behavior of time series but without modelling them as long memory processes (see [1, 2, 3]). Then, a simple approach to forecast long memory time series could be to use a *CI* method (e.g., artificial neural networks or genetic programming) including a larger number of lagged variables as input variables to approximate the long term behavior of the series. However, if the number of input lagged variables increases too much relative to the number of training samples (in-sample set size), the search space becomes too large to search efficiently, as there are many possible models for the given training data set, but most of them are not likely to capture the underlying data generating process of the series. That is, there are too many lagged variables regarding the amount of training samples as to relate these variables in such a way that the obtained model correctly captures the underlying data generating process. This is known as the *curse of dimensionality* problem (see [4, 5, 6, 7]). Moreover, if obtained models become excessively complex[1] in relation to the amount of training samples, they

[1] As a consequence of the increase in the number of input lagged variables.

could lack the appropriate generalization (overfitting). In addition to this, on incrementing the number of input lagged variables, run time and memory requirements of the *CI* method increase and the obtained model is more difficult to understand. Note that when using only short-lagged variables as input, past information which is useful to forecast the future behavior of the series is lost.

To overcome these problems, this work proposes the first *CI* method designed specially to forecast long memory time series, which is a hybrid combining genetic programming and the fractionally integrated (long-memory) component of the autoregressive fractionally integrated moving average (*ARFIMA*) models. It is named fractionally integrated genetic programming and will be denoted by FI-GP. *This approach basically consists of including in the terminal set a new type of variable, named long-memory variables*. Genetic programming (GP) is a random search algorithm based on the process of evolution, in which solutions are tree structures representing computer programs. Here, a search is carried out in the space of possible computer programs defined by the terminal and function sets, evolving both the functional form and the parameters of models (see [8] for a detailed description on GP).

Another objective of this study is the discovery of useful comprehensible novel knowledge, represented as time series predictive models. There are two general kinds of obstacles which make it difficult to obtain comprehensible models on using CI methods in time series forecasting. Those that arise when the time series is not well-understood and those due to limitations of the CI method used to obtain the model. Here, the first type is overcome by understanding the time series main characteristics and by using the FI-GP method mentioned above. Regarding the limitations of the CI method used to obtain a model, difficulties specific to the method, in this case GP, are identified and an approach to overcome them is proposed. The main limitation to obtain comprehensible models using GP is the structural complexity (tree size or code size) of evolved solutions and the difficulty of evolving modular models of which sub-models (modules) are comprehensible. As mentioned above, the FI-GP method contributes to obtaining comprehensible modular models. On the other hand, the problems of bloat and over-fitting, and the search space characteristics are directly related to the general complexity problem of GP evolved solutions. Bloat is the tendency of the average size of individuals in the population to grow uncontrolled after some generations without corresponding increases in fitness (see [9, 10, 11]). This may be caused by the proliferation of introns, which are inactive parts of GP models (i.e., subtrees) that do not affect calculation's results, for example the term $y_{t-1} \cdot 0$ in the expression $y_{t-2} + y_{t-1} \cdot 0$ (see [12]). Another reason may be inefficient GP code (e.g., $y_{t-2} + 1 + 1 + 1$ instead of $y_{t-2} + 3$) or the fact that the crossover and mutation operators have a higher probability to destroy good low complexity solutions than larger ones (see [12, 13, 14]). In addition to requiring more computer resources, high complexity solutions (relative to the problem to be solved) generalize worse than short ones ([15, 16]). The over-fitting problem arises when obtained models fit too much into the in-sample data (training data), generalizing poorly to new unseen (out-of-sample) data (see [17, 18, 19]). Time series data are made up by two general

terms, a pattern, corresponding to the underlying generating process, and a random or error component which is the unpredictable part of the series, as shown in (1).

$$Data = Pattern + Randomness \tag{1}$$

An over-fit model includes randomness as part of the generating process. Then, when over-fitting occurs in GP, the complexity of evolved solutions increases since models fit random data. Finally, in a search space with more high complexity models correctly fitting the pattern than low complexity models (also properly fitting the pattern), there is a tendency to generate high complexity models ([20, 21]).

To overcome these problems, a new method (named radial basis function genetic programming, denoted by RBF-GP) to limit the complexity of evolved solutions and improve predictive quality is proposed here. On decreasing the average structural complexity of the population, obtained model complexity decreases (which could make them easier to understand), and run time and memory requirements of GP are reduced. On the other hand, on limiting complexity, the bloat and over-fitting problems are avoided. *The method presented here is an evolutionary multi-objective search approach based on a new fitness function, which, in addition to forecasting performance, includes structural complexity as an objective whenever the average structural complexity of the population is beyond a given threshold. The innovative component of this method consists of using, in the fitness function, certain probabilities* P_1, P_2 *associated to the forecasting performance and structural complexity respectively*.

Using these two methods (FI-GP and RBF-GP) allows for obtaining more comprehensible models with high predictive quality, keeping run time and memory requirements low, and avoiding bloat and over-fitting. The methods are assessed on five real-world long memory time series and their performance is compared to that of AR, ARMA, and ARFIMA models reported in [3, 22]. Experimental results show the proposed methods' advantages in long memory time series forecasting.

The rest of this work is organized as follows. In Sect. 2, an introduction to long-memory time series and ARFIMA models is presented. Sections 3 and 4 describe and analyze the proposed methods, FI-GP and RBF-GP respectively. Section 5 shows experimental results obtained in the comparative study. Finally, conclusions and future research avenues are given in Sect. 6.

2 Long memory time series and ARFIMA models

Long memory behavior in time series was first observed by the hydrologist H.E. Hurst [23], who studied the annual minima of the water level in the Nile river (Fig. 1, top). For this time series he observed that the value in a given year depends not only on the values in recent past years but also on values from many previous years. The rather slow decay of the sample autocorrelation function for this series (Fig. 1, bottom), suggests the possible presence of long memory. For this type of series, the autocorrelation function (ACF) decreases at a much slower rate than the exponential rate, finally converging to 0. Since then, evidence on the presence of

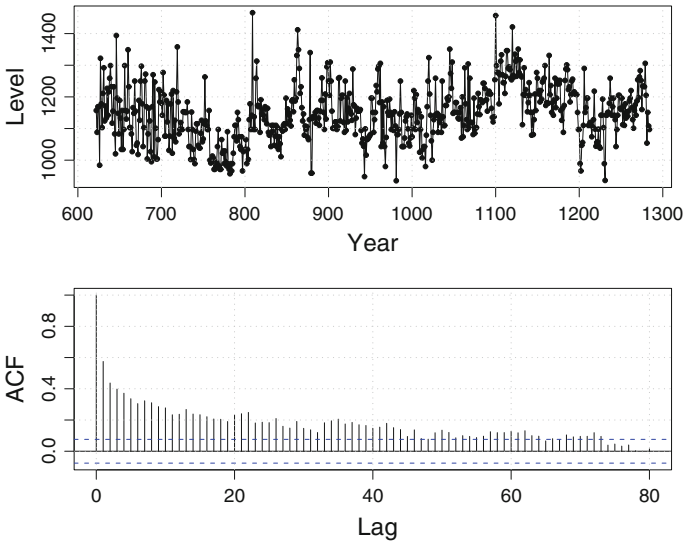

Fig. 1 *Top*: annual minimal water levels of the Nile river for the years 622–1,281, measured at the Roda gauge near Cairo. *Bottom*: sample autocorrelations for the annual minimum water levels of the Nile river time series

long memory in time series has been found in subject areas as diverse as Finance (e.g., [24, 25, 26, 27]), Economics (e.g., [28, 29, 30]), Climatology (e.g., [31, 32, 33]), or Hydrology (e.g., [34, 35, 36]).

Long memory behavior is usually defined as follows. Let y_t be a stationary time series with autocorrelation function $\gamma(k)$, then y_t has long memory if

$$\sum_{k=-\infty}^{\infty} |\gamma(k)| = \infty \tag{2}$$

This asymptotic definition describes the behavior of the autocorrelations when $k \to \infty$. There are several alternative definitions of long memory, all being asymptotic. Then, the long memory behavior is characterized by a slow autocorrelation convergence rate to zero, but there are no restrictions regarding autocorrelations magnitude.

The fractional Gaussian noise model, introduced by Mandelbrot and Van Ness ([37, 38]), was the first model designed to account for the long term behavior of time series. The second long memory model, the integrated (or fractionally differenced) series model, was proposed independently by Granger and Joyeux [39] and by Hosking [40]. Then, in [3], the integrated series model and the fractional Gaussian noise model are generalized, and the equivalence of general fractional Gaussian noise and general integrated series is demonstrated. The autoregressive fractionally integrated moving average (*ARFIMA*) model is a generalization of the integrated series model described in [39, 40].

