Scalable structural break detection

Tamas Elteto, Nikolaus Hansen, Cecile Germain-Renaud, Pascal Bondon

Scalable structural break detection

T. Éltető[a,*], N. Hansen[a], C. Germain-Renaud[a], P. Bondon[b]

[a] *Laboratoire de Recherche en Informatique, Bt 490 Universitè Paris-Sud 11, 91405, Orsay, Cedex France.*
[b] *Laboratoire des Signaux et Systèmes, Supelec, CNRS, UMR 8506, 3 rue Joliot-Curie, 91192, Gif-sur-Yvette, Cedex France*

Abstract

This paper deals with a statistical model fitting procedure for non-stationary time series. This procedure selects the parameters of a piecewise autoregressive model using the Minimum Description Length principle. The existing chromosome representation of the piecewise autoregressive model and its corresponding optimisation algorithm are improved. First, we show that our proposed chromosome representation better captures the intrinsic properties of the piecewise autoregressive model. Second, we apply an optimisation algorithm, the Covariance Matrix Adaptation - Evolution Strategy, with which our setup converges faster to the optimal fit. Our proposed method achieves at least one order of magnitude performance improvement compared to the existing solution.

Keywords: Minimum Description Length principle, Covariance Matrix Adaptation - Evolution Strategy

1. Introduction

The so-called data deluge highlights two issues for data analysis: firstly, modelling non-stationary behaviour; second, scalability. This paper addresses a specific case for scaling model selection for structural break detection in time series. More precisely, building on the the AutoPARM procedure [9] for piecewise autoregressive (AR) time series defined by Davis et al., we propose an alternative optimisation technique for the model selection step. The performance improvement of our alternative solution scales the size of the data sets by one to two orders of magnitude at constant computational budget.

In the Minimum Description Length context, whether in the perspective of machine learning or model fitting, the learning of fitting goal encompasses two tasks: the first one is to define an encoding of the data using some hypotheses/models; the second one is enacting an optimisation procedure to explore the associated hypothesis/model space. The optimisation problem for structural break detection is difficult, because of high dimensionality and a complex objective function. Then, evolutionary algorithms are a method of choice. Our approach consists to revisit the optimisation strategy in the light of, first general advances in evolutionary computation, second the opportunities for a separable representation, and finally, an analysis of the limits of the AutoPARM optimisation method proposed in [9].

*Corresponding author

Email addresses: Tamas.Elteto@lri.fr (T. Éltető), Nikolaus.Hansen@lri.fr (N. Hansen), Cecile.Germain@lri.fr (C. Germain-Renaud), Pascal.Bondon@lss.supelec.fr (P. Bondon)

The major contributions of this paper are the following

1. The single-level optimisation problem is decoupled into a bilevel optimisation. The upper level is the problem of finding the optimal number and location of the break points. The lower level optimises the autoregressive models given the number and locations of break points.
2. At the upper level, our optimisation strategy exploits the state-of-the-art CMA-ES (Covariance Matrix Adaptation - Evolutionary Strategy) instead of a relatively straightforward Genetic Algorithm. The associated representation addresses an important shortcoming of [9]: the distance in the chromosomes space better maps to the distance in the model space.
3. The representation becomes scalable. More precisely, it scales linearly with the length of the data set, independently of the cost of the objective function.
4. At the lower level, finding the optimal autoregressive models is solved by an exhaustive search, and a significant amount of computations can be spared using the Yule-Walker method.
5. Regarding the problem of finding the optimal number and location of the break points, an alternative optimisation algorithm, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), and an adapted chromosome representation is proposed. This representation expresses the complexity of a piecewise autoregressive model better in the sense that a small change in the model maps to a small change in the representation.
6. The problem of finding the optimal autoregressive models in the lower level of the decoupled optimisation problem is solved by an exhaustive search, and a significant amount of computations can be spared using the Yule-Walker method.

The content of the paper is organised as follows. Section 2 provides some background information for the piecewise stationary models, the Minimum Description Length (MDL) principle, its application to piecewise autoregressive model fitting, the actual model fitting procedure to be improved and briefly summarises the CMA-ES optimisation algorithm. Section 3 presents the main contributions of this paper. Section 4 provides the details of the numerical experiments comparing the performances of the different model fitting procedures. Section 5 concludes the paper, and the Appendix contains detailed numerical results of the performance comparisons.

2. Background

Stationarity plays a fundamental role in time series analysis: meaningful statistical inference about a stochastic process requires that its joint probability distribution does not change when shifted in time [6].

In many cases however, real-life observations are not compatible with the stationarity assumption. Trends and periodicity are not an issue, for that they can be eliminated through simple data transformations. A considerably more challenging issue is structural non-stationarity, where the time series is represented by a parametric model in which the parameters change values at unknown times called break points. In a nutshell, structural break analysis is required when the hypothesis does not stand that, at least at some relevant scale, the underlying source that creates the observed time-series either does not change at all, or evolves along a slow drift.

Some important applications of structural break analysis can be highlighted as follows.

1. The objective is to explain the structural properties of the time series. For example, whether or not the system responds to a known/suspected change in the environment. In this case, the location of the change is very important.

2. The correlation or spectral structure of the stationary segments are analysed using classical methods. Then, better quality conclusions can be drawn compared to the case when the structural breaks are neglected. Furthermore, the change of these conclusions from one segment to the other can also be studied.
3. The fitted model is used for generating "artificial" time series and the random series can be used as an input for a system study, a simulator or any other further analysis.

Objective 1 is demonstrated by an example in [9] with a time series analysis focusing on the location of a change. The goal of the analysis is whether or not the introduction of the seat-belt legislation has an effect on the monthly deaths and serious injuries on UK roads. Another example in [9, 17] demonstrating the significance of the search for locations of structural changes is the analysis of speech signals, which are clearly non-stationary as a whole but can be broken into approximately stationary intervals corresponding to the phonemes.

The analysis of structural breaks is the main subject of [22], where the goal is to detect malicious use of computer resources by looking for changes in the structure of consecutive best fitting models over a nontstationary time series. [22] refers to its approach as Dynamic Model Selection and uses an MDL coding scheme to find the best break point locations.

Objective 1 is also important in a structural break analysis of log returns of closing stock prices for 12 companies between 1993 and 2009, that is presented in [1]. According to [1], the detected changes can be readily associated with major historical events like the Asian financial crisis, the Russian financial crisis, September 11 attacks or the beginning of the recent financial crisis.

Objective 2 is demonstrated in [16] by the analysis of the spectral structure of electroencephalograms (EEGs) recorded during an epileptic seizure. Since the series cannot be regarded to be stationary, its Fourier transform cannot reveal useful information on the power conditions in various frequencies, consequently one should look for stationary segments in the time series whose spectral properties can be properly analysed.

Structural break detection related to Objective 3 is presented in [12, 11] where Grid workload measurements are analysed for structural breaks. The goal is to establish a robust workload generator model that can be used in Grid architecture studies. Besides fitting a structural break model to the workload measurements, they also present empirical results on the robustness of the estimated parameters.

The rich literature on the approach of non-stationarity by structural break models (besides the above references, other recent results can be found for example in [3, 4, 5, 7]) underlines the importance of this subject. Once such a model is established, it significantly improves the quality of the further analysis. However, to carry out the parameter fitting, one should face of costly and time consuming (e.g. in the order of 1-3 hours as our examples show) calculations as the structural break models (particularly when only a few prior assumptions are used) usually have very rich parameter space. The exploration of the parameter space is of course essential from the sake of practical usefulness of these models. Improvements of their parameter fitting methods can greatly extend the scope of the models and it is therefore an important task on its own right.

2.1. Piecewise AR models

The piecewise AR model, formalised in (1), describes a finite length discrete-time locally stationary time series as segments of stationary time series that are concatenated. The stationary time series of each segment is assumed to be an AR process. The edges of the segments are referred to as break points. An important argument for focusing on piecewise AR models is that they are

dense in the class of locally stationary processes with regular time varying spectrum and because efficient algorithms exist for fitting an AR model to a stationary process.

More precisely, let n be the length of the time series, m the number of break points. For all t such that $1 \leq t \leq n$,

$$Y_t = X_{t,j},\ \tau_{j-1} \leq t < \tau_j,$$

$$X_{t,j} = \gamma_j + \phi_{j,1} X_{t-1,j} + \ldots + \phi_{j,p_j} X_{t-p_j,j} + \sigma_j \epsilon_{t,j}, \tag{1}$$

where τ_j, $j = 1, \ldots, m$ denotes the break point between the j^{th} and $(j+1)^{\text{th}}$ AR process, $\tau_0 = 1$, $\tau_{m+1} = n$, $(\gamma_j, \phi_{j,1}, \ldots, \phi_{j,p_j}, \sigma_j^2)$ is the parameter vector corresponding to a causal autoregressive process with order p_j, and $\epsilon_{t,j}$ are independent, identically distributed random variables with mean 0 and variance 1.

The time series is explained as a set of consecutive segments, where an autoregressive model describes the evolution of the time series within each segment. Section 2.3 will describe corresponding model selection task in the framework of Minimum Description Length optimisation.

2.2. The MDL principle

The principal motivation of inductive inference is to search for regularity in the data. From a machine learning perspective, what is needed is a decision rule – connected to a set of hypotheses – that uses this regularity. Given a training data, a hypothesis is then selected such that the decision rule generalises to some test data the best with respect to a loss function. Among the many kinds of loss functions, there is one, the so called log-loss, whose "optimisation" leads to a decision rule that best compresses the test data.

In a nutshell, the Minimum Description Length (MDL) principle [14] also seeks for regularity in the data where regularity is interpreted directly as ability to compress. The hope is that, once such compression scenario is found by selecting a proper hypothesis in the class, the inferred regularity generalises or predicts well. Furthermore, if the selected hypothesis is happen to be the "real" data source then the MDL principle leads to consistent model fitting methods.

[14] categorizes data compression data w.r.t. a model in four main types of universal codes. We defer the discussion of the relevance of these categories to the problem at hand to Section 2.4.

