COMISEF
Computational Optimization Methods in Statistics, Econometrics and Finance

- Marie Curie Research and Training Network funded by the EU Commission through MRTN-CT-2006-034270 -

COMISEF WORKING PAPERS SERIES

WPS-003 23/09/2008

Meta-heuristic Methods for Outliers Detection in Multivariate Time Series

D. Cucina
A. di Salvatore
M. Protopapas

Noname manuscript No.
(will be inserted by the editor)

Meta-heuristic Methods for Outliers Detection in Multivariate Time Series

Domenico Cucina · Antonietta di Salvatore · Mattheos Protopapas

Received: date / Accepted: date

Abstract In this article we use meta-heuristic methods to detect additive outliers in multivariate time series. The implemented algorithms are: simulated annealing, threshold accepting and two different versions of genetic algorithm. All of them use the same objective function, the generalized AIC-like criterion, and in contrast with many of the existing methods, they don't require to specify a vector ARMA model for the data and are able to detect any number of potential outliers simultaneously. We used simulated time series and real data to evaluate and compare the performance of the proposed methods.

Keywords Genetic algorithm · Simulated annealing · Threshold accepting

1 Introduction

It is well known that outliers can arise for different reasons in time series data. They may be due to occasional unexpected events. Outliers are defined as observations which appear to be inconsistent with the remainder of the data set. The detection of outliers is an important problem in time series analysis because it can have adverse effects on model identification, parameter estimation (Chang and Tiao, 1983) and forecasting (Chen and Liu, 1993). The presence of just few items of anomalous data can lead to model misspecification, biased parameter estimation, and poor forecasts. Therefore, it is essential to identify them, estimate their magnitude and correct the time series. Several approaches have been proposed in the literature for handling outliers in univariate time series. Among these methods we can distinguish those based on an explicit model (parametric approach) from those that use non-explicit models (nonparametric approach). For parametric approach, (Fox, 1972) developed a likelihood ratio test for detecting outliers in a pure autoregressive model, (Chang and Tiao, 1983; Chang et al., 1988; Tsay, 1986; 1988) extended this test to an autoregressive integrated moving-average (ARIMA) model and proposed an iterative procedure for detecting multiple outliers. However, these procedures may fail to detect multiple outliers due to masking effects. They can also misspecify "good" data points as outliers, resulting in what is commonly referred to as the swamping or smearing effect. (Chen and Liu, 1993) proposed a modified iterative procedure to reduce masking effects by estimating jointly the model parameters and the magnitudes of outlier effects. This procedure may also fail since it starts with parameter estimation that assumes no outliers in the data. For non-parametric approach, (Ljung, 1989; 1993; Peña, 1990; Gómez et al., 1993; Baragona and Battaglia, 1989; Battaglia and Baragona, 1992) have proposed specific procedures based on the relationship between the additive outliers and the linear interpolator, while (Baragona et al., 2001) used a genetic algorithm.

For multivariate time series, (Tsay et al., 2000) extend well established ARIMA-model-based procedures for univariate time series to the multivariate framework, (Galeano et al., 2006; Baragona and Battaglia, 2007) find the linear combination of a multivariate time series that maximizes one objective function; the first paper

Domenico Cucina · Antonietta di Salvatore · Mattheos Protopapas
Department of Statistics, Probability and Applied Statistics, University of Rome La Sapienza, Piazzale Aldo Moro 5, I-00100 Roma, Italy
E-mail: antonietta.disalvatore@uniroma1.it

[1] Financial support from the EU Commission through MRTN-CT-2006-034270 COMISEF is gratefully acknowledged.

used projection pursuit techniques while the latter employs independent component analysis (ICA); (Barbieri, 1991) used a Bayesian method and finally a graphical method was explored by (Khattree and Naik, 1987).

In this article we propose three meta-heuristic algorithms for identifying additive outliers in univariate and multivariate time series. In particular we implemented the simulated annealing, the threshold accepting and two different versions of genetic algorithm that distinguish themself according to the solution encoding. Using the meta-heuristic algorithms for outlier detection seems attractive because several outliers may be processed simultaneously. The path through different solutions could lead to the exploration of promising regions of the solution space. Further refinement may be accomplished by examining, for instance, the significance levels of the potential outlier estimated magnitudes. Note that almost all available methods for outlier detection are iterative, but there is a crucial difference with respect to the meta-heuristic algorithms. In this latter case, any potential location may change through the iterations. In existing methods, once a location has been selected, it remains fixed in the subsequent iterations. So, the meta-heuristic algorithms seem able to provide more flexibility and adaptation to the outlier detection problem. On the other hand, the computational efficiency is very important for the meta-heuristic algorithms to be useful.