Granger and Joyeux observed that differencing certain type of series apparently non stationary, in order to obtain stationarity, could have negative consequences. Therefore, for these series, neither differencing, such as on using

ARIMA(p, $d = 1$, q) models, nor not differencing, such as on using ARIMA(p, $d = 0$, q) models, are appropriate. Then, they proposed a kind of models in which the differencing (integration) order d is fractional. The model of an *ARFIMA* process of order (p, d, q), denoted by *ARFIMA*(p, d, q), is defined as

$$\Phi(B)\,(1-B)^{d}\,y_t = \Theta(B)\,\varepsilon_t \tag{3}$$

where p is the order of the autoregressive polynomial $\Phi(B) = (1 - \Phi_1 B - \ldots - \Phi_p B^p)$, q is the order of the moving average polynomial $\Theta(B) = (1 + \theta_1 B + \ldots + \theta_q B^q)$, d is the fractional differencing (integration or long memory) parameter, B is the backshift (lag) operator defined by $B^i \;\; y_t = y_{t-i}$, and ε_t is white noise with mean zero and variance σ_ε^2. The fractional differencing operator $(1 - B)^d$, used in (3), is defined by the following binomial expansion

$$(1-B)^{d} = \sum_{k=0}^{\infty} \binom{d}{k} (-B)^{k} = \sum_{k=0}^{\infty} \frac{\Gamma(k-d)}{\Gamma(k+1)\,\Gamma(-d)} B^{k} \tag{4}$$

where Γ is the gamma (or generalized) factorial function[2]. The ARFIMA(p, d, q) model can also be found in the literature specified as

$$\Phi(B)\,(1-B)^{d}\,(y_t - \mu_y) = \Theta(B)\,\varepsilon_t \tag{5}$$

where μ_y is the mean of y_t. For d fractional, greater than zero and lower than 0.5, the process exhibits long memory. The long term decay in the autocorrelation function of an *ARFIMA* process is determined by the parameter d, for $0 < d < 0.5$ and d near 0.5, these processes have a strong persistence (long memory), which decreases as d approaches zero. The autoregressive and moving average terms model the short-term behavior of the time series, while the fractionally integrated *FI*(d) component accounts for the long-term behavior. *ARFIMA*(p, d, q) models differ from standard autoregressive integrated moving average (*ARIMA*(p, d, q)) models in that for the latter the parameter d is only allowed to take integer values.

In addition to these long memory models, there are others such as the fractional exponential (*FEXP*) model, proposed by Bloomfield [41]. A description of these and other long memory models can be found in [42]. Also, there are several methods proposed in the literature for estimating the parameters of long memory models (e.g., [3, 43, 44, 45]), as well as for testing for the presence of long memory in time series (e.g., [3, 26, 46, 47]). ARFIMA models' estimation methods are usually grouped into two categories: parametric methods, in which the short and long term correlation structures are known allowing all parameters to be simultaneously estimated, and semiparametric methods, in which first the long term behavior is specified by estimating d, and then the autoregressive and moving average parameters are estimated. For example, the log periodogram regression of Geweke and Porter-Hudak (GPH, [3]) estimation method operates in two steps. First it estimates d and then, given this estimation, it fits an ARMA model to the series $(1 - B)^{\hat{d}}\, y_t$. In the exact maximum likelihood (EML, [48]) estimation method, the

[2] The gamma function, denoted by Γ, is an extension of the factorial function to real and complex numbers.

long memory parameter d is estimated simultaneously with the autoregressive and moving average coefficients. On the other hand, the forecasting performance of long memory time series models has been reported in [3, 22, 30, 49, 50, 2], among others. Interested readers may consult [42, 51] for more detailed background information on long memory processes.

3 Fractionally integrated genetic programming

The main purpose of this research is the development, application, and evaluation of a computational intelligence method specifically tailored for long memory time series forecasting, with emphasis on many-step-ahead prediction. Here, the aim is to obtain useful novel CI models with a forecasting performance comparable with or even better than that of statistical methods reported in the long memory literature. To overcome the problems mentioned in Sect. 1, this work proposes the first *CI* method designed specially to forecast long memory time series, which is a hybrid combining genetic programming and the fractionally integrated (long-memory) component of *ARFIMA* models. It is named fractionally integrated genetic programming (denoted FI-GP). The long memory component is the fractionally differenced time series model (denoted by $I(d)$, $FI(d)$, or $ARFIMA(0, d, 0)$), proposed by Granger and Joyeux [39] and by Hosking [40], defined by (6) where y_t is the value of variable y at time t, B is the backshift (lag) operator[3], ε is white noise, and d (the fractional differencing parameter) is a fractional number.

$$(1 - B)^d y_t = \varepsilon_t \tag{6}$$

The proposed *GP* approach includes two types of input variables in the terminal set, short lagged variables and long-memory variables. Short lagged variables are variables lagged by short periods which account for the short-term behavior of the series. The values of long-memory variables are forecasts from $FI(d)$ models, where d is either chosen at random ($0 < d < 0.5$), estimated by using methods proposed in the literature such as the log periodogram regression of Geweke and Porter-Hudak (GPH, [3]) or the exact maximum likelihood (EML, [48]) estimation methods, or optimized (adjusted) by *CI* methods such as genetic algorithms (GA, [52]), evolutionary programming (EP, [53]), or differential evolution (DE, [54]). These variables account for the long-memory behavior of the series, the aim is to integrate (compress, gather, resume) all past information about long-range dependencies scattered along the series into one or more predictive variables. Those models which include long-memory variables have several input lagged variables, but most of them are encapsulated in comprehensible $FI(d)$ sub-models, which cannot be destroyed by crossover. This allows for a comprehensible and modular design, given that larger models are built by combining smaller comprehensible sub-models. The method contributes to improving forecasting performance by giving the searching process an initial approximate solution, and to increasing comprehensibility of obtained solutions by means of a comprehensible modular design.

[3] Defined by: $B^i \; y_t = y_{t-i}$.

3.1 One and two step alternatives

In a similar way as with ARFIMA models' estimation methods, on using the FI-GP method to obtain a predictive model, it is possible to proceed in one or two steps. The two-step alternative consists in estimating (optimizing, adjusting) d by using one or more statistical (or computational intelligence[4]) methods, in such way as to obtain one or more long-memory variables $\mathrm{FI}(\hat{d}_i)$ for $i = 1, 2, \ldots, n$ in the first step. Then, in the second step, in-sample forecasts from $\mathrm{FI}(\hat{d}_i)$ sub-models estimated (optimized) in the first step are used as long-memory variables during the GP evolutionary process. The one-step alternative consists in running simultaneously the GP evolutionary process and a CI method used to optimize d. On generating the initial population, d values are randomly selected, and then they are optimized by a CI method such as GA or DE, in each generation (or only in some generations) of the GP evolutionary process. After executing the evolutionary process, the final GP model is obtained by replacing each long memory variable in the selected GP solution by its corresponding $\mathrm{FI}(\hat{d}_i)$ sub-model.

3.2 Obtaining forecasts from FI(d) sub-models

Once the value of d has been determined by using statistical or CI methods, forecasts from FI(d) models can be obtained by computing the first T coefficients in the binomial expansion of $(1 - B)^{\hat{d}}$ and using them to form (build) an autoregressive model of order T AR(T) as described as follows. Let $\delta_k = \binom{d}{k}(-1)^k = \frac{\Gamma(k-d)}{\Gamma(k+1)\,\Gamma(-d)}$ in (4), the fractional differencing operator $(1 - B)^d$ can be written as

$$(1 - B)^d = \sum_{k=0}^{\infty} \delta_k B^k = 1 + \sum_{k=1}^{\infty} \delta_k B^k = 1 + \delta_1 B^1 + \delta_2 B^2 + \ldots \tag{7}$$

In order to use an estimated $\mathrm{FI}(\hat{d})$ model to forecast the series value at time $t + 1$, given that it is not possible to have an autoregressive representation of infinite order $\mathrm{AR}(\infty)$, instead an autoregressive model of order T AR(T) is used, where T is an integer corresponding to the amount of available past observations (or truncated at the time period $t - T + 1$) $y_t, y_{t-1}, y_{t-2}, \ldots, y_{t-T+1}$. Then, according to (6), the estimated $\mathrm{FI}(\hat{d})$ model can be described as

$$(1 - B)^{\hat{d}}\, y_t = \sum_{k=0}^{T} \delta_k B^k\, y_t = (1 + \delta_1 B^1 + \delta_2 B^2 + \ldots + \delta_T B^T)\, y_t = \varepsilon_t \tag{8}$$

Then, distributing y_t and applying the backshift (lag) operator (defined by $B^i\, y_t = y_{t-i}$) we have,

[4] In this case, in order to improve run time, a CI method that generates one or more $\hat{d}$ solution values, is used.

$$\varepsilon_t = y_t + \delta_1 y_{t-1} + \delta_2 y_{t-2} + \ldots + \delta_T y_{t-T} \tag{9}$$

Finally, increasing the subscripts by one and taking the expected value of future random errors (ε_{t+1}) as zero, the autoregressive representation of the FI($\hat{d}$) predictive model can be written as follows

$$\hat{y}_{t+1} = -(\delta_1 y_t + \delta_2 y_{t-1} + \ldots + \delta_T y_{t-T+1}) \tag{10}$$

Forecasts h-steps-ahead $(\hat{y}_{t+h})$ can be computed recursively as with an AR model. In what follows, FI(d) will denote either the modeling or the forecasting representation of the fractionally differenced time series model, depending on the context.

3.3 Multi-step-ahead forecasting

To perform multi-step-ahead forecasting with the FI-GP method, recursive and direct methods can be used. In the recursive method (iterated prediction) models are evolved to carry out single-step-ahead forecasting. Then the obtained solution is used recursively to forecast h-steps ahead by iteratively taking single-step-ahead forecasted values as input to forecast the next time step value, until the desired forecast horizon h is reached. Given that the FI(d) sub-models which are part of a FI-GP model, are also models that can be used independently, there are two alternative approaches on using the recursive method with the FI-GP method. The first alternative, named FI-GP-R, is the standard recursive method, that is, the FI(d) sub-models take single-step-ahead forecasted values from the FI-GP model as input to forecast the next time step value. The second alternative, named FI-GP-R^2, only differs from the first in that the FI(d) sub-models take their own single-step-ahead forecasted values as input to forecast the next time step value. In the direct method, instead of forecasting the next time step value, FI-GP models are directly evolved to forecast the h-th period ahead. There are two alternatives on using the direct method with the FI-GP method. In the first one, named FI-GP-D, h-step-ahead forecasts from FI(d) sub-models are obtained by a direct method such as best linear unbiased prediction (readers may consult [42, §8.7] for a detailed description on the best linear unbiased prediction method). In the second alternative, named FI-GP-DR, h-step-ahead forecasts from FI(d) sub-models are obtained by the recursive method.

