Evolutionary Computing in Statistical Data Analysis

Roberto Baragona and Francesco Battaglia

Abstract. Evolutionary computing methods are being used in a wide field domain with increasing confidence and encouraging outcomes. We want to illustrate how these new techniques have influenced the statistical theory and practice concerned with multivariate data analysis, time series model building and optimization methods for statistical estimates computation and inference in complex systems. The distinctive features all these subject topics have in common are the large number of alternatives for model choice, parametrization over high dimensional discrete spaces and lack of convenient properties that may be assumed to hold at least approximately about the data generating process. Evolutionary computing proved to be able to offer a valuable framework to deal with complicated problems in statistical data analysis and time series analysis and we shall draw a wide though by no means exhaustive list of topics of interest in statistics that have been successfully handled by evolutionary computing procedures. Specific issues will be concerned with variable selection in linear regression models, non linear regression, time series model identification and estimation, detection of outlying observations in time series as regards both location and type identification, cluster analysis and grouping problems, including clusters of directional data and clusters of time series. Simulated examples and applications to real data will be used for illustration purpose through the chapter.

1 Introduction

Evolutionary computing is a general approach for simulating evolution on computers. In a statistical framework, evolutionary computing provides a

Roberto Baragona
Sapienza University of Rome Department of Sociology and Communication Via Salaria 113 Rome Italy
e-mail: roberto.baragona@uniroma1.it

Francesco Battaglia
Sapienza University of Rome Department of Statistics Probability and Applied Statistics Piazzale Aldo Moro 5 00100 Rome Italy
e-mail: francesco.battaglia@uniroma1.it

A. Abraham et al. (Eds.): Foundations of Comput. Intel. Vol. 3, SCI 203, pp. 347–386.
springerlink.com

class of methods useful for identification, estimation, validation and prediction of models that describe relationships of interest among variables linked to real world data sets. Many evolutionary computing methods are referenced in [41] but the usual nowadays classification includes evolutionary programming, evolution strategies, genetic algorithms, estimation of distribution algorithms, differential evolution. The difference between these methods is often subtle, the unifying framework consists in assuming that several potential solutions exist that may solve a problem but the optimal one has to be discovered through an evolution process. Such a process develops along the same guidelines that drive the natural adaptation to the environment typical of the biological populations. So the optimal solution may not even be present in the set of solutions that are considered at the beginning. It is built gradually by selection, recombination and mutation of the solutions that enter the population in several iterative steps usually called generations. In this chapter we will focus on genetic algorithms (GAs) and to a minor extent to the estimation of distribution algorithms (EDAs). For an introduction to GAs see, for instance, [65] and [73], and refer to [61] for EDAs. GAs-based methods have been proposed often to solve problems in the field of statistics. Obviously we may find countless applications in the wide framework of the artificial intelligence (AI) and methods developed for AI problems have found their application in statistics as well. The meta-heuristic methods are general purpose heuristics that include as special case the GAs and the other evolutionary computing methods but extend to cover methods such as threshold accepting (TA), simulated annealing (SA), tabu search (TS) and many others ([85, 86]). Often hybridization has been proposed among meta-heuristic methods to exploit useful features of interest for the problem at hand (see [47] for an example of hybrid algorithm which combines TS and GAs, and [88] who considered SA and GAs).

As far as GAs-based heuristics are concerned, [29], [28] and [68] discussed several applications in statistics, for instance the variable selection and parameter estimation in linear regression model. The standard errors of the estimates are approximately evaluated by processing with the GA several bootstrap samples from the data. Applications of genetic algorithms were proposed as well for component and discriminant analysis ([76]), graphical model identification ([75]), model selection ([2]), subset regression ([55]), crisp and fuzzy cluster analysis ([69], [63], [54]), outlier identification for independent data ([36]), non linear optimization to determine the wavelet filter for M-band wavelet decomposition scheme ([34]), exponential power distribution parameter estimation ([83]) and mixture models investigation ([10]). Promising applications of GAs have been proposed in statistical sampling ([56]) and design of experiments ([27], [42], [51]). In machine vision and pattern recognition framework cluster algorithms were proposed based on SA by [6] and on GAs by [3], [4] and [5].

A comprehensive account on meta-heuristic methods for time series analysis may be found in [9]. Special topics for applications are autoregressive moving

average (ARMA) model identification ([72]), subset ARMA (SARMA) model identification ([43], [64]) and subset vector autoregressive (VAR) model ([17]), cluster of time series ([7, 8]), outlier detection in time series ([14]), threshold autoregressive (TAR) model identification ([87]), modeling structural breaks ([37, 38]), identification of transfer function models ([33]).

The paper is organized as follows. In the next section 2 the GAs and EDAs will be outlined in some details. The remaining sections will illustrate examples of different implementations of GAs depending on the problem. Section 3 is devoted to present two examples of variable selection in multivariate linear regression and in time series SARMA models. Parameter estimation with GAs and EDAs will be considered in section 4 using as an illustration a logistic regression model on the coronary heart disease data taken from [52]. In section 5 comparison is made between meta-heuristic methods and algorithms based on gradient optimization and indirect inference methods for the exponential autoregressive (EXPAR) time series model. Then the identification and estimation of time series threshold models is considered concerned with threshold autoregressive (TAR) models and double threshold autoregressive heteroscedastic (DTARCH) and double threshold generalized ARCH (DT-GARCH) models. The outlier detection problem is accounted for in section 6. In section 7 an example of GAs-based cluster analysis is discussed concerned with time series data and directional data in comparison with TA, SA and TS methods. Section 8 outlines directions for further research and conclusions are drawn.

2 GAs and EDAs Implementations

In the next sections we shall illustrate methods essentially based on GAs. A special attention will be reserved to EDAs which is in the same domain of the GAs and similar in many aspects. For GAs in particular there is a general schema that may be implemented in several ways. The more common will be outlined in this section.

2.1 The GAs Procedure

The basic GAs procedure is usually illustrated by assuming that the potential solutions are encoded as binary strings. A real number $x \in (a, b)$ may be encoded as a binary string of length ℓ as follows

$$x = a + c(b - a)/(2^{\ell} - 1), \tag{1}$$

where c is the non negative integer that may be decoded from the binary string. Given a positive integer s, a set of s binary strings $(i_1, i_2, \ldots, i_\ell)$ of pre-specified length ℓ are generated at random. The s strings form the "initial

population." This latter is a subset of the set of all admissible solutions. Let the initial population be the current population, and perform N iterations in each of which the s strings are allowed to change according to the "genetic operators" selection, crossover and mutation.

1. *Selection.* The objective function to be optimized is called in this context the "fitness function" and it is taken to be maximized. If the problem requires that the objective function f has to be minimized, then we may define the fitness function as $f^* = \exp(-f)$, for instance. Then, for s times, a string in the current population is chosen (with replacement) with probability proportional to its fitness function. We obtain this way s strings that are taken to replace the current population.
2. *Crossover.* For the sake of simplicity, let us assume that s is an even integer. Then, the strings in the current population are paired at random to form $s/2$ pairs. Each pair is examined in turn, and crossover takes place if $U < p_c$, where p_c is a pre-specified crossover probability and U is a uniform random number in the interval $(0, 1)$. The crossover acts as follows. (i) An integer j is chosen uniformly randomly in the interval $(1, \ell - 1)$. The number j is called the "cutting point." (ii) The bits from $j + 1$ to ℓ are exchanged between the strings that are paired.
3. *Mutation.* All chromosomes in the population are allowed to change their values with probability p_m. The choice of p_m was proven to influence the performance of the GA considerably. A high p_m may serve to maintain the diversity between the individuals into the population, but this is likely to produce premature convergence as well. The probability p_m is usually assumed rather small in the interval $(0.001, 0.1)$. A popular rule consists in choosing p_m equal to $1/\ell$, where ℓ is the chromosome length.

The "elitist strategy" (see [73] for motivation) applies, that is the best string that may be found in each of the N iterations is always maintained in the current population unless an even better string appears due to the genetic operators. If this latter is not the case, then such string replaces the string with the worst fitness function. This replacement is done so that the population retains the same number of strings s in each iteration. At the end of each iteration the strings obtained by using the genetic operators replace all the existing strings, and the new population is assumed as the current one. The best string in the population after N iterations is taken as the final solution.

2.2 The EDAs Procedure

These algorithms are best explained, and were originally derived, in the case that the chromosomes are real vectors $x = (x_1, x_2, \ldots, x_\ell)'$, though they have been extended to more general settings. In the real vector case, the problem may be formulated as that of maximizing a fitness function $f(x)$ where x is a real vector $x \in R^\ell$.

The proposal originates from the attempt of explicitly taking into account the correlation between genes of different loci (components of the vector x), that may be seen in good solutions, assuming that such correlation structure could be different from that of the less fitted individuals. The key idea is to deliver an explicit probability model and associate to each population (or a subset of it) a multivariate probability distribution.

An initial version of the estimation of distribution algorithm was originally proposed by [66], and then many further contributions developed, generalized and improved the implementation. A thorough account may be found in [57] and in a second more recent book ([61]).

The estimation of distribution algorithm is a regular stochastic population based evolutionary method, and therefore evolves populations through generations. The typical evolution process from one generation to the next may be described as follows:

1. Generate an initial population $P^{(0)} = \{x_i^{(0)}, i = 1, \ldots, N\}$; $c = 0$.
2. If $P^{(c)}$ denotes the current population, select a subset of $P^{(c)}$: $\{x_j^{(c)}, j \in S^{(c)}\}$ with $|S^{(c)}| = n < N$ individuals, according to a selection operator.
3. Consider the subset $\{x_j^{(c)}, j \in S^{(c)}\}$ as a random sample from a multivariate random variable with absolutely continuous distribution and probability density $p^{(c)}(x)$, and estimate $p^{(c)}(x)$ from the sample.
4. Generate a random sample of N individuals from $p^{(c)}(x)$: this is the population at generation $c + 1$, $P^{(c+1)}$.
5. If a stopping rule is met, stop; otherwise $c + 1 \rightarrow c$ and return to 2.

The originally proposed selection operator was the truncation selection, in other words only the n individuals with the largest fitness out of the N members of the population are selected. Later, it was proposed that other common selection mechanism such as the roulette wheel (proportional selection) or the tournament selection (choice of the best fitted inside a group of k individuals chosen at random) may be adopted.

3 Variable Selection in Linear Regression and ARMA Models

A typical issue in linear model identification is variable selection. A linear relationship is postulated between a dependent variable y and a set of independent variables $\{x_1, x_2, \ldots, x_p\}$. Let n observations be available so that we may write the usual linear regression model

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \ldots + \beta_p x_{pi} + u_i, \qquad i = 1, \ldots, n,$$

where $\beta = (\beta_0, \beta_1, \ldots, \beta_p)'$ is the parameter vector and $u = (u_1, \ldots, u_n)'$ is a sequence of independent and identically distributed random variables with zero mean and unknown variance σ_u^2. Let $y = (y_1, \ldots, y_n)'$ and

$X = [1, x_1, x_2, \ldots, x_p]$ where 1 denotes a column vector of ones. The linear regression model may be written in matrix form

$$y = X\beta + u.$$

If p is large it is desirable to reduce the number of independent variables to the set that includes only the variables really important to explain the variability of y. Common available methods are the iterative forward, backward and stepwise methods. However, we would be more confident about the final result if we could compare several subset alternative simultaneously. The GAs are an easy-to-use tool for performing this comparison exactly.

3.1 Subset Regression

We want to select the variables that really matter in a linear regression. The encoding for using GAs is straightforward as it suffices to define a mapping from a binary string of length p and the parameter sequence $\beta_1, \ldots, \beta_p$. The constant term β_0 is always included in the regression.

Let us illustrate a GAs-based procedure for subset regression on an example of $n = 100$ artificial data and a set of $p = 15$ variables. Both independent and dependent variables are sampled from a standard unit normal distribution. The y are generated by a model with parameters

$$\beta = (.01, 0.7, -0.8, 0.5, .01, .01, .01, -0.7, 0.6, 0.8, .01, .01, .01, .01, .01, .01)'$$

and $\sigma_u^2 = 2.25$. It is apparent that only the variables $1-3$ and $7-9$ impact y significantly.

A GA has been employed to search for the best subset model. The fitness function has been the F statistic. The chromosome is a binary string of length 15, for instance

000110000011100

is decoded to a regression model that includes only the variables $4, 5, 11, 12, 13$ as independent variables to explain y. The population size has been chosen $s = 30$, the maximum number of generations $N = 100$, the generational gap (how many new chromosomes are created) has been set equal to 90% of the past population, $p_c = 0.7$ and $p_m = 1/15$. Roulette wheel rule for selection, single cutting point crossover, binary mutation and the elitist strategy are employed. The fitness function evolution is displayed in figure 1. This is the typical fitness function behavior in the presence of elitist strategy. While fitness function improves quickly in the first iterations, then no better solutions are found and the elitist strategy prevents the fitness function from decreasing. The GA finds the best solution corresponding to $F = 29.8941$ and variables $1-3$ and $7-9$.

