INFORMATICA, 2024, Vol. 0, No. 0, 1–33

DOI: https://doi.org/10.15388/24-INFOR572

On the Improvements of Metaheuristic Optimization-Based Strategies for Time Series Structural Break Detection

Mateusz BURCZANIUK, Agnieszka JASTRZĘBSKA*

Faculty of Mathematics and Information Science, Warsaw University of Technology, ul. Koszykowa 75, Warsaw, 00-662, Poland
e-mail: Mat.Burczaniuk@gmail.com, A.Jastrzebska@mini.pw.edu.pl

Received: October 2023; accepted: September 2024

Abstract. Structural break detection is an important time series analysis task. It can be treated as a multi-objective optimization problem, in which we ought to find a time series segmentation such that time series theoretical models constructed on each segment are well-fitted and the segments are long enough to bear meaningful information. Metaheuristic optimization can help us solve this problem. This paper introduces a suite of new cost functions for the structural break detection task. We demonstrate that the new cost functions allow for achieving quantitatively better precision than the cost functions employed in the literature of this domain. We show particular advantages of each new cost function. Furthermore, the paper promotes the use of Particle Swarm Optimization (PSO) in the domain of structural break detection, which so far has relied on the Genetic Algorithm (GA). Our experiments show that PSO outperforms GA for many analysed time series examples. Last but not least, we introduce a non-trivial generalization of the top-performing state-of-the-art approach to the structural break detection problem based on the Minimum Description Length (MDL) rule with autoregressive (AR) model to MDL ARIMA (autoregressive integrated moving average) model.

Key words: time series, structural break, ARIMA, Genetic Algorithm, Particle Swarm Optimization, Ant Colony Optimization,, Minimum Description Length.

1. Introduction

Structural break detection is a time series analysis problem aiming to detect change points in a sequence of observations forming the time series. It is of particular value for areas such as economics or incident detection, in which we are interested in analysing time series in segments, and we need an algorithm to partition the data.

The structural break detection problem can be naturally formulated as an optimization task, where we wish to minimize a specific cost function defined for the produced time series segments. The rough idea is that each produced segment should represent a sequence of observations without a change point. This is typically measured using residuals of a

*Corresponding author.

theoretical model built on a segment and real time series values, which should be as small as possible. Unfortunately, simplifying the problem into just such an assumption leads to segments composed of single observations. Thus, the problem of structural break detection is typically formulated as a balance between two conflicting criteria: we wish to split the time series into segments without change points and, at the same time, maximize the length of the segments.

We may divide the existing structural break detection approaches into two groups. One uses predominantly statistical procedures. A prominent representative of this group is Bai and Perron (1998, 2003), which is widely utilized despite its simplicity. It uses linear regression as the base model for produced segments and a supplemental statistical test to determine the number of structural breaks. The second group fuses metaheuristic optimization with statistical procedures. Statistical procedures are responsible for evaluating the goodness of fit of theoretical models on time series segments. Metaheuristic optimization carries the entire process forward. The approach was presented by Davis *et al.* (2016, 2006, 2008), Davis and Yau (2013). Later studies, for example, Woody *et al.* (2021), Shi *et al.* (2022a) elaborated on the application of Davis's method, but essentially, on the methodological level, left it untouched.

In this paper, we present novel contributions to the domain of structural break detection using metaheuristic optimization. In particular:

1. We present a suite of new cost functions to be used for structural break detection.
2. We present a new methodology for cost function construction that assumes that the cost function value should contain a proportional weight for the variance of residual errors and take into account the relation between segment order and its length.
3. We present a generalization of Davis's approach to the MDL with the ARIMA model. This generalization at the first sight may seem trivial compared to the initial Davis's model based on AR. However, when broken down into parts, it required introducing model-specific optimizations to manage the computational complexity. The generalized model allowed for increasing accuracy of break detection in comparison to the AR-based approach.
4. To the best of our knowledge, this is also the first paper demonstrating the application of Ant Colony Optimization and Particle Swarm Optimization algorithms in conjunction with the Minimum Description Length rule to detect structural breaks in time series. In particular, we show that Particle Swarm Optimization typically slightly outperforms Genetic Algorithm. In addition, we show that PSO is easier to tune for the structural break detection problem than GA since the tunable hyperparameters are real numbers. We have also enriched PSO with an additional hyperparameter internally limiting the maximal number of detected intervals.

The essential contribution addressed in this paper is the new cost functions. We may summarize them as follows:

- A new function, termed Elastic Minimum Description Length (EMDL), which is an extension of the classical Minimum Description Length (MDL) cost function. It allows the user to incorporate expert knowledge into the cost function. No other cost function provides such a possibility.

- Two new cost functions based on the penalty cost function. Our extensions allow to produce better results for real-world time series by including a penalty for the structural break count, a penalty for the number of fitted model parameters, and emphasize the importance of model residuals.
- An extension applicable to all the cost functions that adds a weight controlling the minimal allowed segment length. The literature of the domain reports that the techniques available so far employ strict rules concerning the minimal segment length. We propose to relax this approach. In the paper, we demonstrate that, in consequence, we gain flexibility because we eliminate the non-intuitive additional parameter that has to be set by a user.
- An extension applicable to all the cost functions that punishes deviations from a linear trend which helps to detect shifts in data with the trend. Experiments have proven that this method works well also for data with a nonlinear trend. We show that including this extension allows detecting the exact point of trend shift (of inflection), even if the underlying theoretical time series model is not sensitive to it.

Proposed framework for structural break detection was tested using a suite of synthetic and real-world time series. In the paper, we address the properties of the proposed approach and compare it with the existing methods.

The remainder of the paper is structured as follows. Section 2 addresses relevant literature positions on the structural break detection problem. Section 3 outlines elementary notions concerning time series models and metaheuristic algorithms that we used. Section 4 addresses the approach introduced in this paper. Section 5 addresses a series of empirical experiments conducted to evaluate the quality of methods introduced in this paper. Section 8 concludes our study.

2. Literature Review on Time Series Structural Break Detection

2.1. *Earlier Papers on Time Series Structural Break Detection*

The problem of detecting structural breaks in time series is well-recognized in the literature. There are two base variants: online and offline, the use of which is determined by the practical setting in which we need to execute this task. There were many works devoted to the online variant such as these by Tartakovsky (2019) or Huang *et al.* (2020). However, the online processing is outside the scope of our interest. We are only focusing on the offline processing mode.

Offline structural break detection has its own subvariants. The most classical division of available methods is into two groups. The first group aims at finding at most one change-point (AMOC) detection, where the aim is to state if a time series contains one structural break and, subsequently, where it is located. The popular methods include cumulative sums (CUSUM) (Yang and Zhang, 2022), (SCUSUM) (Shi *et al.*, 2022b) and likelihood ratio tests (LRT) (Bai *et al.*, 2024). Nonetheless, in this article we will focus on multiple structural breaks detection problem.

We can also divide the existing structural break detection methods according to how the structural break is defined and found. The most straightforward approach is a canonical

segmentation problem, where the only parameter analysed during breakpoint detection is time series mean. Numerous works have been dedicated to this approach, and they have been thoroughly discussed by Cho and Kirch (2024). The more advanced approaches consider changes in time series variance or model data with the use of one of the common time series modelling methods. When it comes to the latter aspect, we shall underline that different models may be used for this purpose, for example, the autoregressive model (AR) (Behrendt, 2021) or the GARCH model (Cho and Korkas, 2022). Many papers are dedicated to simple and very narrow problem modelling. For instance, Romano *et al.* (2022) use AR(1) model.

This article is devoted to multiple structural break detection in an offline manner, so let us describe analogous leading and popular methods used for the task. As Shi *et al.* (2022b) say, we can distinguish two categories of methods for solving the task. First are methods using recursive partitioning of a time series into smaller segments or other approaches utilizing AMOC methods. Second are methods analysing the entire time series.

A distinct category can be assigned to the methods which sequentially analyse time series and are able to increasingly add new structural breaks, such as the method proposed by Bai and Perron (1998), which is still recognized as the baseline method in economics (for example, Lee *et al.*, 2022). This approach, commonly known as the Bai-Perron method will be mentioned later in this paper.

The first mentioned family of approaches (recursive one) generally uses AMOC methods to detect one structural breaks and later run the same AMOC method for chunked data. The base method in the area is binary segmentation (BS) (Scott and Knott, 1974). It was later extended by Fryzlewicz (2014), who proposed wild binary segmentation (WBS). There are still works in this direction and new modified variants of binary segmentation were being proposed, such as WBS2 (Fryzlewicz, 2020) or seeded binary segmentation (SBS) by Kovács *et al.* (2023). Another interesting approach of this kind is Classification Score Profile (ClaSP) presented by Ermshaus *et al.* (2023).

Many different approaches were proposed that can be described as general methods that jointly analyse entire time series. Some are using Bayesian learning or other probability-theory-based techniques for changepoint detection. In this line of research, the algorithms typically employ recursion to partition the data into segments. We envision that the changepoints of interest separate these segments. Bayesian approaches use different methods to solve the problem. Another technique was delivered by Li *et al.* (2019a) using a linear regression model, AR in particular. Yet other studies use the Markov chain Monte Carlo sampling, specifically the Metropolis-Hastings algorithm, to search for structural breaks. Shaochuan (2020) studied the use of hidden Markov models with continuous-time FFBS and continuous-time Viterbi algorithm. Casini and Perron (2024) proposed a Laplace-based (Quasi-Bayes) procedure utilizing the Monte Carlo method.

Another notable way to solve the problem of locating multiple structural breaks in a time series is Cumulative Sums of Squares (CUSUM), studied by Inclan and Tiao (1994). Among more recent studies considering the CUSUM approach, we can mention the works of Madrid Padilla *et al.* (2022), Kim *et al.* (2022) or Borzykh and Yazykov (2020). The CUSUM approach passes through the time series to search for points where the sum of squares of previous and present values changes.

Other approach to the problem was moving sum (MOSUM) procedure. Their method is essential in canonical segmentation problem. It was recently used, for instance, by Cho and Kirch (2022) or Meier *et al.* (2021). MOSUM was utilized also for more advanced segmentation problems such as in the work of Kirch and Reckruehm (2024).

It is vital to notice that many of known approaches use constraint on the minimum distance between two consecutive structural changes. The approach is very popular and was used, for example, by Davis *et al.* (2016), Doerr *et al.* (2017) or Yan *et al.* (2021).

As a side note, we have observed a growing number of papers elaborating on the benefits of including a structural break detection step in time series forecasting procedures. Among researchers focusing on this aspect, we find Smith (2023), Safikhani *et al.* (2022), and Altansukh and Osborn (2022). Another substantive stream of studies emerged when the existing changepoint detection methods were adapted to temporal panel data, as indicated by Bardwell *et al.* (2019).