2 Meta-heuristic methods

Many optimization problems do not satisfy the necessary conditions to guarantee the convergence of traditional numerical methods. For instance, in order to apply the least square estimation we need a globally convex likelihood function, however there are a number of relevant cases with non convex likelihood functions or functions with several local optima. Another class of "hard" problems is where the solution space is discrete and wide. These problems are known as combinatorial problems. A simple approach to solve an instance of a combinatorial problem is to list all the feasible solutions of a given problem, evaluate their objective function, and pick the best. However, for combinatorial problem of a reasonable size, the complete enumeration of its elements is not feasible, and most available searching algorithms are likely to yield some local optimum as a result (Rayward-Smith et al., 1996).
Meta-heuristic algorithms are often used to solve such problem instances. Meta-heuristic methods don't rely on a set of strong assumptions about the optimization problem, on the contrary, they are robust to changes in the characteristics of the problem. But, on the other side, they don't produce a deterministic solution but a high quality stochastic approximation to the global optimum.
In this work we are interested in the following methods: simulated annealing, threshold accepting and genetic algorithms. The first two are classified as *local search methods*. Classical local search algorithm move from a initial random solution x^c to an other one, chosen amongst the solutions of the neighborhood of x^c, that have a better value of the objective function. The procedure is iterated until a fixed stopping criterion is satisfied. However, those algorithms may get stuck to local optima. To avoid this problem, the local search algorithms we employ here, sometimes accept worse solutions than the current one. Genetic algorithms have been initially developed by (Holland, 1975) and they are classified as *population based* methods, or *evolutionary algorithms*. They work on a whole set of solutions that is adapted simultaneously by imitating the evolutionary process of species that sexually reproduce.
We give a brief sketch of the three methods.

2.1 Simulated annealing

Simulated annealing (SA) is a random search technique based on an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system. The ideas that form the basis of this method were first published by (Metropolis et al., 1953) in an algorithm to simulate the annealing process. Only thirty years later, (Kirkpatrick et al., 1983) suggested that this type of simulation could be used to search the feasible solutions of an optimization problem, with the objective of converging to an optimal solution which minimizes the objective function.
In analogy with the annealing process, simulated annealing is characterised by the presence of a control parameter T called temperature. One simulation must starts by choosing the initial temperature T_0, the final temperature T_f and a "cooling schedule" whereby the parameter T is decreased. The simulation stops when the temperature T assumes the value T_f. Different cooling schedules are suggested in the literature, in our work we use the geometric schedule :

$$T_t = \alpha T_{t-1} \tag{1}$$

where α is a constant close to 1.
The algorithm proceeds by choosing a random initial solution x^c as the current solution. A new potential solution x^n is drawn randomly in the neighborhood of x^c

and it will always be accepted as the new current solution if it decreases the objective function. Moreover the algorithm accepts also an increase of the objective function but only with a given probability. The acceptance probability is a function of T, in that way it decreases during the course of the simulation to a number close to zero. Every value assigned to T (and affecting the acceptance probability) will be used for SA_{iter} different choices of potential solutions. The pseudo code of the simulated annealing we use here is as follows:

Algorithm 1. Pseudo code for simulated annealing

1. Initialize T_0, T_f e SA_{iter}
2. Generate initial solution x^c
3. $T = T_0$
4. **while** $T > T_f$ **do**
5. **for** $r = 1$ **to** SA_{iter} **do**
6. Compute $x^n \in N(x^c)$ (neighbor to current solution)
7. Compute $\Delta = f(x^n) - f(x^c)$ and generate u from a uniform random variable between 0 and 1
8. **if** $(\Delta < 0)$ or $e^{-\Delta/T} > u$ **then** $x^c = x^n$
9. **end for**
10. T = αT
11. **end while**

2.2 Threshold accepting

The threshold accepting (TA) algorithm is an optimization algorithm which is a modification of the simulated annealing where the sequence of temperatures T is replaced by a sequence of thresholds τ. Statement 8 of algorithm 1 is replaced by:

$$\textbf{if} \quad \Delta < \tau_h \quad \textbf{then} \quad x^c = x^n$$

and in statement 10 we use the threshold $\tau \geq 0$ instead of the temperature T. That is, TA uses a deterministic acceptance criterion instead of the probabilistic one in simulated annealing for accepting worse solutions. During the optimization process the threshold level is gradually lowered like the temperature in simulated annealing.

It was introduced by (Dueck and Scheuer, 1990). They applied the algorithm to a Travelling Salesman Problem and argued that their algorithm is superior to classical simulated annealing. The main advantages of TA are its conceptual simplicity, easy parameterization and its excellent performance for many problem instances. Threshold accepting has been successfully applied to different areas of statistics and econometrics (Winker and Fang, 1997; Fang et al., 2000; Winker, 2000; 2001; Gilli and Winker, 2004). An extensive introduction to TA is given in (Winker, 2001).

2.3 Genetic Algorithm

Genetic algorithms (GA), inspired by (Holland, 1975), imitate the evolution process of biological systems, to optimize a given function. In a GA there is a set of candidate solutions denoted by a population of chromosomes (binary vectors or different encoding) each one having a fitness value that is derived from the success of the chromosome in optimizing the function (i.e. the function value at the corresponding solution). That fitness value determines the probability of a chromosome to be selected as a parent. Children are formed by recombining the genetic material of their two parents (*crossover*) and perhaps after a random alteration of some of the genes (single digits of the chromosome) which is called *mutation* (see Holland, 1975; Goldberg, 1989, for a detailed description). The pseudo-code for the versions of the GA we use here is as follows:

Algorithm 2. Pseudo code for genetic algorithm

1. Set the values of the parameters regarding population size (*pop*), probability of crossover (*pcross*), probability of mutation (*pmut*), number of generations (*gen*) and all the other parameters relevant to this application.
2. Generate initial population of chromosomes representing possible outlier combinations.
3. Calculate the fitness values of these chromosomes.
4. Select two of these chromosomes to become parents, with probability proportional to their fitness.
5. If crossover is used (depending on the probability for crossover mentioned above), combine the genes of these chromosomes using the crossover operator to form two children chromosomes. In the case no crossover is applied, the children chromosomes will be initially, just copies of the parent chromosomes.
6. Then apply the mutation operator to the children chromosomes, so that some (if any) random bits of the children chromosomes are altered.
7. Repeat steps 4-6, until *pop* children chromosomes have been formed.
8. Repeat steps 3-7 until the specified number of generations (*gen*) have passed.