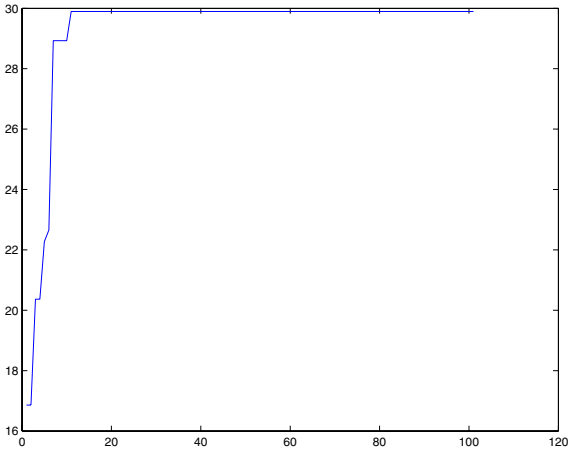

Fig. 1 Fitness function in the subset regression problem versus the number of iterations

A GAs approach to subset regression based on information criteria has been suggested by [55] while GAs have been suggested in [78] to evaluate the bias in parameter estimates produced by omitting variables from the regression model.

3.2 Autoregressive Moving Average Models

Models of the ARMA and ARIMA class are most popular in time series analysis essentially for two reasons: first, they are a natural generalization of regression models, and may be easily interpreted using similar concepts as in regression analysis; and, second, they may be seen as universal approximation for a wide class of well behaved stationary stochastic processes.

An ARMA model may be written

$$x_t - \phi_1 x_{t-1} - \phi_2 x_{t-2} - \ldots - \phi_p x_{t-p} = c + u_t - \theta_1 u_{t-1} - \theta_2 u_{t-2} - \ldots - \theta_q u_{t-q} \quad (2)$$

where $c = \mu(1 - \phi_1 - \ldots - \phi_p)$. Equation (2) is called an ARMA(p, q) model and p and q are known as the autoregressive and the moving average *orders* of the model. If the observed time series $\{x_t\}$ is not stationary, while, for some positive integer d, the differenced series $\{y_t\}$ of order d is stationary, then we have an ARIMA model by replacing x_t with y_t and setting $c = 0$ in (2).

Parsimony is universally accepted as precept among time series analysts, so that the ARMA model building problem may be seen as choosing the model with the smallest number of parameters given an approximation level. An additional way of reducing the number of parameters is considering incomplete models, where some of the parameters $\phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q$ are constrained to zero. Such models are usually referred to as *subset ARMA* models ([30]).

The canonical model building procedure runs iteratively through the following three steps:

1. *Identification.* Selection of the orders p and q, and, if a subset model is considered, choice of which parameters are allowed to be non-zero.
2. *Parameter estimation.* Conditional on identification, the estimation of parameters may be performed through classical statistical inference.
3. *Diagnostic checking.* Once the model is completely specified and estimated, it is customary to check whether it fits the data sufficiently well. This is generally accomplished by computing the residuals. A model is generally accepted provided that the residual are approximately uncorrelated and have zero mean and constant variance.

The most difficult step of the model building procedure is identification. Two different codings have been proposed, in either case a maximum search order has to be selected, both for the autoregressive part (P say) and for the moving average part (Q). The simplest coding amounts to reserving one gene to each possible lag, filling it with 1 if the parameter is free, and with 0 if the parameter is constrained to zero. For example, if we take $P = Q = 6$, the following subset model:

$$x_t = \phi_1 x_{t-1} + \phi_4 x_{t-4} + \phi_5 x_{t-5} + u_t - \theta_2 u_{t-2} - \theta_4 u_{t-4} - \theta_6 u_{t-6}$$

is coded by means of the following chromosome

$$\underbrace{100110}_{\text{AR lags}} \qquad \underbrace{010101}_{\text{MA lags}}$$

This coding system was adopted by most authors, and has the advantage of simplicity and fixed-length chromosomes, but it is not particularly efficient.

An alternative coding based on variable-length chromosomes, was proposed by [64]. The chromosomes consist of two different gene subsets: the first one is devoted to encode the autoregressive and moving average orders, by specifying the number of relevant predictors, i. e., the number of non-zero parameters, respectively for the autoregressive part, p^*, and for the moving average part, q^*. This part consists of eight binary digits and encodes two integer numbers between 0 and 15 (therefore this coding allows for models that have up to 15 non-zero autoregressive parameters and 15 non-zero moving average parameters). The other genes subset is devoted to specifying the lags which the non zero parameters correspond to: it comprises $5(p^* + q^*)$ binary digits, and encodes, consecutively, $p^* + q^*$ integer numbers between 1 and 32. Therefore according to this implementation $P = Q = 32$ and all models containing a maximum of 15 non zero parameters, both for the AR and the MA structures, may be coded. For example. the chromosome corresponding to the previous model is:

$$0011 \;\; 0011 \quad 00001\,00100\,00101\,00010\,00100\,00110\,.$$

The first part of the chromosome contains 8 digits, while the second part has a variable length from zero (white noise) to 150 binary digits. Encoding with the same order limitations would require, in the fixed-length scheme introduced before, 64-digit chromosomes. It appears that the advantage of using the variable length scheme might be sensible if we are searching for relatively sparse models, because the average length depends directly on the number of non zero parameters (for example, a coding admitting no more that 4 non zero parameters in each AR or MA structure, with a maximum lag of 64, would require a fixed chromosome 128 long, and a variable length chromosome with 6 to 54 binary digits).

In any case, the structure of the genetic algorithm depends on the choice of a maximum possible order: it may be based on a-priori considerations, or even on the number of available observations. A more data-dependent choice of the maximum admissible lag could also be based on an order selection criterion computed only on complete models (with all parameters free) as proposed in [22]: in this case a relatively "generous" criterion like the asymptotic information criterion (AIC) (see, for instance, [84], p. 153) seems advisable to avoid reducing the solution space excessively.

Since all proposed codings are binary, each proposal corresponds to usually slight modifications of the canonical genetic algorithm. The selection procedure is obtained by means of the roulette wheel methods, except for [17] who adopt rank selection. The mutation operator is employed in its standard form, while for the cross-over operator some authors propose a random one cut point ([43]), others a random two point cross-over ([67]) or a uniform cross-over ([17]). All authors use unit generational gap and slightly different elitist strategies: saving the best chromosome, or the two best chromosomes, or even the 10% best chromosomes in the population. When dealing with the alternative coding based on subsets of genes that encode the AR and MA order, [64] suggest that the cross-over operator should be modified in order to apply on entire fields (representing the integer numbers denoting order or lag) rather to the single binary digits. Mutation and cross-over probabilities are generally in accordance with the literature on canonical genetic algorithms, proposed values are between 0.001 and 0.1 for mutation, and from 0.6 to 0.9 for cross-over. Not many suggestions are offered concerning the population size, and the chromosomes composing the initial generation (they are generally selected at random in the solution space). It may be reasonably assumed that a good starting point, with satisfying mixing properties, would require initial individuals which may exploit the fitting ability of each single possible parameter. Therefore, advantageous chromosomes in the initial population are those encoding models with just one non zero parameter (i. e., in the first coding scheme, chromosomes with only one gene equal to one). The minimum population size that allows to fully develop this idea is obviously equal to the total number of possible parameters, or $P + Q$.

Much more relevant differences are found in the literature on ARMA model building by means of genetic algorithms, as far as the fitness function is

concerned. Essentially, the fitness function is linked to some version of identification criterion or penalized likelihood. A favorite choice is adopting one of the most popular identification criteria in time series, such as the AIC or the BIC (also called Schwartz) criteria:

$$AIC(M) = N \log\{\hat{\sigma}^2_{(M)}\} + 2p(M)$$

$$BIC(M) = N \log\{\hat{\sigma}^2_{(M)}\} + p(M) \log N$$

where N is the series length, $\hat{\sigma}^2_{(M)}$ is the residual variance estimate for model M, and $p(M)$ is the number of free parameters of model M. Alternatively, [23] suggest their identification criterion called $ICOMP$. Rather than simply considering the number of non zero parameters, $ICOMP$ measures the model complexity through a generalization of the entropic covariance complexity index of [82]. The criterion $ICOMP$ is computed by estimating the Fisher's information matrix for the parameters and by adding to the likelihood- proportional term, $N \log \hat{\sigma}^2$, the quantity $C(\hat{F}^{-1})$:

$$ICOMP(M) = N \log \hat{\sigma}^2_{(M)} + C(\hat{F}^{-1})$$

where

$$C(X) = \dim(X) \log\{\text{trace}(X)/\dim(X)\} - \log|X|\,.$$

Finally, [64] use an AIC-like criterion where the residual variance is replaced by a prediction error variance:

$$s^2(\ell) \propto \sum_t [x_{t+\ell} - \hat{x}_t(\ell)]^2$$

where $\hat{x}_t(\ell)$ is the predictor based on the model. Obviously, for the forecast horizon $\ell = 1$ there is no difference with AIC, [64] try their criterion also for $\ell = 2$ and 3.

A common problem to all these implementations is that the proposed criteria are to be minimized, thus they cannot be employed directly as fitness function (which has, on the contrary, to be maximized). Solution of two kinds have been proposed: [23] avoid the problem of defining a fitness proportionate selection by adopting an ordered fitness selection rule: chromosomes are ordered according to the decreasing values of the $ICOMP$ criterion, and each chromosome is selected with a probability proportional to its rank. An alternative is defining the fitness function by a monotonically decreasing transformation of the identification criterion. Most natural candidates are a simple linear transformation:

$$fitness(M) = W - criterion(M)$$

which is possible if an estimate of the worst possible value of the criterion, W, is available ([43] suggests to compute W on the current population), or a negative exponential transformation:

$$fitness(M) = \exp\{-criterion(M)/d\}$$

where d is a scaling constant. A Boltzman selection procedure is proposed by [43] using the last expression for the fitness, but with a progressively decreasing "temperature" $d_k = (0.95)^k$, where k is the generation number.

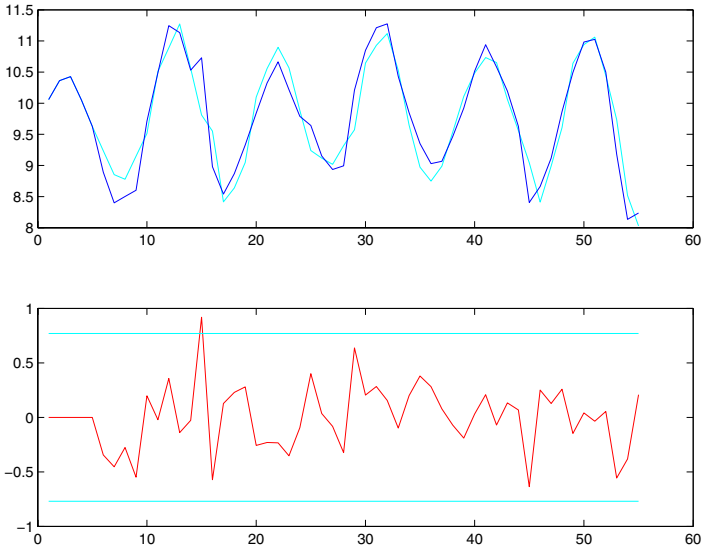

Fig. 2 Yearly number of lynx pelts sold by Hudson's Bay Company in Canada from 1857 to 1911 (top panel) and residuals from the best subset ARMA model (bottom panel)

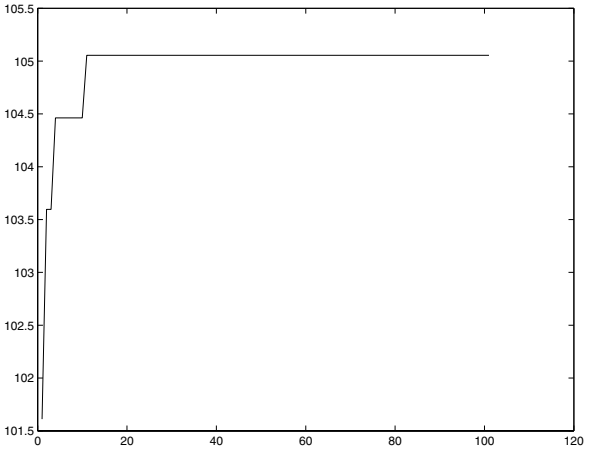

Fig. 3 Fitness function display against iterations of the genetic algorithm

Table 1 Comparison between ARMA and subset ARMA models

Model	c_0	ϕ_1	ϕ_2	ϕ_3	θ_1	θ_2	θ_3	σ_ϵ^2
ARMA	7.08	0.87	0.13	−0.73	0.15	0.35	−0.57	0.0958
	(1.30)	(0.29)	(0.46)	(0.29)	(0.32)	(0.29)	(0.17)	
SARMA	6.77	0.96		−0.64	0.23	0.27	−0.55	0.0962
	(0.70)	(0.06)		(0.06)	(0.15)	(0.15)	(0.16)	

The simulation studies presented in literature suggest satisfying results of each implementation, with slight substantial differences, indicating that for univariate time series with most encountered series lengths, the genetic algorithm yields good models after just a few generations, and converges nearly always, though after many more generations, to the true simulated model.