3.4 Combining and improving FI(d) initial solutions

When the FI-GP method is used with only one long-memory variable (FI(d) sub-model) together with one or more short-lagged variables, this FI(d) sub-model can be considered as an initial partial solution, which is then improved by an evolutionary process. On the other hand, on using more than one long-memory variable (FI(d) models with different values of d) and no short-lagged variables, the FI(d) models can be considered as final solutions which are combined by means of evolution, that is, the FI-GP method provides a way of combining forecasts from these FI(d) predictive models. Finally, when using both short and long-memory variables, the FI-GP method provides a way to combine and improve these FI(d) solutions.

3.5 Setting d randomly

On the other hand, on using the FI-GP method alternative where values of d are randomly set ($0 < d < 0.5$), we could obtain models with high predictive quality, without using statistics (or CI) methods to estimate (optimize) d. Even though it's very likely that those FI(d) sub-models for which d is randomly selected won't have good predictive quality, on combining them with other FI(d) sub-models and with short-lagged variables during the evolutionary process, we could obtain FI-GP models with high forecasting performance. The long term decay in the autocorrelation function of an FI(d) process is determined by the parameter d, for $0 < d < 0.5$ and d near 0.5, these processes have a strong persistence (i.e, a strong association between observations widely separated in time), which decreases as d approaches zero. Then, the aim of incorporating long-memory variables, which take forecasts from *FI*(d) models where the value of d is randomly selected, is that the evolutionary process selects and combines those long-memory variables that better suit the underlying data generating process of the series. That is, those individuals made up of FI(d) sub-models with values of d that better approximate the value best representing the series persistence will tend to be favoured by the selection process, and so, these FI sub-models will thrive through out generations. FI(d) sub-models with values of d not close enough to the value best representing the series persistence will tend to disappear.

4 An evolutive multi-objective searching method

There are several ways to approach the GP complexity problem. Simple approaches consist in using automatically defined functions (ADFs, to encapsulate parts of GP programs so that they can't be destroyed by crossover) or selecting a population big enough so as to generate a satisfactory solution before individual complexity increases too much ([55]), although, generally, they aren't sufficient. Another approach consists in starting with low complexity individuals and progressively considering larger individuals in order to find the lowest complexity satisfactory solution (e.g., see [56]), however, in GP it isn't possible to consider all low complexity solutions. Another possibility is to use a fitness function that combines performance and complexity, but finding proper weights for these functions can be as difficult as setting an appropriate model complexity (see [57]). As an alternative, in [58] these weights are modified according to properties of the evolutionary process, nevertheless, the desired level of training accuracy must be set. Instead of using a weighted function, the approach proposed in [59] separately selects according to either performance or complexity in different tournaments. This requires to set parameters determining the relative importance of performance and complexity. Other approaches consist in using editors to delete introns (non-functional parts of GP code), using a threshold to limit solution complexity (by bounding the number of nodes or tree depth), penalizing (by means of a penalty factor in the fitness function) long solutions or solutions with too many introns, or adapting the crossover operator to restrict code growth (see [55]). Finally, other

authors have used multi-objective search approaches with performance and complexity as objectives (e.g., [60, 61, 62]), however, this can lead to evolutionary searches converging to solutions with low complexity but with poor performance (see e.g., [63, 64]). To avoid this, in [65], diversity is included as an objective in addition to performance and complexity. In this last approach, the complexity of obtained solutions will depend not only on the in-sample data set, but also on other general GP control parameters, which will indirectly influence complexity.

In the time series forecasting framework, different approaches have been used to solve the GP's model complexity problem. In [66], Akaike Information Criterion (AIC, defined in [67]) is utilized to solve model complexity with generalization capacity on using linear genetic programming. Several techniques for overfitting avoidance in GP approaches producing tree-structured polynomials are provided in [68]. In [69], an extension of GP, which uses two re-sampling techniques (i.e., Bagging and Boosting) is proposed and authors assert that the re-sampling techniques are successful in reducing the tree size. [70] studies over-fitting and premature convergence, on a standard GP approach applied to the Mackey-Glass time series prediction. According to results presented in this article, the standard GP approach does not exhibit the over-learning reported in [71] and the premature convergence problem can be corrected by modifying the crossover operator. On the other hand, [72] proposed an approach to forecast in non-static environments that can automatically set the correct analysis window size (i.e., the number of historical data to be analyzed in predicting a future value) without human intervention. Here, two methods to overcome the problem of bloat in this approach are proposed. Other publications approaching the GP's model complexity (over-fitting and bloating) in the time series forecasting framework are [73, 74, 75, 76], among others.

According to the literature reviewed, the right (or approximate) choice of the model's complexity is too important and difficult a task as to be carried out automatically, considering currently available methods. It is important to emphasize that in GP, due to introns and/or inefficient code, there is no optimal model complexity which can prevent both underfitting and overfitting for a given data set. Instead, appropriate model complexities are those which allow to minimize the out-of-sample forecast error, due to both underfitting and overfitting. Selection of appropriate model complexities highly depends on the size and randomness of the in-sample data set. Then, the right choice of model complexity and of in-sample data sets become important, difficult, and highly related top level decisions.

4.1 Overview of the proposed approach

In this work, a new method to limit the complexity of evolved solutions and improve predictive quality is proposed. On decreasing the average structural complexity of the population, the obtained model's complexity decreases (which could make it easier to understand), and run time and memory requirements of GP are reduced. On the other hand, on limiting complexity, the bloat and/or over-fitting problems are avoided. The method presented here is an evolutionary multi-objective search approach, which, in addition to forecasting performance, includes structural complexity as an objective whenever average structural complexity of the

population is beyond a given threshold. Here, the aim is to allow model complexity to increase as much as to capture the whole time series pattern (avoiding underfitting), but not so much as for the models to include randomness as part of the generating process (avoiding overfitting). This way, after a given number of generations have passed, the search will focus on hypothesis regions with complexities within a certain range. This approach has been designed to be used with the FI-GP method presented in Sect. 3, although it could be useful on its own.

4.2 Fitness function for the multi-objective search process

In multi-objective (multi-criteria or multi-attribute) optimization problems, two or more conflicting objectives subject to certain constraints have to be simultaneously optimized (e.g., minimizing the cost while maximizing the performance of a product, or minimizing the weight while maximizing the strength of a material). Considering, without loss of generality, the minimization of all the objectives, the multiobjective optimization problem can be defined in mathematical terms as follows: Let X be an n-dimensional solution search space of decision variable vectors $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$, find a vector $\mathbf{x}^*$ that minimizes a given set of k objective functions $f(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})]$, which satisfies m inequality constraints $g_i(\mathbf{x}) \geq 0, i = 1, 2, \ldots, m$ and p equality constraints $h_i(\mathbf{x}) = 0, i = 1, 2, \ldots, p$. Readers may consult [77, 78, 79, 80] for a detailed description on multiobjective optimization. *In this work, the in-sample forecasting error (mean square error)* f_1 *and the structural complexity* f_2 *are the objectives to be minimized.*

4.2.1 Proportional selection method

Let $\mathbf{x}$ be an individual from the current population. The objectives are combined by using the following Gaussian Radial Basis Function (RBF), which can be used as performance and fitness measure:

$$Fitness(\mathbf{x}) = \exp\left(-\frac{\|P(\mathbf{x}) - \mathbf{c}\|^2}{\sigma^2}\right) = \exp\left(-\frac{\sum_{i=1}^{2}(1 - P_i(\mathbf{x}))^2}{\sigma^2}\right) \tag{11}$$

where $P(\mathbf{x}) = \langle P_1(\mathbf{x}), P_2(\mathbf{x}) \rangle, P_i(\mathbf{x})$ is the probability that $\mathbf{x}$ has of winning a comparison (assuming minimization) according to the objective function f_i against another solution randomly selected from the current population, $\mathbf{c} = \langle 1_1, 1_2 \rangle$, and $\sigma \in \mathbb{R}$ is the spread around the center which determines the ratio of the function decay with its (Euclidean or 2-norm) distance from $\mathbf{c}$. On using this fitness function with proportional fitness (roulette wheel) selection, the spread σ determines the selective pressure. Figure 2 shows the graphical representation of this RBF fitness measure for $\sigma = 0.75$.