1. **NML (Normalised Maximum-Likelihood) code** The code length is described as the sum of
   (a) the negative log-likelihood of the maximum likelihood estimate in a model class given the observation and
   (b) the complexity of the model class (when finite).
2. **Bayes code.** The code length is the weighted sum of log-likelihood of the data using the parametric distributions of the models in the model class $\mathcal{M}$, where the weight is a prior distribution over the models in the class.
3. **Two-part code.** Given an observation $\boldsymbol{y}$
   - Choose a parametric distribution $\hat{\mathcal{F}}$ from a model class $\mathcal{M}$ of distributions and encode $\hat{\mathcal{F}}$.
   - Encode $\boldsymbol{y}$ using $\hat{\mathcal{F}}$.
   - Select the model $\hat{\mathcal{F}}$ that gives the shortest total description length which is the sum of the description length of the model and the description length of $\boldsymbol{y}$.

4. **Sequential predictive.** [14] names it prequential: the encoding is done sequentially. After N observation steps, the maximum-likelihood model is selected for calculating the negative log-likelihood for the $N + 1^{\text{st}}$ observation.

We now turn to some consideration related to Information Theory, which will be useful in the next sections for motivating our choices of both a two-part code, and CMA-ES as the optimisation procedure.

First, the relationship between code lengths on one hand, and likelihood maximization on the other hand, is rooted in the standard interpretation of complete prefix codes as probability distributions. For the sake of completeness, we recall Kraft's inequality: given an alphabet $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ and the corresponding code lengths $l_1, l_2, \ldots, l_n$ of a uniquely decodable binary code (that is the number of bits needed to encode s_k is l_k), $\sum_{k=1}^{n} 2^{-l_k} \leq 1$. When equality holds, the code is called complete, and a natural distribution over $\mathcal{S}$ gives probability 2^{-l_k} to symbol s_k.

Conversely, the Shannon-Fano code corresponding to this distribution has code lengths $l_1, l_2, \ldots, l_n$. Therefore the description length function can be regarded as the negative log-likelihood of this probability source given the observation for which the code length is calculated.

The second important idea derives from the well-known fact that, under certain regularity conditions, the Hessian matrix of the log-likelihood function at the optimum (that is, at the maximum likelihood estimate) is a consistent estimate of the Fisher information matrix. This way, the observed information approximates the asymptotic covariance matrix of the maximum likelihood estimate, see for instance [10].

The maximum likelihood estimation motivates the idea that the optimisation of the fitness functions considered in certain MDL contexts can also be used for assessing the reliability of the model selection. In the case of two-part codes with continuous parametrisation, the parameter vector minimising the negative log-likelihood of the model is per se a Maximum-Likelihood estimate, thus its Observed Information can be used for constructing confidence intervals for the parameter estimates. As a matter of fact, Section 14.4 of [14] has a relevant note, that is, it is important to examine the code length estimates in the neighbourhood of the selected model. For continuous parametrisation this is explained by the Hessian matrix at the optimum, because this provides valuable information on the quality of the model selection.

The considerations so far can be summarized as follows:

- The MDL principle formulates the model (or model class) selection task as an optimisation problem of a fitness function representing a code length.
- If properly normalised, the code length function can be considered as a log-likelihood function of a probabilistic model.
- The neighbourhood of the optimum is also important with respect to the quality of the model selection. In particular, the Hessian matrix of the fitness function at the optimum is useful and we prefer optimisation algorithms that can provide an estimate for it.

2.3. The AutoPARM representation of piecewise AR process

[9] proposes the following code length function, which will be called AutoPARM-code in the following.

$$CL_{\mathcal{F}}(\boldsymbol{y}) = CL_{\mathcal{F}}(\hat{\mathcal{F}}) + CL_{\mathcal{F}}(\hat{\mathbf{e}}|\hat{\mathcal{F}}), \tag{2}$$

where $CL_{\mathcal{F}}(\hat{\mathcal{F}})$ denotes the code length of the fitted model $\hat{\mathcal{F}}$ and $CL_{\mathcal{F}}(\hat{\mathbf{e}}|\hat{\mathcal{F}})$ denotes the code length of the time series using $\hat{\mathcal{F}}$. In other words according to [9], $CL_{\mathcal{F}}(\hat{\mathbf{e}}|\hat{\mathcal{F}})$ is what is not explained by the fitted model: the residuals of the AR process when the underlying piecewise AR model is $\hat{\mathcal{F}}$.

More precisely:

$$CL_{\mathcal{F}}(\hat{\mathcal{F}}) = \log_2 m + (m+1)\log_2 n + \sum_{j=1}^{m+1} \log_2 p_j + \sum_{j=1}^{m+1} \frac{p_j+2}{2} \log_2 n_j, \tag{3}$$

where $\log_2 m$ and $\log_2 n$ are the length for encoding m and a segment size, respectively, $\log_2 p_j$ refers to the encoding of the AR order p_j, $n_j = \tau_j - \tau_{j-1}$ is the length of segment j, and $\frac{p_j+2}{2}\log_2 n_j$ corresponds to the encoding of the AR parameters.

For the unexplained part,

$$CL_{\mathcal{F}}(\hat{\mathbf{e}}|\hat{\mathcal{F}}) \approx \sum_{j=1}^{m+1} \frac{n_j}{2} \log(2\pi\hat{\sigma}_j^2), \tag{4}$$

where $\hat{\sigma}_j^2$ is the Yule-Walker estimate (see [6] for details on this estimate) of the variance σ_j^2 of the j^{th} segment.

[9] shows that, when the number of break points is known, the relative break point locations selected by the two-part AutoPARM-code converge almost surely to their true values.

Function (3) encodes the descriptions of the length of the segments by $(m+1)\log_2 n$. This encoding implicitly bears the assumption that the length of the segments are independent and identically distributed, because the common log-likelihood (the code length) of lengths of the segments is the sum of the individual log-likelihoods.

We note, for further use, that the code length estimation of the time series can be decomposed into a sum of $m+1$ terms in (2) with each term referring to a segment. Consequently, if the code length estimate (2) is viewed as an objective function with AR orders as parameters, then the minimisation problem of this function is additively separable, that is, each segment only contributes summands that are independent of the others. More details on this observation are elaborated in Section 3.1.

2.4. Discussion

As said before, model selection covers both representation and optimisation. The resulting palette of possible methods is thus large, amounting to combining two relatively independent research directions. [14] shows both theoretically and in practice that the accuracy of the parameter selection method highly depends on the quality of the compression method, i.e. the calculation of the code length. This section will briefly discuss alternative representations, both in the perspective of the four universal coding methodologies presented in 2.2, and within AutoPARM-code. However, exploring these code alternatives goes beyond the scope of the present paper, which focuses on the performance of the alternative optimisation algorithm described in Section 3.

Considering the coding methodology, AutoPARM-code is a two-part coding scheme and [9] contains a proof of consistency of this approach. We note that if one is willing to choose among alternative encodins (to be detailed below) the same consistency property have to be proved as well.

[13] presents Normalised Maximum-Likelihood codes for the AR and ARMA model classes, and give empirical evidence of high efficiency for model selection. However, the calculation of the

complexity of the model class involves Monte-Carlo integration, which is not a good candidate for scalability. Moreover, we found it difficult to estimate the values above AR order 6, an observation already presented in [13]. There is an alternative encoding, which is to be based on the Conditional Normalised Maximum-Likelihood code of [19] for generalised Gaussian regression families for AR models. Finally, [18] presents an ARMA model selection method based on a predictive MDL code. The model parameters have to be fitted step-by-step, therefore this method would be too slow for our scenarios with many long AR segments.

Focusing on the AutoPARM-code as described by (3) and (4), some improvements, that would probably improve the rate of convergence in the asymptotics, might be as follows.

First, the number of break points is encoded using $\log_2 m$ bits. However, the prefix encoding of an arbitrary integer needs more bits, for example, $2\log_2 m$ is a better estimation. The same applies to the encoding of the AR order p_j. Second, the encoding of the break point locations by $(m+1)\log_2 n$ bits is not the shortest one. Assuming that the length of all segments is greater than 1, the break point locations could be encoded using $\log_2 \binom{n}{m}$ bits.

Last, but not least, according to (3), the number of bits to encode the model parameters in the j^{th} segment is $(p_j+2)\frac{1}{2}\log_2 n_j$.

Theorem 10.1 in [14] states that under certain conditions (e.g. on the parameter set of an exponential family with k parameters), the regret (the amount of needed extra code length compared to the shortest maximum-likelihood code) of a two-part code has a particular form, which is asymptitoctically $\frac{k}{2}\log\frac{n}{2\pi}$ if n, the length of the time series, goes to infinity. The extension of this Theorem to the case of AR models would theoretically justify the above outlined approach, but to the best of our knowledge, this has not yet been done in the literature.

2.5. AutoPARM-GA

AutoPARM carries out the model fitting task by a genetic algorithm, which will be named AutoPARM-GA in the rest of the paper. Here, we briefly discuss its chromosome representation and some other details of the optimisation procedure.

Definition 1. *A chromosome is a vector of integers $\delta = (\delta_1, \ldots, \delta_n)$ of length n where δ_t is defined as*

$$\delta_t = \begin{cases} -1, & \text{if no break point at } t, \\ p_j, & \text{if there is a break point at } t \text{ and the AR order for the } j^{th} \text{ segment is } p_j. \end{cases}$$

AutoPARM-GA follows the island strategy [21]: rather than running the search in the population as a whole, it considers N islands with a fixed number of chromosomes in each island. After generating the offspring, the new and old chromosomes are compared and the best chromosomes are kept independently in each island. After a certain number of generations, migrations occur among the islands. Together with the migrations, the best chromosome (representing the model with the shortest description length among all chromosomes seen so far) is updated. The stopping criterion is that the best chromosome does not change after a certain number of consecutive migrations, or the number of migrations reaches a limit. In the following, this particular kind of genetic algorithm will be termed NI.