3.3 An Example of Subset ARMA Fitted to a Real Data Set

Let us consider the yearly number of lynx pelts sold by Hudson's Bay Company in Canada from 1857 to 1911. The data set is composed of 55 observations and is reported in [84], p. 449, as Series W7. Following [84] (p. 150) we use the natural logarithm of the observations for ARMA model identification. In figure 2 the logarithms of the data are displayed (solid line) in the top panel with the predicted values (dotted line) computed from the best subset ARMA model. In the bottom panel of figure 2 the residuals (dotted line) are displayed with the 95% normal bounds (straight lines). The GAs are used for identifying the best subset model. The encoding used to obtain the best subset model is the first one. The fitness function was set equal to the appropriate transform of the AIC criterion. In figure 3 the fitness function evolution is displayed.

In table 1 a comparison is made between the ARMA model and the subset ARMA model fitted to Series W7. The ARMA(3,3) model has a smaller residual variance but its AIC (−103) is slightly greater than the subset model (−105). The difference between the two model is concerned with a single parameter and it seems that little is gained as for diagnostic checking values. Nonetheless there is a considerable advantage as regards the standard errors of the estimates which display a sharp decrease if the subset model is used.

4 The Logistic Regression Model

We give an example of application of the logistic regression model as it has been fitted to the data by GAs by [29] and [68] and by EDAs by [74]. So it

seems of interest to use a logistic regression model to show an application of GAs and EDAs for nonlinear model parameter estimation.

Let y denote a binary dependent variable and $\{x_1, x_2, \ldots, x_p\}$ a set of independent variables. The logistic regression ([52]) assumes a non linear relationship between y, called in this context the response variable, and the covariates $x_1, x_2, \ldots, x_p$. Let Y denote a binary random variable and assume that $y = (y_1, \ldots, y_n)'$ is a sample from Y. Let π be the probability $P(Y = 1|x_1, \ldots, x_p)$ and define the logit transform

$$\text{logit}(\pi) = \log \frac{\pi}{1-\pi}$$

As an example, we fitted a logistic regression model to a real data set, namely the coronary heart disease data from [52]. We used several algorithms to estimate the two parameters β_0 and β_1 of the logistic regression model

$$\text{logit}(\pi_i) = \beta_0 + \beta_1 x_i, \qquad i = 1, 2, \ldots, 100.$$

The observed response variable is $y_i = 1$ if insurgence of coronary heart disease in the ith patient has been recorded and $y_i = 0$ otherwise. The covariate x_i is the ith patient's age (years). The iterative re-weighted least squares (IRLS), the GAs and EDAs algorithms implemented for maximizing the logarithm of the likelihood

$$L = \sum_{i=1}^{n} y_i(\beta_0 + \beta_1 x_i) - \sum_{i=1}^{n} \log\{1 + \exp(\beta_0 + \beta_1 x_i)\}$$

are outlined as follows. Upper and lower bounds for the two parameters have been chosen $(-10, 10)$ for β_0 and $(-2, 2)$ for β_1. The input matrix is defined $X = [1, x]$ where 1 denotes a column vector of ones and the parameter vector is $\beta = [\beta_0, \beta_1]$.

- *IRLS.* This iterative algorithm implements the Newton method applied to the problem of maximizing the likelihood of a response variable y given X. Let a preliminary guess of the model parameter $\hat{\beta}^{(0)}$ be available. Then the following steps describe the move from $\hat{\beta}^{(k)}$ to $\hat{\beta}^{(k+1)}$ at iteration k.

  1. Set, for $i = 1, \ldots, n$,

  $$\pi_i^{(k)} = \exp\left(\hat{\beta}_0^{(k)} + \hat{\beta}_1^{(k)} x_i\right) / \left\{1 + \exp\left(\hat{\beta}_0^{(k)} + \hat{\beta}_1^{(k)} x_i\right)\right\}.$$

  2. Define the weights matrix $W^{(k)} = \text{diag}(w_1^{(k)}, \ldots, w_n^{(k)})$ where $w_i^{(k)} = \pi_i^{(k)}(1 - \pi_i^{(k)})$.
  3. Compute $z^{(k)} = X\hat{\beta}^{(k)} + \left(W^{(k)}\right)^{-1}(y - \pi^{(k)})$.
  4. Solve with respect to $\hat{\beta}^{(k+1)}$ the weighted linear regression problem

$$\left(X'W^{(k)}X\right)\hat{\beta}^{(k+1)} = X'W^{(k)}z^{(k)}$$

5. Replace $\hat{\beta}^{(k)}$ with $\hat{\beta}^{(k+1)}$ and repeat from step 1 until some termination condition is met.

- *GA-1* (binary encoding). The potential solutions to the maximization problem have been encoded as two binary strings of length 20 each. Equation (1) has been used to obtain the value of each of the two parameters given the integer c encoded as a binary string. The Gray code has been used. The population size has been taken equal to 30, the number of iterations has been 300 and 30 bootstrap samples have been generated to compute the estimates as the average estimates and the standard errors. The stochastic universal sampling method has been used for selection, then the single point crossover with $p_c = 0.7$ and the binary mutation with $p_m = 1/20$.
- *GA-2* (real encoding). The potential solutions to the maximization problem have been encoded as floating-point numbers. The population size has been taken equal to 30, the number of iterations has been 300 and 30 bootstrap samples have been generated to compute the estimates as the average estimates and the standard errors. The stochastic universal sampling method (see, for instance, [65], p. 166) has been used for selection, then the line recombination crossover with $p_c = 0.7$ and the floating-point mutation with $p_m = 1/20$.
- *GA-3* ([29]). Equation (1) has been used to obtain the value of each of the two parameters given the integers c_0, c_1 encoded as binary strings. The chromosome is the binary string $c = [c_1, c_2]$. The population size has been taken rather large, 1000 chromosomes, and the number of generations has been chosen equal to 30. Tournament selection with $p_s = 0.7$, single point crossover with $p_c = 0.65$, and binary mutation with $p_m = 0.1$ have been used. An additional operation, the inversion ([65]), has been applied with probability $p_i = 0.75$ to each chromosome in each generation. Inversion consists in choosing two points ℓ_1 and ℓ_2 at random in the chromosome and taking the bits from ℓ_1 to ℓ_2 in reverse order. The standard errors of the parameter estimates are computed by applying the GA on 250 bootstrap samples of the data.
- *GA-4* ([68]). The potential solutions to the maximization problem have been encoded as two pairs of numbers, a real number $r \in (0, 1)$ the first one and an integer in a given interval (N_a, N_b) the second one. The parameter estimates are obtained as $\hat{\beta}_0 = r_0 N_0$ and $\hat{\beta}_1 = r_1 N_1$. The population size has been taken equal to 100, the number of iterations has been 100 and 30 bootstrap samples have been generated to compute standard errors of the parameter estimates. The binary tournament has been used as selection process. Then special crossover (modified uniform crossover) and mutation are suggested. For crossover, the chromosomes are paired at random and the integer parts exchange. If the integer parts are equal, then the exchange

takes place as regards the real part of the chromosome. Only the offspring with better fit is placed in the new generation. Mutation applies, with probability $p_m = 0.1$, only to the real parts of each chromosome. This part, r say, is multiplied by a random number between 0.8 and 1.2, namely r is multiplied by $0.8 + 0.4U$, where U is an uniform random number in $(0, 1)$. If mutation yields a number greater than 1 the first component is set to 1.

- *EDA* ([74]). The potential solutions to the maximization problem are represented by s vectors of 2 real numbers, the parameters β_0 and β_1. The initial population is generated at random. Then the s chromosomes are evaluated according to the likelihood function and the better s^* are retained. Other than using the likelihood, also the AUC ([24]) criterion is suggested as an alternative fitness function. This is recommended when the logistic regression model is used as a classifier and the AUC is the area under the receiver operating characteristic (ROC) curve, a graphical device to describe the predictive behavior of a classifier. The s^* best vectors found are used to estimate a bivariate probability density function. From this latter distribution s new chromosomes are generated and a new iteration starts. We adopted a bivariate normal distribution so that only the mean vector and the variance-covariance matrix are needed to estimate the distribution in each iteration. We assumed $s = 50$ and $s^* = 30$, 300 iterations and 30 bootstrap samples.

The results are reported in table 2 for the 6 algorithms. Estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ are displayed with standard errors enclosed in parentheses. As a measure of adaptation the logarithm of the likelihood L computed on the average estimates is reported.

According to the figures displayed in table 2 the best algorithm is GA-1 as it shows the largest log-likelihood. The second best would be the algorithm GA-4. In this case however the estimates are markedly biased and exhibit the largest standard errors. This result may be explained by the encoding method. Splitting each parameter in two parts has the immediate consequence of increasing the variability of the estimates. Moreover, the fitness function should be linked to the parameters as directly as possible while for algorithm

Table 2 Parameter estimates for logistic regression fitted to the CHD data by using IRLS, 4 GAs and an EDA-based algorithms

	IRLS	GA-1	GA-2	GA-3	GA-4	EDA
$\hat{\beta}_0$	− 5.3195	− 5.1771	− 6.0010	− 5.4360	− 3.1658	− 5.4985
	(1.1337)	(1.3775)	(1.6355)	(1.2580)	(1.9609)	(1.5062)
$\hat{\beta}_1$	0.1109	0.1070	0.1248	0.1130	1.3995	0.1147
	(0.0241)	(0.0290)	(0.0331)	(0.0270)	(0.6422)	(0.0301)
L	−53.6774	−53.1873	−53.8577	−53.6886	−53.4000	−53.6907

GA-4 there is an intermediate step that separates the chromosomes decoding from the fitness function evaluation. The other algorithms lead to slightly smaller values of the log-likelihood and yield results similar to the algorithm GA-1. The algorithms IRLS and GA-3 have the advantage to attain the smallest standard errors of both estimates $\hat{\beta}_0$ and $\hat{\beta}_1$. As an overall result it seems that binary encoding performs better than the real one for evolutionary computing algorithms.

5 Multi-regimes Model Parameter Estimation

Stationary time series do not change their characteristic features through time. This behavior is not always observed in real time series data. A model that fits well the time series in a time interval may prove inadequate in other time intervals. Often the mean of the time series changes with time but this may be found true as well for the variance, the autocorrelation function, the spectral density in some frequency intervals, for instance. On the other hand linear models are specially useful to model time series data. The ARMA specification takes advantage of a well established theoretical background and well known effective procedures for identification, estimation, diagnostic checking, validation and forecasting. The multi-regime models may take non-stationarity into account and use at the same time the ARMA models as different ARMA models apply to different subsets of data. The smooth transition autoregressive models ([81]) are an example of useful multi-regime models where switching from an AR model to another takes place gradually. The threshold models (see [79]) have been developed along the same guidelines except that a step transition replaces the smooth passage between regimes. [37, 38] employed threshold autoregressive processes for modeling structural breaks, and used a GA for identifying the model. As in each regime the ARMA modeling may be used conveniently, the main problem with multi-regime models is the transition parameter or threshold. In this section we examine some algorithms, including meta-heuristics and GAs, that aim at locating the thresholds as accurately as possible. We assume the EXPAR model as a special example in the class of the smooth transition autoregressive models. The threshold models will be examined in the next sections.

5.1 The Exponential Autoregressive Model

The EXPAR(p) model may be written

$$y_t = \{\phi_1 + \pi_1 \exp(-\gamma y_{t-d}^2)\} y_{t-1} + \ldots + \{\phi_p + \pi_p \exp(-\gamma y_{t-d}^2)\} y_{t-p} + e_t, \quad (3)$$

where d is the delay parameter. Unlike linear models, a change of the error variance σ_e^2 by multiplying the $\{e_t\}$ by a constant k, say, does not imply

that the $\{y_t\}$ turn into $\{ky_t\}$. The order of magnitude of $\{y_t\}$ in (3) depends on γ too, in the sense that we may obtain the time series $\{ky_t\}$ by both multiplying σ_e^2 by k^2 and dividing γ by k^2.