4.2.2 Tournament selection method

For the tournament selection method, selective pressure is adjusted by changing the tournament size, then the following fitness function can be used:

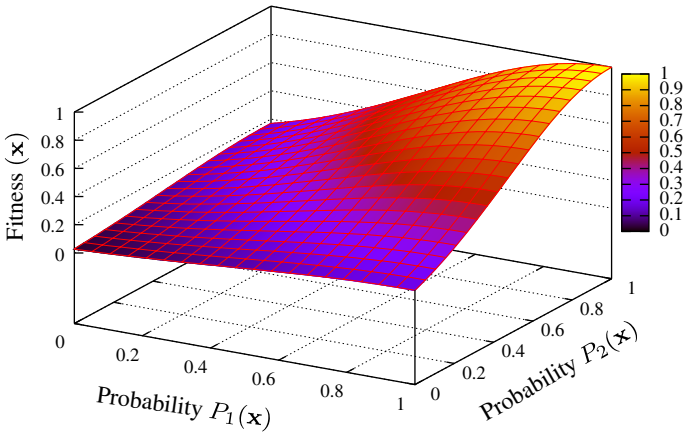

Fig. 2 Graphical representation of the radial basis function given in (11) with $\sigma = 0.75$

$$Fitness(\mathbf{x}) = \left(\sum_{i=1}^{2} (1 - P_i(\mathbf{x}))^2 \right)^{\frac{1}{2}} \tag{12}$$

4.2.3 Summary of the RBF-GP method

The approach used in this work consists of using a threshold for the average complexity of the population. Each time the average complexity is beyond this threshold, complexity is included as an objective (in addition to forecasting performance) by using (11) or (12); otherwise, a standard (mono-objective) fitness function is used. The innovative component of this method consists of using, in the fitness function, the probabilities $P(\mathbf{x}) = \langle P_1(\mathbf{x}), P_2(\mathbf{x}) \rangle$ and the reference point $\mathbf{c} = \langle 1_1, 1_2 \rangle$ defined on the space of probabilities.

4.2.4 Run example

Once the threshold is reached, the average complexity of the population will oscillate within a given range of values as shown in Fig. 3. In this figure it can be seen that, after a certain number of generations, the average population complexity oscillates between 110 and 150 (approx.), thus the range of individual complexities is broad. In this run, which uses elitism, the complexity of the best individual is decreased from a value near 400 (in the 24th generation) to near 130 (in the 79th generation) improving the in-sample forecasting performance (without overfitting the data series). So, the choice of the threshold value is rather intuitive and it does not restrict complexity to only one value, which makes setting this parameter easier.

4.3 How and why the RBF-GP approach could work

The RBF-GP approach works as follows. At the beginning of the evolutionary process, models are generated with relatively low structural complexity and include

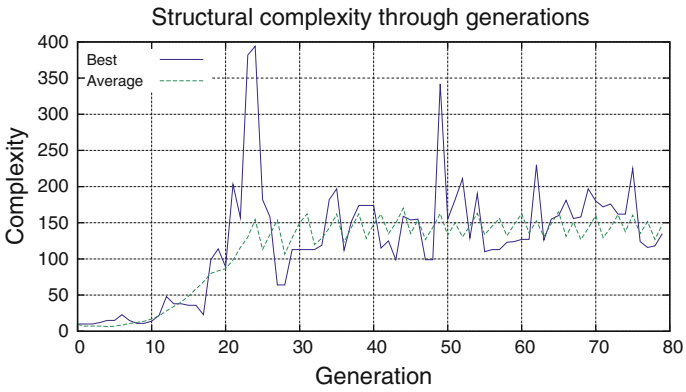

Fig. 3 Average population complexity and best solution complexity through the generations

randomness and pattern components. Then, as generations go by, average structural complexity increases. Then we have the following hypothesis:

Hypothesis 1: the increase in structural complexity is due to over-fitting. After enough generations have elapsed, in standard GP, the complexity of individuals could increase until individuals learn the whole pattern and memorize all of the series random component. Then, it's necessary to bound structural complexity during evolution. This way, average population complexity increases, and the individuals learn the pattern and memorize the series randomness until average complexity reaches the threshold. From then on, the RBF-GP multiobjective search approach will include complexity as an objective to minimize. Then, the learning process (guided by fitness) will progressively remove randomness from the models undergoing evolution. This is given that, on reaching the threshold, the only way in which fitness can increase is replacing the random component by the pattern. The random component will be replaced by pattern due to the complexity necessary to learn the pattern is less than that needed to memorize the randomness. For example, consider the series $y = 2, 4, 8, 16, 32, \ldots, 2^n$, the complexity necessary to memorize this series is much greater than that needed to learn the pattern $y_i = 2^i$ (assuming a big enough in-sample data set size). Thus, this approach allows for prevent and overcome over-fitting.

Hypothesis 2: the increase in complexity is due to code bloat. After enough generations have elapsed, in standard GP, the complexity of individuals could increase until it consumes considerable resources. Suppose that most code growth is due to proliferation of introns (which are inactive parts of GP models that do not affect calculation's results), and thus, doesn't contribute to memorization/over-fitting. Then, average population complexity increases until it reaches the threshold. From then on, the RBF-GP multiobjective search approach will include complexity as an objective to minimize. Then, the learning process (guided by fitness) will progressively remove introns from the models undergoing evolution and/or will avoid them, given that, the only way in which fitness can increase is replacing the introns by the pattern. Thus, this approach allows for saving the considerable

computational resources that additional code (of code growth) consumes and, given that code growth may interfere with exploration of the search space, it also allows for improving the forecasting performance.

5 A comparative study

The purpose of this section is to evaluate the forecasting performance of the approaches proposed in this work, comparing it with that of statistical methods published in the literature (and of standard GP[5]) in several-step-ahead forecasting, using real-world economic long memory time series and out-of-sample forecasts (where no data beyond the point of forecast is used either for analysis, model estimation, nor model evolution). The GP approaches proposed in this work and assessed in this section, are: FI-GP (Sect. 3), RBF-GP (Sect. 4), and the FI-RBF-GP method, which uses both the FI-GP and RBF-GP methods. In these experiments, h-step-ahead out-of-sample forecasts, with horizons $h = 1, 2, \ldots, 100$ are analyzed. All estimations and forecasts for statistical models, including those submodels of models generated by the FI-GP and FI-RBF-GP methods, are performed using the object-oriented matrix programming statistical system Ox version 6.00 (see [81]) and the Arfima package version 1.00 ([82]). The kernel system for the standard GP, FI-GP, RBF-GP, and FI-RBF-GP approaches is based on the public domain genetic programming system Gpc++ Version 0.40 [83] (modified as to use a generational genetic programming scheme instead of steady state genetic programming). The statistical models and estimation methods, and the long memory time series used are those of [3, 22].

The rest of this section is organized as follows. Section 5.1 details the FI-GP and RBF-GP methods' alternatives (variants) used. Statistical methods and data sets used in these experiments are given in Sect. 5.2. The setup of experiments is described in Sect. 5.3. Finally, results and performance analysis are given in Sect. 5.4.

5.1 FI-GP and RBF-GP methods' alternatives

On using the FI-GP and RBF-GP methods, it's necessary to choose from several alternatives (or variants) as described in Sects. 3 and 4. For these experiments, the two-step FI-GP alternative, using statistical methods to estimate d, has been chosen. Forecasts h-steps-ahead $(\hat{y}_{t+h})$ are computed recursively as with an AR model, using the alternative named FI-GP-R (see Sect. 3). The statistical methods used to estimate d are the log periodogram regression of Geweke and Porter-Hudak (GPH, [3]), exact maximum likelihood (EML, [48]), and nonlinear least squares (NLS, see [30]) estimation methods. Standard GP will be denoted by STD-GP or STD-GP(n), where n is the number of lagged variables $y_t, y_{t-1}, \ldots, y_{t-n+1}$. Similarly, the FI-GP, RBF-GP, and FI-RBF-GP approaches can be denoted as FI-GP(n, m), RBF-GP(n), and FI-RBF-GP(n, m), where m is the number of long memory variables. For $m = 1$

[5] This approach is only used as a reference.

EML is used, for $m = 2$ EML and NLS are used, and for $m = 3$ the EML, NLS, and GPH estimation methods are used. The forecasting performance measure used by all GP approaches during evolution is the normalized mean square error (NMSE) defined by $NMSE = \frac{MSE}{S_y^2} = \frac{N^{-1}\cdot\sum(y_i-\hat{y}_i)^2}{(N-1)^{-1}\cdot\sum(y_i-\bar{y})^2}$ where N is the number of forecast cases, y_i and $\hat{y}_i$ are the observed series value and forecasted value respectively, for the forecast case i, and, $\bar{y}$ and S_y^2 are the sample mean and sample variance of the series respectively, computed over all forecast cases $i = 1, 2, \ldots, N$. The complexity of a GP solution is measured by its structural complexity.

Structural complexity is defined, for this paper, as the number of terminals and functions in the GP tree.

The FI-GP and STD-GP methods use the following fitness function

$$Fitness(\mathbf{x}) = \frac{1}{1 + NMSE_{\mathbf{x}}} \tag{13}$$

where $\text{NMSE}_{\mathbf{x}}$ is the NMSE of individual $\mathbf{x}$. The RBF-GP and FI-RBF-GP methods use the fitness function given in (12), Sect.(4), each time the average complexity is beyond the threshold; otherwise (13) is used as fitness function.

5.2 Statistical methods and data sets

The two published studies on long memory time series forecasting chosen as references for the comparative evaluation are [3, 22]. Experiments in these publications are reproduced and their results are compared to those of methods proposed in this work. The comparative analysis is carried out using the following long memory monthly time series data sets: the Consumer Price Index for Food - All Urban Consumers (denoted by Food-CPI, from January 1947 to July 1978), the Producer Price Index - All Commodities (denoted by PPI, from January 1947 to February 1977), the Consumer Price Index - All Urban Consumers (denoted by CPI, from January 1947 to February 1976), the Consumer Price Index for Food for wage earners and clerical workers (denoted by Food-CPI-WECW, from January 1947 to July 1978), and the Retail Price Index Inflation Rate for UK (denoted by UK-Inflation-Rate, from February 1969 to September 1992). The first three series have been taken from [3] and the last two from [22], evidence of the presence of long memory in these series can be found in the respective publications.