3 Algorithm Features

In this section we describe the features of the algorithms we used for outlier detection.

3.1 Solution Coding

For TA and SA we use the same coding for the solution as suggested in (Baragona et al., 2001). Briefly, a solution x^c is a binary string with length N, where N is the number of observations of the time series. So let

$x^c = (x_1^c, x_2^c, ..., x_N^c)$ where x_i^c takes the value 1 if at the time i there is an outlier and 0 if i is outlier-free. We use this coding for both univariate and multivariate time series, in the latter case we just indicate the times where there are outliers but we don't specify in which components they occur. For the genetic algorithms we use two different forms of chromosome encoding. The first is the same as in the cases of TA and SA, i.e. the solutions are denoted as binary encoded strings. In second the chromosome consists of g integer fields where g is the maximum number of outliers that can be present in the time series. Each field can take all integer values from $1 : N$, that state the position of the outlier in the time series. We also consider the chromosome corresponding to no outliers in the time series. The binary coding implies that the a priori solution space Ω_0 is composed of $\sum_{k=0}^{N} \binom{N}{k}$ distinct elements. However, if the total number of outliers is limited to a constant g, as it seems reasonable, the solution space Ω consists of $\sum_{k=0}^{g} \binom{N}{k}$ distinct elements. We can see that Ω is really large even if g is considerably lower than the length of time series. All algorithms either hardly penalize chromosomes that have more than a maximum number of outliers g, or do not consider these chromosomes at all. TA and SA algorithms are built so that they do not evaluate solutions with more than g outliers.

As concerning the GA algorithms, in the case of integer encoding, it is impossible to access a chromosome that belongs to Ω_0 but does not belong to Ω, and therefore the solution space is actually Ω. In the binary case, chromosomes that do not belong to Ω are severely penalized, and the algorithms tend to avoid these chromosomes. As we have mentioned the binary encoding suffers from the fact that chromosomes with more outliers than the maximum needed are active in the population, increasing the total time required for the execution of the algorithm. The integer encoding scheme suffers from another complication, the possible replications of solutions in the set of chromosomes. That is, if more than one gene in a chromosome holds the same value, we assume they all "point" to the same outlier, i.e. the chromosomes say $[1, 15, 1]$ and $[1, 1, 15]$ denote the same solution, the one with outliers at data positions 1 and 15. The comparison of the efficiency of the two encoding schemes can be studied by the introduction of two indices of efficiency(E_B and E_I). In the binary case, one can define an efficiency measure, as the ratio between the number of admissible chromosomes (chromosomes having no more than the maximum number of outliers) and the total number of chromosomes. So if we denote by N the number of available observations of the time series, and g the maximum number of outliers, then

$$E_B = \frac{\sum_{j=0}^{g} \binom{N}{j}}{2^N}$$

In the case of integer encoded chromosomes, the efficiency measure should be the ratio between the number of chromosomes needed to encode the possible outlier combinations in a "one-to-one" correspondence (which is equal by definition to the number of possible outlier combinations) and the total number of chromosomes, and therefore

$$E_I = \frac{\sum_{j=1}^{g} \binom{N}{j}}{N^g}$$

The relative efficiency of the binary versus the integer encoding scheme can be defined as the ratio of the two

$$R_{B>I} = \frac{E_B}{E_I} = \frac{N^g}{2^N} \cdot \frac{\sum_{j=0}^{g} \binom{N}{j}}{\sum_{j=1}^{g} \binom{N}{j}} \approx \frac{N^g}{2^N}$$

So the binary encoding scheme will be more efficient if $R_{B>I} > 1$ or

$$\log_2 \frac{N^g}{2^N} > 0 \Rightarrow g \log_2 N - N > 0 \Rightarrow g > \frac{N}{\log_2 N}$$

i.e. when the maximum number of outliers is greater than the ratio between the number of observations and the number of bits required to represent that number in the binary system.

3.2 Neighbourhood search in simulated annealing and threshold accepting

Each solution $x^c \in \Omega$ has an associated set of *neighbours*, $N(x^c) \subset \Omega$, called the neighbourhood of ξ where every $x^{c'} \in N(x^c)$ can be reached directly from x^c by an operation called *move*. We consider three different moves: add an outlier; remove an outlier; change the position of an outlier. Our target is to not allow x^c to have more than g outliers, so moves are applied according to the current solution in the following ways: if x^c doesn't contain outliers (i.e., it is a string where every bit takes the vale 0), algorithms can only introduce an outlier; if x^c contains less than g outliers, algorithms can add or remove an outlier with probability of 0.5; if x^c contains already g outliers, algorithms cannot proceed adding an outlier but can only remove or change the position of one of them again with the probability of 0.5.

3.3 TA: Sequence of threshold

The implementation of the TA algorithm requires setting of the following control parameters: the definition of neighborhoods for the choice of x^n, the total number of iterations and the sequence of thresholds τ_h. For the TA implementation we used the neighborhood definition described in section 3.2. (Althöfer and Koschnick, 1991) demonstrated the convergence of the TA algorithm under the hypothesis that an appropriate threshold sequence exists . But in their proof they do not provide a way to construct an appropriate sequence. Consequently, several approaches have been proposed in the literature. In extreme cases of threshold settings, the algorithm behaves as a classical local search algorithm (if all threshold values are set equal to zero) or like a random walk (if all values of the threshold sequence are set to a very large value).