The ability of the EXPAR to account for limit cycles depends whether some conditions on the parameters in equation (3) be fulfilled. If the time series is believed to exhibit limit cycles behavior, then the estimation procedure needs to be constrained in some way.

A brief description of the basic parameter estimation procedure proposed by [50] for the estimation of (3) follows. It may be considered as a natural benchmark for competitive alternatives because it is quite straightforward and unlike to fail to yield a solution. It does not ensure, however, that the limit cycles conditions be fulfilled.

The algorithm requires that an interval (a, b), $a \geq 0$, be pre-specified for the γ values in (3). This interval is split in N sub-interval, so that a grid of candidate γ values is built. Let $s = (b - a)/N$ and $\gamma = a$. Then, for N times, the following steps are performed.

1. Set $\gamma = \gamma + s$
2. Estimate ϕ_j e π_j by ordinary least squares regression of y_t on y_{t-1}, $y_{t-1}\exp(-\gamma y_{t-d}^2)$, y_{t-2}, $y_{t-2}\exp(-\gamma y_{t-d}^2)$, ...
3. Compute the AIC criterion and repeat steps 1 and 2 for $N - 1$ times.

Final estimated parameters are taken that minimize the AIC.

For the existence of limit cycles, the following conditions (see, for instance, [71]) are required to hold:
(1) all the roots of

$$z^p - \phi_1 z^{p-1} ... - \phi_p = 0$$

lie inside the unit circle,
(2) some of the roots of

$$z^p - (\phi_1 + \pi_1) z^{p-1} ... - (\phi_p + \pi_p) = 0$$

lie outside the unit circle,
(3)

$$\frac{1 - \sum_{j=1}^{p} \phi_j}{\sum_{j=1}^{p} \pi_j} > 1 \text{ or } < 0.$$

Several algorithms are available for nonlinear models parameter estimation in the presence of either likelihood function or residual sum of squares difficult to maximize or minimize respectively in the presence of many local optima. We select some algorithms to perform a simulation experiment in comparison with the GAs, namely the grid search, indirect inference ([49]), TS, SA and TA.

We carried out a simulation experiment to compare the performance of these methods and of GAs for parameter estimation of the EXPAR model

Table 3 Average estimates for 100 replications from an EXPAR(2)

parameter	ϕ_1	ϕ_2	π_1	π_2	γ			
true value	1.95	−0.96	0.23	−0.24	1.0			
500 obs	$\hat{\phi}_1$	$\hat{\phi}_2$	$\hat{\pi}_1$	$\hat{\pi}_2$	$\hat{\gamma}$	d^2	$\hat{\sigma}^2$	MSE
Grid search	1.76	− .77	− .35	− .32	.71	3.99	1.64	1.64
	(.17)	(.17)	(1.65)	(.28)	(.80)		(.96)	(1.63)
Indirect inf	1.92	−.94	−.40	−.66	1.53	7.14	1.45	1.40
	(.05)	(.05)	(2.0)	(1.38)	(.59)		(.81)	(1.23)
Tabu search	1.91	−.92	.05	−.08	.79	3.73	.99	.97
	(.16)	(.15)	(1.69)	(.25)	(.82)		(.06)	(.20)
Simul anneal	1.62	−.63	.64	−.63	1.05	4.63	1.41	1.52
	(.77)	(.77)	(1.28)	(.75)	(.84)		(.54)	(1.44)
Thre accept	1.84	−.85	−.03	−.45	.88	1.44	1.25	1.25
	(.11)	(.11)	(.86)	(.19)	(.70)		(.36)	(.39)
Genetic alg	1.89	−.91	.07	−.10	.46	3.47	.99	.97
	(.20)	(.18)	(1.58)	(.26)	(.72)		(.06)	(.21)
1000 obs	$\hat{\phi}_1$	$\hat{\phi}_2$	$\hat{\pi}_1$	$\hat{\pi}_2$	$\hat{\gamma}$	d^2	$\hat{\sigma}^2$	MSE
Grid search	1.85	−.86	−.02	−.24	.79	1.03	1.25	1.35
	(.10)	(.10)	(.43)	(.15)	(.83)		(.41)	(1.02)
Indirect inf	1.94	−.95	.37	−.42	1.40	7.58	1.36	1.19
	(.04)	(.04)	(2.19)	(1.45)	(.66)		(.50)	(.41)
Tabu search	1.95	−.96	.20	−.07	.66	1.31	1.0	.99
	(.02)	(.01)	(.76)	(.12)	(.76)		(.05)	(.13)
Simul anneal	1.85	−.86	.36	−.37	1.16	1.19	1.18	1.26
	(.15)	(.15)	(.73)	(.20)	(.70)		(.32)	(.99)
Thre accept	1.85	−.86	.25	−.38	1.20	1.09	1.0	1.0
	(.20)	(.20)	(.70)	(.23)	(.62)		(.05)	(.05)
Genetic alg	1.95	−.96	.19	−.07	.45	1.32	1.0	1.0
	(.02)	(.02)	(.69)	(.12)	(.70)		(.05)	(.13)

with $\phi_1 = 1.95$, $\phi_2 = -0.96$, $\pi_1 = 0.23$, $\pi_2 = -0.24$, $\gamma = 1$ and $d = 1$. This model has been proposed as an example by [50]. We simulated 100 series of 1550 observations by using standard unit Gaussian deviates. For each series, the first 1000 observations have been discarded, and the last 50 set apart for out-of-sample one-step-ahead forecasts. So, for estimation we used 500 observations. Further, 100 series of 2100 observations were generated as well. For each series the first 1000 observations were discarded, but 100 observations were set apart for out-of-sample forecasts. The observations left for estimation purpose were 1000. The results are displayed in table 3. The parameter estimates, averaged over 100 replications, are reported, and their standard errors are enclosed in parentheses. The index d^2 is computed as the average squared Euclidean distance between the two sets of estimated and true parameters. Then, the residual variance $\hat{\sigma}^2$ and the mean square error forecast (MSE) have been computed. The structure of the model has been assumed known, so that the number of parameters has been held fixed.

As far as the residual variance and MSE are concerned, TS and GAs for 500 and TS, TA and GAs for 1000 observations give the best performances as their values are close to one. Parameters ϕ_1 and ϕ_2 are estimated fairly well by all methods and standard errors are small, with the only exception of SA for 500 observations. On the other hand, estimates of π_1, π_2 and γ are often severely biased, and standard errors are large. The smallest squared difference d^2 between true and estimated parameters averaged over 100 replications is obtained by using TA for 500 and grid search and TA for 1000 observations.

5.2 The Generalized EXPAR Model

We may consider the more general EXPAR model (see, for instance, [32])

$$y_t = \{\phi_1 + \pi_1 \exp(-\gamma_1 y_{t-d}^2)\} y_{t-1} + \ldots + \{\phi_p + \pi_p \exp(-\gamma_p y_{t-d}^2)\} y_{t-p} + e_t,$$

where $\gamma_1, \ldots, \gamma_p$ are positive constants and d is the delay parameter. Advantages that may come from this model may consist in greater flexibility, better fit and improved forecasts. On the other hand a grid search for estimating $\gamma_1, \ldots, \gamma_p$ is less efficient and may become infeasible if the model order is large. Then the GAs may constitute a convenient device for estimating the parameters ϕ_j, π_j and γ_j. By using a GAs-based algorithm [15] fitted several "generalized" EXPAR models to the well known Canadian lynx data and sunspot numbers (see [79], chapter 7, for a detailed analysis) and obtained satisfactory results for both time series. We shall report some results concerned with the sunspot numbers.

Computations have been performed on the mean-deleted transformed data $2\{(1 + y_t)^{1/2} - 1\}$ as suggested in [79], p. 420. We considered the AR(9) model reported by [79], p. 423, and the self-excited threshold autoregressive SETAR$(2; 11, 3)$ model proposed by [45], p. 247. Then we estimated using GAs the EXPAR(2), the EXPAR(6) and the EXPAR(9) models with one γ and with 2, 6 and 9 γ's respectively. For estimating the parameters of each model we used the observations from 1700 to 1979, while the observations from 1980 to 1995 were reserved for the multi-step forecasts. The data have been downloaded from the URL: http://www.sidc.be/sunspot-data/ (SIDC-team, Royal Observatory of Belgium, Ringlaan 3, 1180 Brussel, Belgium, The International Sunspot Number, Monthly Report on the International Sunspot Number, online catalogue, yearly data 1700-2007). The results are displayed in table 4. Models are compared by means of the residual variance and the forecasts MSE. Time origins are 1979, 1984, 1987 and lead times $1, 2, \ldots, 8$.

The best forecasts not always are obtained by using models that have the smallest residual variance. The EXPAR(9) model with 9 γ's, for instance, yields the smallest residual variance, but the SETAR$(2; 11, 3)$ model provides the best multi-step forecasts for the years 1980-1987. The results change, however, if different time intervals are considered. Thus, the least mean square forecasts error is observed for the EXPAR(9) with 9 γ's in 1985-1992, for the

Table 4 Sunspot numbers: comparison among AR, SETAR, EXPAR and generalized EXPAR

model	residual variance	*mse* 1980-87	*mse* 1985-92	*mse* 1988-95	*mse* 1980-92
AR(9)	4.05	3.60	16.5	9.01	16.19
SETAR(2; 11, 3)	3.73	1.82	33.51	17.34	22.27
EXPAR(2) one γ	4.90	7.08	65.28	31.39	32.97
EXPAR(2) 2 γ's	4.83	3.77	85.33	29.32	38.46
EXPAR(6) one γ	4.47	7.64	54.74	19.46	21.11
EXPAR(6) 6 γ's	4.34	11.85	42.01	20.62	21.89
EXPAR(9) one γ	3.66	4.99	20.43	8.21	13.02
EXPAR(9) 9 γ's	3.57	2.62	16.34	10.65	10.27

EXPAR(9) with a single γ in 1988-1995. In the wider time span 1980-1992, the EXPAR(9) with 9 γ's is able to produce the best multi-step forecasts. The cyclical behavior of this time series is changing over time, and our models may describe it better in certain years than others. It seems that the EXPAR(9) model with 9 γ's almost always yields the most accurate forecasting performance.

5.3 Threshold Autoregressive Models

A self-exciting TAR (SETAR) model may be written

$$y_t = c_j^{(i)} + \sum_{j=1}^{p} \phi_j^{(i)} y_{t-j} + \varepsilon_t \qquad \text{if} \quad y_{t-d} \in (r_{i-1}, r_i],$$

where $\{\varepsilon_t\}$ is white noise, d is a given positive integer and the k disjoint intervals $(r_{i-1}, r_i]$, $i = 1, \ldots, k$, partition the real axis .

The GAs-based TAR model identification procedure in [87] needs the preliminary specification of the maximum number of "regimes" K ($K \geq 2$), the largest autoregressive model order P, the number of candidate threshold parameters H ($H \geq K-1$) and the number of delay parameters D. If all models had to be enumerated exhaustively their number should be computed

$$(P+1)^K \binom{H}{K-1} D.$$

Such number of candidate solution may obviously become very large. The GA solution ([87]) is based on encoding each of the tentative models as a string which is composed of several "fragments." The first one encodes the delay parameter, the second one the candidate threshold parameters (H "percentiles" were chosen from the ordered observations), then the orders of each of the autoregressive models are encoded. For instance, if $D = 4$, $H = 3$, the

number of regimes is taken equal to 2 and the maximum autoregressive order is $P = 3$, then a string of 9 bits would suffice to represent each and every potential solution. For example, the string

$$01|011|10|11$$

means that $d = 1$, the third percentile value is taken as the threshold parameter, the autoregressive order for the first regime is 2 and that for the second regime is 3. The fitness function is chosen as a modified version of the AIC. The effectiveness of the method is shown by means of both a simulation experiment and some empirical studies for investigating the changing exchange rate of Thailand.

The TAR model easily generalizes to a threshold ARMA model if at least in some regimes an MA part is specified. Obviously in some regimes the model may be a pure MA without an AR part. A GAs-based algorithm may be developed along the same guidelines by defining a chromosome augmented to take the MA part into account.

In the multivariate framework a hybrid algorithm which combines GAs and SA has been proposed by [88] for estimating a threshold vector error correction model. The GA is implemented by using the real encoding and suitable genetic operators for crossover and mutation. In addition each chromosome in the current population is updated only if the offspring is accepted according to the Metropolis rule.