The statistical methods used for CPI, Food-CPI, and PPI are: an ARFIMA (0, d, 0) model (see (3), Sect. 2) estimated by the log periodogram regression of Geweke and Porter-Hudak method (denoted by ARFIMA-GPH) and an autoregressive model of order fifty AR(50) estimated by nonlinear least squares (denoted by AR50-NLS). The statistical methods used for the Food-CPI-WECW and UK-Inflation-Rate series are: an ARFIMA(0, d, 0) model (see (5), Sect. 2) estimated by exact maximum likelihood (denoted by ARFIMA-EML) and an ARMA(2,2) model also estimated by EML (denoted by ARMA-EML). Data series

transformations used in the out-of-sample forecasting experiments for these statistical methods are those of [3, 22].

5.3 Test experiments setup

This section details the out-of-sample forecasting procedure, the GP parameters setting, the reasons to use the recursive method, the problem found using it, and the proposed solution.

5.3.1 The forecasting procedure

For each series, the procedure to carry out out-of-sample forecasts using the statistical and GP methods, is as follows. The first T series observations are used to estimate the statistical models and to obtain the GP models. Then, each of these models is used to forecast the series at time $T + h(\hat{y}_{T+h})$, computing the forecast errors for $h = 1, 2, \ldots, 100$ or until $T + h$ reaches the end of the series. Forecasts are computed by using the recursive method. This procedure is repeated increasing the forecast origin T by one at a time, until T reaches a given time point T_{max}. The statistical models' parameters are re-estimated and GP models re-obtained each time an additional observation is included into the in-sample data set. The initial forecast origin will be denoted by T_{init}. Then, for each horizon $h = 1, 2, \ldots, 100$ the forecasting performance measures (MSE, RMSE, NMSE, and MAE) are computed. For each time series the number of observations, T_{init} and T_{max} are as follows: for Food-CPI $T_{init} = 251$ and $T_{max} = 330$; for PPI $T_{init} = 239$ and $T_{max} = 318$; for CPI $T_{init} = 227$ and $T_{max} = 306$; for Food-CPI-WECW $T_{init} = 251$ and $T_{max} = 330$; for UK-Inflation-Rate $T_{init} = 181$ and $T_{max} = 260$. The maximum number of out-of-sample forecasts ($T_{max} - T_{init}$) is set to 80, this way there will not be an excessive difference between the numbers of samples used to compute the performance measure for each horizon. On using a GP method, the first 36 observations (the first three years) are reserved to be used as lagged values to carry out forecasts.

5.3.2 The recursive method and forecast exceptions

In the out-of-sample forecast experiments, forecasts h-steps ahead are computed by using the recursive method (iterated prediction). The recursive method has been chosen instead of the direct method to make a fairer comparison against results of statistical methods published in the literature, which use it, and since it's more interesting to use the recursive method given that the forecasting performance of models is expected to improve on using the direct method (as explained next). In preliminary runs, it was observed that, some of the GP obtained models had high performance for the first forecast horizons, but, from certain forecast horizon on, they performed very badly. This problem worsened for further forecast horizons. Then, on applying the recursive method with a given GP model, the forecasting errors could be normal for the first horizons, and then, (from a given horizon) errors could suddenly increase. This error could quickly increase in the following horizons until it produces a floating-point exception. This is not due to overfitting,

underfitting, or a change in the underlying data generating process of the series, but to from what now on will be called *forecast exceptions*. These forecast exceptions are generated by inactive parts of GP models (i.e., subtrees) that do not affect calculation's results for the in-sample data set, but which, on using the recursive method to forecast unseen (out-of-sample) observations, become active even if there is no change in the underlying data generating process. This happens because, on using the recursive method to forecast the time period $T + h$, *unexpected innovations* are produced in the time series formed by concatenating the in-sample data series and the recursive forecasts: $y_1, y_2, \ldots, y_T, \hat{y}_{T+1}, \hat{y}_{T+2}, \ldots, \hat{y}_{T+h-1}$. Forecast exceptions are produced for forecast horizons $h > 1$, they expand to following horizons (chain reaction) until either forecasting values stabilize or forecasting errors increase and produce a floating-point exception. A forecast exception can also happen for only one forecast horizon, without having an effect on the next.

When a forecast exception is generated during the *out-of-sample forecast experiments*, it's important to stop the chain reaction given that, either on producing a floating-point exception or on getting too-high errors, all the sample runs would become useless. Here, the following method to detect and deal with forecast exceptions is proposed and used. To detect an out-of-sample forecast exception on the time period $T + h$, the time series formed by concatenating the data series until time T and the out-of-sample recursive forecasts $\hat{y}_{T+1}, \hat{y}_{T+2}, \ldots, \hat{y}_{T+h-1}$ is considered. Out-of-sample forecasts lying over or under $r \in \mathbb{R}$ standard deviations σ from the mean of the series $y_1, y_2, \ldots, y_t, \hat{y}_{t+1}, \hat{y}_{t+2}, \ldots, \hat{y}_{t+h-1}$, are deterministically considered exceptions. An out-of-sample forecast exception (at time $t + h$) will be replaced by the out-of-sample forecast at time $t + h - 1$ (the previous out-of-sample forecast), which is the naïve random walk model $(\hat{y}_{t+1} = y_t)$ forecast (using the recursive method). Based on preliminary runs, in the comparative experiments r will be set to 5 for all series, except for UK-Inflation-Rate for which it will be set to 1.

5.3.3 GP parameters and data transformations

GP configuration and control parameters shared by STD-GP, FI-GP, RBF-GP, and FI-RBF-GP approaches in the out-of-sample forecasting experiments are the following. Creation type: ramped half and half. Maximum depth at creation: 6. Maximum depth at crossover: 17. ADFs: not used. Crossover probability: 0.9. Mutation probability: 0.01. Selection method: Tournament (the tournament size is set to 5). Elitism is used[6]. The generational genetic programming scheme is used. Function set: $\{*, +, -, \%, Sqrt, Sine, Cosine, e\text{−Exp}, \text{e−Ln}\}$, where the protected division operator % returns 1.0 if the denominator is zero, the protected operator *Sqrt* returns the square root of the absolute value of its argument, the protected exponential operator $e - Exp$ returns e^{10} if its argument is greater than 10.0, the protected operator *e-Ln* returns the natural logarithm of the absolute value of its argument if its argument is not equal to zero, and zero otherwise. Ephemeral random constant range: [−1.000, 1.000]. Termination criterion: the run terminates

[6] Here, the best solution from the previous population is included in the current population.

Table 1 Configuration and control parameters specific to each GP method

TS	GP method	NumGens	PopSize	Threshold
CPI	STD-GP(12)	60	800	–
	RBF-GP(12)	60	800	600
	FI-GP(10,1)	20	1,000	–
	FI-RBF-GP(10,1)	30	1,000	30
Food-CPI	STD-GP(10)	64	800	–
	RBF-GP(10)	64	800	800
	FI-GP(0,2)	20	1,000	–
	FI-RBF-GP(0,2)	30	1,000	30
PPI	STD-GP(15)	60	800	–
	RBF-GP(15)	60	800	600
	FI-GP(0,2)	20	1,000	–
	FI-RBF-GP(0,2)	30	1,000	30
CPI-WECW	STD-GP(10)	64	800	–
	RBF-GP(10)	64	800	800
	FI-GP(5,3)	20	1,200	–
	FI-RBF-GP(5,3)	30	1,200	30
UK-Inf-Rate	STD-GP(10)	50	1,000	–
	RBF-GP(10)	50	1,000	600
	FI-GP(2,2)	20	50	–
	FI-RBF-GP(2,2)	50	1,000	0

when a given number of generations (*NumGens*) have been run. Result designation: designate the solution with the best forecasting performance, that ever appeared in any generation of the population, as the result. Table 1 shows lagged and long memory variables (column Method, see Sect. 5.1 for notation), number of generations (column *NumGens*), population size (column PopSize), and the RBF-GP threshold parameter (column Threshold), which are specific to each GP method for each time series (column TS). Data series transformations used in out-of-sample forecasting experiments are the following. CPI: 1-st-order difference $(y'_t = y_t - y_{t-1})$. Food-CPI: 1-st-order difference. PPI: 1-st-order difference and mean adjustment $(y'_t = y_t - \bar{y})$. Food-CPI-WECW: 1-st-order difference. UK-Inflation-Rate: none. Thirty independent runs were performed for each configuration setting (series and GP method shown in Table 1), computing the mean and standard deviation of forecasting performance (SSE, MSE, NMSE, RMSE, and MAE), run CPU time, and solution structural complexity. According to statistical hypothesis tests done with these 30 samples and a significance level $\alpha = 0.05$, all the results of the experiments presented in this study are statistically significant. That is, the differences in performance, including forecasting error, structural complexity and run time, are statistically significant for all algorithms at all forecast horizons (in the forecasting error case) for all problems. Details of the statistical tests are available on request from the corresponding author, or in the Genetic Programming and

Evolvable Machines journal web site. The statistical tests have been carried out by using the software environment for statistical computing and graphics named R (see [84]). Next subsection presents and analyses these out-of-sample forecasting experiments' results.

5.4 Results and analysis of the performance

First, a graphical analysis of the performance of the different approaches, considering each series separately, is done. Then, a general analysis is carried out, taking into account simultaneously all series, methods, and forecast horizons. Finally, model complexity and run CPU time are compared.