A first limitation of the representations of Definition 1 is the associated computational complexity. The crossover and mutation operations step through the whole chromosome (or a pair of chromosomes). Since the length of a chromosome is equal to the length of the dataset to be segmented, and the crossover and mutation operations are considering the elements of the chromosome

vector almost one-by-one, the number of computation steps in these operations are proportional to the length of the dataset. Thus, the number of computation steps per objective function evaluation during the optimisation scales linearly with the length of the dataset even when the cost of the objective function evaluation and the total number of function evaluations are disregarded. This is a limitation when the algorithm is going to be applied to a large series. The island model offers opportunity for speedup through parallelisation. The speedup factor would be related to the number of islands or chromosomes depending on the level of parallelisation.

A more fundamental, limitation of AutoPARM-GA is related to the chosen combination of encoding and the operators in the algorithm. Particularly for long samples, the exact location of a break point is not at all critical: a small shift of the break point location implies only a small change in the overall piecewise AR model. For instance, in the experimental study [11], bootstrapped samples from segmented large time series showed some small variability in break point locations, but consistent behaviour.

The issue with the representation of Definition 1 is that the change of the location of a break point involves (at least) two operations on a chromosome. Assume, for example, that there is a break point at t

$$\delta_t = p \text{ and } \delta_{t+1} = -1.$$

Shifting this break point from t to $t+1$ involves two independent operations: deleting the current break point, i.e., $\delta_t = -1$; and the new break point should be established, i.e., $\delta_{t+1} = p$. It can be expected, that if the chromosome representation is modified, such that the change of break point location can be performed in one simple operation, then the efficiency of the optimisation should improve.

2.6. The CMA-ES optimisation algorithm and the probabilistic interpretation of its output

The goal of CMA-ES [15] is to minimise an objective function $f : \mathbb{R}^n \to \mathbb{R}$. The only accessible information on f are function values at evaluated search points. CMA-ES performs a stochastic search in the parameter space by adapting the direction and step size using inferred dependencies between the parameters. These dependencies between all parameter pairs are learned by updating a covariance matrix of a multivariate normal search distribution.

After generating the next offspring of λ samples in the parameter space, the best[1] μ of them are selected and are used to update the mean and the covariance matrix of the sampling distribution. The updated mean is a weighted average of the selected samples. The covariance matrix is updated by averaging the old covariance matrix estimate, the covariance matrix estimate coming from the evolution path history (rank-one update) and the covariance matrix estimate from the selected samples (rank-μ update).

When the CMA-ES algorithm has converged, the mean of the last update of the multivariate normal distribution is as close to the optimum as possible depending on the stopping rule, while the covariance matrix approximates, in case of a smooth objective function, the second order smoothness properties of the objective function around the optimum. If a negative log-likelihood function of a parametric probability distribution satisfying certain regularity requirements is optimised such that the maximum likelihood estimation is consistent and asymptotically normal, then the covariance matrix estimate of CMA-ES will also be an estimate of the observed information (that is the Hessian

[1]The samples are ordered according to the corresponding objective function values.

matrix of the negative log-likelihood function, see [10]) apart from a constant term. Thus, CMA-ES satisfies the 3$^{\text{rd}}$ requirement on the list of considerations ath the end of Section 2.2.

3. Bilevel optimisation and the role of CMA-ES

This section introduces two alternative optimisation methods for fitting a piecewise AR model. The first one is the bilevel method presented in Section 3.1. Empirical evidence (see Section 4.2) shows that it has superior performance over AutoPARM-GA, which is a single-level method using the NI optimisation algorithm. The second one comes from our observation that its chromosome representation has a significant limitation in that multiple operations are needed for exploring models in the parameter space that are otherwise close to each other. Section 3.2 introduces the application of CMA-ES and the corresponding chromosome representation that do not have this limitation. The additional performance gain will be presented in section 4.3.

3.1. The bilevel method

In AutoPARM-GA, the break point locations and the AR orders are optimised at the same time. In the following we shall refer to this as the single-level method. The search space here is rather large, and consequently it can be costly to find the optimal solution. We saw in Section 2.1 that the optimisation problem is additively decomposable. Because of this, the problem can be formulated as a special bilevel optimisation problem (a recent overview on bilevel optimisation is given in [8]).

We first present the decomposition idea as follows. The objective function is the code length function of (2). We make the parameters in $CL_{\mathcal{F}}(\boldsymbol{y})$ explicit by

$$CL_{\mathcal{F}}(\boldsymbol{y}) = f(m, \kappa, p),$$

where m is the number of break points, κ denotes the vector of relative break point locations, $\kappa = (\kappa_1, \ldots, \kappa_m)$ where $\kappa_j = \tau_j/n$, and $p \in \{0, 1, \ldots, p_{\max}\}^{m+1}$ denotes the vector of AR orders of the segments.

Then, the optimisation problem can be formalised as

$$\min_{m,\kappa,p} f(m, \kappa, p).$$

In the bilevel setup, we formulate this optimisation problem as

$$\min_{m,\kappa}\{\min_{p} f(m, \kappa, p)\}. \tag{5}$$

The graphical interpretation of the proposed optimisation method is shown in Figure 1.

3.1.1. Upper level: localising the break points

As it was indicated at the beginning of the section, the upper level optimisation is the selection of the number and locations of the break points. We address this problem in two different ways. First, we used the same NI algorithm as in AutoPARM-GA, described in 2.5, in order to investigate the contribution of the bilevel setting in the original scenario. We keep the problem encoding in the upper level essentially unchanged: each locus in the chromosome δ_t, for $t = 1, \ldots, n$, determines whether a break point is realized, which is the case if $\delta_t \neq -1$. The AR order is provided by the lower level.

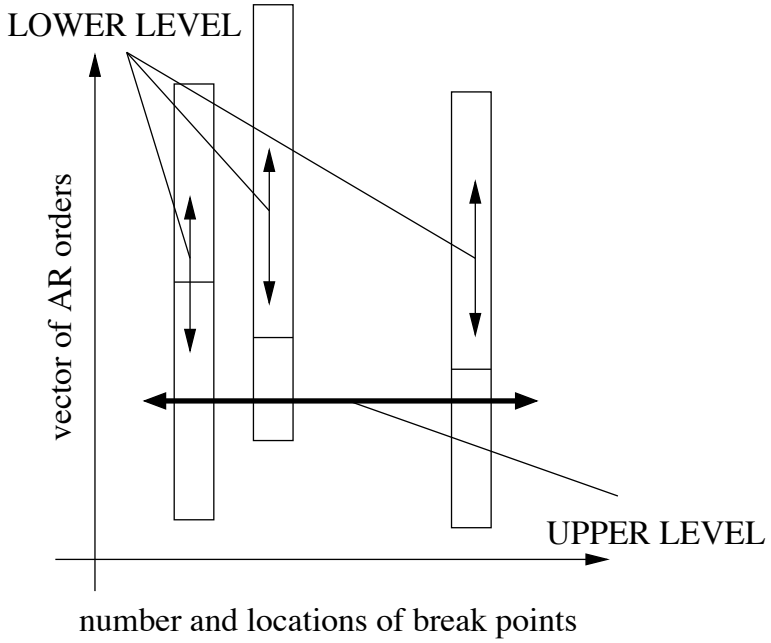

Figure 1: Graphical interpretation of the bilevel optimisation method.

Second, we encode each break point location as an integer. The number of integers defines the number of break points and also the dimension of a search problem with a fixed number of break points. Changing the number of break points means to change the search space dimension of this search problem. In this approach, we repeatedly choose a break point number and run CMA-ES, as explained in Section 3.2.2.

This problem can be solved using a generic optimisation method. In this paper, we used the same NI algorithm as in AutoPARM-GA, described in 2.5, in order to highlight the specific contribution of the bilevel setting.

In any case, the important information in a chromosome of the NI algorithm is whether the elements δ_t are -1 or not for $t = 1, \ldots, n$, where n is the length of the chromosome. The AR order, which is important in the crossover and mutation operations as it might impose constraints on the length of the segment, is not randomly chosen but it is provided by the lower level.

3.1.2. Lower level: determining the AR models within the segments

When a particular number and location of break points are selected, then a lower level optimisation is performed. The objective function detailed in (3) and (4) can be minimised segment by segment. Let

$$f^*(\kappa_j, \kappa_{j+1}, p_j) = \log_2 n + \log_2 p_j + \frac{p_j + 2}{2} \log_2 n_j + \frac{n_j}{2} \log(2\pi\hat{\sigma}_j^2), \tag{6}$$

we have

$$f(m, \kappa, p) = \log_2 m + \sum_{j=0}^{m} f^*(\kappa_j, \kappa_{j+1}, p_{j+1}),$$

and then,

$$\min_{p} f(m, \kappa, p) = \log_2 m + \min_{p_1} f^*(\kappa_0, \kappa_1, p_1) + \ldots + \min_{p_{m+1}} f^*(\kappa_m, \kappa_{m+1}, p_{m+1}). \tag{7}$$

Focusing on segment j, the code length estimate for each order p_j, $0 \leq p_j \leq p_{\max}$ is calculated. The order giving the shortest code length estimate for segment j is selected and the lower level optimisation procedure steps over to segment $j+1$. The constraint on the length of the segment (in order to avoid high order AR parameter fitting on short segments) is taken into consideration by limiting $p_{\max}$.

The most expensive part of the code length estimate for segment j in terms of computational complexity is the last summand in (6). In particular, the calculation of the variance estimate $\hat{\sigma}_j^2$ is the most demanding. The variance estimates are calculated by the Durbin-Levinson algorithm which fit AR models of successively increasing orders $1, 2, \ldots, p_{\max}$ to the data, see [6] Section 8.2. The complexity of this algorithm is $\mathrm{O}(p_{\max}^2)$.

3.2. Piecewise AR model fitting using CMA-ES

This Section shows how CMA-ES can be incorporated into the piecewise AR model fitting framework. There are two important issues that need further considerations: the varying number of parameters and the discrete versus continuous parameter space.