5.4 Double Threshold ARCH and GARCH Models

The autoregressive conditional heteroscedastic (ARCH) models and generalized ARCH (GARCH) have been introduced for modeling volatility clustering. The self-exciting threshold autoregressive ARCH (SETAR-ARCH) is a generalization that accounts for asymmetries in levels. Asymmetries both in levels and volatility may be modeled by the double threshold ARCH (DTARCH) and double threshold GARCH (DTGARCH) models. References for ARCH and GARCH models are [39] and [20], see [58], [59] and [26] for threshold ARCH and GARCH models. The GAs have been considered by [1] for identifying the optimal structure of a GARCH model. For the identification and estimation of DTARCH and DTGARCH models GAs-based methods have been developed by [11] and [13].

A GARCH model takes the form

$$x_t = m(I^t, \theta) + \varepsilon_t \sqrt{v(I^t, \theta)}$$

$$I^t = \{x_{t-1}, x_{t-2}, \ldots, , \varepsilon_{t-1}, \varepsilon_{t-2}, \ldots\} \text{ information at time t,}$$

$$m(I^t, \theta) : \text{conditional mean,}$$

$$v(I^t, \theta) : \text{conditional variance.}$$

If the conditional mean $m(I^t, \theta)$ follows a multi-regime SETAR(p) model and the conditional variance $v(I^t, \theta)$ follows a multi-regime ARMA model SETARMA(s, q) then we may specify the DTGARCH model

$$x_t = \phi_0^{(u)} + \sum_j \phi_j^{(u)} x_{t-j} + a_t \quad \text{if } x_{t-d} \in R_u$$

$$v_t = \alpha_0^{(u)} + \sum_i \beta_i^{(u)} v_{t-i} + \sum_j \alpha_j^{(u)} a_{t-j}^2 \quad \text{if } x_{t-d} \in R_u$$

where R is a partition : $(-\infty, \infty) = R_1 \cup R_2 \cup \ldots \cup R_k$, d is the *delay*, a_t are independent Gaussian zero mean random variables and $E(a_t^2) = v(I^t, \theta)$. The partition sets R_j are always intervals: $R_1 = (-\infty, r_1), R_2 = (r_1, r_2)$, $R_3 = (r_2, r_3)$,... , $R_k = (r_{k-1}, \infty)$, where $r_1 < r_2 < \ldots < r_{k-1}$ are the *thresholds*. In practice the DTGARCH model parameters may be distinguished in *structural parameters*, i.e.

1. delay parameter d
2. regime number k
3. thresholds $(r_1, r_2, \ldots, r_{k-1})$
4. autoregressive orders $(p_1, p_2, \ldots, p_k)$
5. GARCH orders $(q_1, q_2, \ldots, q_k, s_1, s_2, \ldots, s_k)$

and the *equation coefficients*

$$\phi_j^{(u)} \, , \, \alpha_j^{(u)} \, , \, \beta_j^{(u)}$$

subject to stationarity and variance non-negativity constraints.

If the *structural parameters* are given, the equation coefficients may be estimated by maximum likelihood:

$$\log L(x_1, x_2, \ldots, x_n | \phi, \alpha, \beta) = \text{const} - \frac{1}{2} \sum_{u=1}^{k} \{\log(v_t) + a_t^2 / v_t\} i_t^{(u)},$$

where $i_t^{(u)}$ denotes the indicator function of $x_{t-d} \in R_u$. But for the *structural parameters* there is no analytic method available. The only possible procedure which may yield an exact solution consists in:

1. enumerating all possible models
2. estimating the coefficients of each and every model
3. performing diagnostic checking and evaluating all models
4. selecting the best model.

A more viable alternative is determining sets of structural parameters by means of heuristic methods and estimating and comparing the full models.

We shall describe here a hybrid GA for estimating a DTGARCH model where the GA is used for searching for optimal structural parameters d, k,

r_j, p_j, s_j, q_j while the coefficients of the model equations $\phi_j^{(u)}$, $\alpha_j^{(u)}$, $\beta_j^{(u)}$ are estimated by maximizing the log-likelihood.

Chromosome encoding

Let K denote the maximum number of regimes and M the minimum number of observations required in each regime. The chromosome c consists in two separate parts, since orders and thresholds are separately encoded.

- *orders part*
  - The first part c_1 encodes $d, p_1, \ldots, p_k, q_1, \ldots, q_k, s_1, \ldots, s_k$,
  - the binary coding is used to represent integers and
  - mutation and crossover are assumed as in simple GA.
- *thresholds part*
  - The second part c_2 of the chromosome c encodes $(g_1, g_2, \ldots, g_{K-1})$ where g_i represents the number of observations in the ith regime and $g_i \in (M, n)$.
  - This encoding is motivated by the convenience in avoiding *legalization* problems, namely any c_2 is a valid chromosome fragment.

Chromosome decoding

Decoding c_1 is straightforward. For c_2 we have to specify a rule to compute the number of regimes k and the threshold $r_1, \ldots, r_{k-1}$ from $(g_1, g_2, \ldots, g_{K-1})$. The requirement $g_i \in (M, n)$ is needed to ensure that estimation may be performed in each regime easily. We compute the thresholds as follows:

$$r_1 = x_{(g_1)}, r_2 = x_{(g_1+g_2)}, r_3 = x_{(g_1+g_2+g_3)}, \ldots$$

where $(x_{(1)}, x_{(2)}, \ldots, x_{(n)})$ are the ordered data. The number of regimes k is computed as

$$k = \arg\max_{\kappa} \{g_1 + \ldots + g_\kappa < n - M\} + 1 \leq K.$$

In practice only the first $k-1$ out of the $K-1$ integers g_i's have to be computed and we may stop decoding as soon as $g_k \geq n - M$. Then we assume k regimes and thresholds $r_1, \ldots, r_{k-1}$. Note that both the chromosome fragment c_1 and c_2 have fixed length, $\ell_1 = (3k+1)\nu$ the first one and $\ell_2 = (K-1)\nu$ the second one, where ν is the number of bits we adopt to encode the integers as binary strings. The complete chromosome c has fixed length $\ell = \ell_1 + \ell_2$. The genes that do not contribute to the decoding procedure still belong to c and may turn useful for recombination in the later crossover steps.

Genetic operators

The crossover may be performed as usual in c assuming that the genes are integer numbers. Mutation in c_1 may be performed as binary mutation, while mutation in c_2 needs a special procedure. We adopted the following device: if gene g_i mutates then its new value is a uniform random number in the interval $(\max\{M, g_i - M/2\}, g_i + M/2)$.

Fitness function

We have adopted the AIC criterion for fitness function evaluation. Given the chromosome c which encodes the structural parameters, let us maximize the likelihood with respect to ϕ, α, β, i.e.

$$L^*(c) = \sup_{\phi,\alpha,\beta} L(x|\phi, \alpha, \beta)$$

Then the AIC is computed

$$AIC(c) = -2 \log L^*(c) + 2\,(\text{number of parameters}).$$

In order to obtain a positive non decreasing fitness function f the following transform

$$f(c) = \exp\{-AIC(c)/n\}$$

may be used.

5.5 An Application to the Daily Hong Kong Stock Exchange (Hang Seng) Index

As a first example we considered the daily Hang Seng index data from January 1987 to December 1991. If x_t denotes the original data, the *return* series has been computed as

$$y_t = \log(x_t/x_{t-1}).$$

The data behavior suggests to fit a different model to the data recorded and transformed in each of the five years. For instance in 1987 there are 260 observations available. The time series for the year 1987 is displayed in figure 4.

The GAs-based algorithm applied to the Hang Seng index data recorded in year 1987 yields a DTGARCH model with delay parameter $d = 2$ and $k = 2$ regimes with threshold parameter $r_1 = -0.0044$. Model is

$$y_t = -0.0044 + \sum_{j=1}^{4} \phi_j y_{t-j} + a_t$$

$$v_t = 0.0153$$

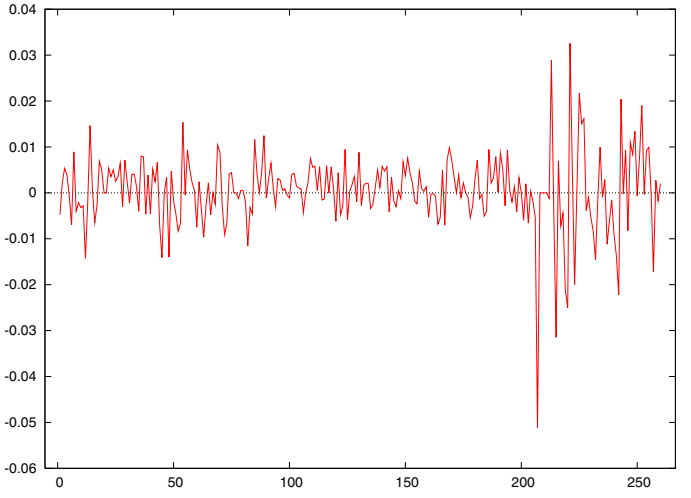

Fig. 4 Differenced logarithmic transform of the daily Hang Seng index data (1987)

if $y_{t-2} < -0.0044$, and

$$y_t = 0.01 + \sum_{j=1}^{7} \phi_j y_{t-j} + a_t$$

$$v_t = 0.002 + 0.0028 v_{t-1} + \sum_{j=1}^{6} \alpha_j a_{t-j}^2$$

if $y_{t-2} \geq -0.0044$. Note that the process is stationary AR homoscedastic when returns are low while the process is heteroscedastic when returns are high.

5.6 An Application to the Daily Exchange Rate Yen/Dollar

As a second example we consider the daily exchange rates yen/dollar from January 1, 1983 to January 28, 1985. Data have not been transformed. There are 541 observations available. The time series data are plotted in figure 5.

We obtained from the GAs-based procedure the following DTGARCH model.

- delay = 2
- two regimes : $x_{t-2} \leq 237 \quad , \quad x_{t-2} > 237$
- first regime: $p = 3, q = 3, s = 3$
  - $x_t = 6.85 + \sum_{j=1}^{3} \phi_j x_{t-j} + a_t$

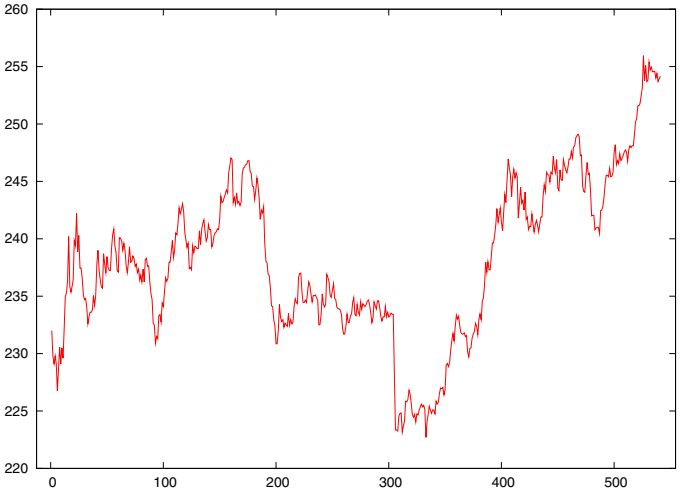

Fig. 5 Daily exchange rate Yen/Dollar from January 1, 1983 to January 28, 1985

  - $v_t = 0.864 + \sum_{i=1}^{3} \beta_i v_{t-i} + \sum_{j=1}^{3} \alpha_j a_{t-j}^2$

- second regime : $p = 1, q = 0, s = 0$
  - $x_t = 2.74 + 0.989 x_{t-1} + a_t$
  - $v_t = 1.151$

Note that when the exchange rate is low the process is AR(3) heteroscedastic while the process is a random walk when the exchange rate is high.

6 Multiple Outliers in Data Sets

Data sets are often affected by unexpected observations or gross errors that may negatively impact data analysis, model estimation and forecasting. Usually a preliminary investigation is performed to identify outlying observations. These latter are generally closely related to missing data treatment and validation procedures. The approach of robust statistics aims essentially at ensuring that reliable estimates may be obtained from the data even in the presence of outliers. We consider here the alternative approach that consists in discovering the outliers and performing some appropriate action. A comprehensive review of statistical methods for the treatment of outliers in data sets is [16]. In the next sections we shall address the identification of outliers in time series as some additional difficulties are involved due to the correlation structure. Though GAs-based methods for outlier detection have been originally introduced for independent data a great deal of work has been devoted to develop evolutionary computing methods for detecting outliers in time series.