Figure 4a shows the results, as root mean squared errors (RMSE), for the CPI series. In general, it's observed that FI-RBF-GP has the best forecasting performance followed by FI-GP and these are followed by the statistical methods. The RBF-GP approach slightly outperforms STD-GP for most horizons. Figure 4b shows the results for the Food-CPI series. The best forecasting performance is achieved by FI-RBF-GP and by FI-GP, FI-RBF-GP being better for most forecast horizons. The RBF-GP approach outperforms the STD-GP approach by a noticeable difference from horizon 1 to horizon 63, from which on both have very similar performance. For the PPI series (see Fig. 4c), the best performance is achieved by the FI-RBF-GP followed very closely by FI-GP, and then, in general, by a greater margin, by the ARFIMA-GPH method. In general, the STD-GP and RBF-GP approaches have very similar forecasting performance, with slight advantages for one or the other, for horizons between 1 and 20. For the Food-CPI-WECW series (see Fig. 4d), for $h = 1$ to $h = 20$, the best performance is achieved with the FI-RBF-GP and ARFIMA-EML methods, FI-RBF-GP being better for most horizons. These are followed by FI-GP. From $h = 20$ on, the best forecasting performance is generally achieved by FI-RBF-GP and FI-GP, followed by ARFIMA-EML (which ranks first from $h = 24$ to $h = 49$ approx.). RBF-GP outperforms STD-GP for all horizons. Figure 4e shows results for the UK-Inflation-Rate series. From $h = 1$ to $h = 7$ the best forecasting perfromance is achieved by ARMA-EML and FI-RBF-GP, from $h = 8$ to $h = 92$ (approx.), by ARFIMA-EML, and from then on, by FI-RBF-GP. RBF-GP and STD-GP perform similarly, but STD-GP outperforms RBF-GP for most horizons. Its important to remark that, even if ARFIMA-EML ranks last for $h = 1$ to $h = 5$, it has the best performance for most other horizons. The two problems that arise with the UK-Inflation Rate series are the series characteristics and the size of the data set, which, for this particular case, affects more the approaches based on GP than the statistical methods. With a total of 284 observations, the UK-Inflation Rate is the one with the least amount of data of the 5 series used in the experiments. Given this, the in-sample preliminary data set and the in-sample validation data set used to set (adjust) the parameters of the approaches based on GP are smaller than for the other series. That is, the insufficient amount of data is a problem that affects more the methods based on GP than the statistical approaches (for which it is not necessary to separate and divide the in sample data set into a preliminary in-sample data set and an in-sample validation data set to estimate parameters). On the other hand, it can be observed in Fig. 4e,

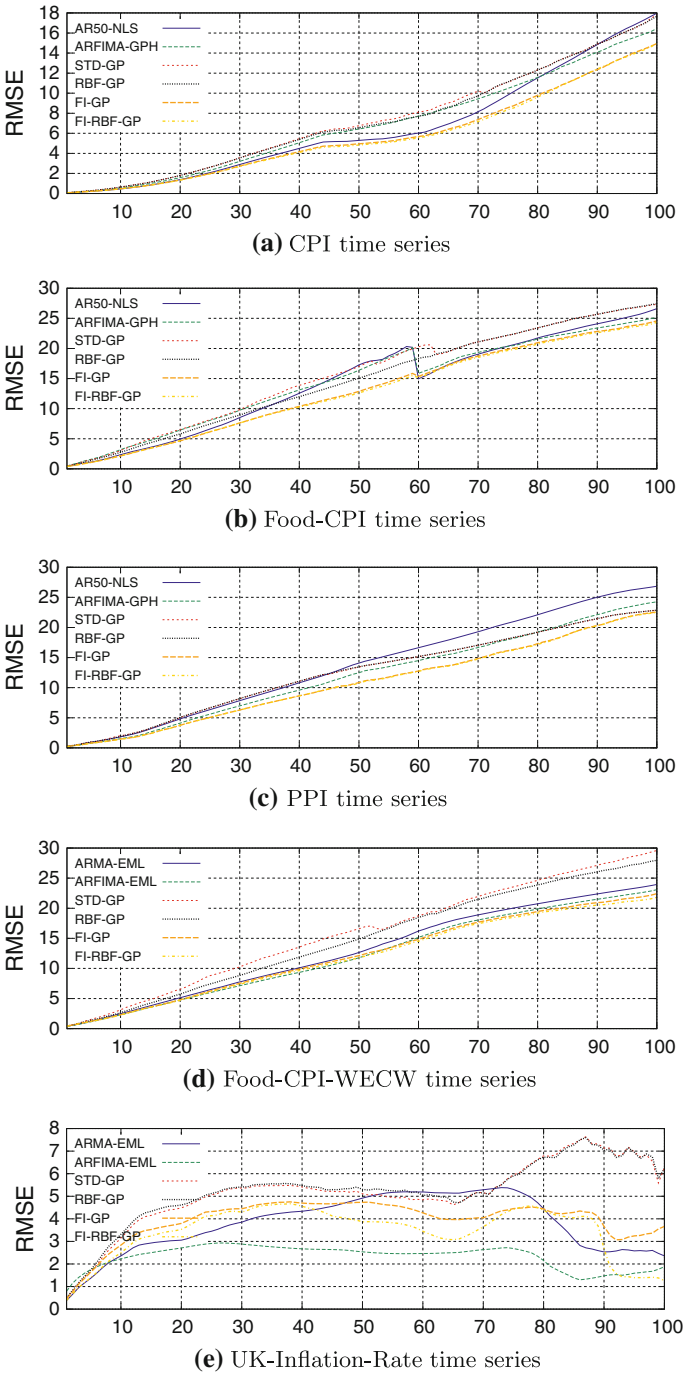

Fig. 4 Methods' RMSE forecasting performance. Horizons 1 to 100. **a** CPI time series. **b** Food-CPI time series. **c** PPI time series. **d** Food-CPI-WECW time series. **e** UK-Inflation-Rate time series

that when considering some horizon intervals, the error tends to reduce on increasing the horizon. This is due to: (1) changes in the variance and the level (mean) of the UK-Inflation-Rate series, and (2), the period of time chosen to perform the out-of-sample forecasts.

In order to compare the different methods for all forecast horizons it's necessary to apply a comparison criterion and a performance measure. Then, the following measure is used to compare forecasting performances:

$$Performance(\mathbf{x}) = \sqrt{k} - \sqrt{\sum_{i=1}^{k} (1 - P_i(\mathbf{x}))^2} \tag{14}$$

where $k = 100, \mathbf{x} \in S_M = \{$ARFIMA-GPH, AR50-NLS, ARFIMA-EML, ARMA-$EML, STD - GP, RBF - GP, FI - GP, FI - RBF - GP\}$, and $P_i(\mathbf{x})$ is the probability that $\mathbf{x}$ has of winning a comparison (assuming minimization) according to the objective function f_i (i.e., the forecasting error for horizon h_i) against another method randomly selected from S_M. The best possible performance value is $\sqrt{k}$, an the worst is zero. Figure 5 shows the forecasting performance of each method for all the time series. It can be observed that, in general, the FI-RBF-GP method has the best forecasting performance, followed by FI-GP, ARFIMA-GPH/EML, and AR50-NLS/ARMA-EML, in that order. On the other hand, it could be said that RBF-GP performs better than STD-GP, given that the former outperformed the latter for three of the five series, and never ranked last for all horizons.

Table 2 reports methods' run CPU time for each time series. The ARFIMA-GPH/EML method has by far the best run CPU time, followed by AR50-NLS/ARMA-EML. FI-GP and FI-RBF-GP rank third (FI-GP does better time than FI-RBF-GP for all but one series). These last two methods reduce the run time required to obtain models, by a great margin regarding STD-GP and RBF-GP. On the other hand, the RBF-GP method takes less run time than STD-GP for all the series.

Table 3 shows the GP methods' structural complexity for each series. The FI-RBF-GP (which ranks first) and FI-GP approaches have by far the lowest

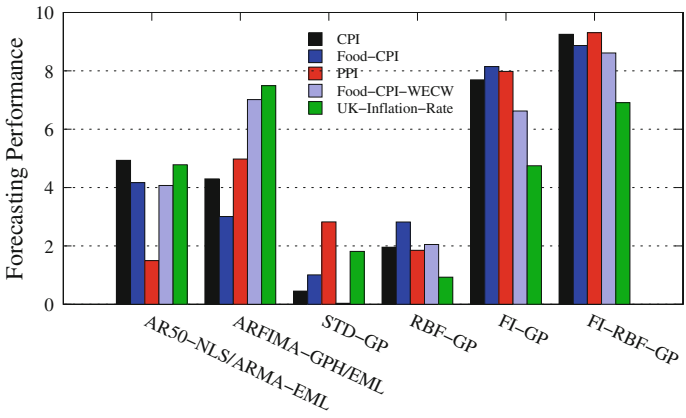

Fig. 5 Forecasting performance comparative bar graph

Table 2 Methods' run CPU time for each time series (in seconds)

Series	AR/ARMA	ARFIMA	STD-GP	RBF-GP	FI-GP	FI-RBF-GP
CPI	1.04588	0.003	177.095	149.866	3.94579	6.46088
Food-CPI	1.06925	0.00325	208.31	183.821	9.33866	9.80201
PPI	0.976375	0.00275	155.331	113.498	10.0162	9.08566
Food-CPI-WECW	0.145625	0.012125	203.57	180.489	7.42333	9.43142
UK-Inflation-Rate	0.064625	0.018	118.328	107.319	0.221656	1.70318

Table 3 Methods' solution structural complexity for each time series

Series	STD-GP	RBF-GP	FI-GP	FI-RBF-GP
CPI	625.438	495.604	53.0081	51.185
Food-CPI	883.004	663.479	91.4719	53.2644
PPI	700.237	443.283	115.417	54.485
Food-CPI-WECW	810.229	612.154	78.33	54.9769
UK-Inflation-Rate	474.6	391.442	18.1	2.27875

structural complexity. The RBF-GP approach reduces complexity between 20% and 40% (approx.) regarding STD-GP.