3.2.1. The problem formulation for CMA-ES

The following alternative chromosome representation, that contains the same information as Definition 1, is established for the optimisation procedure using CMA-ES.

Definition 2. *A chromosome is an ordered pair of an integer and a vector of ordered real numbers in* $(0, 1)$*:*

$$\delta = (m, \kappa), \quad \kappa \in (0, 1)^m, \quad m \in \{0, \ldots, m_{\max}\},$$

where $m_{\max}$ *is the maximum number of break points allowed. If* $m = 0$*, the chromosome refers to an AR model without break point.*

The chromosome representation of Definition 2 is an ordered pair whose first component is the number m of break points. The dimension of the vector in the second component of the chromosome is m. Therefore, the objective function, that calculates the code length estimate for an arbitrary chromosome, has varying number of input parameters depending on the number of break points. However, CMA-ES is designed for objective functions with real vector arguments of a fixed dimension. Consequently, the optimisation procedure using CMA-ES should be carried out with a fixed number of break points. That is, several optimisation steps using different number of break points should be done.

The relative locations of the break points are discrete because of the discrete-time AR model, but CMA-ES is a continuous-space optimisation method therefore the real parameters are rounded for the evaluation in the objective function. In other words, the continuous-space objective function is piecewise constant.

3.2.2. The optimisation procedure

This Section gives a high level description of the optimisation procedure; Table 1 gives the pseudo-code. Denote by $\overline{f} : \mathcal{N} \to \mathcal{R}^+$ the following function:

$$\overline{f}(m) = \min_{\kappa \in (0,1)^m} \min_{p \in \{0,\ldots,p_{\max}\}^{m+1}} f(m, \kappa, p).$$

The value of $\overline{f}(m)$ is obtained by fitting piecewise autoregressive models with m break points to the time series. The model fit uses CMA-ES to find the optimal location of the break points by searching in the parameter space $(\kappa_1, \ldots, \kappa_m)$, where the constraint $0 < \kappa_1 < \ldots < \kappa_m < 1$ is enforced by penalisation.

The basic idea of the search algorithm is that starting from $m_{\text{pointer}} = 2$, we always test three cases with $m_{\text{pointer}} - 1$, m_{pointer} and $m_{\text{pointer}} + 1$ number of break points in one loop. When the estimated code length shows a decreasing trend then m_{pointer} is increased. E.g. if

$$\overline{f}(1) > \overline{f}(2) > \overline{f}(3),$$

then the number of analysed break points is doubled, $m^{\text{next}}_{\text{pointer}} = 2 \cdot 2 = 4$, and the new $\overline{f}(\cdot)$ are tested in the new loop. (That is, we test the neighbourhood of 2, 4, 8, 16, ...)

Oppositely, when the estimated code length values show an increasing trend then m_{pointer} is reduced. E.g. if we found decreasing trend for 8, but increasing for 16, that is

$$\overline{f}(7) > \overline{f}(8) > \overline{f}(9),$$

and

$$\overline{f}(15) < \overline{f}(16) < \overline{f}(17),$$

then we set the pointer to the middle of the pointers already visited, that is, $m^{\text{next}}_{\text{pointer}} = \frac{1}{2}(8+16) = 12$ and go on with the next loop.

The loops are finished when $\overline{f}(m_{\text{pointer}})$ is not larger than the neighbouring $\overline{f}(\cdot)$ values:

$$\overline{f}(m_{\text{pointer}} - 1) \geq \overline{f}(m_{\text{pointer}}) \leq \overline{f}(m_{\text{pointer}} + 1).$$

The number of break points is selected to be the one which gives the smallest code length estimation L_m among all that were tested.

Some technical issues should be discussed here. Firstly, the rounding inside $f(m, \kappa, p)$ may divert CMA-ES for short time series. However, in the range of 1k – 2k or above, this effect is not apparent. Next, given $\overline{f}(.)$ is unimodal (and noiseless), we can easily prove that the algorithm will find the global optimum for the number of break points fast. Otherwise, only a local optimum is guaranteed to be found. In fitting piecewise AR models to empirical time series, unimodality seems to be a prerequisite for the overall goal to be reasonable: the underlying hypothesis is that there is some physical process driving the ruptures. For instance, our experiments in bootstrapping segmented time series [12] exhibited correlation between times series corresponding to the activity of distributed agents, corresponding to a common driving mechanism. Finally,we note, that in order to avoid models with generally too short segments, an upper limit on the possible number of break points was set to $\frac{1}{100}$ of the length of the series.

Table 1: Pseudo-code of the search for the optimal number of break points.

1.	$m_{\text{pointer}} = 2$, $m_{\text{upper limit}} = \frac{\text{series length}}{100}$, $m_{\text{low}} = 1$, $m_{\text{high}} = m_{\text{upper limit}}$.
2.	$CL_{-1} = \overline{f}(m_{\text{pointer}} - 1)$, $CL_0 = \overline{f}(m_{\text{pointer}})$, $CL_1 = \overline{f}(m_{\text{pointer}} + 1)$
3.	Store the $\overline{f}(\cdot)$ values together with the # of break points in Table A.
4.	If not$\big((CL_{-1} \leq CL_0 \geq CL_1)$ or $(CL_{-1} \leq CL_0 \leq CL_1)\big)$ then
4.1	$m_{\text{low}} = m_{\text{pointer}}$.
4.2	If $CL_{-1} > CL_0 > CL_1$ then $m_{\text{pointer}} = 2m_{\text{pointer}}$.
4.3	While not$(CL_{-1} \geq CL_0 \leq CL_1)$ and $(m_{\text{low}} < m_{\text{high}})$
4.3.1	$CL_{-1} = \overline{f}(n_{\text{pointer}} - 1)$, $CL_0 = \overline{f}(m_{\text{pointer}})$, $CL_1 = \overline{f}(m_{\text{pointer}} + 1)$
4.3.2	Store the $\overline{f}(\cdot)$ values together with the # of break points in Table A.
4.3.3	If $CL_{-1} \geq CL_0 \geq CL_1$ then
4.3.3.1	If $m_{\text{high}} = m_{\text{upper limit}}$ then
4.3.3.1.1	$m_{\text{low}} = m_{\text{pointer}}$
4.3.3.1.2	$m_{\text{pointer}} = \min\{2m_{\text{pointer}}, m_{\text{upper limit}}\}$
4.3.3.2	else
4.3.3.2.1	$m_{\text{low}} = m_{\text{pointer}}$
4.3.3.2.2	$m_{\text{pointer}} = \left[\frac{m_{\text{pointer}} + m_{\text{high}}}{2}\right]$
4.3.4	else if $\big((CL_{-1} \leq CL_0 \geq CL_1)$ or $(CL_{-1} \leq CL_0 \leq CL_1)\big)$
4.3.4.1	$m_{\text{high}} = m_{\text{pointer}}$
4.3.4.2	$m_{\text{pointer}} = \left[\frac{m_{\text{low}} + m_{\text{pointer}}}{2}\right]$
5	Select the m_{pointer} value giving the smallest code length taken from Table A.

4. Experimental results

The goal of the experiments is to assess, respectively, the contribution of the bilevel methodology first, and then of the introduction of a new encoding for the break points and of CMA-ES as the optimisation strategy. Thus, the comparison was split into two parts.

First, we show the empirical comparison results concerning the single-level method (AutoPARM-GA) versus the bilevel method using the NI algorithm. Though in this case the problem parametrisation was different, the optimisation algorithm is the same: the NI algorithm. The question in this case is which one of the different exploration strategies exhibit better performance. The numbers of function evaluations are not comparable due to the fundamentally different objective functions. Therefore we chose the target performance metric to be the CPU time versus the optimisation history. This metric is indeed comparable for the two optimisation methods if they are run in exactly the same computer environment and the same implementation of the NI algorithm implementation is used for both cases.

Second, we show the empirical comparison results concerning the bilevel optimisation using the NI algorithm versus the CMA-ES one. Because of the different implementations of the optimisation algorithms, a fair comparison should exclude the implementation-related effects of the evolutionary algorithms. Thus, in this case we compared the number of objective function evaluations versus the history of the best objective function values.

The technical details of the experiments are given in Section 4.6.

4.1. The benchmarks

All experiments are conducted on the following six test cases:

- **PAR_many** – Piecewise AR model with many segments
- **PAR_dyad** – Piecewise stationary process with dyadic structure
- **SlowAR** – Slowly varying AR(2) process
- **P_ARMA** – Piecewise ARMA process
- **Tvar_MA** – Time varying MA(2) process
- **Short** – Short segments

PAR_many is motivated by [12], with a relatively large number of segments. The other ones have been defined by Davis in [9]; for the sake of completeness, they are recalled in Appendix A. PAR_many model has 8 segments defined by

$$Y_t = \begin{cases} 0.9Y_{t-1} + \epsilon_{t,1}, & \text{if } 1 \le t \le 320, \\ -0.3Y_{t-1} + \epsilon_{t,2}, & \text{if } 321 \le t \le 512, \\ 1.69Y_{t-1} - 0.81Y_{t-2} + \epsilon_{t,3}, & \text{if } 513 \le t \le 768, \\ 1.32Y_{t-1} - 0.81Y_{t-2} + \epsilon_{t,4}, & \text{if } 769 \le t \le 1024, \\ -0.3Y_{t-1} + \epsilon_{t,5}, & \text{if } 1025 \le t \le 1310, \\ 0.53Y_{t-1} - 0.23Y_{t-2} + \epsilon_{t,6}, & \text{if } 1311 \le t \le 1460, \\ -0.75Y_{t-1} + \epsilon_{t,7}, & \text{if } 1461 \le t \le 1832, \\ -0.23Y_{t-1} + 0.35Y_{t-2} + \epsilon_{t,8}, & \text{if } 1833 \le t \le 2048. \end{cases}$$