2.2. *Evolutionary Methods in Structural Break Detection Problem*

The last group of methods for solving the task we want to discuss are evolutionary algorithms. In particular, it is worth mentioning the works of Davis *et al.* (2016, 2006, 2008), Davis and Yau (2013), who conducted several studies on the application of the genetic algorithm in the discussed context. They have referred to their new method as Auto-PARM. The chromosome in this algorithm contained information about whether a structural change was present at a given location. The method uses relatively simple mutation, various crossover variants, and proportional selection. In addition, the island model of the genetic algorithm was employed to improve the results.

The genetic algorithm was also considered in the works of Doerr *et al.* (2017). As already mentioned, their algorithm was constructed for long time series up to one million data points. The results of this study indicated that using both single-point and uniform crossover simultaneously improves the quality of the results. Furthermore, the effectiveness of enhancing the algorithm by using multi-start and champions league strategies was noted. A similar outcome was documented by Suárez-Sierra *et al.* (2023). On the other hand, Lim *et al.* (2020) presented a memetic algorithm for multivariate time series segmentation dedicated to long time series.

Éltetö *et al.* (2012) proposed using the highly acclaimed CMA-ES (Covariance Matrix Adaptation Evolution Strategy) algorithm. They divided their method into two steps. The first step aims at finding the number of structural changes. The second step optimizes the locations of each structural change. They did this by analysing solutions for different, algorithmically chosen numbers of breakpoints. CMA-ES operates in a continuous space, while the change point detection problem is embedded in a discrete space. Therefore, it was necessary to move the problem to another space, which was done by processing the structural change locations and rounding them for solution evaluation.

2.3. *The Issue of the Target Function Definition*

Many of the methods used for structural break detection require an objective function that allows comparing particular structural change positions. Choosing an appropriate objec-

tive function is not easy because of the need to simultaneously fit the data model and minimize the number of model parameters, as state (Shi *et al.*, 2022b). Different objective functions were used in the literature. Some were very simple and failed to reflect the nuances of model fitting. This was due to various reasons. In some cases, the computational complexity was decisive, as it was in the approach of Doerr *et al.* (2017).

Penalty function methods have been analysed as well. For example, Hall *et al.* (2013) used a penalty function in which three parameters of a time series segment model corresponded to one structural change. Another variant of the penalty function was considered by Doerr *et al.* (2017), whose penalty function was the sum of the penalty for the number of detected breakpoints and the penalty for the fitting error calculated separately for each segment of the time series as the area of a rectangle whose sides were: the beginning and the end of the studied segment, and the minimum and maximum value of the series on the given segment.

For more complex approaches, a fairly obvious choice was to check known information criterion models. Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), or modified Bayesian Information Criterion (mBIC) can be employed (Zhang and Siegmund, 2007). An intensive comparison of these criteria and MDL (see next paragraph) was delivered by Shi *et al.* (2022b).

In recent years, the principle of Minimum Description Length (MDL) formulated by Rissanen (1978) has gained substantial popularity in this field. It was applied to the problem of structural change detection by Davis *et al.* (2006). This solution was originally adapted for use together with the time series autoregressive (AR) model. Later works made it possible to transfer this method to other time series models: Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) (Davis *et al.*, 2008), periodic AR (Lu *et al.*, 2010) or AutoRegressive Moving Average (ARMA) (Davis and Yau, 2013). As mentioned, Li *et al.* (2019b) formulated a modification of the MDL method known as the Bayesian Minimum Description Length rule. The cost function based on the MDL principle was later used in many different works, such as by Ditzen *et al.* (2021) or Lan *et al.* (2022). A modification of the MDL rule was also shown in Bayesian Minimum Description Length (BMDL) method proposed by Li *et al.* (2019a).

3. Preliminaries

In this section, we provide basic information related to the background knowledge behind the proposed approach.

3.1. *ARIMA for Time Series Modelling*

Time series can be simply described as a sequence of values observed in time. There are many different ways to model time series. One of the very popular is the use of models from the AutoRegressive Integrated Moving Average (ARIMA) family. The simplest model belonging to this family is the autoregressive model (AR), which can be written as

$$y_t = c + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \cdots + \beta_p y_{t-p} + \varepsilon_t, \tag{1}$$

where:

- $c, \beta_1, \ldots, \beta_p$ – model parameters,
- ε_t – white noise,
- p – order of the AR model.

The second important model is moving average (MA), given as

$$y_t = c + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} + \cdots + \phi_q \varepsilon_{t-q} + \varepsilon_t, \tag{2}$$

where:

- $c, \phi_1, \ldots, \phi_q$ – model parameters,
- q – order of the MA model.

If we join the AR and MA models, we get the ARMA model. If we add a differencing operation to the model, we get the ARIMA model. The differencing operation is performed on time series d times, where d usually equals 0, 1, or 2. Thus, by putting together the described components of ARIMA, we obtain the following model formulation:

$$y_t = c + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \cdots + \beta_p y_{t-p} + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} + \cdots + \phi_q \varepsilon_{t-q} + \varepsilon_t. \tag{3}$$

The model can be briefly expressed as ARIMA(p, d, q), where p is the order of the autoregressive part, d is the degree of differencing, and q is the order of the moving average part.

A time series structural break is a place in the time series where data changes its characteristics. It can be a change in the time series mean, variance, or a point where a trend has emerged or disappeared. The most often-mentioned definition says that it is a place in the time series where the time series model or one of its parameters changes (for example, Cheng *et al.*, 2022). Obviously, time series can have several structural breaks or not have them at all.

3.2. *Metaheuristic Algorithms*

In this work, we use metaheuristic optimization-based algorithms to solve the problem of time series structural break detection. We selected Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA). The algorithms mentioned are all metaheuristics, which means that they do not guarantee that an optimal solution will be found. However, they can find a good-enough solution in a relatively short time. This property makes them indispensable in many engineering applications, see Abdel-Basset *et al.* (2018).

ACO, PSO, and GA are based on an analogous idea that the algorithm operates on a collection of candidate solutions called population. The procedure is iterative. In each iteration, the population undergoes specific modifications, and in this manner, candidate solutions are constructed or modified. An essential part of these algorithms is a cost function that evaluates how good a particular candidate solution is. Since these are heuristic optimization approaches, termination criteria concern either a count of performed iterations or a count of iterations without any improvement.

3.3. *Ant Colony Optimization*

ACO is a method proposed and developed by Dorigo and Stützle (2019). It is patterned on the behaviour of ants. The population of the algorithm is interpreted as a swarm of ants that create solutions by walking on a graph. All solutions found in a given iteration are evaluated, and ants put pheromones on the edges of a graph used in these solutions according to their fitness.

Ants construct solutions in a probabilistic, step-wise manner. If we have an iteration t and an ant, which built a part of a solution and is in a vertex i, the probability of the ant's move from the ith to the jth vertex is given as

$$p_{ij}^{k}(t) = \begin{cases} \frac{(\tau_{ij}(t))^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{a \in N} (\tau_{ia}(t))^{\alpha} \cdot \eta_{ia}^{\beta}}, & j \in N, \\ 0, & j \notin N, \end{cases} \tag{4}$$

where:

- $\tau_{ij}(t)$ – pheromone count between vertices i and j at the moment t,
- η_{ij} – heuristic cost of i–j edge,
- N – neighbourhood of the vertex i,
- α, β – algorithm parameters.

It is important to note that the ACO algorithm works in a discrete space. It is perfect for graph-based problem representations.

The general scheme of ACO is presented in Algorithm 1.

Algorithm 1 General schema of ACO

```
Initialize pheromone τ_ij(1), i = 1, ..., n.
while stop condition is not fulfilled do
    for each ant k = 1, ..., m do
        Set ant's path to empty set: S_k(t) = ∅.
        for all decision points i = 1, ..., n do
            Select an edge according to Eq. (4).
            Add the selected edge to the ant's path S_k(t).
        end for
    end for
    Evaluate fitness function for S_k, for k = 1, ..., m.
    Update pheromone paths.
end while
```

3.4. *Particle Swarm Optimization*

PSO is patterned on the behaviour of birds' flocks or schools of fish. In contrast to ACO, the algorithm operates in a continuous space. The population is interpreted as a collection

of particles. Each particle has its own location, memory, and velocity. The location of a particle i is denoted as x_i, the velocity is denoted as v_i, the best solution found by the particle is denoted as s_i, and the best global solution is denoted as s_b. During each iteration the velocity and location of each particle are modified according to the formula:

$$v_i(k) = \omega v_i(k-1) + c_1 r_1 \big[s_i(k-1) - x_i(k-1)\big] + c_2 r_2 \big[s_b(k-1) - x_i(k-1)\big], \quad (5)$$

where:

- ω – inertia coefficient,
- c_1 – best local solution influence coefficient,
- c_2 – best global solution coefficient,
- r_1, r_2 – random numbers, where $r_1, r_2 \in [0, 1]$, they can be different for each iteration and each particle.

The location of a particle is updated according to:

$$x_i(k) = x_i(k-1) + v_i(k). \quad (6)$$

The general flow of the PSO is presented in Algorithm 2.

Algorithm 2 General schema of PSO

```
Initialize the location and velocity of each particle.
Initialize s_i to null value (best solution).
while stop condition is not fulfilled do
    for each particle i do
        Evaluate the fitness value of particle i.
        if the fitness value of particle i is better than s_b then
            Set current fitness as s_i.
        end if
    end for
    Choose the particle with s_i as s_b (best particle).
    for each particle i do
        Update particle velocity.
        Update particle location.
    end for
end while
```

3.5. *Genetic Algorithm*

GA is a long-known evolutionary algorithm that uses a population of solutions. Each solution is represented by a chromosome, which consists of genes. The population is modified

by three fundamental operations: selection, crossover, and mutation, which are performed during each iteration. These operations are defined very generally. Thus, so many variants of the GA have arisen. The first operation is selection, which aims to pick the best individuals from the population. The two most common selection mechanisms are proportional selection and tournament selection. A mutation usually makes a minor change to the solution. For example, in the popular bit flip mutation for binary encoded chromosomes, one or more bits are randomly selected and inverted. There are other types of mutations, such as swap mutation, truncation mutation, or inversion mutation. Crossover initially results in several parent solutions that combine to form one or more offspring. Popular crossover methods are uniform crossover, one-point crossover, or multi-point crossover. In many situations, applying standard mutation or crossover methods is impossible, and the need to modify them or create custom, problem-specific crossover or mutation operators arises.

The general scheme of the GA is presented in Algorithm 3.