6.1 The Outlier Problem in Time Series

Outliers in time series are observations that do not conform to the behavior of the majority of the neighboring time series data. An account may be found in [21], chapter 12. It is customary to distinguish four outlier type, i.e. the additive outlier (AO), the innovation outlier (IO), the transient change (TC) and the level shift (LS). The AO impacts the time series only at the time point where it occurs (a recording error, for instance), but it does not influence any other observation. Special methods are concerned with sequences of consecutive AO's (see, for instance, [53]). The IO affects the random shocks process underlying the time series, and its influence is present in time series points even after its occurrence. TC is a temporary deviation from the expected pattern, while LS is a permanent change in the mean. Iterative procedures such as in [31] are common practice for outlier treatment in time series. The 'skipping' approach (using the Kalman filter to eliminate from computations either AO or missing data) has been compared to the AO approach (replace missing or outlying observations with interpolation estimates) by [48] and the two approach are shown to be equivalent. This method has been implemented in the software TRAMO-SEATS (URL: http://www.bde.es/servicio/software/econome.htm)

In the multivariate framework, detecting the four types of outliers, AO, IO, TC and LS, has been considered by [80]. Vector ARMA modeling forms the basis of their procedure. A different approach consists in projecting the multivariate time series data along low-dimensional or univariate directions and using the computed low-dimensional time series to identify the potential outliers dates. Projection pursuit has been suggested by [44] and independent component analysis by [12].

Using GAs has been proposed by [14] for univariate time series and by [22] for detecting influential observations in dynamic multivariate linear models.

The outlier detection procedures are quite effective if outliers, either isolated or occurring as a "patch," are not too close each others in the time series. If it is not the case, then masking (an outlier may hide another one which is close to it) and smearing (an outlier may impact some subsequent observations so as these latter are recognized in turn as outliers, though they are not) effects, that arise because the time series observations are correlated, make particularly difficult both outlier detection and estimation. Evolutionary computing methods allow several complete outlier patterns to be considered and compared. We assume that there exists a maximum number of outliers K, say, as outliers are to be considered "rare events" and only a limited number, 5% of observations, for instance, may occur. If we want to distinguish m outlier types, the number of outlier patterns is

$$mn + m^2 \binom{n}{2} + m^3 \binom{n}{3} + \ldots + m^K \binom{n}{K}.$$

Searching such a large "solution space," however, is the natural task for a evolutionary computing or meta-heuristic method. We have to adopt appropriate encoding and fitness function to obtain from an evolutionary computing algorithm the optimal (or at least near optimal) solution.

6.2 Genetic Algorithms for Outlier Detection in Time Series

Binary encoding is usually adopted in GAs-based algorithms for outlier detection in time series. A binary string of length $\ell = n$ has to be defined and time points are associated to the bits in the binary sequence. A bit is equal to 1 for an outlying observation and 0 otherwise. A different encoding consists in preparing a list of the time points labels where an outlying observation is supposed to occur, and make it to precede the list of the remaining time point labels, in any order. Such order-based encoding has been suggested by [36] for independent observations, but extension to correlated time series data is straightforward. If, in addition, we want to distinguish the outlier types, we may use m binary strings, each string for each type, under the constraint that in any time point there is no more than a single 1. As an alternative, we may resort to the integer encoding which has been proposed for grouping problems ([40]), by associating an integer code to any outlier type, for example 1 for AO and 2 for IO. Let, for instance, the time series $y=(y_1, y_2, \ldots, y_n)'$ have $n = 30$ observations, and let outliers (any type) be located at $t = 9, 20, 21, 22$. Then, the two encodings may look as follows

$$\begin{array}{cc} \text{binary} & 000000001000000000011100000000 \\ \text{order}-\text{based} & 9 \quad 20 \quad 21 \quad 22 \quad |\ldots\text{other time point labels}\ldots \end{array}$$

Consider instead the case that we want to distinguish between the AO and IO types. If there is a IO at $t = 9$ and there are AO's at $t = 20, 21, 22$, then we may use either the binary or integer encoding

$$\begin{array}{cc} \text{binary} & \begin{cases} 000000000000000000011100000000 \\ 000000001000000000000000000000 \end{cases} \\ \text{integer} & 000000002000000000011100000000 \end{array}$$

For evolutionary computing methods to work in a reasonable time, the fitness function has to be chosen so that it may be properly and quickly computed. We may attempt to minimize the sum of squares computed from some ARMA model fitted to the data. An identification stage, in necessarily automatic way, has to be performed, however, which requires in most cases a considerable computational effort. Then, a valid course of action consists in exploiting the relationship between the linear interpolator and the AO (see [70], p. 237; see also [35]). Only the inverse covariance function ([84], p. 123) is needed which may be easily estimated from the data. Let us consider,

for the sake of simplicity, only the AO type. Let k outliers be located at $t = t_1, \ldots, t_k$. Let us define the $n \times k$ "design matrix" X, where $X_{j,h} = 1$ if $j = t_h$ (that is, the h-th outlying observation is located at time $t = j$) and 0 otherwise. Let $z{=}(z_1, \ldots, z_n)'$ be the observed time series and y the unobserved outlier free realization. Then, the relationship

$$z = X\omega + y$$

holds, where $\omega = (\omega_1, \ldots, \omega_k)'$ is the outlier size array. The likelihood to be maximized is approximately, under Gaussianity assumption,

$$L(X, \omega | y) = (2\pi)^{-\frac{n}{2}} \sqrt{\det(\Gamma i)} \exp\left\{-\frac{1}{2}(z - X\omega)' \Gamma i (z - X\omega)\right\}. \qquad (4)$$

The matrix Γi of the inverse autocovariances may be estimated from the data by using robust techniques (see [46]).

For an illustration of the procedure a simulation experiment is reported. A set of 200 observations have been generated from the ARMA(0,2) model with parameters $\theta_1 = 0.7$ and $\theta_2 = -0.5$ and Gaussian white noise with mean zero and unit variance. By discarding the first 40 artificial observations, we obtain 160 outlier free observations, whose standard error is about 1.5. This time series has been modified by adding 4 to its values at $t = 60$, $t = 62$ and $t = 64$, by subtracting 5 from its value at $t = 100$ and adding 5 to its value at $t = 101$. Time series data are displayed in figure 6. This is a very difficult pattern to detect, and common procedures fail to perform the identification task correctly. In order to discover the outliers in the data, the GA has been employed with crossover probability $p_c = 0.75$ while several mutation probabilities p_m have

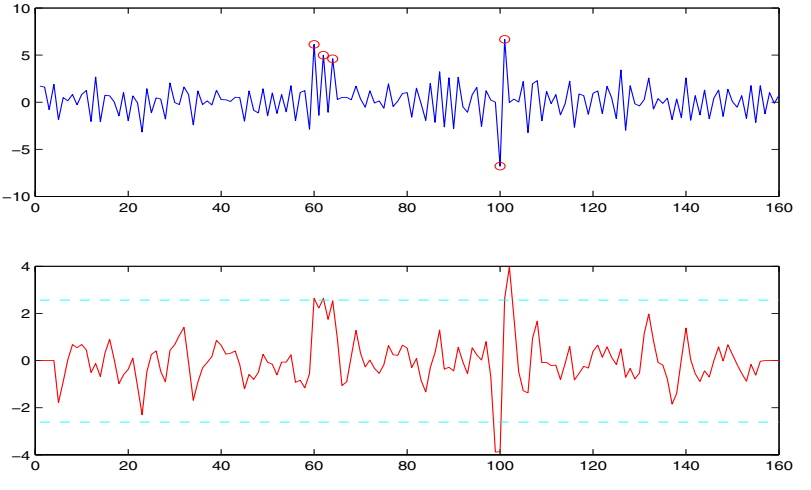

Fig. 6 Top panel, 160 observations simulated from a MA(2) model with additive outliers (circles). Bottom panel, residuals and confidence limits (straight lines)

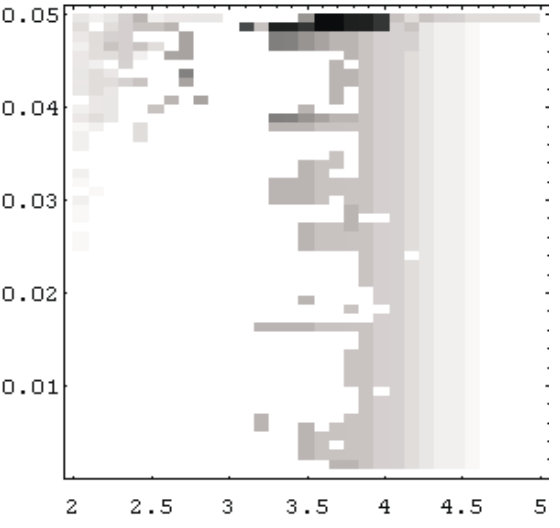

Fig. 7 Density plot of the distance, after 500 iterations, of the fitness from the global maximum, as a function of the constant c and probability of mutation p_m. The probability of crossover is $p_c = 0.75$. The distance (that is, the error) ranges from 0 (white) through 13 (black). The gray levels vary linearly between

been tried, from 0.001 to 0.05 with step 0.001. The inverse variance covariance matrix has been assumed known, so that, if the "right" solution is assumed, then the global maximum of the fitness function equals the logarithm of the maximized likelihood (4), that is $\log(L) = 66.7$, minus $2ck$, where c is a proportionality constant. For $2.52 < c < 4.67$ the maximum of the fitness function coincides with the correct outlier identification while if $c < 2.52$, the global maximum of the fitness function is attained by including, in addition, the "spurious" outlier at $t = 23$. If $c > 4.67$, then the global maximum of the fitness function is attained by considering only the observations at $t = 100$ and $t = 101$ as outlying ones. This circumstance supports the choice $c = 3$, for

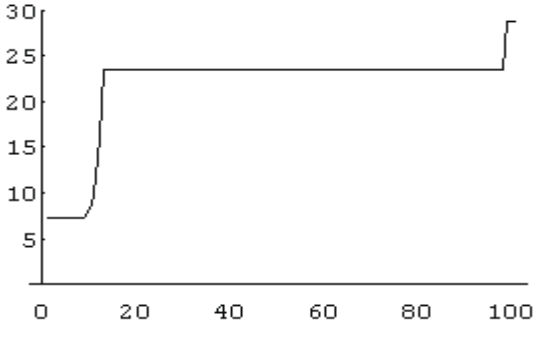

Fig. 8 Fitness as a function of the number of iterations. $p_c = 0.75$, $p_m = 0.015$ and $c = 3.8$. Outliers were found at $t = 102$ (iterations $0-8$); $t = 100$, 101, 102 and 148 (iteration 9); $t = 45$, 100, 101 and 102 (iteration 10); $t = 100$, 101 and 102 (iteration 11); $t = 100$ and 101 (iterations $12-97$); $t = 60$, 62, 64, 100 and 101 (from iteration 98 on: the elitist strategy prevents losing the best chromosome)

instance. In figure 7 the difference between the global optimum of the fitness function and its best value obtained in 500 generations is plotted as a function of both p_m and c, this latter varying from 2 to 5 with step 0.1. We may see that the choice of the mutation probability does not impact the quality of the solution, while the choice of c is very important for the fitness function to characterize the "right" solution. Note that in figure 7 we do not consider the "true" outlier set as the objective of the search, but only the maximization of the fitness function is taken into account. In such perspective, we may observe that the GA performance deteriorates only if c takes values in a rather narrow band. The choice of c may be done by following the usual guidelines, that is c has to be taken low to allow for "high sensitivity," and large for "low sensitivity." As far as the relationship between the fitness function and the number of iterations of the GA is concerned, we found the typical behavior displayed in figure 8. Searching for outliers has been performed by assuming $p_c = 0.75$, $p_m = 0.015$ and $c = 3.8$. The solution has been reached after 98 iterations out of the 500 iterations allowed as a maximum. The persistence of the algorithm in the local maximum corresponding to the outliers at $t = 100$ and $t = 101$ is apparent. The usefulness of the mutation operator is clearly shown, because, at this stage, the searching procedure moves towards the global maximum by mutation. The size of the population seem to be the other parameter to take under control if we want to obtain a near optimal solution in the shortest time.

7 Genetic Algorithms for Cluster Analysis

Methods for cluster analysis are the object of a large literature and are an active research field specially in connection with data mining techniques. For a comprehensive review see [18] and references therein. Let a set of n objects be given and let p measurements concerned with real variables be available for each and every object in the set. Objects and measurements define the usual $n \times p$ data matrix. A line of the matrix is an observation of p variables that characterize the corresponding object and a column is a variable. We assume that a genuine cluster structure exists in the data set. Further, we assume that a similarity (or dissimilarity) measure between each and every pair of objects may be computed from the $n \times p$ measurements. An optimality criterion is assumed to evaluate the internal cluster cohesion and the external cluster dissimilarity (see, for instance, [19]). In multiobjective cluster analysis two or more indexes are used to decide the cluster membership of an object.