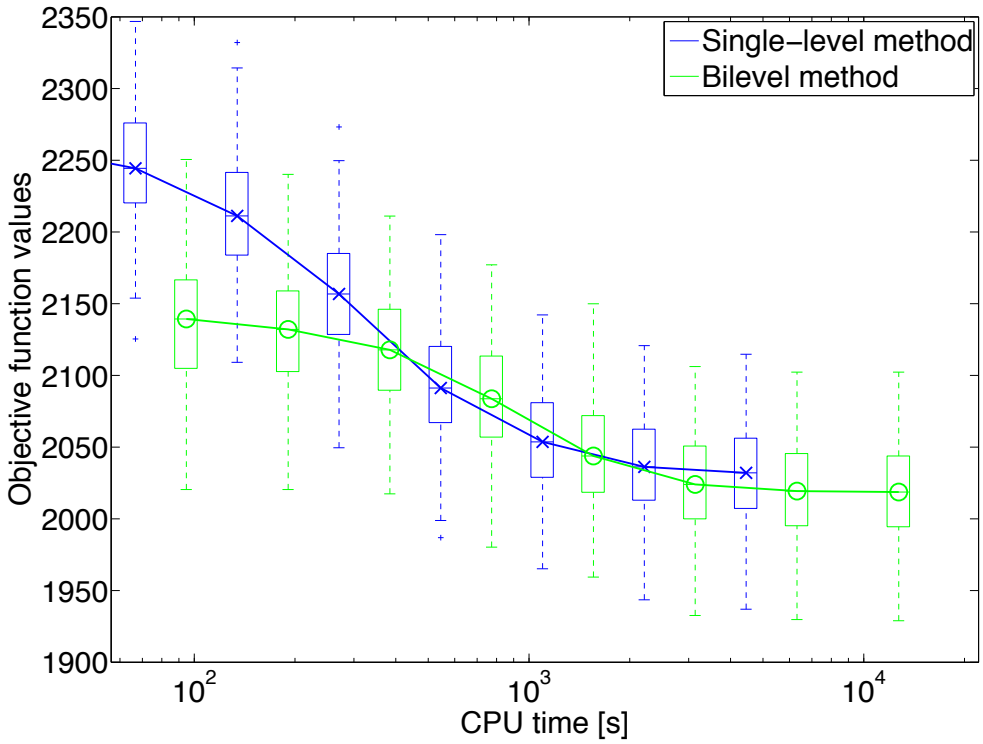

Figure 2: Performance comparison of the single-level and the bilevel method in the case of the PAR_many model (200 independent runs – 8 AR segments were generated, 7–8 were found; ca. 30K samples of objective function values in the optimisation history per one run; the boxes represent the 0.25, 0.5, 0.75 quantiles, the whiskers represent ca. the 0.07, 0.993 quantiles of the objective function values at chosen sample points). The average code length of the random generated resies using the PAR_many model evaluated at the true parameter values was 2052 (std: 42) over the 200 independent runs. The NI algorithm was used here.

4.2. Performance comparison of the single- and bilevel methods

Figures 2 to 7 show that the single-level and bilevel optimisation methods explore the parameter space with similar efficiency after an initial start up phase. Furthermore, at the beginning of the optimisation, the bilevel method is significantly more efficient as the range of explored code length values is lower compared to its single-level counterpart. This effect is particularly apparent in Figure 2: the bilevel method reaches the 2100-2150 range of code length values at ca. 100 seconds of CPU running time, while the single-level method reaches the same range at ca. 300 seconds. After ca. 400 seconds, the two historical curves run together.

We also note, that both experiments use the NI algorithm, whose running time scales together with the data length and this is also reflected in these performance figures.

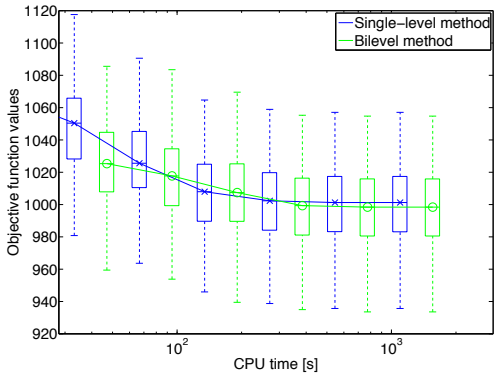

Figure 3: PAR_dyad

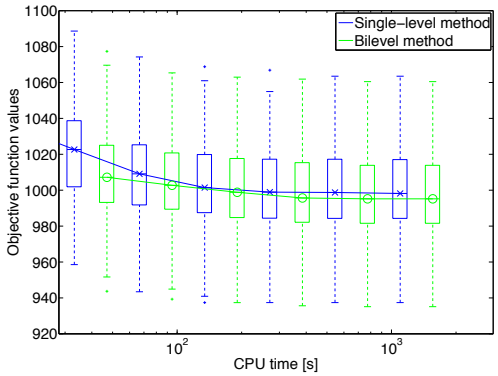

Figure 4: SlowAR

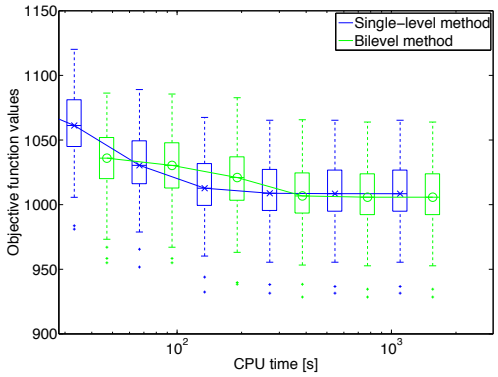

Figure 5: P_ARMA

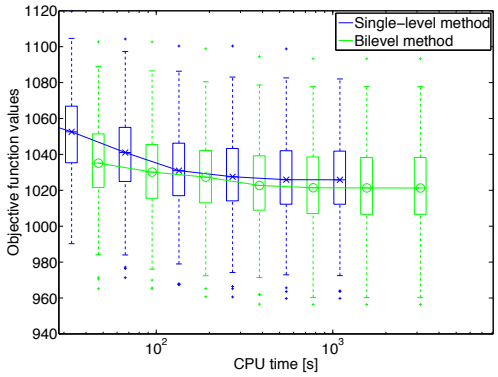

Figure 6: Tvar_MA

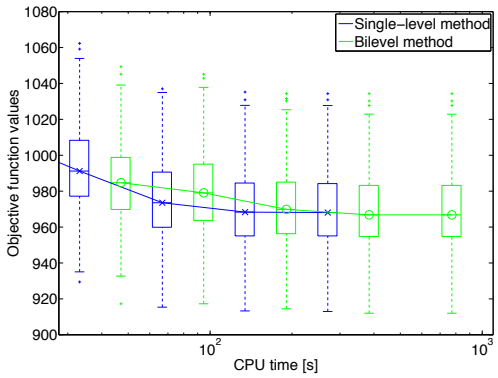

Figure 7: Short

4.3. Performance comparison of the NI and CMA-ES algorithms

The straightforward way to find the number of break points minimising the code length is to test all possible values $m = 0, \ldots, m_{\max}$. However, depending on the value of $m_{\max}$ this might be rather time-consuming. If instead our search algorithm is used, then we will be able to find the optimal number of break points m when $\overline{f}(\cdot)$ is unimodal. In this case, the optimal number of break points can be found within $\log_2 m_{\max}$ steps by halving the region to be explored in each test of the $\overline{f}(\cdot)$ values.

The number of chromosomes, λ, used by CMA-ES should be chosen carefully, keeping efficiency in mind. According to our practical experience, a moderate (e.g. in the order of 40–60) population size is sufficient for finding the optimal number of break points, because the exact break point locations are not yet needed for this task. Instead, after the number of break points has been selected, a second optimisation run can be performed with an increased population size (e.g. in the order of 500-1500) so that the break point location estimations are refined.

As a further enhancement of the model fitting method, after the optimisation has converged, it is restarted with increased population sizes [2] until a limit on the cumulative number of function evaluations is reached.

According to our performance tests, CMA-ES needs considerably smaller number of function evaluations than the NI algorithm. Figure 8 shows the performance comparison of the bilevel method with NI and CMA-ES through different number of function evaluations. One can see in Figure 8, that though initially the NI algorithm performs better, after the start up phase, the code length estimates of the bilevel method with the CMA-ES algorithm can reach the range of 2000-2080 in ca. 14K function evaluations, while the bilevel method with the NI algorithm can reach the same range after ca. 520K function evaluations. Considering the slight performance gain that was experienced for the same model in Section 4.2 we can conclude that, in this test example, the performance of the bilevel method with the CMA-ES algorithm is superior by more than 1 order of magnitude over the single-level method with the NI algorithm (AutoPARM-GA). This performance gain appears to be even more appealing if we remember that the search space dimension for the NI algorithm, as proposed by [9], is proportional to the length of the time series whereas CMA-ES is independent of this length.