Algorithm 3 Pseudocode of GA

```
Generate the initial population randomly.
while stop condition is not fulfilled do
    Select a subset of individuals to create a new population.
    Set the number of crossover nc operations to perform.
    for i = 1, ..., nc do
        Randomly select individuals for crossover.
        Generate a new individual by applying the crossover operation.
        Save the new individual for the new generation.
    end for
    Set the number of mutation nm operations to perform.
    for i = 1, ..., nm do
        Randomly select an individual.
        Perform mutation.
        Save the mutated individual for the new generation.
    end for
    Evaluate the new population.
end while
```

3.6. *Island Model in Heuristic Optimization*

The island model is one of the approaches to perform multi-population optimization. In the model, there are several populations located on several islands. After every t iterations, we perform a migration that allows replacing the worst solutions from one population with the best solutions from another population. The count of iterations between migrations, population size, and the count of migrated elements are parameters of the model.

The island model is a simple, yet effective method to get good results fast. It is often used with the genetic algorithm but can also be applied to other evolutionary algorithms.

3.7. *Cost Functions Used in the Literature to Optimize Structural Break Count*

Selecting a cost function to measure the quality of solutions is a critical problem in metaheuristic optimization. Constructing a suitable cost function for the structural break detection problem is difficult because we need to minimize the model errors and the number of structural breaks and balance the two.

The literature offers various solutions to this problem; more information is available from Farsi *et al.* (2020). One of the most acclaimed ones was the use of the minimum description length (MDL) rule. This rule states that a model is only as good as long as it needs little space to save its data. We can write it as

$$CL(y) = CL(M) + CL(e|M), \tag{7}$$

where CL is a code length, y is the analysed time series, M is the model describing it, and $e|M$ is the error of fitting M to y. $CL(M)$ is simple to calculate compared to $CL(e|M)$. Rissanen (1978) proposed to calculate it using the log-likelihood function:

$$CL(e|M) = -\log L(M). \tag{8}$$

Davis *et al.* (2016) introduced an adaptation of the MDL technique to the problem of structural breaks detection for the AR model:

$$\begin{aligned} &MDL\big(m, p_0, (\tau_1, p_1), \ldots, (\tau_m, p_m)\big) \\ &\quad = \log_2 m + m \log_2 n + \sum_{i=0}^{m} \log_2 p_i + \sum_{i=0}^{m} \frac{p_i + 2}{2} \log_2 n_i \\ &\qquad + \sum_{i=0}^{m} \frac{n_i + n_i \log_2(2\pi\sigma_i^2)}{2}, \end{aligned} \tag{9}$$

where:

- m – structural change count,
- n – time series length,
- $\tau_1, \ldots, \tau_m$ – structural change locations,
- $p_1, \ldots, p_m$ – orders of AR model on successive segments,
- σ_i^2 – variation of model error on segment number i.

4. Proposed Approach

Let us recall the key contributions of this paper:

- a generalization of Davis's approach to the MDL with the ARIMA model,
- new cost functions: Elastic MDL – a variant of MDL giving more flexibility to the user and new functions based on penalty cost suitable to process real-world time series,

- including a weight for punishing (controlling) minimal allowed interval length,
- an extension of the cost function that takes into account deviations from a linear trend observed on time series segments,
- new cost functions using relations between segment's order and length,
- in this study, PSO and ACO algorithms, in conjunction with the Minimum Description Length rule, are used for the first time in the task of structural break detection in time series.

In the following subsections, we present them in detail.

4.1. *Generalization of the Approach: Using the New MDL Function with the ARIMA Model*

Let us define the MDL function for the ARIMA model based on Eqs. (7) and (8) as

$$\begin{aligned} &MDL\big(m, (\tau_1, p_1, q_1), \ldots, (\tau_m, p_m, q_m)\big) \\ &\quad = \log_2 m + m \log_2 n + \sum_{i=0}^{m} \log_2 (p_i + d_i + q_i) \\ &\qquad + \sum_{i=0}^{m} \frac{p_i + d_i + q_i + 2}{2} \log_2 n_i - \max_i \log L(M_i), \end{aligned} \tag{10}$$

where $\log L(M_i)$ is the log-likelihood function.

The defined function in Eq. (10) relies on the addition of four components. The first component ($\log_2 m$) represents the number of structural breaks. It is a natural number. The logarithm is used to represent this value in terms of bits, according to the idea of the original MDL function. The second component is $m \log_2 n$. It represents the space needed to save the location of each structural break. Each location is kept as a natural number that is not larger than the series length n. The third component represents the ARIMA specification (p, d, q) for each segment in the time series. The final component keeps the values of ARIMA model parameters for each segment. These values are computed based on n_i elements.

4.2. *Introducing New Cost Functions for the Optimization Problems Dealing with Structural Break Detection*

In this subsection, we address new cost functions. Each of them was defined in two variants – for the AR and ARIMA model.

4.2.1. *Penalty Function, Modification #1*

This extension relies on penalties for structural breaks count, model parameter count, and variation of model fitting error. The version for the AR model is defined as:

$$SPF1_{AR}\big(m, (\tau_1, p_1), \ldots, (\tau_m, p_m)\big) = \mu m^2 + \sum_{i=1}^{m} \big(\sigma_i^2 + \nu p_i^2\big), \tag{11}$$

where

- μ – parameter describing structural change count impact on model choice,
- ν – parameter describing parameter count impact on model choice.

The cost function proposed for the ARIMA model is

$$SPF1_{ARIMA}\big(m, (\tau_1, p_1, d_1, q_1), \ldots, (\tau_m, p_m, d_m, q_m)\big) = \mu m^2 + \sum_{i=1}^{m} \big(s_i + \nu(p_i + \rho d_i + q_i)^2\big), \tag{12}$$

where ρ describes the differencing count impact on model choice, and s_i is a sum of the squared estimate of errors (SSE) for the segment fitting.

4.2.2. *Penalty Function, Modification #2*
We defined the second cost function to give residual errors higher importance. The variant delivered for the AR model looks as follows:

$$SPF2_{AR}(m, \big(\tau_1, p_1, \ldots, (\tau_m, p_m)\big) = \mu m^2 + \sum_{i=1}^{m} \big(\sigma_i^4 + \nu p_i^2\big), \tag{13}$$

while for the ARIMA model it is

$$SPF2_{ARIMA}\big(m, (\tau_1, p_1, d_1, q_1), \ldots, (\tau_m, p_m, d_m, q_m)\big) = \mu m^2 + \sum_{i=1}^{m} \big(s_i^2 + \nu(p_i + \rho d_i + q_i)^2\big). \tag{14}$$

The rationale was to partition the time series such that the segments are as long as possible, but short enough to fit the theoretical model well. The shorter the segment, the easier it is to fit a theoretical model. At the same time, for practical reasons, too many segments may not be helpful.

These conflicting goals (long segments with good theoretical models) were coded differently by means of different objective functions.

4.2.3. *Elastic MDL*
The third idea was to use the MDL function to give users more flexibility and allow them to tune it to their preferences. The version for the AR model is defined as:

$$EMDL_{AR}\big(m, p_0, (\tau_1, p_1), \ldots, (\tau_m, p_m)\big) = \log_2 m + m \log_2 n + \mu \sum_{i=0}^{m} \left(\log_2 p_i + \frac{p_i + 2}{2} \log_2 n_i \right) + \nu \sum_{i=0}^{m} \frac{n_i + n_i \log_2(2\pi\sigma_i^2)}{2}, \tag{15}$$

and for ARIMA model as:

$$EMDL_{ARIMA}\big(m, (\tau_1, p_1, q_1), \ldots, (\tau_m, p_m, q_m)\big) = \log_2 m + m \log_2 n + \mu \sum_{i=0}^{m} \log_2(p_i + d_i + q_i) + \sum_{i=0}^{m} \frac{p_i + d_i + q_i + 2}{2} \log_2 n_i - \nu \max_i \log L(M_i). \tag{16}$$

4.2.4. *Division Cost Function*
The following cost function is based on the assumption that structural break selection is satisfactory when the variance of residual errors is small, and the orders of assigned models are small compared to the adjusted segment's length. The version for AR was defined as

$$DCF_{AR}\big(m, p_0, (\tau_1, p_1), \ldots, (\tau_m, p_m)\big) = \log_2 m + \sum_{i=0}^{m} \log_2\left(\mu\left(\frac{p_i}{n_i}\right)^{\delta} + 1\right) + \nu \sum_{i=0}^{m} \log_2\big(\sigma_i^2\big), \tag{17}$$

where δ is a parameter. The formula for ARIMA is defined as

$$DCF_{ARIMA}\big(m, (\tau_1, p_1, q_1), \ldots, (\tau_m, p_m, q_m)\big) = \log_2 m + \sum_{i=0}^{m} \left[\log_2\left(\left(\mu \frac{p_i + d_i + q_i}{n_i}\right)^{\delta} + 1\right) - \nu \max_i \log L(M_i)\right]. \tag{18}$$

4.2.5. *Penalty-Division Cost Function*
The final cost function is a combination of penalty and division cost functions. We defined it for the AR model as

$$PDCF_{AR}\big(m, p_0, (\tau_1, p_1), \ldots, (\tau_m, p_m)\big) = m^2 + \sum_{i=0}^{m} \left(\mu\left(\frac{p_i}{n_i}\right)^{\delta} + 1 + \nu \log_2\big(\sigma_i^2\big)\right), \tag{19}$$

and for the ARIMA model as

$$PDCF_{ARIMA}\big(m, (\tau_1, p_1, q_1), \ldots, (\tau_m, p_m, q_m)\big) = m^2 + \sum_{i=0}^{m} \left(\mu\left(\frac{p_i + d_i + q_i}{n_i}\right)^{\delta} - \nu \max_i \log L(M_i)\right). \tag{20}$$

Comparing this cost function with the plain MDL (see Eq. (9)), the key difference is that it includes the possibility of influencing the outcome by leveraging the introduced parameters. It can be particularly beneficial when the user has domain-specific knowledge.

4.3. *Extensions of Cost Functions Used to Solve Structural Break Detection Problem*

We proposed two extensions to the cost functions that can increase their capabilities.

The first extension is a weight for enforcing minimal allowed interval length. A common pitfall of structural break detection algorithms is that they find too short, insignificant time series segments. So far, the problem has mostly been solved by inserting a strict ban on generating time series segments shorter than a certain threshold. Such a tough, binary solution is not clear and justified. Thus, we propose a new attempt to solve this problem. We suggest adding to cost functions a penalty term defined as follows:

$$PFS(s_i) = \begin{cases} (b - n_i)^2, & n_i < b, \\ 0, & n_i \geqslant b, \end{cases} \tag{21}$$

where:

- $PFS(s_i)$ – function value for a segment s_i,
- n_i – length of s_i segment,
- b – a threshold.

The use of weight in the cost function gives more flexibility than the use of a strict threshold.

The second extension was dedicated to the methods that do not account for trends. We propose to solve this deficiency by adding a weight to a cost function to estimate the deviation from a linear trend. We can define it as

$$MSE(s_i) = \sum_{j=0}^{n_i - 1} (s_{ij} - aj - b), \tag{22}$$

where

- s_{ij} – j-th element of i-th time series segment,
- a, b – parameters of resulting linear regression applied to the segment.