A cluster analysis algorithm aims at grouping the objects so that the resulting cluster structure satisfies the optimality criterion. Every object belongs to a cluster and if none is similar to any other it forms a cluster on its own. This is a hard or crisp partition. We may consider fuzzy clustering by allowing an object to belong to more than a cluster according to some 'membership degree'.

The evolutionary computing methods and in general the methods in the class of meta-heuristics are appropriate for dealing with cluster analysis as the number of groups that may be formed with n objects is large even if n is rather small. If the number of groups may vary from 1 to a pre-specified maximum G, then

$$N(G,n) = \sum_{g=1}^{G} \frac{1}{g!} \sum_{j=1}^{g} (-1)^{g-j} \binom{g}{j} j^n$$

is the number of possible clusters.

7.1 Genetic Clustering Algorithms

Many GAs-based procedures have been developed to solve cluster analysis problems. Such procedures take several different features into account, for instance large data sets, cluster constraints and special data structures. GAs-based algorithms combined with the well known k-means and k-medoids algorithms have been considered in [69]. Variable length chromosomes have been suggested by [63] in the context of fuzzy cluster analysis. [54] deals with cluster analysis of panel data. Multi-objective GAs-based cluster analysis has been introduced in [4].

As a typical example of implementation of GAs in a cluster analysis problem we shall give a brief description of the algorithm Genetic Clustering for Unknown K (GCUK) developed by [3]. GCUK essentially combines the k-means algorithm with a GA procedure. There are two main improvements with respect to the basic k-means algorithm, namely the number of groups is unknown and does not have to be pre-specified, and the possibility that the procedure yields a local optimum as a result is greatly reduced. The GCUK algorithm requires that a suitable interval $[g_{\min}, g_{\max}]$, where $g_{\min} > 1$ and $g_{\max} \leq n$ has to be pre-specified. A fixed length chromosome is assumed and $\ell = pg_{\max}$, where p is the number of measurements. The characteristic features of GCUK may be summarized as follows.

- *Encoding.* Any solution is coded as a string of $g_{\max}$ sets of centroid coordinates. These latter are vectors of p floating-point numbers each of which represents a cluster. Some of the centroids may correspond to empty clusters. In this case the symbol # ('don't care') is used to make such circumstance clear. As $g > 1$ the number of symbols # cannot be greater than $g_{\max} - 2$. In general, the chromosome will contain, arranged in any order, g centroids and $g_{\max} - g$ symbols #.
- *Fitness function.* Each chromosome is associated, as a measure of adaptation to the environment, an index of cluster validity. As evolutionary algorithms usually maximize the fitness function the reciprocal of the index is assumed if the optimum corresponds to the smallest index value.

- *Initial population.* Let s denote the population size. For $i = 1, 2, \ldots, s$ an integer g_i is generated uniformly randomly in $[g_{\min}, g_{\max}]$ and g_i objects are chosen at random. Each one of these g_i objects are assigned to a set of p consecutive genes selected at random within the chromosome. The genes that are left unassigned are marked with the symbol #.

The genetic operators implemented by GCUK are the roulette wheel rule for selection, the single point crossover and mutation. However, it has to be noticed that the crossover is implemented by assuming a centroid as undivided, i.e. recombination is performed by exchanging centroids. Moreover, mutation is performed as usual in the presence of floating-point encoding. Each and every measurement may change with probability p_m of a small amount δ around its present value. δ is a number generated from the uniform distribution in $(0, 1)$ and the $+$ or $-$ sign occurs with equal probability.

7.2 Cluster of Time Series

Grouping a set of time series in smaller subsets may provide us with interesting information about the time series structure. For example, time series that follow similar models, or are strongly correlated in some sense may be assumed to belong to the same subset (cluster). Several different measures of similarity (or dissimilarity) have been proposed. A comprehensive review of cluster of time series may be found in [60]. Genetic algorithms were applied by [7, 8] for clustering time series according to either time series cross correlations or phase spectrum dissimilarities. In this latter case, the statistics for directional data introduced by [62] has been used.

The problem that we want to examine here in some detail is concerned with finding a partition of a set of time series according to their cross correlations computed after pre-whitening. Each set of the estimated partition is a cluster which groups together time series that may, for instance, be joint modeled, or are sharing properties of interest, such as correlation with some composite indicator. In this context, we shall define a cluster as a set (group) of time series that satisfy the following condition ([89]). Given a set of k stationary time series $\{x_1, \ldots, x_k\}$, where $x_i{=}(x_{i,1}, \ldots, x_{i,n})'$, $i = 1, \ldots, k$, a subset C which includes k' series ($k' < k$) is said to form a group if, for each of the $k'(k'-1)/2$ cross-correlations $\rho_{i,j}(\tau)$, we have

$$|\rho_{i,j}(\tau)| > c(\alpha) \tag{5}$$

for at least a lag τ between $-m$ and m, and $i, j \in C, i \neq j$. A positive integer m has to be pre-specified which denotes the maximum lag. The cross-correlations $\rho_{i,j}(\tau)$ have to be computed from the pre-whitened time series (see, for instance, [25], p. 232). If all time series have n as a common number of observations, then choosing the significance level $\alpha = 0.05$, say, gives the

figure $c(\alpha) = 1.96/\sqrt{n}$ in (5). The previously stated definition does not exclude that a time series may belong to more than a single group. Then there are possibly several allowable partitions to consider, and their number may be very large. In [7] a GA is developed to find the optimal partition that fulfills equation (5) in each cluster.

The fitness function is based on the k-min cluster criterion ([77]) and may be defined

$$f^+(C_1, C_2, \ldots, C_g; g) = \sum_{\omega=1}^{g} \sum_{i,j \in C_\omega, i \neq j} d^+_{i,j}, \tag{6}$$

where

$$d^+_{i,j} = \max_{\tau \in (-m,m)} (1 - |\rho_{i,j}(\tau)|)$$

and g is assumed unknown. When using (6) each and every cluster needs to be a group, according to (5), for, otherwise, any algorithm, unless prematurely ended, will put together all time series into a single cluster.

It looks convenient to code any admissible time series partition in permutation form. Each time series is labeled with a positive integer number between 1 and k. Then, let $(i_1, i_2, \ldots, i_k)$ be a permutation of $(1, 2, \ldots, k)$. Given the significance level α, the permutation will be given its proper meaning as follows.

1. The first time series, labeled i_1, is taken as the first element of the first cluster.
2. Let i_2 be considered. If the maximum absolute value cross-correlation between the time series i_1 and i_2, computed after pre-whitening, is greater than $c(\alpha)$, then i_2 joins i_1 into the first cluster. Otherwise, the time series i_2 is to become the first element of the second cluster.
3. The i_j-th time series joins an existing cluster if (5) turns true for all pairs belonging to it. If such a circumstance applies for more than one cluster, then the cluster ω is chosen for which $\sum_{i \in C_\omega} d^+_{i,i_j}$ is greatest.
4. The decoding procedure ends as soon as each time series belongs to a cluster.

The choice criterion included into step 3 may look somewhat arbitrary, but it proved necessary, for if, for instance, the time series were assigned so as to maximize the overall criterion, then some undesirable penalization of small clusters would be introduced.

In [7] three procedures were proposed for solution each of which was designed by implementing TS, SA and GAs respectively and several simulation experiments were carried out. Results showed that the three algorithms may be considered effective in recovering correctly the cluster of time series. Further computations on the same artificial data sets by using an implementation of TA produce similar results (not shown here).

8 Concluding Remarks

Evolutionary computing methods provide useful tools for handling many difficult problems that arise in statistical data analysis. However, as for the other fields of application, their usefulness is best exploited if the particular problem involves the search in a finite, but very large, set of discrete parameters. In order that a problem may really require that an evolutionary computing method be implemented for its solution, essentially three circumstances have to be verified.

1. The space of the solutions is quite large.
2. The problem may be coded directly in a natural meaningful way.
3. The objective function to be optimized has to be readily and quickly computed.

In general, it is convenient to resort to evolutionary computing when the objective function to optimize does not meet the usual mathematical requirements, such as continuity, differentiability and convexity.

In statistical data analysis, we could see that there are problems that are suited for use with evolutionary computing techniques, whilst others had better solved by gradient-based techniques. For instance, it is not advisable to employ an evolutionary computing method to estimate the parameters of an ARMA model, but it is convenient to use evolutionary computing or other meta-heuristics if subset ARMA models have to be identified. We reviewed some important problems that are commonplace in statistical data analysis and may need evolutionary computing techniques for reliable solution. These are the estimation of some special non linear models, the identification of threshold parameters in AR and ARCH models, the identification and estimation of subset ARMA and VAR models, detection of location and type of outlying observations, cluster of time series.

Other topics may be envisaged where evolutionary computing methods may turn useful, though not always specific and detailed approach have been fully developed. These are, for instance, the identification and estimation of more general time series state dependent models, the filter design and wavelet filtering, the detection of outliers in vector time series and in non linear time series, the development of new methodological tools for statistical design of experiments. For some of these problems guidelines were provided, however. The algorithms that have been designed for threshold autoregressive model identification and estimation may be extended to include multivariate models. The filter design by genetic algorithm may be extended to wavelet filtering. In the GA framework, development of symbolic regression systems has been considered. In symbolic regression the algorithm is designed to find both the functional specification from a given set of suited functions and the parameter estimates. This same principle applies for selecting wavelets and parameters to optimize the fitness function. Another interesting application for wavelet filtering design is using evolutionary computing-based techniques to select

the coefficients to be set to zero in the wavelet signal expansion. Moreover, a promising field for applications may be the outlier detection in vector time series linear models, and in non linear time series, either univariate or multivariate. Some procedures exist that may be investigated, generalized, and checked by simulation studies. Needless to say, even the fields where evolutionary computing proved to be particularly useful in dealing with statistical data were not yet fully studied. Better understanding is needed on how the evolutionary computing-based techniques work in some specific problems, such as clustering time series, and better encoding and design are likely to be able to greatly improve their performance.

Acknowledgements. Support from EU Commission under contract MRTN-CT-2006-034270 Marie Curie Research and Training Network "COMISEF" Computational Optimization Methods in Statistics, Econometrics and Finance, and from Sapienza University of Rome is gratefully acknowledged.

References

1. Adanu, K.: Optimizing the garch model - An application of two global and two local search methods. Computational Economics 28, 277–290 (2006)
2. Balcombe, K.G.: Model selection using information criteria and genetic algorithms. Computational Economics 25, 207–228 (2005)
3. Bandyopadhyay, S., Maulik, U.: Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognition 35, 1197–1208 (2002)
4. Bandyopadhyay, S., Maulik, U., Mukhopadhyay, A.: Multiobjective Genetic Clustering for Pixel Classification in Remote Sensing Imagery. IEEE Transactions on Geoscience and Remote Sensing 45, 1506–1511 (2007)
5. Bandyopadhyay, S., Mukhopadhyay, A., Maulik, U.: An improved algorithm for clustering gene expression data. Bioinformatics 23, 2859–2865 (2007)
6. Bandyopadhyay, S., Saha, S., Maulik, U., Deb, K.: A Simulated Annealing Based Multi-objective Optimization Algorithm: AMOSA. IEEE Transaction on Evolutionary Computation 12, 269–283 (2008)
7. Baragona, R.: A simulation study on clustering time series with metaheuristic methods. Quaderni di Statistica 3, 1–26 (2001)
8. Baragona, R.: Further results on Lund's statistic for identifying cluster in a circular data set with application to time series. Communications in Statistics – Simulation and Computation 32(3) (2003)
9. Baragona, R.: General local search methods in time series. Contributed paper at the International Workshop on Computational Management Science, Economics, Finance and Engineering, Limassol, Cyprus, March 28-30, 2003, vol. 2003(10), pp. 28–59 (October 2003), `http://www.sciencedirect.com/preprintarchive`
10. Baragona, R., Battaglia, F.: Multivariate mixture models estimation: a genetic algorithm approach. In: Schader, M., Gaul, W., Vichi, M. (eds.) Between Data Science and Applied Data Analysis, Series: Studies in Classification, Data Analysis and Knowledge Organization, pp. 133–142. Springer, Berlin (2003)