In our implementation we used the method proposed by (Winker and Fang, 1997). The pseudo-code of this procedure is reported below.

Algorithm 3. Pseudocode for generate threshold sequence

1. Initialize N_t, α e M
2. **for** $r = 1$ **to** M **do**
3. Randomly choose solution x_r^c
4. Randomly choose neighbor solution $x_r^c \in N(x_r^c)$
5. Compute $\Delta_r =\mid f(x_r^c) - f(x_r^n) \mid$
6. **end for**
7. Sort $\Delta_1 \leq \Delta_2 \leq ... \leq \Delta_M$
8. Use $\Delta_{N_t}, ..., \Delta_1$ as threshold sequence

The method uses a two step process to construct the threshold sequence. For the first step a large number (M) of possible solutions x^c is generated at random. Then, we compute the distances between values of the objective function at random points x^c and its neighbors x^n, $\Delta_r =\mid f(x_r^c) - f(x_r^n) \mid, r = 1, 2, .., M$. The second step of the construction consists in sorting these values in descending order. So an approximation to the distribution of local relative changes of the objective function is obtained. Only α-lower fraction of sorted sequence is used as the threshold sequence. Otherwise, we can provide percentiles $P_i, i = 1, ..., N_t$ and compute the corresponding quantiles $Q_i, i = 1, ..., N_t$. So, the threshold sequence $\tau_i, i = 1, ..., N_t$ corresponds to $Q_i, i = 1, ..., N_t$. In both cases, the threshold sequence will be monotonically decreasing to zero.

3.4 Objective function

Let $\mathbf{z}_t = [z_{1,t}, ..., z_{s,t}]'$ be a s-dimensional jointly second order stationary real-valued vector time series, with mean zero for each component, covariance matrix $\mathbf{\Gamma}_u$ and inverse covariance matrix $\mathbf{\Gamma i}_u$ for integer lag u.

Suppose that k disturbances $\omega_t = [\omega_{1,t}, ..., \omega_{s,t}]$ happen to be located at time points $t_1, ..., t_k$. Then, given N observations $z_1, ..., z_N$ of time series $\mathbf{z}_t$ the outlier configuration may be described by means of the pattern design matrix $\mathbf{X}$. Let i_j be the number of contaminated component series at time $t_j, j = 1, ..., k$, and let $h = i_1 + ... + i_k$. In our implementation we suppose that all outliers are global, then the integer h that represents the number of the scalar outliers is equal to $h = ks$. For each $t = 1, ..., N$ consider the rows $(t-1)s+r, r = 1, ..., s$ of $\mathbf{X}$. All entries in such rows are zero unless an outlier is present at time t. In this case, if i denotes such scalar outlier, the entries in column i of these rows will be unity. The $Ns \times h$ matrix $\mathbf{X}$ contains all information about the given outlier pattern. In order to evaluate the plausibility of every string the meta-heuristic algorithm uses a objective function that assigns a real number to every solution.

Let $\mathbf{z} = [\mathbf{z}_1', ..., \mathbf{z}_N']'$ be the vector obtained by stacking the s component observations at each time point, and let $\mathbf{\Gamma}$ denote the $Ns \times Ns$ block Toeplitz matrix with $\mathbf{\Gamma}_{i-j}$ as the (i,j)-th block. Assume, at this moment, that both $\mathbf{\Gamma}$ and $\mathbf{X}$ are known. Then, the natural logarithm of the likelihood for $\mathbf{z}$ may be written, by omitting all constant terms,

$$L = -\frac{1}{2}(\mathbf{z} - \mathbf{X}\omega)'\mathbf{\Gamma}^{-1}(\mathbf{z} - \mathbf{X}\omega) \tag{2}$$

Maximization of (2) with respect to ω yields:

$$\hat{\omega} = (\mathbf{X}'\mathbf{\Gamma}^{-1}\mathbf{X})\mathbf{X}'\mathbf{\Gamma}^{-1}\mathbf{z} \tag{3}$$

if we approximate $\mathbf{\Gamma}^{-1}$ with $\mathbf{\Gamma i}$ (Shaman, 1976), where $\mathbf{\Gamma i}$ denotes the $Ns \times Ns$ block Toeplitz matrix with $\mathbf{\Gamma i}_{i-j}$ as the (i,j)-th block; then, the maximum likelihood estimate (3) of ω takes the form:

$$\hat{\omega} = (\mathbf{X}'\mathbf{\Gamma i}\mathbf{X})\mathbf{X}'(\mathbf{I} \otimes \mathbf{\Gamma i}_0)\mathbf{e} \tag{4}$$

where $e = [e'_1, ..., e'_N]$ is the interpolation error (Bhansali and Ippoliti, 2005). Equality (4) establishes in the multivariate framework the relationship between the additive outlier and the linear interpolator. There are different ways to estimate the inverse covariance matrices, but we used the formula suggested by (Battaglia, 1984).