Both extensions can be applied with the use of additional parameters, giving them suitable impact.

4.4. *New Adaptations of Metaheuristic Algorithms to the Problem of Structural Break Detection*

4.4.1. *Ant Colony Optimization Adaptations*

To implement the ACO algorithm for the structural break detection problem, we have modified the rule of ranking possible edges from that presented in Eq. (4) to the form:

$$p_{ij}^k(t) = \begin{cases} \frac{(\tau_{ij}(t))^\alpha * c_{ij}^\beta * d_{ij}^\gamma}{\sum_{a \in N} (\tau_{ia}(t))^\alpha * c_{ia}^\beta * d_{ij}^\gamma}, & j \in N, \\ 0, & j \notin N, \end{cases} \tag{23}$$

where:

- c_{ij} – cost function value for the segment that starts at the i-th time series element and ends at the j-th element, calculated with a given cost function,
- d_{ij} – count of structural breaks in solution,
- β, γ are parameters.

The crucial issue is that we shall know how many structural breaks will occur after choosing j. We can not specify it exactly, since we select the next structural break location with some randomness, but we can estimate it. Let us denote the count of time series segments generated after the k-th time series element as $v(k)$. Let us use the letter n to denote the length of the time series. The $v(k)$ function is defined for k in the range from 0 to $n+1$, where the letter is the element after the last element of the time series. It is apparent that $v(n+1) = 0$. Similarly, $v(n) = 1$. Later, after the $(n-1)$-th element, we can add a structural break at the data point in n or cease adding more structural breaks. We can assume that these two events are equally probable. So, $E[v(n-1)] = 1+\frac{1}{2}E[v(n)]+\frac{1}{2}E[v(n+1)]$. This way, we can formulate the method to estimate $v(k)$ value.

$$E\big[v(k)\big] = \begin{cases} 0, & k = n+1, \\ 1 + \frac{1}{(n+1)-k} \sum_{i=k+1}^{n+1} E[v(i)], & k < n+1. \end{cases} \tag{24}$$

The method allows estimating the count of structural breaks after the k-th time series element in time $O(n)$.

4.4.2. *Particle Swarm Optimization Adaptations*

Since PSO is an algorithm operating in a continuous space, its application to the structural break detection problem requires adaptations. We have solved this problem by creating an upper limit on the produced structural breaks (k) count. A solution was represented as a list of k structural break locations. A similar solution to the problem was presented by Éltetö *et al.* (2012), but they restricted that elements in the array will always be different and sorted. We did not add such a limitation. It gave us the advantage that our PSO implementation could detect fewer structural breaks, even zero.

4.4.3. *Genetic Algorithm Adaptation*

We used the traditional genetic algorithm formulation with tournament selection, with two-point crossover. We used mutation similar to bit flip mutation, where each bit described if an i-th point of time series is a structural break. Still, there was a minor modification because the probability to change 0 to 1 and 1 to 0 could not be equal – we proposed a parameter which was promoting mutations that removed structural breaks and discouraging mutations that were adding structural breaks.

4.5. *Time Complexity of Analysed Algorithms*

While designing the new solutions, we paid a lot of attention to the issue of computational complexity. It was critical since the problem at hand involved intensive computations. The

algorithms we used needed information about cost function value linked with time series segments. The information could be calculated dynamically during the computation of a structural break or eager (earlier) before the computation. The second strategy seems more straightforward as we divide our task into two sub-tasks: computation of a cost function for all possible segments and finding structural breaks. It is viable only for short time series because executing it for long time series would induce many redundant calculations.

Let us address in more detail the computational complexity of finding structural breaks. We will mark population size as P, iteration count as N, length of time series as n, and time needed to get cost function for the segment as C. If we use precalculated values for it, C is constant. In PSO, two procedures: modification of the solution and its evaluation, are performed in each iteration. The time complexity of modification is proportional to the maximum count of structural breaks in the solution, which is a parameter of the algorithm, and we will denote it as K. So, the time complexity of evaluation is proportional to the product of K and C. Thus, the time complexity of PSO can be expressed as:

$$t_{PSO} = O(P \cdot N \cdot K \cdot C). \tag{25}$$

In each iteration, GA performs selection, mutation, crossover, and evaluation. The time complexity of two-point crossover and mutation is $O(n)$ and the time complexity of tournament selection is $O(T)$, where T is a tournament size. The time series evaluation cost is $K \cdot C$. However, K is not constant, but it is a number not larger than n. Thus, the time complexity of GA is given as:

$$t_{GA} = O\big(P \cdot N \cdot (T + n \cdot C)\big). \tag{26}$$

ACO performs three operations: construction of a new solution, evaluation, and updating pheromone trace in each iteration. During solution construction, an ant calculates probabilities and needs knowledge of cost function at analysed time series segments, which is done in $O(n \cdot C)$ time. The maximum count of detected structural breaks is $O(n)$. Thus, the construction of a single solution is calculated in $O(n^2 \cdot C)$ time. The solution evaluation is in $O(n \cdot C)$ time. The update of the pheromone trace is made for all ants together and has three important parts:

1. Sorting solutions calculated by ants in time $O(P^2)$.
2. Pheromone evaporation in $O(n^2)$ time.
3. Putting new pheromone int time $O(B \cdot n)$, where B is the count of ants putting pheromone (we know that $B < P$).

Thus, the pessimistic complexity of the ant algorithm is given as:

$$t_{ACO} = O\big(N \cdot P \cdot n^2 \cdot C + N \cdot \big(P^2 + n^2 + B \cdot n\big)\big) = O\big(P \cdot N \cdot \big(n^2 \cdot C + P\big)\big). \tag{27}$$

5. Experimental Evaluation

5.1. *Experiment Methodology*

In this section, we present the experiments' outcomes to validate the quality of the proposed novel methods. The scope of the study covered both artificial and real-world time series representing different situations. Most of the analysed time series were relatively short, since conventional structural break detection tasks are executed for such cases. We analysed eight synthesized time series, which included instances with different numbers of structural breaks, appearing and disappearing trends, and changing time series mean and variance. There was also an example of a white noise time series with added variation, in which the algorithms should not detect any structural breaks. We have defined "test cases" of synthetic time series. We assumed some underlying time series properties for each test case and then generated each test case multiple times, maintaining the same underlying properties but with some random noise. This entailed that the experiments were repeated multiple times for each test case. Specific differences between particular time series instances within a single "case" arise only because we draw values, but the parameters of distributions we use stay the same.

On top of ten synthetic series, we used two real-world time series representing the power demand of a fridge freezer in a kitchen. They come from the Electrical Load Measurement dataset by Murray *et al.* (2015).[1] Example time series from the synthetic test cases are displayed in Fig. 1. Processed real-world time series are illustrated in Fig. 2.

Let us list and briefly discuss synthetic test cases:

1. Time series composed of two segments with the same segment variance but different segment mean (an example is given in Fig. 1(a)).
2. Time series composed of two segments with the same mean and different segment variance (an example is given in Fig. 1(b)).
3. Time series composed of two segments generated with the use of different AR models ($y_t = 0.9y_{t-1} + \varepsilon_t$ and $y_t = -0.9y_{t-1} + \varepsilon_t$). In the first three examples, two segments have equal length (like in Fig. 1(c)).
4. Test case without structural breaks. It is a white noise series with a small random distortion (example is in Fig. 1(d)).
5. Test case where time series start with a long and low-variance segment, but at the end there is a short, but intense, rise of values (example is in Fig. 1(e)).
6. Time series composed of three segments, where the first and the third segments are identical and have a fixed mean and variance. The segment in the middle has a larger variance than the other two segments (see Fig. 1(f)).
7. Test case, which follows a parabolic pattern with low variance of data points. An example is given in Fig. 1(g)).

[1] The dataset is available at https://www.timeseriesclassification.com/description.php?Dataset=FreezerRegularTrain

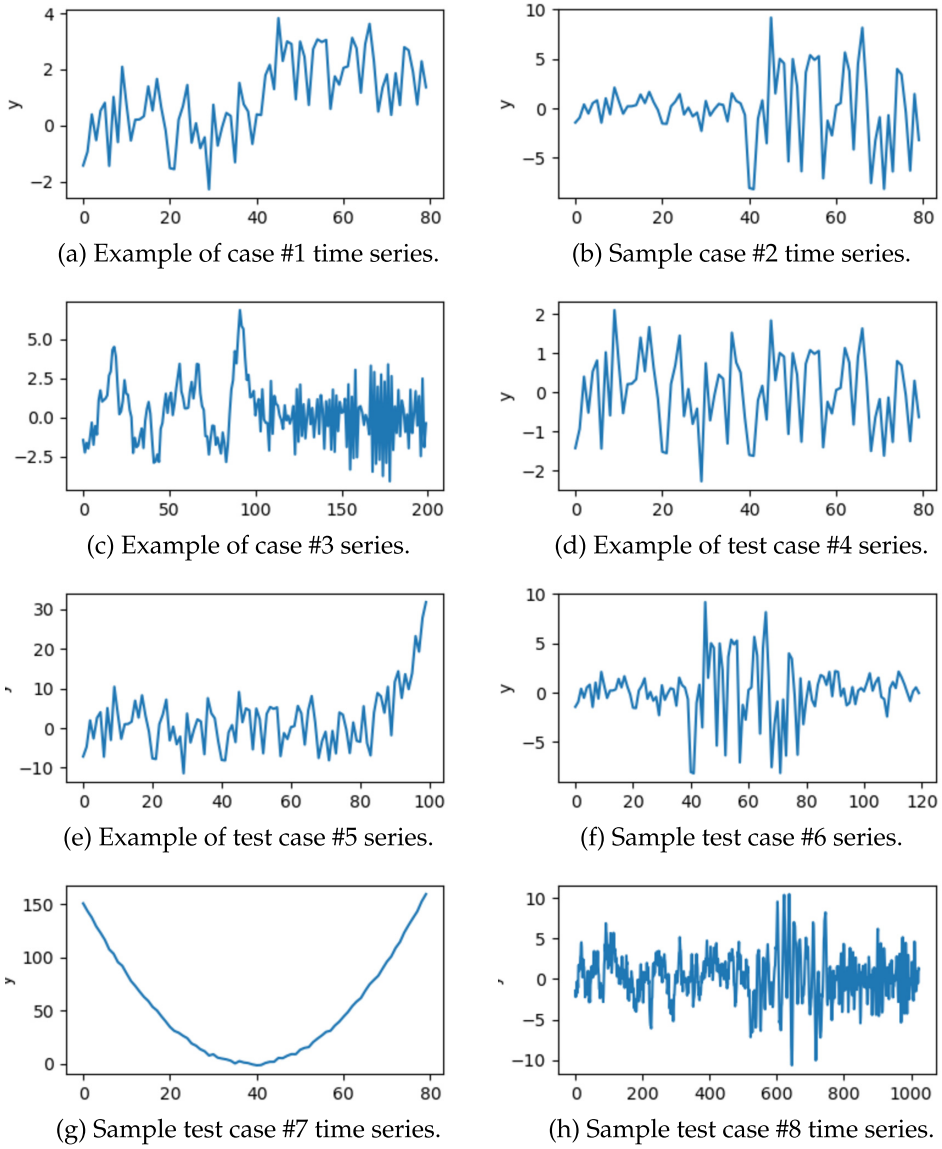
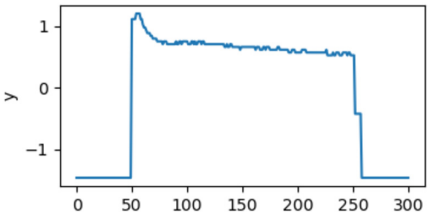


Fig. 1. Example time series coming from artificial test cases analysed in the paper.