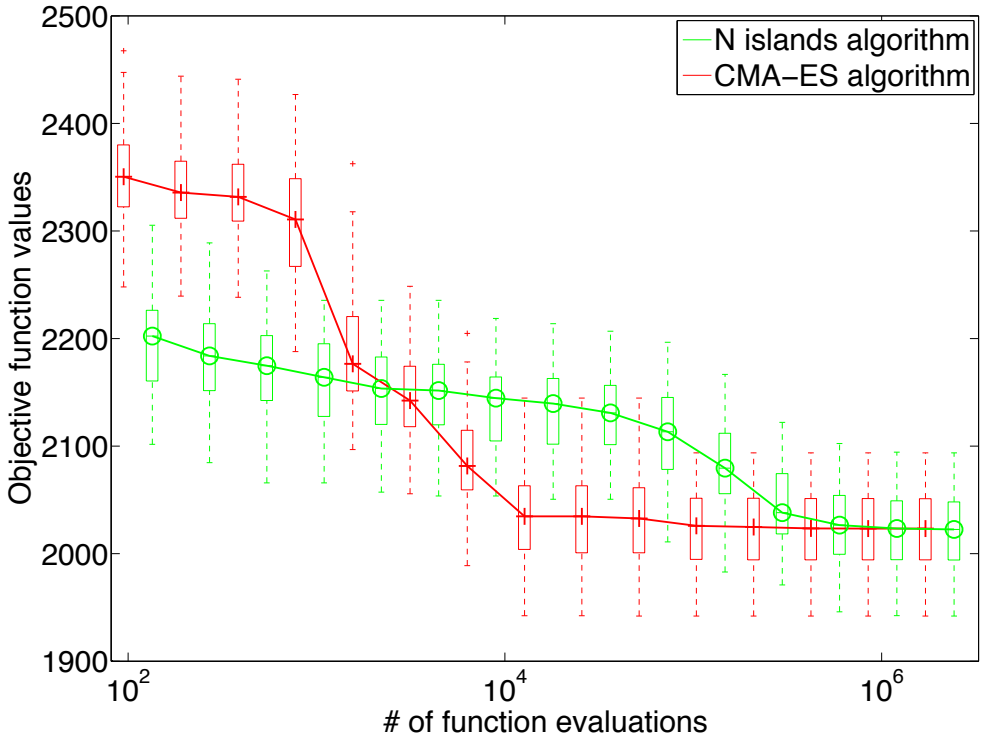

Figure 8: Performance comparison of the NI and the CMA-ES algorithm in the case of PAR_many model (200 independent runs – 8 AR segments were generated, 7–8 were found; ca. 20K samples of objective function values in the optimisation history per one run; the boxes represent the 0.25, 0.5, 0.75 quantiles, the whiskers represent ca. the 0.07, 0.993 quantiles of the objective function values at chosen sample points).

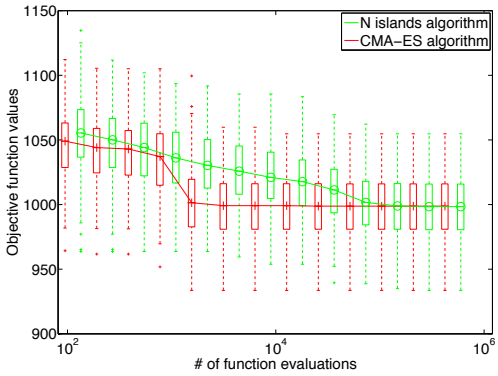

Figure 9: PAR_dyad

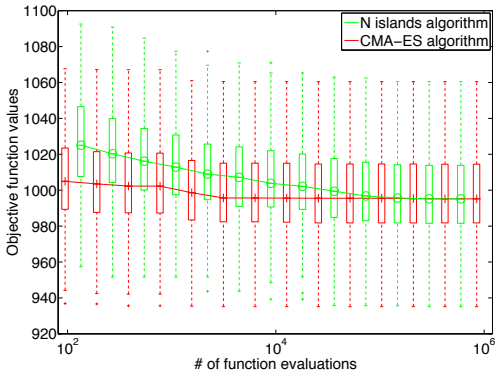

Figure 10: SlowAR

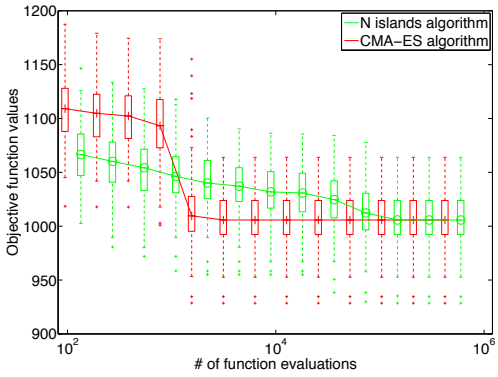

Figure 11: P_ARMA

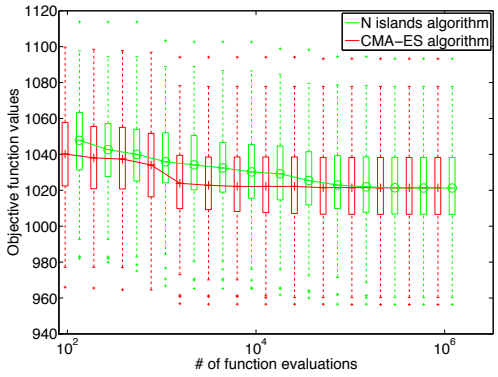

Figure 12: Tvar_MA

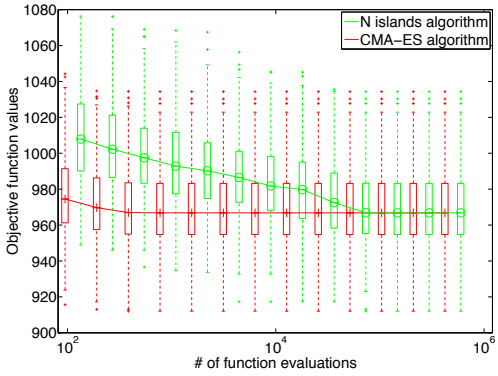

Figure 13: Short

4.4. Average CPU times per function evaluation

This section presents empirical comparison results concerning the average CPU time per function evaluation for all three model fitting methods and highlight the main effects that lead to our performance observations. Table 2 shows the results. The values were obtained by dividing the total CPU running times by the total number of function evaluations for each run, where a run is defined by the triple test case, algorithm, method.

Table 2: Comparison of CPU seconds per objective function evaluation. The values are the aggregate CPU time divided by the total number of function evaluations for each experiment.

Model	**NI alg.**		**CMA-ES alg.**
	single-level method	bilevel method	
PAR_dyad	5.7e-4	21.7e-4	8.5e-4
SlowAR	6.0e-4	22.1e-4	9.7e-4
P_ARMA	5.3e-4	21.3e-4	8.0e-4
Tvar_MA	7.1e-4	29.2e-4	12.6e-4
Short	5.9e-4	17.6e-4	6.5e-4
PAR_many	12.6e-4	47.2e-4	17.7e-4

The most significant effect that determines the CPU time per function evaluation is, of course, the length of the series. Table 2 clearly demonstrates this, as the average CPU time for the PAR_many model is about twice as much as for the other models in all the columns. The reason for this is, that for this model, the length of the series is 2048, while it is 1024 for the rest of the models.

The effect of the difference between the single-level method and the bilevel method can also be studied by comparing the first and second columns of Table 2. The bilevel method performs an exhaustive search over all possible AR orders up to $p_{\max} = 20$, while the single-level method does not. This increases the CPU time per function evaluation for the bilevel method by at most a factor of ca. 4 as it can be seen in Table 2.

Another effect is the running time overhead of the optimisation procedure. This can be quite significant as it can be seen by comparing the second and third columns of Table 2, where the objective functions are exactly the same (in fact, the same C implementation was used in the tests), but the optimisation algorithms are different. Though a large part of the NI algorithm was implemented in C and the CMA-ES algorithm was completely implemented in Matlab, the comparison shows that the CPU time, and thus the overhead of the optimisation algorithm, is smaller for CMA-ES.

Finally, we note that the rows of Table 2 show different values for the different models. The main reason for this is that the contribution of the optimisation algorithm and the contribution of the objective function evaluation to the average CPU time depend very much on the optimisation path (e.g. the number and locations of the break points, the AR orders, etc.), which is different for each model.

4.5. Accuracy

So far we showed, that the minimal code length estimate, and thus the optimal model fit, is more efficiently approached by our proposed solution. Table 4 shows the parameter values such as

Table 3: Comparisons of the number of function evaluations until the bilevel optimisation method with NI and CMA-ES algorithms reached the final range of code length values.

Model	NI algorithm	CMA-ES algorithm	Figure
PAR_dyad	140K	2K	Figure 9
SlowAR	140K	4K	Figure 10
P_ARMA	140K	4K	Figure 11
Tvar_MA	140K	2K	Figure 12
Short	70K	1K	Figure 13
PAR_many	520K	14K	Figure 8

the number of break points, ranges of break point locations and distribution of AR orders selected by the different runs for our illustrative scenario, the PAR_many test case (Appendix B gives similar tables for the other test cases). For the sake of brevity, we show only the distribution of AR orders collected from runs with the most frequent number of break points. Also, in case of many segments, we show only the results for a few of them. The fitted values are quite similar for each of the three procedures. We note, however, that the bilevel method with the NI algorithm produces consistently slightly better solution than the single-level algorithm and the CMA-ES algorithm is slightly even more precise than the two others.

4.6. Technical details

The length of the series for all the benchmark examples were 1024, except for the first one, where it is 2048. The maximum of the AR orders to be considered was set to 20.

The dimensionality of the problem for the NI algorithm with single-level optimisation is 1000-2000, where each single parameter can take 22 different values. The dimensionality of the NI algorithm with bilevel optimisation is the same, however with effectively only two different values. The dimensionality of the problem for CMA-ES with bilevel optimisation depends on the number of break points explored on the optimisation path (it is at most 10-20).

The parameters for the NI algorithm are the parameters proposed in [9] except that the number of islands is increased to 100 (from 40), the population size on each island is increased to 100 (from 40) and the limit on the number of migrations is set to a large value so that the optimisation always stops due to that the overall best chromosome has not changed for 10 consecutive migrations. Our reasons for changing these parameters (e.g. the approximately 4x increase in the overall size of the population) is that, according to our experience, this speeded up tho optimisation process because of the following main reasons:

- It turned out, that the larger population resulted generally so much gain in the objective function value that the same level could be reached using 90%-95% number of function evaluations compared to an optimisation with the original population sizes.
- It turned out that the original population sizes sometimes led the optimisation stuck into a sub-optimum. Then, neither the crossover nor the mutation operations moved the process out of this sub-optimum (at least on the intended timescale of our runs).

Regarding CMA-ES, we use the default optimisation parameters except that the covariance matrix estimate is set to be diagonal [20]. The population size is set to 50 during the phase when

Table 4: Comparison of some statistics of the experiments (PAR_many model).