The natural logarithm of the maximized likelihood is obtained, by replacing, in (2), ω with (4) and $\mathbf{\Gamma}^{-1}$ with $\mathbf{\Gamma i}$:

$$L = -\frac{1}{2}\{\mathbf{X}(\mathbf{I}\otimes\mathbf{\Gamma i}_0)\mathbf{e}\}'(\mathbf{X}'\mathbf{\Gamma i X})^{-1}\mathbf{X}'(\mathbf{I}\otimes\mathbf{\Gamma i}_0)\mathbf{e} \qquad (5)$$

where all constant terms are omitted. As L increases with the number of outliers, a generalized Akaike's information criterion (Bhansali, 1986) is used. So, the function which has actually to be minimized is:

$$AIC = -2L + ch \qquad (6)$$

where c is an arbitrary constant and h is the actual number of outliers. The function AIC, however, depends on both the integer h and the matrix $\mathbf{X}$ in (5). The integer parameter h varies from 0 to the maximum prespecified allowed number of scalar outlying observations g.

3.5 Parent selection and other implementation issues in the Genetic Algorithms

In both the binary and the integer cases, we use ordered fitness, i.e. we rank the chromosomes in descending order of AIC (with an additional penalty factor in the binary case) and then assign a fitness value of "1" to the chromosome with the highest AIC value in the population, "2" to the chromosome with the second highest AIC value and so on.

We don't use the "standard" random generated initial populations (Goldberg, 1989). In the algorithms used here, the initial populations consist of chromosomes with just one outlier, different each other (which implies that the size of the population is less than the number of observations). At the beginning, all possible single-outlier chromosomes are generated and sorted in terms of AIC value. Then the initial population is created, consisting of the chromosomes that have the lowest AIC values.

The "roulette wheel" rule is used for parent selection. The probability of a chromosome to be selected as a parent is proportional to its fitness. The crossover operator used is "uniform crossover" (Goldberg, 1989). For any given gene of the first child one of the parents is selected at random (with equal probability) and its corresponding gene is inherited at the same position of the child chromosome. The other parent is used to determine the second child's corresponding gene. Finally, a fixed probability is defined for randomly changing the value of each gene of the child-chromosome (mutation). The entire population of chromosomes is replaced from the offspring created by the crossover and mutation processes at each generation.

In the binary encoding case, where we have only two admissible values for a gene ("0" and "1") the application of the mutation operator is pretty straightforward. In the integer case however, the mutation operator must be adjusted to the fact that the set of possible gene values contains more than two elements. So, in case of a mutation, we have chosen to assign to the gene a uniformly distributed random value, from the set of allowable values $(0 \dots N)$. The two variants do not differ when it comes to the other issues discussed here (crossover, stopping criterion, ecc.). Parents selection and crossover is done as in the binary case, each chromosome has selection probability proportional to its fitness, and the uniform crossover operator is employed for crossover. The same policy is used for the population of the subsequent generations as well. The entire population of a subsequent generation consists of the children chromosomes, i.e. no elitism is used.

4 Examples of application

We applied the proposed methods to three time series: the first is a simulated time series and the last two are real data. Each of them has already been analysed in the literature.

There are two types of parameters we have to determine. One concerning the outlier problem on itself and the other regarding the parameters of the meta-heuristic algorithms. The parameters of the outlier detection problem are three: the constant c in (6), the order of the linear interpolator m, and the maximum number of outliers g. When it comes to the genetic algorithms, choices have to be made regarding the crossover probability ($pcross$), mutation probability ($pmut$), population size (pop) and number of generations -or termination criterion- (gen). For the binary encoded GA we also have to choose the factor for penalizing the objective function in case we have more outliers than the desired number. For the simulated annealing algorithm we have to determine the initial temperature (T_0), final temperature (T_f), number of internal loop iterations at every temperature (SA_{iter}), and the constant α in (1), that characterizing the cooling schedule. Threshold accepting requires two parameters: the number of thresholds (N_t) and the number of internal loop iterations at every threshold (TA_{iter}).

For each time series under consideration, we use c=10 and m=4. Since there are no rules for deciding the values of the parameters of the meta-heuristic methods, we used an empirical process to determine their values. We observed for a number of different parameter sets the evolution of the best solution's value of the objec-

tive function and chose the parameter set that seems to bring the best convergence properties to those values.

Example 1. The first example is the simulated time series used in (Tsay et al., 2000). The trivariate AR(1) model is in the form $x_t = \Phi x_{t-1} + \epsilon_t$ with parameters given by

$$\Phi = \begin{pmatrix} 0.2 & 0.3 & 0.0 \\ -0.6 & 1.1 & 0.0 \\ 0.2 & 0.3 & 0.6 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 1.0 & 0.2 & 0.2 \\ 0.2 & 1.0 & 0.2 \\ 0.2 & 0.2 & 1.0 \end{pmatrix}$$

where Σ is the covariance matrix of $\{\epsilon_t\}$. The eigenvalues of the matrix parameter are (0.5, 0.6, 0.8).
The simulated time series we employ has 200 observations. We have applied our methods to 500 realizations of the model, introducing two outliers at time indices t=100 and t=150, respectively, both having a magnitude of $\omega = (3.5\ 3.5\ 3.5)'$.

We ran the simulated annealing algorithm with T_0= 3.0, T_f= 0.05, $SA_{iter} = 100$, $\alpha = 0.95$, threshold accepting with $N_t = 30$ and $TA_{iter} = 333$, the binary encoded genetic algoritm with $pcross = 1$, $pmut = 0.001$, $pop = 30$, $gen = 300$, penalty factor $pen = 1000$ and the integer encoded GA with $pcross = 1$, $pmut = 0.04$, $pop = 30$, $gen = 300$. The parameter $g = 5$ for all algorithms.