6 Conclusions and future work

The forecasting performance of statistical long memory time series models has been reported in several publications. The main purpose of this research was the development, application, and evaluation of a computational intelligence method specifically tailored to forecast (univariate) long memory time series, with emphasis on many-step-ahead prediction. Here, the aim has been to obtain useful comprehensible novel CI models with a forecasting performance comparable with or even better than that of statistical methods reported in the long memory literature. However, as mentioned in Sect. 1, on using GP to evolve forecasting models, the problems of curse of dimensionality, bloat, over-fitting, and the tendency to generate high complexity models, may arise. Then, two methods to overcome these problems, namely FI-GP and RBF-GP, have been proposed in this work. FI-GP (the first CI method designed specially to forecast long memory time series) contributes to improving forecasting performance by giving the searching process an initial approximate solution, and to increase comprehensibility of the obtained solutions by means of a comprehensible modular design. It does not introduce any parameters requiring finely tuned settings; it's only necessary to choose the long memory variables. Another objective of this study is the discovery of useful comprehensible novel knowledge, represented as time series predictive models. In this respect, a new evolutionary multi-objective search method (named RBF-GP) to limit complexity of evolved solutions and improve predictive quality has been proposed.

The threshold parameter introduced in this method does not require a finely tuned setting; it can be intuitively set from few possible values. The FI-GP and RBF-GP methods allow one to limit complexity of evolved solutions preventing it from impairing model comprehensibility, improving predictive quality, decreasing run time and memory requirements, and avoiding bloat and (possibly) over-fitting.

The forecasting performance of the FI-GP (Sect. 3), RBF-GP (Sect. 4), and FI-RBF-GP approaches have been evaluated by comparing them with that of statistical methods published in the literature in several-step-ahead forecasting, using real-world economic long memory time series and out-of-sample forecasts. In general, the FI-RBF-GP method has the best forecasting performance, followed by FI-GP, ARFIMA-GPH/EML, and AR50-NLS/ARMA-EML, in that order. On the other hand, it could be said that RBF-GP performs better than STD-GP, given that the former outperformed the latter for three of the five series, and never ranked last for all horizons. Regarding run time, the ARFIMA-GPH/EML method has by far the best run CPU time, followed by AR50-NLS/ARMA-EML. FI-GP and FI-RBF-GP rank third (FI-GP does better time than FI-RBF-GP for all but one series). These last two methods reduce the run time required to obtain models, by a great margin regarding STD-GP and RBF-GP. On the other hand, the RBF-GP method takes less run time than STD-GP for all the series. In addition, both fractionally integrated GP approaches have a very competitive run time, together with the best general forecasting performance. As for structural complexity, the FI-RBF-GP (which ranks first) and FI-GP approaches have by far the lowest complexity. The RBF-GP approach reduces complexity between 20% and 40% (approx.) in comparison to STD-GP. In general, structural complexity of FI-GP and FI-RBF-GP obtained models does not impair nor prevent their comprehensibility. These methods allow for obtaining useful (and possibly comprehensible) novel knowledge, different from what can traditionally be obtained with standard GP or statistical methods.

Future works could include the evaluation of the forecasting performance, run time, structural complexity, and comprehensibility of the FI-GP alternatives described in Sect. 3, such as adjusting the d parameter by using CI methods or choosing it at random ($0 < d < 0.5$), instead of using statistical methods. One of the conclusions of this work is that, on using the recursive method, in spite of *forecasts exceptions* (a concept introduced in this work), it's possible to obtain high (good) forecasting performance with the FI-GP and FI-RBF-GP approaches. However, using the direct method could lead to better results, and then further experiments could use this method. In these experiments a simple method has been applied to identify and deal with forecast exceptions (on using the recursive method). In this respect, future research could consider applying methods from computer science and statistics fields used to identify and deal with outlying observations (which are those that appear to deviate markedly from other members of the sample in which they occur). To further study the performance of the proposed methods (and of standard GP) in long memory time series forecasting, their performance in forecasting simulated long memory time series with different degrees of persistence could be evaluated.

The methods proposed in this work increase performance, reduce computational requirements, and at the same time, allow for obtaining useful (possibly)

comprehensible novel knowledge (represented as forecasting models). The FI-GP and FI-RBF-GP approaches allow for obtaining models specially designed to forecast long memory time series, contributing to the improvement of forecasting performance by giving the searching process an initial approximate solution, and to the increase of the comprehensibility of obtained solutions by means of a comprehensible modular design. Thus, they provide an effective alternative to conventional methods. Finally, we suggest that further research in above mentioned directions could contribute substantially to the state of the art in time series forecasting.

Acknowledgments I would like to thank the anonymous reviewers for their review, comments, and suggestions. The author acknowledges the financial support, offered through a doctoral fellowship, from CONICET (Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina).

References

1. B. Ray, Modeling long-memory processes for optimal long-range prediction. J. Time Ser. Anal. **14**(5), 511–525 (1993)
2. J. Barkoulas, C. Baum, Long-memory forecasting of US monetary indices. J. Forecast. **25**(4) (2006)
3. J. Geweke, S. Porter-Hudak, The estimation and application of long memory time series models. J. Time Ser. Anal. **4**(4), 221–238 (1983)
4. J. Reisinger, K. Stanley, R. Miikkulainen, *Evolving reusable neural modules. Lecture Notes in Computer Science* (Springer, Berlin/Heidelberg, 2004), pp. 69–81
5. P. Evangelista, M. Embrechts, B. Szymanski, Taming the curse of dimensionality in kernels and novelty detection. in *Applied Soft Computing Technologies: The Challenge of Complexity*, ed. by A. Abraham, B.D. Baets, M. Koppen, B. Nickolay (2006), pp. 431–444
6. M. Verleysen, D. Francois, *The Curse of Dimensionality in Data Mining and Time Series Prediction. Lecture Notes Computer Science* (Springer, Berlin/Heidelberg, 2005), pp. 758–770
7. A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, A. Lendasse, Methodology for long-term prediction of time series. Neurocomputing **70**(16–18), 2861–2869 (2007)
8. J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, Cambridge, MA, 1992)
9. S. Luke, L. Panait, A comparison of bloat control methods for genetic programming. Evol. Comput. **14**(3), 309–344 (2006)
10. P. Monsieurs, E. Flerackers, *Reducing Bloat in Genetic Programming. Lecture Notes in Computer Science* (Springer, Berlin/Heidelberg, 2001), pp. 471–478
11. W. Langdon, R. Poli, Genetic programming bloat with dynamic fitness. in *Proceedings of the First European Workshop on Genetic Programming* (Springer, London, 1998), pp. 97–112
12. P. Nordin, W. Banzhaf, Complexity compression and evolution. in *Proceedings of the 6th International Conference on Genetic Algorithms* (Morgan Kaufmann Publishers Inc., San Francisco, CA, 1995), pp. 310–317
13. T. Blickle, L. Thiele, Genetic programming and redundancy. in *Genetic Algorithms within the Framework of Evolutionary Computation* (Workshop at KI-94, Saarbrucken), (Max-Planck-Institut für Informatik, 1994), pp. 33–38
14. W. Langdon, Evolution of size in variable length representations. in *The 1998 IEEE International Conference on Evolutionary Computation*, ICEC'98 (1998), pp. 633–638
15. K. Kinnear Jr, Generality and Difficulty in Genetic Programming: Evolving a Sort. in *Proceedings of the 5th International Conference on Genetic Algorithms* (Morgan Kaufmann Publishers Inc. San Francisco, CA, 1993), pp. 287–294
16. J. Rosca, Generality versus size in genetic programming. in *Genetic Programming 1996: Proceedings of the First Annual Conference* (The MIT Press, Cambridge, MA, 1996), pp. 381–387