	NI algorithm		**CMA-ES algorithm**
	single-level method	bilevel method	
relative frequencies where different number of break points were found			
4> break points	0.0000	0.0000	0.0000
4 break points	0.0000	0.0000	0.0591
5 break points	0.0246	0.0049	0.0246
6 break points	0.3892	0.1232	0.0296
7 break points	0.5468	0.7931	0.8030
8 break points	0.0394	0.0788	0.0690
9 break points	0.0000	0.0000	0.0148
other	0.0000	0.0000	0.0000
relative locations of break points in cases where 7 break points were found			
	average (standard deviation)		
break point 1	0.1566 (2.0e-3)	0.1566 (2.1e-3)	0.1566 (2.1e-3)
break point 2	0.2499 (9.4e-4)	0.2499 (1.0e-3)	0.2499 (1.1e-3)
break point 3	0.3765 (5.4e-3)	0.3756 (4.2e-3)	0.3769 (6.4e-3)
break point 4	0.5000 (1.4e-3)	0.5000 (1.3e-3)	0.5000 (1.3e-3)
break point 5	0.6398 (5.7e-3)	0.6399 (3.7e-3)	0.6399 (4.1e-3)
break point 6	0.7130 (2.7e-3)	0.7130 (2.5e-3)	0.7130 (2.3e-3)
break point 7	0.8975 (1.6e-2)	0.8976 (1.0e-2)	0.8978 (1.0e-2)
relative frequencies of AR orders when 7 break points were found			
	Segment 1		
order <1	0.0000	0.0000	0.0000
order 1	0.9882	0.9882	0.9882
order 2	0.0118	0.0118	0.0118
order >2	0.0000	0.0000	0.0000
	Segment 2		
order 0	0.0235	0.0118	0.0118
order 1	0.8588	0.9765	0.9765
order 2	0.0941	0.0118	0.0118
order 3	0.0118	0.0000	0.0000
order 4	0.0118	0.0000	0.0000
order >4	0.0000	0.0000	0.0000

the number of break points are determined and it is set to 1000 during the phase when the break point locations are refined. When the CMA-ES optimisation is restarted, the population sizes are increased by a factor of 2.

We randomly generate 200 time series samples following the prescribed models for each experiment. Then, we fit piecewise AR models using the single-level and bilevel optimisation methods with the NI algorithm and using the bilevel method with the CMA-ES algorithm. The code length values of the model fittings are recorded for each experiment during the optimisation and the observations are stored in historical tables. One historical entry contains the CPU-time, the number of function evaluations and the code length value at that instant. Approximately, 20K-30K historical entries are saved. Finally, each historical curve is interpolated and sampled at certain time instants and the quantile statistics of the CPU seconds and the number of function evaluations are collected

and plotted at ca. 0.003 (lower whiskers), 0.25 (bottoms of the boxes), 0.5 (middle points of the boxes), 0.75 (tops of the boxes) and at ca. 0.993 (upper whiskers).

Appendix B provides the numerical details for all experiments. Generally we observe that the bilevel method using the NI algorithm shows the same or superior performance over the single-level method and the final objective function values after convergence are always better for the bilevel optimisation method.

4.7. Extending the segmentation model

The code length function of the residual part (4) uses an approximation of the maximum likelihood estimation by using the Yule-Walker estimate. For large segment sizes, this approximation is satisfactory. However, the Yule-Walker estimator might be replaced by another estimator for several reasons. On the one hand, particularly for short segments, the maximum likelihood estimate might be a better choice. On the other hand, the use of the Durbin-Levinson algorithm is restricted to AR models. In the case of autoregressive and moving average (ARMA) models or nonlinear models like conditionally heteroscedastic models, effective closed-form parameter estimation methods similar to the Yule-Walker estimator do not exist in general.

In these cases, one should use numerical optimisation techniques to obtain relevant estimates. In the framework of the NI algorithm, this means that for each chromosome and for each segment one should run a second- (or third-) level optimisation and then use the obtained negative log-likelihoods for calculating the code length estimate. This can be rather time consuming and therefore the selection of an efficient optimisation algorithm is essential. In such cases, CMA-ES is a promising alternative especially for optimisation problems involving a large number of parameters and difficult optimisation functions.

5. Conclusions

In this paper, we consider the problem of fitting a piecewise autoregressive model to a time series using the MDL principle. We are dealing with two main aspects of the fitting procedure. The first one is the method that explores the parameter space of piecewise autoregressive models. The second one is the algorithm that actually carries out the optimisation.

Our study is based on simulations on five test examples given in [9] and on a sixth example which is motivated by practical experience from [12] with a longer series and larger number of break points.

We show that the performance of the single-level method using the NI algorithm can be improved to a moderate extent by replacing it with a bilevel method. An empirical comparison of the total CPU-time of the single-level and bilevel method using the NI optimisation algorithm is presented. We find that the CPU-time until convergence is the same or slightly smaller in favour of the bilevel method compared to its single-level counterpart.

The same optimisation problems are used to compare the bilevel method using the NI and CMA-ES algorithms. In these cases, the number of upper level objective function evaluations are presented. Here, the upper level objective function incorporates the (lower level) optimisation of the autoregressive models. We find that CMA-ES algorithm has at least one order of magnitude faster convergence in the examples. Therefore, based on our empirical experience, we propose the bilevel method using CMA-ES for piecewise autoregressive model fitting purposes.

6. Acknowledgements

This work was partially supported by the Île-de-France council/DIGITEO under the Software and Complex Systems program. We thank the anonymous referees for their helpful recommendations.

Appendix A. Experiment details

Appendix A.1. PAR_dyad

This example follows a piecewise AR model given as

$$Y_t = \begin{cases} 0.9Y_{t-1} + \epsilon_{t,1}, & \text{if } 1 \le t \le 512, \\ 1.69Y_{t-1} - 0.81Y_{t-2} + \epsilon_{t,2}, & \text{if } 513 \le t \le 768, \\ 1.32Y_{t-1} - 0.81Y_{t-2} + \epsilon_{t,3}, & \text{if } 769 \le t \le 1024. \end{cases}$$

Table B.5 shows the detailed results.

Appendix A.2. SlowAR

This example follows a time-dependent AR model given as

$$Y_t = a_t Y_{t-1} - 0.81Y_{t-2} + \epsilon_t, \quad t = 1, 2, \ldots, 1024$$

where $a_t = 0.8(1 - 0.5\cos(\pi t/1024))$ and $\epsilon_t \sim$ iid $N(0, 1)$. Table B.6 shows the detailed results.

Appendix A.3. P_ARMA

This example follows a P_ARMA model given as

$$Y_t = \begin{cases} -0.9Y_{t-1} + \epsilon_{t,1} + 0.7\epsilon_{t-1,1}, & \text{if } 1 \le t \le 512, \\ 0.9Y_{t-1} + \epsilon_{t,2}, & \text{if } 513 \le t \le 768, \\ \epsilon_{t,3} - 0.7\epsilon_{t-1,3}, & \text{if } 769 \le t \le 1024. \end{cases}$$

Table B.7 shows the detailed results.

Appendix A.4. Tvar_MA

This example follows a time-dependent MA(2) model given as

$$Y_t = \epsilon_t + a_t\epsilon_{t-1} + 0.5\epsilon_{t-2}, \quad t = 1, 2, \ldots, 1024$$

where $a_t = 1.122(1 - 1.781\sin(\pi t/2048))$ and $\epsilon_t \sim$ iid $N(0, 1)$. Table B.8 shows the detailed results.

Appendix A.5. Short

This example follows a piecewise AR model given as

$$Y_t = \begin{cases} 0.75Y_{t-1} + \epsilon_{t,1}, & \text{if } 1 \le t \le 50, \\ -0.50Y_{t-1} + \epsilon_{t,2}, & \text{if } 51 \le t \le 1024. \end{cases}$$

Table B.9 shows the detailed results.

Appendix B. Detailed results for the fitted parameters

Table B.5: PAR_dyad

	NI algorithm		**CMA-ES algorithm**
	single-level method	bilevel method	
relative frequencies where different number of break points were found			
1 break point	0.0000	0.0000	0.0000
2 break points	0.9600	0.9450	0.9900
3 break points	0.0400	0.0550	0.0100
4 break points	0.0000	0.0000	0.0000
other	0.0000	0.0000	0.0000
relative locations of break points in cases where 2 break points were found			
	average (standard deviation)		
break point 1	0.4973 (1.0e-2)	0.4985 (1.0e-2)	0.4969 (1.1e-2)
break point 2	0.7512 (1.0e-2)	0.7502 (9.4e-3)	0.7514 (1.2e-2)
relative frequencies of AR orders when 2 break points were found			
	Segment 1		
order 0	0.0000	0.0000	0.0000
order 1	1.0000	1.0000	1.0000
order >1	0.0000	0.0000	0.0000
	Segment 2		
order <2	0.0000	0.0000	0.0000
order 2	0.9572	0.9893	0.9893
order 3	0.0321	0.0053	0.0053
order 4	0.0107	0.0053	0.0053
order >4	0.0000	0.0000	0.0000
	Segment 3		
order <2	0.0000	0.0000	0.0000
order 2	0.9091	0.9840	0.9893
order 3	0.0856	0.0160	0.0107
order 4	0.0053	0.0000	0.0000
order >2	0.0000	0.0000	0.0000

Table B.6: SlowAR

	NI algorithm		**CMA-ES algorithm**
	single-level method	bilevel method	
relative frequencies where different number of break points were found			
1 break point	0.0000	0.0000	0.0000
2 break points	0.4700	0.3350	0.4150
3 break points	0.5200	0.6450	0.5850
4 break points	0.0100	0.0200	0.0000
other	0.0000	0.0000	0.0000
relative locations of break points in cases where 2 break points were found			
	average (standard deviation)		
break point 1	0.3591 (7.2e-2)	0.3574 (7.1e-2)	0.3545 (7.0e-2)
break point 2	0.6801 (7.2e-2)	0.6827 (7.5e-2)	0.6817 (6.9e-2)
relative frequencies of AR orders when 2 break points were found			
	Segment 1		
order <1	0.0000	0.0000	0.0000
order 2	0.9688	0.9688	0.9792
order 3	0.0208	0.0208	0.0104
order 4	0.0104	0.0104	0.0104
order >4	0.0000	0.0000	0.0000
	Segment 2		
order <2	0.0000	0.0000	0.0000
order 2	0.9479	0.9583	0.9688
order 3	0.0521	0.0313	0.0208
order 4	0.0000	0.0104	0.0104
order >4	0.0000	0.0000	0.0000
	Segment 3		
order <2	0.0000	0.0000	0.0000
order 2	0.9583	0.9688	0.9688
order 3	0.0313	0.0208	0.0208
order 4	0.0104	0.0104	0.0104
order >4	0.0000	0.0000	0.0000

Table B.7: Comparison of some statistics of the experiments (P_ARMA).