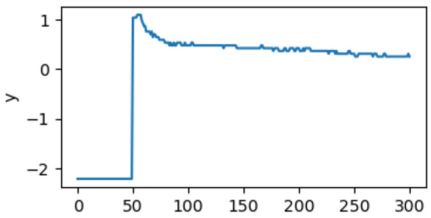
8. Time series generated using the AR model based on the equation:

$$y_t = \begin{cases} 0.9y_{t-1} + \varepsilon_t, & 0 \leqslant t \leqslant 512, \\ 1.69y_{t-1} - 0.81y_{t-2} + \varepsilon_t, & 512 \leqslant t < 768, \\ 1.32y_{t-1} - 0.81y_{t-2} + \varepsilon_t, & 768 \leqslant t < 1024 \end{cases} \tag{28}$$

(example is in Fig. 1(h)).



(a) Example of test case #9 series.



(b) Example of test case #10 series.

Fig. 2. Two real-world time series describing fridge freezer power demand.

To eliminate the possible bias arising due to the randomness of the empirical methodology of this study, we have been performing multiple repetitions of the experiments. For each synthesized test case, we generated 10 time series instances with different random number generator seeds. Experiments were executed for each generated time series. As a result, we performed 10 repetitions of each experiment for each test case. Our motivation was to ensure a reasonable variability of the data by maintaining the same data generation process that relies on drawing from distributions. To keep the same level of objectivity in result evaluation, in the case of the real-world time series, we were also repeating the experiments 10 times.

The outcomes of the experiments that were conducted involving different algorithms were compared with each other and with the outcomes obtained by other researchers. We assume three state-of-the-art methods, which we use for such comparisons. The first is the method proposed by Bai and Perron (1998), Bai and Perron (2003), and the second is ClaSP by Ermshaus *et al.* (2023). The third is the method by Davis *et al.* (2016), which can be seen as the immediate predecessor of the approach introduced in this paper. In the first two cases, we run the algorithm to produce the results, but in the third case, we compare our results to those presented by Davis *et al.* (2006) using analogous time series examples.

We propose two metrics for the evaluation of the outcomes of structural break detection procedures:

- break point difference (BPD) – absolute value of the difference between the found and correct count of structural breaks,
- score value (SC):

$$SC = \sqrt{k * BPD^2 + \max(t_m - m, 0) \cdot \frac{1}{4} + \sum_{i=0}^{m} \left(\frac{x_i - c(x_i)}{n} \right)^2}, \tag{29}$$

where:

- k is a user parameter, which in our experiments was set to 4 based on our experience with this measure,
- t_m is the target count of structural breaks,
- x_i are locations of consecutive structural breaks,
- $c(x_i)$ is a function that returns location of correct structural break closest to x_i.

The proposed method of results evaluation is our novel contribution, which is introduced in this paper.

5.2. *Experimental Evaluation of Analysed Cost Functions*

At first, we compared results obtained with the MDL, SPF1, SPF2, EMDL, DCF, and PDCF cost functions. We used both the AR and ARIMA models and both versions of the cost function: with and without a weight for deviation from a linear trend. The experiments concerned seven synthetic time series and two real-world time series. The only omitted case is Fig. 1(h), which will be discussed in detail in Section 7.

Experiments were performed using PSO with parameters C_{11}: $\omega = -0.1832$, $c_1 = 0.5287$, $c_2 = 3.1913$, population size 47. These parameters come from Hvass Pedersen (2010). We have tested 188 variants of cost functions: 164 for the AR model and 22 for the ARIMA model. In principle, we have employed a grid search of procedure parameters for SPF1, SPF2, EMDL, DCF, and PDCF functions with the AR model. In the case of ARIMA, we have performed fewer tests due to a more considerable computational complexity – we tested a subset of configurations close to the configurations, for which results for the corresponding functions with the AR model were good.

A detailed discussion on optimization algorithms hyperparameters and their fitting is provided in Section 6.

Our results were evaluated with the use of BPD and SC metrics and conclusions attained with use of both were very similar. The best results (assessed with the use of BPD and SC metrics) were obtained consecutively for configurations: EMDL_{AR} ($\mu = 3$, $\nu = 1$), EMDL_{AR} ($\mu = 3$, $\nu = 1$), $\text{MDL}_{\text{ARIMA}}$, all without involvement of the MSE modification. There were also other cost functions which led to noteworthy results. The most interesting configurations are presented in Table 1 and Table 2.

Let us discuss the behaviour of various cost functions in more detail. For the simplest cases, analysed functions perform very well. For test case 1(a) most of the cost functions in almost each run correctly detect places where the average value of time series changes.

Table 1
BPD score for the best-performing cost functions.

Cost function		Test case									
	MSE	1(a)	1(b)	1(c)	1(d)	1(e)	1(f)	1(g)	2(a)	2(b)	Mean
$\text{EMDL}_{\text{ARIMA}}$ ($\mu = 3$, $\nu = 1$)	–	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.1	0.3	0.27
EMDL_{AR} ($\mu = 3$, $\nu = 1$)	–	0.0	0.0	0.0	0.0	0.0	0.1	1.0	1.2	0.3	0.29
$\text{MDL}_{\text{ARIMA}}$	–	0.1	0.0	0.0	0.0	0.0	0.1	1.0	1.3	0.2	0.30
MDL_{AR}	–	0.0	0.0	0.0	0.0	0.0	0.0	2.5	1.2	0.5	0.47
$\text{DCF}_{\text{ARIMA}}$ ($\mu = 1$, $\nu = 0.1$, $\delta = 0.5$)	–	0.0	0.0	0.0	0.9	0.0	0.0	0.8	2.0	1.0	0.52
$\text{PDCF}_{\text{ARIMA}}$ ($\mu = 10$, $\nu = 0.1$, $\delta = 2$)	–	0.0	0.0	0.0	1.0	0.0	0.0	0.7	2.0	1.0	0.52
SPF_{AR} ($\mu = 0.1$, $\nu = 1$)	–	0.8	1.0	0.3	0.0	1.0	2.0	0.4	0.5	0.1	0.68
SPF2_{AR} ($\mu = 0.1$, $\nu = 10$)	–	0.0	1.0	0.2	0.0	0.5	2.0	2.8	0.3	0.0	0.76
SPF_{AR} ($\mu = 10$, $\nu = 10$)	+	0.2	0.7	0.8	0.1	1.4	0.4	3.7	0.0	0.0	0.81
SPF2_{AR} ($\mu = 10$, $\nu = 1$)	+	0.1	0.9	0.8	0.1	1.0	0.8	3.9	0.0	0.1	0.86

Table 2
SC score for the best-performing cost functions.

Cost function		Test case									
Type	MSE	1(a)	1(b)	1(c)	1(d)	1(e)	1(f)	1(g)	2(a)	2(b)	Mean
$EMDL_{ARIMA}$ ($\mu = 3, \nu = 1$)	–	0.019	0.001	0.007	0.000	0.013	0.004	2.062	2.265	1.203	0.619
$EMDL_{AR}$ ($\mu = 3, \nu = 1$)	–	0.006	0.001	0.009	0.000	0.011	0.224	2.036	2.465	1.181	0.659
MDL_{ARIMA}	–	0.225	0.001	0.007	0.000	0.028	0.218	2.062	2.673	1.051	0.659
MDL_{AR}	–	0.019	0.001	0.009	0.000	0.006	0.013	5.023	2.465	1.432	0.996
DCF_{ARIMA} ($\mu = 1, \nu = 0.1\ \delta = 0.5$)	–	0.050	0.008	0.009	1.800	0.060	0.010	1.679	4.062	2.062	1.082
$PDCF_{ARIMA}$ ($\mu = 10, \nu = 0.1, \delta = 2$)	–	0.050	0.008	0.008	2.000	0.081	0.018	1.473	4.062	2.062	1.085
SPF_{AR} ($\mu = 0.1, \nu = 1$)	–	1.652	2.062	0.624	0.000	2.062	4.062	1.087	1.042	0.293	1.431
$SPF2_{AR}$ ($\mu = 0.1, \nu = 10$)	–	0.006	2.062	0.505	0.000	1.045	4.062	5.659	0.630	0.000	1.552
SPF_{AR} ($\mu = 10, \nu = 10$)	+	0.421	1.517	1.639	0.200	2.866	0.891	7.418	0.010	0.000	1.662
$SPF2_{AR}$ ($\mu = 10, \nu = 1$)	+	0.215	1.902	1.640	0.200	2.015	1.699	7.844	0.009	0.217	1.749

The exception was SPF_{AR} ($\mu = 0.1$, $\nu = 1$) with no MSE modification, which did not find any structural break.

Most of analysed methods correctly find the place where the variance changes (test case 1(b)). Problems have only SPF and SPF2 methods. They generally did not detect a break in such a case. The SPF_{AR} ($\mu = 10$, $\nu = 10$) with MSE is an exception, because it finds structural breaks but locates them in incorrect places.

Most of presented cost functions detect change between two different AR models very well (case 1(c)). The exception are methods involving MSE modification. They detect usually two structural breaks and only small part of them is close to correct structural break location.

Most of presented cost functions correctly solved the task of processing white noise time series (case 1(d)). The expected outcome in this scenario is not to return any structural breaks. The only cost functions which almost always failed the test are the PDCF and DCF cost functions.

Test case 1(e) contained series composed of two segments: white noise and a segment with a trend. In this scenario, the classical MDL and the elastic MDL rule produced excellent outcomes. PDCF and DCF cost functions were less precise, but they detected one structural break near the expected location (slightly too early). Methods based on penalty cost function in most cases returned more than one structural break. Usually one of them was in the expected location (with the exception of configuration SPF_{AR} ($\mu = 0.1$, $\nu = 1$) without MSE where no structural breaks were found.

Result obtained for the test cases composed of three segments of the same length (case 1(f)) were generally good. Only SPF and SPF2 cost functions had problems. The configurations did not detect any structural break or detected (often more than one) in incorrect places.