17. L. Becker, M. Seshadri, *Comprehensibility and Overfitting Avoidance in Genetic Programming for Technical Trading Rules*. Worcester Polytechnic Institute, Computer Science Technical Report (2003)
18. D. Robilliard, C. Fonlupt, *Backwarding: An Overfitting Control for Genetic Programming in a Remote Sensing Application. Lecture Notes in Computer Science* (Springer, Berlin/Heidelberg, 2002), pp. 245–254
19. N. Nikolaev, L. de Menezes, H. Iba, Overfitting avoidance in genetic programming of polynomials. in *Proceedings of the Evolutionary Computation on 2002. CEC'02. Proceedings of the 2002 Congress-Volume 02* (IEEE Computer Society, Washington, DC, 2002), pp. 1209–1214
20. W. Langdon, R. Poli, *Fitness Causes Bloat: Mutation. Lecture Notes in Computer Science* (Springer, Berlin/Heidelberg, 1998), pp. 37–48
21. W. Banzhaf, W. Langdon, Some considerations on the reason for bloat. Gene. Prog. Evol. Mach. **3**(1), 81–91 (2002)
22. K. Man, Long memory time series and short term forecasts. Int. J. Forecast. **19**(3), 477–491 (2003)
23. H. Hurst, Long-term storage capacity of reservoirs. in *Transactions of the American Society of Civil Engineers*, vol. 116 (American Society of Civil Engineering, 1951), pp. 770–808
24. Z. Ding, C. Granger, R. Engle, A long memory property of stock market returns and a new model. Econ. Soc. Monogr. **33**, 349–372 (2001)
25. N. Crato, de P. Lima, Long-range dependence in the conditional variance of stock returns. Econ. Lett. **45**(3), 281–285 (1994)
26. A. Lo, Long-term memory in stock market prices. Econom. J. Econ. Soc. 1279–1313 (1991)
27. T. Bollerslev, H. Ole Mikkelsen, Modeling and pricing long memory in stock market volatility. J. Econom. **73**(1), 151–184 (1996)
28. U. Hassler, J. Wolters, Long memory in inflation rates: International evidence. J. Bus. Econ. Stat. 37–45 (1995)
29. R. Baillie, C. Chung, M. Tieslau, Analysing inflation by the fractionally integrated ARFIMA-GARCH model. J. Appl. Econom. **11**(1), 23–40 (1996)
30. J. Doornik, M. Ooms, Inference and forecasting for ARFIMA models with an application to US and UK inflation. Stud. Nonlinear Dyn. Econom. **8**(2), 1208–1218 (2004)
31. R. Baillie, S. Chung, Modeling and forecasting from trend-stationary long memory models with applications to climatology. Int. J. Forecast. **18**(2), 215–226 (2002)
32. R. Caballero, S. Jewson, A. Brix, Long memory in surface air temperature detection, modeling, and application to weather derivative valuation. Climat. Res. **21**(2), 127–140 (2002)
33. H. Hamisultane, *Pricing the Weather Derivatives in the Presence of Long Memory in Temperatures*. Technical report, Working paper, EconomiX, Nanterre (2006)
34. J. Hosking, Modeling persistence in hydrological time series using fractional differencing. Water Resour. Res. **20**(12), 1898–1908 (1984)
35. M. Ooms, P. Franses, A seasonal periodic long memory model for monthly river flows. Environ. Modell. Softw. **16**(6), 559–569 (2001)
36. W. Wang, P. Van Gelder, J. Vrijling, Long-memory in streamflow processes of the Yellow river. in *IWA International Conference on Water Economics, Statistics, and Finance* (2005), pp. 8–10
37. B. Mandelbrot, J. Van Ness, Fractional Brownian motions, fractional noises and applications. SIAM Rev. 422–437 (1968)
38. B. Mandelbrot, A fast fractional gaussian noise generator. Water Resour. Res. **7**(3), 543–553 (1971)
39. C. Granger, R. Joyeux, An Introduction to long-memory time series models and fractional differencing. J. Time Ser. Anal. **1**(1), 15–29 (1980)
40. J. Hosking, Fractional differencing. Biometrika **68**(1), 165–176 (1981)
41. P. Bloomfield, An exponential model for the spectrum of a scalar time series. Biometrika **60**(2), 217–226 (1973)
42. J. Beran, *Statistics for long-memory processes* (Chapman & Hall/CRC, London, 1994)
43. Y. Yajima, On estimation of long-memory time series models. Aust. NZ. J. Stat. **27**(3), 303–320 (1985)
44. R. Fox, M. Taqqu, Large-sample properties of parameter estimates for strongly dependent stationary gaussian time series. Annal. Stat. **14**(2), 517–532 (1986)
45. W. Li, A. McLeod, *Fractional time series modelling* (Biometrika Trust, Great Britain, 1986)
46. P. Robinson, Testing for strong serial correlation and dynamic conditional heteroskedasticity in multiple regression. J. Econ. **47**(1), 67–84 (1991)
47. P. Robinson, Efficient tests of nonstationary hypotheses. J. Am. Stat. Assoc. (1994), 1420–1437

48. F. Sowell, Maximum likelihood estimation of stationary univariate fractionally integrated time series models. J. Econom. **53**(1), 2 (1992)
49. G. Bhardwaj, N. Swanson, An empirical investigation of the usefulness of ARFIMA models for predicting macroeconomic and financial time series. J. Econom. **131**(1-2), 539–578 (2006)
50. J. Barkoulas, C. Baum, Long memory and forecasting in Euroyen deposit rates. Asia-Pac Financ. Mark. **4**(3), 189–201 (1997)
51. R. Baillie, Long memory processes and fractional integration in econometrics. J. Econom. **73**(1), 5–59 (1996)
52. J. Holland, *Adaptation in Natural and Artificial Systems* (University of Michigan Press, Ann Arbor, MI, 1975)
53. L. Fogel, A. Owens, M. Walsh, *Artificial Intelligence Through Simulated Evolution* (Wiley, New York, 1966)
54. R. Storn, K. Price, *Differential Evolution-a Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces*. International Computer Science Institute-Publications-TR (1995)
55. P. Smith, *Controlling Code Growth in Genetic Programming. Advances in Soft Computing* (2000), pp. 166–171
56. R. Olsson, Inductive functional programming using incremental program transformation. Artif. Intell. **74**(1), 55–81 (1995)
57. T. Soule, J. Foster, Effects of code growth and parsimony pressure on populations in genetic programming. Evol. Comput. **6**(4), 293–309 (1998)
58. B. Zhang, H. Mühlenbein, Balancing accuracy and parsimony in genetic programming. Evol. Comput. **3**(1), 17–38 (1995)
59. S. Luke, L. Panait, *Fighting Bloat with Nonparametric Parsimony Pressure. Lecture Notes in Computer Science* (2003), pp. 411–421
60. W. Langdon, Data structures and genetic programming. Adv. Genet. Prog. **2**, 395–414 (1996)
61. E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)
62. E. De Jong, R. Watson, J. Pollack, Reducing bloat and promoting diversity using multi-objective methods. in *Proceedings of the Genetic and Evolutionary Computation Conference* (2001), pp. 11–18
63. W. Langdon, J. Nordin, *Seeding Genetic Programming Populations. Lecture Notes in Computer Science* (2000), pp. 304–315
64. A. Ekárt, S. Nemeth, Selection based on the pareto nondomination criterion for controlling code growth in genetic programming. Gene. Prog. Evol. Mach. **2**(1), 61–73 (2001)
65. E. De Jong, J. Pollack, Multi-objective methods for tree size control. Gene. Prog. Evol. Mach. **4**(3), 211–233 (2003)
66. A. Guven, Linear genetic programming for time-series modelling of daily flow rate. J. Earth Syst. Sci. **118**(2), 137–146 (2009)
67. H. Akaike, A new look at the statistical identification model. IEEE Trans. Auto. Cont. **19**(6), 716–723 (1974)
68. N. Nikolaev, L.M. de Menezes, H. Iba, Overfitting avoidance in genetic programming of polynomials. in *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002* (IEEE Press, 2002), pp. 1209–1214
69. H. Iba, Bagging, boosting, and bloating in genetic programming. in *Proceedings of the Genetic and Evolutionary Computation Conference*, Vol. 2. (1999), pp. 1053–1060
70. B.S. Mulloy, R.L. Riolo, R.S. Savit, Dynamics of genetic programming and chaotic time series prediction. in *Genetic Programming 1996: Proceedings of the First Annual Conference*, Stanford University, CA, (MIT Press, 1996), pp. 166–174
71. H. Iba, de H. Garis, T. Sato, A numerical approach to genetic programming for system identification. Evol. Comput. **3**(4), 417–452 (1995)
72. N. Wagner, Z. Michalewicz, M. Khouja, R.R. McGregor, Time series forecasting for dynamic environments: The DyFor genetic program model. IEEE Trans. Evol. Comput. **11**(4), 433–452 (2007)
73. R. Jagielski, *Genetic programming prediction of solar activity. Intelligent Data Engineering and Automated LearningIDEAL 2000. Data Mining, Financial Engineering, and Intelligent Agents* (2009), pp. 191–210
74. N. Nikolaev, H. Iba, Genetic programming of polynomial harmonic models using the discrete Fourier transform. in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*. Vol. 2. (IEEE, 2002), pp. 902–909

75. N. Nikolaev, H. Iba, Regularization approach to inductive genetic programming. Evol. Comput. IEEE Trans. **5**(4), 359–375 (2002)
76. H. Iba, N. Nikolaev, Genetic programming polynomial models of financial data series. in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*. Vol. 2. (IEEE, 2002), pp. 1459–1466
77. R. Steuer, *Multiple Criteria Optimization: Theory, Computation, and Application* (Wiley, New York, 1986)
78. K. Deb, *Multi-objective Optimization Using Evolutionary Algorithms* (Wiley, New York, 2001)
79. A. Konak, D. Coit, A. Smith, Multi-objective optimization using genetic algorithms: a tutorial. Reliab. Eng. Syst. Safe. **91**(9), 992–1007 (2006)
80. C. Coello, G. Lamont, D. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-objective Problems* (Springer, New York , 2007)
81. J. Doornik, *Object-Oriented Matrix Programming using Ox 6.0* (2007) London: Timberlake Consultants Ltd. See http://www.doornik.com
82. J. Doornik, M. Ooms, Computational aspects of maximum likelihood estimation of autoregressive fractionally integrated moving average models. Comput. Stat. Data Anal. **42**(3), 333–348 (2003)
83. A. Fraser, *Genetic Programming in C++ (Gpc++ Version 0.40). Public Domain Genetic Programming System* (1994)
84. R Development Core Team, R: *A Language and Environment for Statistical Computing*. (R Foundation for Statistical Computing, Vienna, Austria, 2010) ISBN 3-900051-07-0