11. Baragona, R., Battaglia, F.: Genetic algorithms for building double threshold generalized autoregressive conditional heteroscedastic models of time series. In: Rizzi, A., Vichi, M. (eds.) Compstat 2006 - Proceedings in Computational Statistics, 17th Symposium Held in Rome, Italy, pp. 441–452. Springer, Berlin (2006)
12. Baragona, R., Battaglia, F.: Outliers detection in multivariate time series by independent component analysis. Neural Computation 19, 1962–1984 (2007)
13. Baragona, R., Cucina, D.: Double threshold autoregressive conditionally heteroscedastic model building by genetic algorithms. Journal of Statistical Computation and Simulation 78, 541–559 (2008)
14. Baragona, R., Battaglia, F., Calzini, C.: Genetic algorithms for the identification of additive and innovation outliers in time series. Computational Statistics & Data Analysis 37, 1–12 (2001)
15. Baragona, R., Battaglia, F., Cucina, D.: A note on estimating autoregressive exponential models. Quaderni di Statistica 4, 71–88 (2002)
16. Barnett, V., Lewis, T.: Outliers in Statistical Data, 3rd edn. John Wiley & Sons, Chichester (1994)
17. Bearse, P., Bozdogan, H.: Subset selection in vector autoregressive models using the genetic algorithm with informational complexity as the fitness function. Systems Analysis Modelling Simulation 31, 61–91 (1998)
18. Berkhin, P.: Survey of clustering data mining techniques. Technical Report, Accrue Software, San Jose, California (2002), `http://citeseer.nj.nec.com/berkhin02survey.html`
19. Bezdek, J.C., Pal, N.R.: Some new indexes of cluster validity. IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics 28, 301–315 (1998)
20. Bollerslev, T.: A generalized autoregressive conditional heteroskedasticity. Journal of Econometrics 31, 307–327 (1986)
21. Box, G.E.P., Jenkins, G.M., Reinsel, G.C.: Time Series Analysis: Forecasting and Control, 3rd edn. Prentice Hall, Englewood Cliffs (1994)
22. Bozdogan, H.: Information complexity criteria for detecting influential observations in dynamic multivariate linear models using the genetic algorithm. Journal of Statistical Planning and Inference 114, 31–44 (1988)
23. Bozdogan, H., Bearse, P.: ICOMP: A new model-selection criterion. In: Bock, H.H. (ed.) Classification and Related Methods of Data Analysis, pp. 599–608. Elsevier Science Publishers, Amsterdam (2003)
24. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30, 1145–1159 (1997)
25. Brockwell, P.J., Davis, R.A.: Introduction to Time Series and Forecasting. Springer, New York (1996)
26. Brooks, C.: A double-threshold GARCH model for the French Franc Deutschmark exchange rate. Journal of Forecasting 20, 135–143 (2001)
27. Broudiscou, A., Leardi, R., Phan-Tan-Luu, R.: Genetic algorithms as a tool for selection of *D*-optimal design. Chemometrics and Intelligent Laboratory Systems 35, 105–116 (1996)
28. Chatterjee, S., Laudato, M.: Genetic algorithms in statistics: procedures and applications. Communications in Statistics – Theory and Methods 26(4), 1617–1630 (1997)
29. Chatterjee, S., Laudato, M., Lynch, L.A.: Genetic algorithms and their statistical applications: an introduction. Computational Statistics & Data Analysis 22, 633–651 (1996)

30. Chen, C.W.S.: Subset selection of autoregressive time series models. Journal of Forecasting 18, 505–516 (1999)
31. Chen, C., Liu, L.-M.: Joint estimation of model parameters and outlier effects in time series. Journal of the American Statistical Association 88, 284–297 (1993)
32. Chen, R., Tsay, R.S.: Functional-coefficient autoregressive models. Journal of the American Statistical Association 88, 298–308 (1993)
33. Chiogna, M., Gaetan, C., Masarotto, G.: Automatic identification of seasonal transfer function models by means of iterative stepwise and genetic algorithms. Journal of Time Series Analysis 29, 37–50 (2008)
34. Chitre, Y., Dhawan, A.P.: M-band wavelet discrimination of natural textures. Pattern Recognition 32, 773–789 (1999)
35. Choy, K.: Outlier detection for stationary time series. Journal of Statistical Planning and Inference 99, 111–127 (2001)
36. Crawford, K.D., Wainwright, R.L.: Applying genetic algorithms to outlier detection. In: Eshelman, L.J. (ed.) Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 546–550. Morgan Kaufmann, San Mateo (1995)
37. Davis, R.A., Lee, T.C.M., Rodriguez-Yam, G.A.: Structural break estimation for nonstationary time series models. Journal of the American Statistical Association 101, 223–239 (2006)
38. Davis, R.A., Lee, T.C.M., Rodriguez-Yam, G.A.: Break detection for a class of nonlinear time series models. Journal of Time Series Analysis 29, 834–867 (2008)
39. Engle, R.F.: Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econometrica 50, 987–1007 (1982)
40. Falkenauer, E.: Genetic Algorithms and Grouping Problems. Wiley, New York (1998)
41. Fogel, D.B.: Evolutionary computation: toward a new philosophy of machine intelligence. IEEE Press, New York (1998)
42. Forlin, M., Poli, I., De March, D., Packard, N., Gazzola, G., Serra, R.: Evolutionary experiments for self-assembling amphiphilic systems. Chemometrics and Intelligent Laboratory Systems 90, 153–160 (2008)
43. Gaetan, C.: Subset ARMA model identification using genetic algorithms. Journal of Time Series Analysis 21, 559–570 (2000)
44. Galeano, P., Peña, D., Tsay, R.S.: Outlier detection in multivariate time series by projection pursuit. Journal of the American Statistical Association - Theory and Methods 101, 654–669 (2006)
45. Ghaddar, D.K., Tong, H.: Data transformation and self-exciting threshold autoregression. Applied Statistics 30, 238–248 (1981)
46. Glendinning, R.H.: Estimating the inverse autocorrelation function from outlier contaminated data. Computational Statistics 15, 541–565 (2000)
47. Glover, F., Kelly, J.P., Laguna, M.: Genetic algorithms and tabu search: hybrids for optimization. Computers and Operations Research 22, 111–134 (1995)
48. Gomez, V., Maravall, A., Peña, D.: Missing observations in ARIMA models: Skipping approach versus additive outlier approach. Journal of Econometrics 88, 341–363 (1999)
49. Gourieroux, C., Monfort, A., Renault, E.: Indirect inference. Journal of Applied Econometrics 118, S85–S118 (1993)
50. Haggan, V., Ozaki, T.: Modelling nonlinear random vibrations using an amplitude-dependent autoregressive time series model. Biometrika 68, 189–196 (1981)

51. Heredia-Langner, A., Carlyle, W.M., Montgomery, D.C., Borror, C.M., Runger, G.C.: Genetic algorithms for the construction of D-optimal designs. Journal of Quality Technology 35, 28–46 (2003)
52. Hosmer, D.W., Lemeshow, S.: Applied Logistic Regression, 2nd edn. John Wiley & Sons, Hoboken (2000)
53. Justel, A., Peña, D., Tsay, R.S.: Detection of outlier patches in autoregressive time series. Statistica Sinica 11, 651–673 (2001)
54. Kapetanios, G.: Cluster analysis of panel data sets using non-standard optimisation of information criteria. Journal of Economic Dynamics and Control 30, 1389–1408 (2006)
55. Kapetanios, G.: Variable selection in regression models using nonstandard optimisation of information criteria. Computational Statistics & Data Analysis 52, 4–15 (2007)
56. Keskinturk, T., Er, S.: A genetic algorithm approach to determine stratum boundaries and sample sizes of each stratum in stratified sampling. Computational Statistics & Data Analysis 52, 53–67 (2007)
57. Larrañaga, P., Lozano, J.A.: Estimation of distribution algorithms: a new tool for evolutionary optimization. Kluwer, Boston (2002)
58. Li, W.K., Lam, K.: Modelling asymmetry in stock returns by threshold autoregressive conditional heteroscedastic model. The Statistician 44, 333–341 (1995)
59. Li, C.W., Li, W.K.: On a double-threshold autoregressive heteroscedastic time series model. Journal of Applied Econometrics 11, 253–274 (1996)
60. Liao, T.W.: Clustering of time series data - a survey. Pattern Recognition 38, 1857–1874 (2005)
61. Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, G.: Towards a new evolutionary computation. Advances in estimation of distribution algorithms. Springer, Berlin (2006)
62. Lund, U.: Cluster analysis for directional data. Communications in Statistics – Simulation and Computation 28(4), 1001–1009 (1999)
63. Maulik, U., Bandyopadhyay, S.: Fuzzy Partitioning Using Real Coded Variable Length Genetic Algorithm for Pixel Classification. IEEE Transactions on Geosciences and Remote Sensing 41, 1075–1081 (2003)
64. Minerva, T., Poli, I.: Building ARMA models with genetic algorithms. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 335–342. Springer, Heidelberg (2001)
65. Mitchell, M.: An introduction to genetic algorithms. MIT Press, Cambridge (1996)
66. Mühlenbein, H., Paas, G.: From Recombination of Genes to the Estimation of Distributions I. Binary Parameters, Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, September 22-26, 1996, pp. 178–187 (1996)
67. Ong, C.S., Huang, J.J., Tzeng, G.H.: Model identification of ARIMA family using genetic algorithms. Applied Mathematics and Computation 164, 885–912 (2005)
68. Pasia, J.M., Hermosilla, A.Y., Ombao, H.: A useful tool for statistical estimation: genetic algorithms. Journal of Statistical Computation and Simulation 75, 237–251 (2005)
69. Paterlini, S., Minerva, T.: Evolutionary approaches for cluster analysis. In: Bonarini, A., Masulli, F., Pasi, G. (eds.) Soft Computing Applications, pp. 167–178. Springer, Berlin (2003)

70. Peña, D.: Influential observations in time series. Journal of Business & Economic Statistics 8, 235–241 (1990)
71. Priestley, M.B.: Non-linear and Non-stationary Time Series Analysis. Academic Press, London (1988)
72. Qian, G., Zhao, X.: On time series model selection involving many candidate ARMA models. Computational Statistics & Data Analysis 51, 6180–6196 (2007)
73. Reeves, C.R., Rowe, J.E.: Genetic algorithms - Principles and Perspective: A Guide to GA Theory. Kluwer Academic Publishers, London (2003)
74. Robles, V., Bielza, C., Larrañaga, P., González, S., Ohno-Machado, L.: Optimizing logistic regression coefficients for discrimination and calibration using estimation of distribution algorithms. TOP (2008) (published on line) doi:10.1007/s11750-008-0054-3
75. Roverato, A., Poli, I.: A genetic algorithm for graphical model selection. Journal of the Italian Statistical Society 7, 197–208 (1998)
76. Sabatier, R., Reyne's, C.: Extensions of simple component analysis and simple linear discriminant analysis using genetic algorithms. Computational Statistics & Data Analysis 52, 4779–4789 (2008)
77. Sahni, S., Gonzalez, T.: P-Complete approximation problems. Journal of the Association for Computing Machinery 23, 555–565 (1976)
78. Sessions, D.N., Stevans, L.K.: Investigating omitted variable bias in regression parameter estimation: A genetic algorithm approach. Computational Statistics & Data Analysis 50, 2835–2854 (2006)
79. Tong, H.: Non Linear Time Series: A Dynamical System Approach. Oxford University Press, Oxford (1990)
80. Tsay, R.S., Peña, D., Pankratz, A.E.: Outliers in multivariate time series. Biometrika 87, 789–804 (2000)
81. van Dijk, D., Terasvirta, T., Franses, P.H.: Smooth transition autoregressive models - A survey of recent developments. Econometric Reviews 21, 1–47 (2002)
82. Van Emden, M.H.: An analysis of complexity, vol. 35, Mathematical Centre Tracts, Amsterdam (1971)
83. Vitrano, S., Baragona, R.: The genetic algorithm estimates for the parameters of order p normal distributions. In: Bock, H.-H., Chiodi, M., Mineo, A. (eds.) Advances in Multivariate Data Analysis, Series: Studies in Classification, Data Analysis and Knowledge Organization, pp. 133–143. Springer, Berlin (2004)
84. Wei, W.W.S.: Time Series Analysis. Addison-Wesley, Redwood (1990)
85. Winker, P.: Optimization Heuristics in Econometrics: Application of Threshold Accepting. John Wiley & Sons, Chichester (2001)
86. Winker, P., Gilli, M.: Applications of optimization heuristics to estimation and modelling problems. Computational Statistics & Data Analysis 47, 211–223 (2004)
87. Wu, B., Chang, C.-L.: Using genetic algorithms to parameters (d, r) estimation for threshold autoregressive models. Computational Statistics & Data Analysis 38, 315–330 (2002)
88. Yang, Z., Tian, Z., Yuan, Z.: GSA-based maximum likelihood estimation for threshold vector error correction model. Computational Statistics & Data Analysis 52, 109–120 (2007)
89. Zani, S.: Osservazioni sulle serie storiche multiple e l'analisi dei gruppi. In: Piccolo, D. (ed.) Analisi Moderna delle Serie Storiche, Franco Angeli, Milano, pp. 263–274 (1983)