Table 1 Comparison of the algorithms' results' frequencies among meta-heuristic methods and the detection procedure of Tsay, Peña & Pankratz (2000)

Outliers found at	SA	TA	GA_1	GA_2	TPP
t=100, 150	0.832	0.832	0.772	0.668	0.402
t=100	0.006	0.000	0.000	0.000	0.0148
t=150	0.002	0.000	0.000	0.000	0.0164
t=100, 150, ...	0.152	0.152	0.150	0.262	0.000
t=100, ...	0.002	0.000	0.000	0.000	0.000
t=150, ...	0.006	0.000	0.000	0.000	0.000
$\sum$ ...	0.160	0.152	0.150	0.262	0.000
none found	0.000	0.016	0.078	0.070	0.286

In Table 1 we compare the results coming from the four algorithms and the detection procedure of (Tsay et al., 2000)(TPP). In the TPP method we set the empirical quantile of the $J_{max}(A, h_A)$ statistic equal to 33.04 corresponding to a significance level of 0.05% (Tsay et al. (2000), pag 797). The rows show the percentage of the following events happening: outliers found at times 100 and 150, only at 100, only 150, at 100 and 150 as well as others, at 100 and others, at 150 and others, the sum of the events where incorrect positions have been detected and in the last row the event that no outlier was found at all. We can see that SA and TA identify both outlier locations correctly with the higher frequency (83.2% for both). The two GA's algorithms detected them with frequency of 77.2% (GA_1) and 66.8% (GA_2), while frequency of corrected detection for TPP is 40.2%. All four meta-heuristic algorithms find in several runs the outliers at positions 100 and 150 as well as others. We almost always observed that magnitude of the wrong detected outliers is small and its standard error is large. So, a rejection of these wrong detections is highly probable. Assuming that, meta-heuristic methods reach the exact solution in more than 93% of replications. Concerning the TPP method, it does never find wrong solutions but the number of time where none solution is found is significant. We observed that these results are due to a high value of the used empirical quantile of the statistic $J_{max}(A, h_A)$. Indeed using lower values for the $J_{max}(A, h_A)$ the TPP method improves the frequency where both outliers are detected through the frequency of incorrect outliers detections obviously also increases. For example with $J_{max}(A, h_A)$=28.9, corresponding to a significance level of 0.1, the frequency of the corrected detections of both outliers is 64.8%. In Table 2 the estimated magnitude and standard deviation of outliers are reported. The SA and TA provide a better estimation of ω, both the GA's algorithms tend to underestimate the value of magnitude of outliers while the TPP method overestimates ω.

Table 2 Outliers' magnitude and standard deviation estimations

	t=100		
SA	3.4119	3.4424	3.4816
	(0.8444)	(0.7621)	(0.8007)
TA	3.4070	3.4369	3.4791
	(0.8501)	(0.7645)	(0.7997)
GA_1	3.2161	3.2371	3.2555
	(0.8070)	(0.7267)	(0.7847)
GA_2	3.1953	3.2250	3.2534
	(0.8068)	(0.7278)	(0.7946)
TPP	3.7929	3.8859	3.8083
	(0.7656)	(0.6428)	(0.7708)
	t=150		
SA	3.4100	3.4629	3.4814
	(0.8092)	(0.6876)	(0.7814)
TA	3.4071	3.4577	3.4776
	(0.8116)	(0.6888)	(0.7796)
GA_1	3.1702	3.2140	3.2475
	(0.7512)	(0.6480)	(0.7492)
GA_2	3.1688	3.2119	3.2332
	(0.7601)	(0.6479)	(0.7555)
TPP	3.8114	3.8068	3.8018
	(0.7818)	(0.6569)	(0.7888)

Example 2. The second example is the gas–furnace series of (Box et al., 1994). This bivariate time series consists of an input gas rate in cubic feet per minute and the CO_2 concentration in the outlet gas as a percentage, both measured at 9–second time intervals. There are 296 observations. This series, too, has been analyzed by (Tsay et al., 2000) where they found multivariate

Table 3 Meta-heuristic algorithms solutions for the gas–furnace series

Solution	Fitness	Locations
S_1	-86.52	43 55 133 199 235 265
S_2	-86.14	42 55 133 199 235 265
S_3	-81.68	43 55 133 198 235 265
S_4	-86.52	42 55 133 198 235 265

outliers at positions 43, 55, 133, 199, 236, 265, 288 and 287.

When we use 100,000 iterations (T_0= 8.0, T_f= 0.05, SA_{iter} = 200, α = 0.99, *gen*=3000, *pop*=30, N_t=100 and TA_{iter} = 1000) all algorithms converge to the solution with six outliers at positions: 43, 54, 113, 199, 235, 264. The objective function's value is -86.52.

For a more detailed comparison between the algorithms, we did 100 runs of each one of them, but we had to use 10,000 iterations (T_0= 2.5, T_f= 0.05, SA_{iter} = 100, α = 0.96 for simulated annealing, while the values of GA and TA parameters are the same as in the case of artificial time series) due to computational power restrictions. In this case we set g = 15. The results are shown on figure 1. We obtained four different final solutions. The most frequent result for every algorithm is the same fitness function obtained during the experiment with 100,000 iteration (-86.52). The SA and TA algorithms reach this solution with higher frequencies than both GA algorithms. This means that TA and SA require less iterations to reach the optimum, so they explore the solution space more efficiently. Table 3 reports the position of outliers corresponding to the four found final solutions. We can see that the solutions are similar, in particular S_2, S_3 and S_4 differ from S_1 only on the presence of outliers at positions 42 and/or 198 instead of 43 and/or 199. The iterative detection procedure of (Tsay et al., 2000) also considers these positions as possible locations of outliers.