	NI algorithm		**CMA-ES algorithm**
	single-level method	bilevel method	
relative frequencies where different number of break points were found			
1 break point	0.0000	0.0000	0.0000
2 break points	0.9950	0.9800	0.9900
3 break points	0.0050	0.0200	0.0100
other	0.0000	0.0000	0.0000
relative locations of break points in cases where 2 break points were found			
	average (standard deviation)		
break point 1	0.4999 (6.2e-3)	0.5002 (6.3e-3)	0.5001 (6.1e-3)
break point 2	0.7505 (2.9e-3)	0.7507 (2.7e-3)	0.7507 (3.0e-3)
relative frequencies of AR orders when 2 break points were found			
	Segment 1		
order 0	0.0000	0.0000	0.0000
order 1	0.0408	0.0408	0.0408
order 2	0.2194	0.2194	0.2194
order 3	0.4541	0.4592	0.4592
order 4	0.2245	0.2245	0.2245
order 5	0.0561	0.0510	0.0510
order 6	0.0051	0.0051	0.0051
order >6	0.0000	0.0000	0.0000
	Segment 2		
order <1	0.0000	0.0000	0.0000
order 1	0.9796	0.9898	0.9898
order 2	0.0102	0.0102	0.0102
order 3	0.0051	0.0000	0.0000
order 4	0.0051	0.0000	0.0000
order >4	0.0000	0.0000	0.0000
	Segment 3		
order <1	0.0000	0.0000	0.0000
order 1	0.0051	0.0051	0.0051
order 2	0.1888	0.2296	0.2296
order 3	0.4388	0.4184	0.4184
order 4	0.2857	0.2755	0.2755
order 5	0.0612	0.0612	0.0612
order 6	0.0153	0.0102	0.0102
order >6	0.0051	0.0000	0.0000

Table B.8: Comparison of some statistics of the experiments (Tva_MA).

	NI algorithm		**CMA-ES algorithm**
	single-level method	bilevel method	
relative frequencies where different number of break points were found			
1 break point	0.0450	0.0200	0.0300
2 break points	0.8700	0.7850	0.7850
3 break points	0.0850	0.1950	0.1800
4 break points	0.0000	0.0000	0.0050
other	0.0000	0.0000	0.0000
relative locations of break points in cases where 2 break points were found			
	average (standard deviation)		
break point 1	0.2400 (6.4e-2)	0.2395 (6.4e-2)	0.2438 (6.9e-2)
break point 2	0.5525 (7.9e-2)	0.5571 (7.7e-2)	0.5565 (8.3e-2)
relative frequencies of AR orders when 2 break points were found			
	Segment 1		
order <1	0.0000	0.0000	0.0000
order 1	0.1644	0.1644	0.1644
order 2	0.3562	0.3151	0.3014
order 3	0.1849	0.1849	0.1849
order 4	0.2397	0.2534	0.2740
order 5	0.0479	0.0685	0.0685
order >5	0.0068	0.0136	0.0068
	Segment 2		
order <2	0.0000	0.0000	0.0000
order 2	0.4726	0.4247	0.4110
order 3	0.1575	0.1644	0.1849
order 4	0.3082	0.3493	0.3425
order 5	0.0548	0.0548	0.0548
order >5	0.0068	0.0068	0.0068
	Segment 3		
order <2	0.0000	0.0000	0.0000
order 2	0.1507	0.1712	0.1712
order 3	0.0685	0.0822	0.0822
order 4	0.6712	0.6233	0.6301
order 5	0.0685	0.0890	0.0822
order 6	0.0274	0.0205	0.0205
order >6	0.0137	0.0137	0.0137

Table B.9: Comparison of some statistics of the experiments (Short).

	NI algorithm		**CMA-ES algorithm**
	single-level method	bilevel method	
relative frequencies where different number of break points were found			
0 break point	0.0050	0.0050	0.0000
1 break points	0.9950	0.9850	1.0000
2 break points	0.0000	0.0100	0.0000
other	0.0000	0.0000	0.0000
relative locations of break points in cases where 1 break point was found			
	average (standard deviation)		
break point	0.0491 (5.1e-3)	0.0486 (4.5e-3)	0.0486 (4.5e-3)
relative frequencies of AR orders when 2 break points were found			
	Segment 1		
order 0	0.0102	0.0102	0.0102
order 1	0.9492	0.9543	0.9543
order 2	0.0305	0.0254	0.0254
order 3	0.0101	0.0102	0.0102
order >3	0.0000	0.0000	0.0000
	Segment 2		
order 0	0.0000	0.0000	0.0000
order 1	0.9695	0.9949	0.9949
order 2	0.0254	0.0051	0.0051
order 3	0.0051	0.0000	0.0000
order >3	0.0000	0.0000	0.0000

References

[1] A. Aue, S. Hörmann, L. Horváth, and M. Reimherr. Break detection in the covariance structure of multivariate time series models. The Annals of Statistics, 37:4046–4087, 2009.

[2] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In B. McKay et al., editors, The 2005 IEEE International Congress on Evolutionary Computation (CEC'05), volume 2, pages 1769–1776, 2005.

[3] J. Bai and P. Perron. Estimating and testing linear models with multiple structural changes. Econometrica, 66(1):47–78, 1998.

[4] J. Bai and P. Perron. Multiple structural change models: a simulation analysis. In Econometric theory and practice, pages 212–237. Cambridge Univ. Press, Cambridge, 2006.

[5] M. Basseville and N. Nikiforov. The Detection of abrupt changes: Theory and applications. Information and System sciences. Prentice-Hall, 1993.

[6] P. J. Brockwell and R. A. Davis. Time series: Theory and Methods. Springer series in statistics. Springer, New York, NY, 2. ed edition, 1991.

[7] Z. Brodsky and B. Darkhovsky. Nonparametric methods in change-point problem. Kluwer Academic Publishers, 1993.

[8] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. Annals of Operations Research, 153:235–256, 2007.

[9] R. A. Davis, T. C. M. Lee, and G. A. Rodriguez-Yam. Structural break estimation for non-stationary time series models. Journal of the American Statistical Association, 101:223–239, March 2006.

[10] B. Efron and D. V. Hinkley. Assessing the accuracy of the maximum likelihood estimator: Observed versus expected fisher information. Biometrika, 65(3):457–483, December 1978.

[11] T. Elteto, S. Germain-Renaud, P. Bondon, and M. Sebag. Towards non-stationary grid models.

[12] T. Elteto, S. Germain-Renaud, P. Bondon, and M. Sebag. Discovering piecewise linear models of grid workload. In 10th IEEE/ACM Int. Symp. on Cluster, Cloud, and Grid Computing, 2010.

[13] C. D. Giurcaneanu and J. Rissanen. Estimation of ar and arma models by stochastic complexity. IMS Lecture Notes-Monograph Series, Time Series and Related Topics, 52:48–59, 2006.

[14] P. D. Grünwald. The Minimum Description Length Principle, volume 1 of MIT Press Books. The MIT Press, December 2007.

[15] N. Hansen. The CMA evolution strategy: A comparing review. In Jose A. Lozano, Pedro Larrañaga, Iñaki Inza, and Endika Bengoetxea, editors, Towards a New Evolutionary Computation, volume 192 of Studies in Fuzziness and Soft Computing, chapter 4, pages 75–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[16] H. Ombao, R. von Sachs, and W. Guo. SLEX analysis of multivariate nonstationary time series. Journal of the American Statistical Association, 100(470):519–531, June 2005.

[17] H. Ombao, R. von Sachs, and B. A. Marlow. Automatic statistical analysis of bivariate nonstationary time series. Journal of the American Statistical Association, 96:543–560, 2001.

[18] J. Rissanen. Order estimation by accumulated prediction errors. Journal of Applied Probability, 23:55–61, 1986.

[19] J. Rissanen and T. Roos. Conditional nml universal models. In Information Theory and Applications Workshop, 2007, pages 337–341, 29 2007-feb. 2 2007.

[20] R. Ros and N. Hansen. A simple modification in CMA-ES achieving linear time and space complexity. In 10th International Conference on Parallel Problem Solving From Nature, Dortmund Germany, 2008.

[21] Darrell Whitley, Soraya Rana, and Robert B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. Journal of Computing and Information Technology, 7:33–47, 1998.

[22] K. Yamanishi and Y. Maruyama. Dynamic model selection with its applications to novelty detection. IEEE TRANSACTIONS ON INFORMATION THEORY, 53(6):2180–2189, 2007.

Tamás Éltető currently is a postdoc at Paris-Sud University. His research interests are in Probability Theory and Mathematical Statistics.

Nikolaus Hansen is a researcher at The French National Institute for Research in Computer Science and Control (INRIA). Before he joined INRIA, he has been working in applied artificial intelligence, in genomics, in evolutionary computation and in computational science. His main research interests are learning and adaptation in evolutionary computation and the development of algorithms applicable in practice.

Pascal Bondon is a researcher at The French National Center for Scientific Research (CNRS). He works in the Laboratory of Signals and Systems (L2S) at Supélec. His research interests are in time series analysis and prediction theory.

Cécile Germain-Renaud is a full Professor at Paris-Sud University. Her research interests are in models for High Performance Computing, and in Autonomic Grid and Clouds. She is involved with the European Grid Initiative, and leads the Grid Observatory project. Her web site is at

`http://www.lri.fr/~cecile/communications-english.html`.