The next test case concentrated on a rather theoretical example, where time series was the shape of a parabola (see Fig. 1(g)). The parabola case is very difficult as there are many viable locations of structural breaks. In this test case, the introduced BPD and SC scores are not very informative and a valid evaluation must be manual. The experiments have shown, that the methods including the MSE modification and $SPF2_{AR}$ ($\mu = 0.1$, $\nu = 10$)

detected too many structural breaks and were not very practical. Conversely, cost functions employing the ARIMA model had a strong tendency not to detect any structural change. Results generated by $\mathrm{SPF}_{\mathrm{AR}}$ ($\mu = 0.1$, $\nu = 1$) also were very impractical (one structural break detected closely to start or end of time series). Other cost functions usually allowed for detecting a few structural changes near the $\frac{1}{4}$th and the $\frac{3}{4}$th of the time series length. The best results were obtained with the $\mathrm{EMDL}_{\mathrm{AR}}$ ($\mu = 3$, $\nu = 1$) model.

The last two examples were the real-world time series describing fridge freezer power demand. We analysed two examples. First, where the freezer was turned on and turned off after some time (case 2(a)). Second, where the freezer was turned on and was working until the end of the time series (case 2(b)). The best results were obtained for chosen configurations of SPF and SPF2 methods. For case 2(a) they allowed for a correct detection of both structural breaks and for case 2(b) the same methods returned one structural break in the correct location. It is worth to say that cost function variants using MSE modification give more reliable results for case 2(a). All other cost functions listed in the tables did not find any structural break or found one near to the end of the time series. The latter behaviour is a result of a phenomenon which will be discussed in the next subsection.

We have observed that the proposed cost functions enhance time series processing capabilities for structural break detection algorithms. Noteworthy, for all of the proposed cost functions, we can find a set of parameters that works very well for various test cases. Furthermore, we will demonstrate in Section 7 that the proposed cost functions are generally better performing than the state-of-the-art $\mathrm{MDL}_{\mathrm{AR}}$ cost function. Promising results have been achieved with the use of the SPF and SPF2 cost functions. Their superiority was the clearest when we processed real-world time series. Their only weakness is that they may tend to output incorrect structural break count – too many or too few, more often too many, especially when the methods are used with MSE modification.

The empirical experiments have shown that the cost functions using the ARIMA model are very good, and in many cases, they are better than the ones using the AR model. Their weak point is computational complexity, which is higher, and this difference grows as the time series length increases. For example, it took only 0.3s for the AR model to calculate MDL cost function partial values for all possible time series segments for the test case with 50 data points and 190.0s for ARIMA. In contrast, for a case with 100 data points, it took 1.4s and 937.7s, respectively. This shows that cost functions using the ARIMA model are helpful for short time series. In practice, it is not a severe limitation for many real-world domains applying break detection methods (such as economics), in which relatively short time series are of interest.

The proposed objective functions punish for deviations from theoretical models constructed on developed segments. Therefore, if these models cannot reflect the underlying data well, there is no reasonable justification to apply this model. If the data contains a clearly visible and changing trend, the recommended choice is an objective function that punishes deviations from theoretical models. The component that punishes for short intervals should we weighted as less important. Punishing for deviations from a theoretical model is a lousy strategy for high-variance time series or if variance is the crucial distinction for subsequent segments. In this case, punishing for short segments and diminishing the punishment for deviations from theoretical models is much better.

6. Parameter Fitting and a Comparison Between PSO, GA, and ACO

In this section, we present a study on the impact of hyperparameters on the heuristic algorithms. We performed these experiments with the use of classical AR MDL cost function without the MSE modification due to its good reputation and long presence in literature. The objective of these experiments was to demonstrate technical validity of parameter selection for the optimizers. Such a study is necessary in methods utilizing heuristic approaches. While it does not deliver new results concerning the algorithmic level, it shows the behaviour of the procedure and allows to verify its stability and robustness.

6.1. *PSO*

We performed tests for 3 configurations of PSO:

- C_{11}: $\omega = -0.1832$, $c_1 = 0.5287$, $c_2 = 3.1913$ with population size 47 (these parameters come from Hvass Pedersen, 2010),
- C_{12}: $\omega = 0.8$, $c_1 = 2$, $c_2 = 2$ (used by Charalampakis and Dimou, 2010) with population size 20,
- C_{13}: $\omega = 0.5$, $c_1 = 1.8$, $c_2 = 2.29$ with population size 40, being an own modification of the previous set of parameters.

All the experiments were performed with maximal limit of 10 detected structural breaks. Tables 3 and 4 present results for these configurations.

According to the two presented metrics, the best results achieved using PSO were for the parameter configuration C_{11}. All tested configurations produced excellent results for the first six case studies (1(a)–1(f)).

The most interesting observations were made for the real-world time series. All named configurations did not locate structural breaks in the correct locations and found none or

Table 3
BPD score for different PSO configurations.

Config.	Test case									
	1(a)	1(b)	1(c)	1(d)	1(e)	1(f)	1(g)	2(a)	2(b)	Mean
C_{11}	0.0	0.0	0.0	0.0	0.0	0.0	2.5	1.2	0.5	0.467
C_{12}	0.0	0.0	0.0	0.0	0.0	0.0	2.6	1.7	0.9	0.578
C_{13}	0.0	0.0	0.0	0.0	0.0	0.0	2.6	1.9	1.0	0.611

Table 4
SC score for different PSO configurations.

Config.	Test case									
	1(a)	1(b)	1(c)	1(d)	1(e)	1(f)	1(g)	2(a)	2(b)	Mean
C_{11}	0.019	0.001	0.009	0.000	0.006	0.013	5.023	2.465	1.432	0.996
C_{12}	0.019	0.001	0.009	0.000	0.006	0.018	5.224	3.463	1.936	1.186
C_{13}	0.019	0.001	0.009	0.000	0.006	0.014	5.222	3.862	2.062	1.244

one structural break close to the end of the time series. The detection of such structural breaks (near or precisely at the end of a sequence) is a side effect of the algorithm's operations. During the optimization procedure, each structural break location is treated as a variable, and it can change its position in time series. A variable at the end of the time series is treated as if the structural break did not exist. This, however, may result in the observed adverse effect. If a time series is challenging, we may end up with incorrectly identified structural breaks near the end. The described behaviour did not occur for configuration C_{13}, but it appeared for C_{12}, where the population count was smaller (20), and for C_{11}, probably due to the negative inertia.

6.2. *GA*

For the GA we ran the experiments on a grid of parameters. We have tested mutation probabilities from the set {0.0125, 0.0083, 0.0033}, crossover probabilities from the set {0.2, 0.5, 0.8}, tournament sizes {2, 8}, population size 40 and using small direction modification (0.2) or not.

Both metrics indicated the same best-performing parameter configurations listed below:

- C_{21}: mutation probability: 0.0083, crossover probability: 0.5, tournament size: 8, population size: 40, no direction modification,
- C_{22}: mutation probability: 0.0125, crossover probability: 0.5, tournament size: 8, population size: 40, no direction modification,
- C_{23}: mutation probability: 0.0083, crossover probability: 0.2, tournament size: 8, population size: 40, no direction modification,
- C_{24}: mutation probability: 0.0033, crossover probability: 0.5, tournament size: 8, population size: 40, no direction modification.

The results calculated for listed configurations are in Tables 5 and 6.

Table 5
BPD score for different GA configurations.

Config.	Test case									
	1(a)	1(b)	1(c)	1(d)	1(e)	1(f)	1(g)	2(a)	2(b)	Mean
C_{21}	0.0	0.0	0.0	0.0	0.0	0.0	1.5	2.0	1.0	0.500
C_{22}	0.1	0.0	0.0	0.0	0.1	0.0	1.5	2.0	1.0	0.522
C_{23}	0.1	0.0	0.2	0.0	0.0	0.0	1.4	2.0	1.0	0.522
C_{24}	0.1	0.0	0.1	0.0	0.0	0.0	1.6	2.0	1.0	0.533

Table 6
SC score for different GA configurations.

Config.	Test case									
	1(a)	1(b)	1(c)	1(d)	1(e)	1(f)	1(g)	2(a)	2(b)	Mean
C_{21}	0.029	0.030	0.012	0.000	0.049	0.067	3.027	4.062	2.062	1.038
C_{22}	0.235	0.018	0.021	0.000	0.249	0.053	3.028	4.062	2.062	1.081
C_{23}	0.245	0.041	0.417	0.000	0.065	0.081	2.827	4.062	2.062	1.089
C_{24}	0.244	0.028	0.211	0.000	0.039	0.054	3.230	4.062	2.062	1.103

The analysis showed that it was generally better not to use the direction modification. The average value of the SC metrics of the results concerning the direction modification was 1.467, while the average for the results without it was 1.187. For various values of mutation probability, crossover probability, and tournament size, mean SC metrics fluctuations were very little, especially compared to concerning mutation modification influence.

Qualitative analysis of the experiments' results showed that the outcomes were quite similar for all tested GA parameters. There were minor differences in the count of detected structural breaks in a few cases.

6.3. *ACO*

The experiments with ACO were conducted for a grid of parameters. We tested different combinations of parameters assuming the following sets of values of interest: the evaporation parameter {0.5, 0.9}, importance of pheromone count {1, 1.5}, importance of fragment rank {2, 5}, and estimated break count {1, 2}, count of elitist ants used to correct pheromone data {2, 4}, importance of count of pheromone updated at one time {0.05, 0.2} and the limit of detected structural breaks {4, 10}. The best results were obtained for the following configurations:

- C_{32}: evaporation parameter: 0.9, pheromone count parameter: 1, fragment rank parameter: 2, estimated break count parameter: 2, elitist ants count: 4, updated pheromone count: 0.05, detected structural break limit: 4,
- C_{31}: evaporation parameter: 0.9, pheromone count parameter: 1, fragment rank parameter: 2, estimated break count parameter: 2, elitist ants count: 4, updated pheromone count: 0.05, detected structural break limit: 10,
- C_{33}: evaporation parameter: 0.9, pheromone count parameter: 1.5, fragment rank parameter: 2, estimated break count parameter: 2, elitist ants count: 4, updated pheromone count: 0.05, detected structural break limit: 4.

Results obtained for the listed parameter configurations them are presented in Tables 7 and 8.

Table 7
SC score for different ACO configurations.

Config.	Test case									
	1(a)	1(b)	1(c)	1(d)	1(e)	1(f)	1(g)	2(a)	2(b)	Mean
C_{31}	0.0	0.0	0.0	0.0	0.0	0.0	1.8	2.0	1.0	0.53
C_{32}	0.0	0.0	0.1	0.0	0.0	0.0	1.8	2.0	1.0	0.54
C_{33}	0.0	0.0	0.0	0.0	0.0	0.0	1.9	2.0	1.0	0.54

Table 8
SC score for different ACO configurations.