Example 3. Series A data are the concentration readings of a chemical process recorded at two hours interval with a total of 197 readings. Chang et al. (1988) report the identification of an IO at t=64 and of an AO at t=43. Other contributions point out the same location as potential outliers (McCulloch and Tsay, 1994; Luceño, 1998; Baragona et al., 2001). Fig. 2 reports the final results frequencies based on 100 runs for each algorithm. We used the same value of parameters applied for gas–furnace series. The best solution is -444.99 corresponding to the outliers at positions 43 and 64. We can see that SA and TA always found these outliers. GAs algorithms reach that solution with at least a frequency of 0.8. Both found other types of final solution: -444.66 corresponding to outliers at positions 32, 43 and 64 and -443.8 corresponding to outliers at position 43, 64 and 190. Times 190 and 32 correspond to the largest and to the second largest observation.

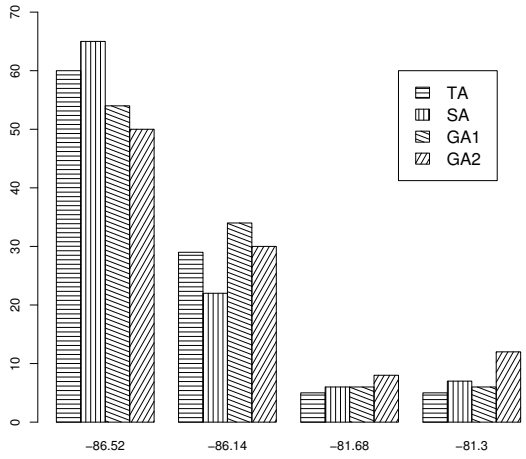

Fig. 1 Best solutions' frequencies of the meta-heuristic methods for gas–furnace series

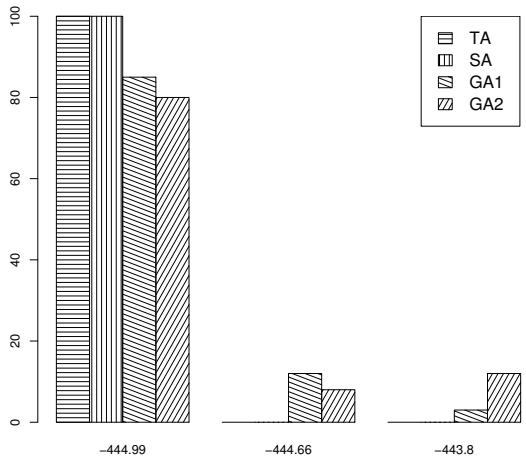

Fig. 2 Best solutions' frequencies of the meta-heuristic methods for series A

5 Conclusions

In this paper three meta-heuristic methods for detecting additive outliers in multivariate time series were proposed. Meta-heuristic algorithms, unlike other proposed methods in literature, do not identify and remove outliers one at a time, but, examine several proposed outlier patterns, where all observations that are possibly outlying ones are simultaneously considered. This feature seems to be effective in handling masking

(meaning that one outlier hides others from being detected) and swamping (when outliers make other clean observations to appear outliers as well) effects caused by multiple outliers. Furthermore, our methods do not require the specification of an adequate multivariate model, which is a difficult task, especially when the data are contaminated by outliers. The procedures were illustrated by analyzing artificial and real data sets. The results obtained from the simulation experiment seem to support the view that the meta-heuristic algorithms constitute a valid approach to detect the time points where potential outliers in vector time series are located. In our experiment the meta-heuristic methods provide better results than the procedure based on the vector ARIMA model for the data. The examination of real time series, the "gas–furnace" data and "series A" of Box and Jenkins yielded satisfactory results. For first time series, the comparison with the results obtained by the detection procedure of (Tsay et al., 2000) showed that, in two cases, both approaches detected the same outliers' locations. This happened in spite of the fact that the two identification procedures are very different. In other cases, discordant detections may be easily explained. In addition, the estimates from both approaches, are very close as far as both the magnitudes and their standard errors are concerned. Meta-heuristic procedures, however, may jointly perform both the outlier detection and the estimation steps, while, in the TPP approach, only the joint outlier estimation may be performed.

For the proposed procedures and TPP method the outlier set often changes if different values of constant c in (6) or quantile of $J_{max}(A, h_A)$ are used. If the critical value $J_{max}(A, h_A)$ (or c) is too large, the percentage of outlier detection will be low. When a smaller critical value is employed, the percentage of outlier detection is increased. However, the frequencies of spurious outliers detections is also increased.

The efficiency of meta-heuristic methods crucially depends on the choice of appropriate values for some control parameters. Several alternatives were tried. The most sensitive GA parameter is the mutation rate. SA and TA algorithms were robust concerning the choice of the initial temperature and the sequence of thresholds respectively. If the initial temperature or threshold are too high, then the algorithms need a large number of iterations.

Acknowledgements Support from the EU Commission through contract Marie Curie Research Training Network COMISEF MRTN-CT-2006-034270 is gratefully acknowledged. The authors would like to thank Prof. Francesco Battaglia and Prof. Roberto Baragona for the numerous conversations on this subject and for their many suggestions during the preparation of the work.

References

I. Althöfer and K.U. Koschnick. On the convergence of threshold accepting. *Applied Mathematics and Optimization*, 24:183–195, 1991.

R. Baragona and F. Battaglia. Outliers detection in multivariate time series by independent component analysis. *Neural Computation*, 19:1962–1984, 2007.

R. Baragona and F. Battaglia. Identificazione e stima di dati anomali in serie temporali per mezzo di interpolatori lineari. Technical Report 19, University of Roma La Sapienza, Italy, 1989.

R. Baragona, F. Battaglia, and C. Calzini. Genetic algorithms for the identification of additive and innovational outliers in time series. *Computational Statistics and Data Analysis*, 2001.

M.M. Barbieri. Outliers in serie temporali multivariate. *Quaderni di Statistica e Econometria*, 13:1–11, 1991.

F. Battaglia. Inverse covariances of a multivariate time series. *Metron*, 42:117–129, 1984.

F. Battaglia and R. Baragona. Linear interpolators and the outlier problem in time series. *Metron*, 50:79–97, 1992.

R. J. Bhansali. A derivation of the information criteria for selecting autoregressive models. *Advances in Applied Probability*, 18:360–387, 1986.

R.J. Bhansali and L. Ippoliti. Inverse correlations for multiple time series and gaussian random fields and measures of their linear determinism. *J.Mathematics and Statistics*, 1:287–299, 2005.

G. E. P. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis. Forecasting and Control (3rd edition)*. Prentice Hall, Englewood Cliffs, New Jersey, 1994.

I. Chang and G. C. Tiao. Estimation of time series parameters in the presence of outliers. Technical Report 8, University of Chicago, Statistics Research Center, 1983.

I. Chang, G. C. Tiao, and C. Chen. Estimation of time series parameters in the presence of outliers. *Technomeirics*, 30:193–204, 1988.

C. Chen and L. Liu. Joint estimation of model parameters and outlier effects in time series. *Journal of the American Statistical Association*, 88:284–297, 1993.

G. Dueck and T. Scheuer. Threshold accepting: A general purpose algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90:161–175, 1990.

K.T. Fang, D. K. J. Lin, P. Winker, and Y. Zhang. Uniform design:theory and application. *Technometrics*, 42:237–248, 2000.

A. J. Fox. Outliers in time series. *Journal of the Royal Statistical Society*, Series B 34:350–63, 1972.

P. Galeano, D. Peña, and R. S. Tsay. Outlier detection in multivariate time series by projection pursuit. *Journal of the American Statistical Association*, 101: 654–669, 2006.

M. Gilli and P. Winker. Applications of optimization heuristics to estimation and modelling problems. *Computational Statistics and Data Analysis*, 47:211–223, 2004.

D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* AddisonWesley, Reading, 1989.

V. Gómez, A. Maravall, and D. Peña. Computing missing values in time series. Working Paper 93-27 Statistics and Econometric Series 21, Departamento de Estadistica y Econometria Universidad Carlos III de Madrid, 1993.

J.M. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and AI.* The University of Michigan, Ann Arbor, MI, 1975.

R. Khattree and D.N. Naik. Detection of outliers in bivariate time series data. *Communications in Statistics - Theory and Methods*, 16(12):3701–3714, 1987.

S. Kirkpatrick, C. D. Gellat, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

G. M. Ljung. A note on the estimation of missing values in time series. *Communications in Statistics - Simulation and Computation*, 18(2):459–465, 1989.

G. M. Ljung. On outlier detection in time series. *Journal of the Royal Statistical Society*, Series B, 55:559–567, 1993.

A. Luceño. Detecting possibly non-consecutive outliers in industrial time series. *Journal of the Royal Statistical Society Series B*, 60:295–310, 1998.

R. E. McCulloch and R. S. Tsay. Bayesian analysis of autoregressive time series via the gibbs sampler. *Journal of Time Series Analysis*, 15:235–250, 1994.

N. Metropolis, A. W. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculation by fast computing machines. *J. of Chem. Phys.*, 21:1087–1091, 1953.

D. Peña. Influential observations in time series. *Journal of Business and Economic Statistics*, 8:235–241, 1990.

V.J. Rayward-Smith, I.H. Osman, C.R. Reeves, and G.D. Smith. *Modern Heuristic Search Methods.* Wiley, Chichester, New york, 1996.

P. Shaman. Approximations for stationary covariance matrices and their inverses with applications to arima models. *Ann. Statist.*, 4:292–301, 1976.

R. S. Tsay. Time series model specification in the presence of outliers. *Journal of the American Statistical Association*, 81:132–141, 1986.

R. S. Tsay. Outliers, level shifts, and variance changes in time series. *Journal of Forecasting*, 7:1–20, 1988.

R.S. Tsay, D. Peña, and A.E. Pankratz. Outliers in multivariate time series. *Biometrika*, 87:789–804, 2000.

P. Winker. Optimized multivariate lag structure selection. *Computational Economics*, 16:87–103, 2000.

P. Winker. *Optimization Heuristics in Econometrics and Statistics: A simple approach for complex problems with Threshold Accepting.* Wiley, New York, 2001.

P. Winker and K.T. Fang. Application of threshold accepting to the evaluation of the discrepancy of a set of points. *SIAM Journal on Numerical Analysis*, 34:2028–2042, 1997.