Config.	Test case									
	1(a)	1(b)	1(c)	1(d)	1(e)	1(f)	1(g)	2(a)	2(b)	Mean
C_{31}	0.029	0.001	0.011	0.000	0.006	0.014	3.625	4.062	2.062	1.090
C_{32}	0.013	0.001	0.211	0.000	0.006	0.024	3.629	4.062	2.062	1.112
C_{33}	0.029	0.003	0.009	0.000	0.006	0.031	3.823	4.062	2.062	1.114

Subsequently, we looked into each parameter and analysed its role in the achieved outcomes. As BPD and SC metrics usually give similar conclusions, but the SC metric provides more information, the discussion below refers to the average value of the SC. The results showed that:

- a more favourable value for the evaporation coefficient is 0.9 (SC 1.289) than 0.5 (SC 1.383),
- a slightly better value of importance of the pheromone parameter is 1.0 (SC 1.335) than 1.5 (SC 1.337),
- 2 is a slightly better value than 5 for the weight of fragment rank (achieving the average SC of 1.321, in comparison to 1.350),
- for the importance of estimated break count parameter value 2 (SC 1.292) is a better choice than 1 (SC 1.379),
- increasing the count of elitist ants had a little contribution towards the improvement of the score (SC 1.345 for value 2 and 1.327 for 4),
- a better value of scale of the pheromone update parameter was 0.05 (SC 1.324) than 0.2 (SC 1.348),
- better results were obtained with the use of a limit of 10 maximum detected structural changes (SC 1.319) than with the use of a limit of 4 (SC 1.353).

The most significant conclusion refers to the structural break count limit. This limit allows for accelerating calculations and achieving better results for test cases where the desirable structural break count is less than this limit. The experiments showed that more favourable results were obtained with a bigger limit, which indicates that the algorithm can very well generalize and detect fewer breaks than this limit, which is an essential feature for many possible real-world applications of structural break detection methods.

7. Comparison with Baseline and State-of-the-Art Methods

The proposed approaches were compared with two selected state-of-the-art methods: Bai and Perron (1998, 2003), ClaSP by Ermshaus *et al.* (2023) and Davis *et al.* method (Davis *et al.*, 2016, 2006, 2008; Davis and Yau, 2013; Davis *et al.*, 2005). The comparison is divided into three parts. First, we compare with the Bai-Perron method, and next with ClaSP. It was performed using cases 1(a)–2(b) without case 1(h). Next, we address the time series used by Davis *et al.* (2005) in their work, denoted in this paper as case 1h. For this time series, we compare our results with those of the Bai-Perron, ClaSP, and Davis methods.

7.1. *Comparison with the Bai-Perron Method (Baseline)*

We performed experiments using the Bai-Perron method with the same test cases as in previous tests. Mean BPD and SC for these examples and general mean are presented in Table 9.

Table 9
BPD and SC results for the Bai-Perron method.

	Test case									
	1(a)	1(b)	1(c)	1(d)	1(e)	1(f)	1(g)	2(a)	2(b)	Mean
BPD	0.1	0	0.7	0.2	0	0.4	4.1	1.0	4.0	1.167
SC	0.335	0.081	1.492	0.400	0.066	0.864	8.214	2.020	8.058	2.392

We can see that the Bai-Perron method was substantially worse than the presented methods. Qualitative analysis showed that the Bai-Perron method achieved moderately satisfying results for cases 1(c), 1(e), or 1(f). However, even in these cases, the structural breaks were inexact. The Bai-Perron method produced even worse outcomes for the remaining test cases, confirming results for individual test cases. It is worth highlighting that the method had problems with real-world cases where it detected too many structural breaks. The experiments proved that the methods presented in this paper guarantee better results than the baseline Bai-Perron method (please compare with Tables 1 and 2 concerning our results).

7.2. *Comparison with ClaSP Algorithm*

We compared our work also to the state-of-the-art ClaSP algorithm by Ermshaus *et al.* (2023). We run the algorithm on our examples. The obtained results were very poor. They are presented in Table 10. The method found any structural breaks rarely. The only example from set 1(a)–2(b) where structural break was detected was example 1(c), but the precision also was not ideal.

The results covered in Table 10 (ClaSP) may be compared with Table 1 and 2 concerning our results. The advantage of the proposed approach is visible.

7.3. *Experiments Using Real-World Time Series*

Contrary to previous experiments, to process the real-world dataset published by Davis *et al.*, we used the island model (40 islands, 20 migrations, 5 iterations between migrations, small population size 40) to make our settings similar to the settings employed by Davis *et al.* (2005).

Table 11 presents mean precision of detected structural break locations and associated standard deviations. Particular hyperparameter configurations used with our optimizers are noted with letter C, and their explanation is provided in Section 6.

Table 10
BPD and SC results for the ClaSP method.

Metric	1(a)	1(b)	1(c)	1(d)	1(e)	1(f)	1(g)	2(a)	2(b)	Mean
BPD	1.0	1.0	0.1	0.0	1.0	2.0	1.0	2.0	1.0	1.011
SC	2.062	2.062	0.235	0.000	2.062	4.062	2.062	4.062	2.062	2.074

Table 11
Comparison with the state-of-the-art methods for case 1h. Mean 1 and Std 1 refer to the mean precision and standard deviation for the first (out of two) true structural breaks in the time series. Mean 2 and Std 2 refer to the mean precision and standard deviation concerning detecting the second true structural break. Results for Bai-Perron and ClaSP algorithms concern only experiments where two structural breaks were detected.

Algo.	Config.	Cost Function	Mean 1	Std 1	Mean 2	Std 2
Davis *et al.*	–	MDL_{AR}	0.50000	0.00200	0.74200	0.00700
Bai-Perron	–	–	0.49898	0.07827	0.73511	0.03329
ClaSP	–	–	0.62666	0.16288	0.73706	0.00674
PSO	C_{11}	$\text{EMDL}_{\text{ARIMA}}$ ($\mu = 3, \nu = 1$)	0.50020	0.00373	0.75039	0.00050
PSO	C_{11}	ARIMA MDL	0.50127	0.00291	0.75000	0.00159
GA	C_{27}	$\text{EMDL}_{\text{ARIMA}}$ ($\mu = 3, \nu = 1$)	0.50068	0.00252	0.75000	0.00260
PSO	C_{11}	$\text{PDCF}_{\text{ARIMA}}$ ($\mu = 10, \nu = 0.1, \delta = 2$)	0.50068	0.00412	0.74932	0.00393

Davis *et al.* (2005) delivered a method that returned two structural breaks for every experiment. Our algorithms also found two structural breaks for every performed test.

Subsequently, we compared our results against those calculated using Bai-Perron's and ClaSP methods. They are also given in Table 11. Please note "Std" values, which show the efficiency of the compared methods (the smaller the better).

It is essential to state that the means and standard deviations presented in Table 11 are calculated only for experiments where two structural breaks were detected. Notably, Davis's algorithm and our configurations consistently detected two breaks each time we ran them. That was the expected behaviour. However, the Bai-Perron method detected two breaks in 80% of repetitions, while ClaSP detected two breaks only in 40% of repetitions. That information is sufficient to claim that both Bai-Perron and ClaSP methods work worse than the other analysed methods.

Still, to provide a deeper analysis, Table 11 presents the results achieved by all four approaches limited to the cases when two structural breaks were detected (which promotes Bai-Perron and ClaSP, as we skipped the cases when these two returned other breaks numbers).

First and foremost, Table 11 allows comparing our results and those of Davis *et al.* In both cases, two (out of two) structural breaks were always detected. However, using our method, the second structural break was detected much better than in Davis's work, while the first was slightly worse. If we take an average of standard deviations for the location of the first and the second structural break, we see that the average for Davis's work is 0.0045, and the average of the best variant of our method is 0.00212. The latter average (our approach) is two times better than the Davis's. This shows that the method presented in this article outperforms the one given by Davis *et al.*

Our approach is very good in comparison with state-of-the-art methods. In our opinion, it is also more versatile and customizable. We have delivered not a single method but a suite of methods. Finally, we may emphasize that our results are much better than those obtained with Bai-Perron's and ClaSP methods. These two algorithms struggle to find two structural breaks. Even if they find two structural breaks, they locate them inaccurately.

8. Conclusion

The discussed experiments have proven the key statements from the introductory section of this paper. In our studies, we have delivered a suite of new cost functions to be used for structural break detection with the help of metaheuristic optimization.

The experimental procedure was designed to ensure that a satisfying number of repetitions of a model construction process was performed. We have paid attention to the three aspects in which properties influence the processing outcome. To eliminate chances of drawing unfounded conclusions, we applied the following schema:

- We have drawn 10 (different) time series for each synthetic time series case. The generative process stayed the same within one case, but time series-specific values were different.
- For real-world time series, each experiment was repeated 10 times.
- We have fixed the metaheuristic methods' hyperparameters to a selected configuration and run the same configuration for different specifications of the break detection algorithm to test one aspect.
- We have fixed the structural break detection algorithm specification for a few configurations and tested the impact of metaheuristic algorithm hyperparameters.

The introduced changes allowed us to improve the qualitative and quantitative outcomes of the structural break detection task. In particular, concerning the state-of-the-art algorithms, we managed to:

- improve numerical accuracy of the results, especially when analysing time series containing trend,
- show that ACO and PSO methods are not worse at solving the task than GA,
- give more flexibility to the experts (potential users of our method) by allowing, optionally, to set parameters that control the count of detected structural breaks.

Each new cost function or modification presented in this paper targeted a different new property. The Elastic MDL method guarantees results similar to the state-of-the-art techniques but introduces the possibility of adapting the model's shape to the user's preferences. Methods using a penalty for deviation from a linear trend work well for real-world time series. DCF and PDCF cost functions capture the new understanding of cost functions designed for structural break detection problems. Let us recall that these two take into account the variance of residuals.

The experiments have shown that using ARIMA often helps to achieve better results than AR. Thanks to the use of metaheuristic approaches, in such cases, the optimization procedure adjusts the model's shape and allows us to obtain more accurate results.

Our studies in this area can be continued. The most interesting future research area is extending this approach to time series with multiple variables.

There is a straightforward solution to this task because the proposed formulas are additive, cf. equations (10)–(20). Because of that, we can simply add more components to these formulas to account for multiple variables – they will be represented with their own

AR/ARIMA models for the same segments. The issue with this simple solution is that the formulas' complexity grows. Consequently, such an objective function may become more challenging to optimize, even for a heuristic method. However, we cannot evaluate the actual influence of this complexity without appropriate empirical tests.

An alternative path worthy of further inspection when working with multivariate time series could be to reduce time series dimensionality (perhaps even to one dimension) and proceed with the approach presented in this paper. The potential problem that may arise when working with such a solution is that the procedure may become intangible. We would have to employ an external algorithm whose properties will heavily influence the processing outcome. Analysing the performance of such a data processing pipeline will be more complex, as we need to cover this additional algorithm.

Finally, the most advanced solution would be to replace ARIMA/AR, predominantly for univariate time series, with a model suitable for dealing with multiple variables. VAR (Vector Autoregression) can be fused to the proposed framework in such a case.

Funding

This research was supported by the Warsaw University of Technology within the Excellence Initiative: Research University (IDUB) programme.

References

Abdel-Basset, M., Abdel-Fatah, L., Sangaiah, A.K. (2018). Metaheuristic algorithms: a comprehensive review. In: *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, pp. 185–231.

Altansukh, G., Osborn, D.R. (2022). Using structural break inference for forecasting time series. *Empirical Economics*, 63, 1–41.

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

Bai, J., Perron, P. (2003). Computation and analysis of multiple structural change models. *Journal of Applied Econometrics*, 18(1), 1–22.

Bai, J., Duan, J., Han, X. (2024). The likelihood ratio test for structural changes in factor models. *Journal of Econometrics*, 238(2), 105631.

Bardwell, L., Fearnhead, P., Eckley, I.A., Smith, S., Spott, M. (2019). Most recent changepoint detection in panel data. *Technometrics*, 61(1), 88–98.

Behrendt, S. (2021). Structural breaks in Box-Cox transforms of realized volatility: a model selection perspective. *Quantitative Finance*, 21(11), 1905–1919.

Borzykh, D.A., Yazykov, A.A. (2020). On the practical applicability of three cusum-methods for structural breaks detection in EGARCH-models. *Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes*, 16(1), 19–30.

Casini, A., Perron, P. (2024). Change-point analysis of time series with evolutionary spectra. *Journal of Econometrics*, 242(2), 105811.

Charalampakis, A., Dimou, C. (2010). Identification of Bouc–Wen hysteretic systems using particle swarm optimization. *Computers & Structures*, 88(21), 1197–1205.

Cheng, Y., Yi, J., Yang, X., Lai, K.K., Seco, L. (2022). A CEEMD-ARIMA-SVM model with structural breaks to forecast the crude oil prices linked with extreme events. *Soft Computing*, 26, 8537–8551.

Cho, H., Kirch, C. (2022). Two-stage data segmentation permitting multiscale change points, heavy tails and dependence. *Annals of the Institute of Statistical Mathematics*, 74(4), 653–684.

Cho, H., Korkas, K.K. (2022). High-dimensional garch process segmentation with an application to value-at-risk. *Econometrics and Statistics*, 23, 187–203.

Cho, H., Kirch, C. (2024). Data segmentation algorithms: univariate mean change and beyond. *Econometrics and Statistics*, 30, 76–95.

Davis, R.A., Yau, C.Y. (2013). Consistency of minimum description length model selection for piecewise stationary time series models. *Electronic Journal of Statistics*, 7, 381–411.

Davis, R., Lee, T., Rodriguez-Yam, G. (2005). Structural breaks estimation for non-stationary time series signals. In: *IEEE Workshop on Statistical Signal Processing Proceedings*, Vol. 2005, pp. 233–238.

Davis, R.A., Lee, T.C.M., Rodriguez-Yam, G.A. (2006). Structural break estimation for nonstationary time series models. *Journal of the American Statistical Association*, 101(473), 223–239.

Davis, R.A., Lee, T.C.M., Rodriguez-Yam, G.A. (2008). Break detection for a class of nonlinear time series models. *Journal of Time Series Analysis*, 29(5), 834–867.

Davis, R.A., Hancock, S.A., Yao, Y.-C. (2016). On consistency of minimum description length model selection for piecewise autoregressions. *Journal of Econometrics*, 194(2), 360–368.

Ditzen, J., Karavias, Y., Westerlund, J. (2021). *Testing and Estimating Structural Breaks in Time Series and Panel Data in Stata*. Discussion Papers 21-14, Department of Economics, University of Birmingham.

Doerr, B., Fischer, P., Hilbert, A., Witt, C. (2017). Detecting structural breaks in time series via genetic algorithms. *Soft Computing*, 21, 4707–4720.

Dorigo, M., Stützle, T. (2019). *Ant Colony Optimization*. MIT Press, Cambridge, MA.

Éltetö, T., Hansen, N., Germain-Renaud, C., Bondon, P. (2012). Scalable structural break detection. *Applied Soft Computing*, 12(11), 3408–3420.

Ermshaus, A., Schäfer, P., Leser, U. (2023). ClaSP: parameter-free time series segmentation. *Data Mining and Knowledge Discovery*, 37(3), 1262–1300.

Farsi, N., Mahjouri, N., Ghasemi, H. (2020). Breakpoint detection in non-stationary runoff time series under uncertainty. *Journal of Hydrology*, 590, 125458.

Fryzlewicz, P. (2014). Wild binary segmentation for multiple change-point detection. *The Annals of Statistics*, *42*(6).

Fryzlewicz, P. (2020). Detecting possibly frequent change-points: wild binary segmentation 2 and steepest-drop model selection. *Journal of the Korean Statistical Society*, 49(4), 1027–1070.

Hall, A.R., Osborn, D.R., Sakkas, N. (2013). Inference on structural breaks using information criteria. *The Manchester School*, 81(S3), 54–81.

Huang, S.-H., Shih, W.-Y., Lu, J.-Y., Chang, H.-H., Chu, C.-H., Wang, J.-Z., Huang, J.-L., Dai, T.-S. (2020). Online structural break detection for pairs trading using wavelet transform and hybrid deep learning model. In: *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 209–216.

Hvass Pedersen, M.E. (2010). *Good Parameters for Particle Swarm Optimization*. In technical report No. HL1001. Hvass Laboratories.

Inclan, C., Tiao, G.C. (1994). Use of cumulative sums of squares for retrospective detection of changes of variance. *Journal of the American Statistical Association*, 89(427), 913–923.

Kim, K., Park, J.H., Lee, M., Song, J.W. (2022). Unsupervised change point detection and trend prediction for financial time-series using a new CUSUM-based approach. *IEEE Access*, 10, 34690–34705.

Kirch, C., Reckruehm, K. (2024). Data segmentation for time series based on a general moving sum approach. *Annals of the Institute of Statistical Mathematics*, 76, 393–421.

Kovács, S., Bühlmann, P., Li, H., Munk, A. (2023). Seeded binary segmentation: a general methodology for fast and optimal changepoint detection. *Biometrika*, 110(1), 249–256.

Lan, N., Geyer, M., Chemla, E., Katzir, R. (2022). Minimum description length recurrent neural networks. *Transactions of the Association for Computational Linguistics*, 10, 785–799.

Lee, T.H., Parsaeian, S., Ullah, A. (2022). Efficient combined estimation under structural breaks. *Advances in Econometrics*, 43A, 119–142.

Li, Y., Cezeaux, R., Yu, D. (2019a). Automating data monitoring: detecting structural breaks in time series data using Bayesian minimum description length.

Li, Y., Lund, R., Hewaarachchi, A. (2019b). Multiple changepoint detection with partial information on change-point times. *Electronic Journal of Statistics*, 13(2), 2462–2520.

Lim, H., Choi, H., Choi, Y., Kim, I.-J. (2020). Memetic algorithm for multivariate time-series segmentation. *Pattern Recognition Letters*, 138, 60–67.

Lu, Q., Lund, R., Lee, T.C.M. (2010). An MDL approach to the climate segmentation problem. *The Annals of Applied Statistics*, 4(1), 299–319.

Madrid Padilla, O.H., Yu, Y., Wang, D., Rinaldo, A. (2022). Optimal nonparametric multivariate change point detection and localization. *IEEE Transactions on Information Theory*, 68(3), 1922–1944.

Meier, A., Kirch, C., Cho, H. (2021). mosum: a package for moving sums in change-point analysis. *Journal of Statistical Software*, 97(8), 1–42.

Murray, D., Liao, J., Stankovic, L., Stankovic, V., Hauxwell-Baldwin, R., Wilson, C., Coleman, M., Kane, T., Firth, S. (2015). A data management platform for personalised real-time energy feedback. In: *Proceedings of the 8th International Conference on Energy Efficiency in Domestic Appliances and Lighting*.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5), 465–471.

Romano, G., Rigaill, G., Runge, V., Fearnhead, P. (2022). Detecting abrupt changes in the presence of local fluctuations and autocorrelated noise. *Journal of the American Statistical Association*, 117(540), 2147–2162.

Safikhani, A., Bai, Y., Michailidis, G. (2022). Fast and scalable algorithm for detection of structural breaks in big VAR models. *Journal of Computational and Graphical Statistics*, 31(1), 176–189.

Scott, A.J., Knott, M. (1974). A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, 30(3), 507–512.

Shaochuan, L. (2020). Bayesian multiple changepoints detection for Markov jump processes. *Computational Statistics*, 35(3), 1501–1523.

Shi, X., Beaulieu, C., Killick, R., Lund, R. (2022a). Changepoint detection: an analysis of the Central England temperature series. *Journal of Climate*, 35(19), 2729–2742.

Shi, X., Gallagher, C., Lund, R., Killick, R. (2022b). A comparison of single and multiple changepoint techniques for time series data. *Computational Statistics & Data Analysis*, 170, 107433.

Smith, S.C. (2023). Structural breaks in grouped heterogeneity. *Journal of Business & Economic Statistics*, 41, 752–764.

Suárez-Sierra, B.M., Coen, A., Taimal, C.A. (2023). Genetic algorithm with a Bayesian approach for the detection of multiple points of change of time series of counting exceedances of specific thresholds. *Journal of the Korean Statistical Society*, 52, 982–1024.

Tartakovsky, A. (2019). *Sequential Change Detection and Hypothesis Testing*. Chapman and Hall/CRC.

Woody, J., Xu, Y., Dyer, J., Lund, R., Hewaarachchi, A.P. (2021). A statistical analysis of daily snow depth trends in North America. *Atmosphere*, 17(7) 820.

Yan, Q., Liu, Y., Liu, S., Ma, T. (2021). Change-point detection based on adjusted shape context cost method. *Information Sciences*, 545, 363–380.

Yang, Q., Zhang, Y. (2022). Change-point detection for the link function in a single-index model. *Statistics & Probability Letters*, 186, 109468.

Zhang, N.R., Siegmund, D.O. (2007). A modified Bayes Information Criterion with applications to the analysis of comparative genomic hybridization data. *Biometrics*, 63(1), 22–32.

M. Burczaniuk received the BSc degree in computer science and the MSc degree in computer science and information systems from the Warsaw University of Technology, Warsaw, Poland, in 2020 and 2021, respectively. His research interests include machine learning, evolutionary computation and swarm intelligence.

A. Jastrzębska received the BSc degree in information technology from the University of Derby, Derby, UK, in 2009, the MScEng degree in computer engineering from the Rzeszow University of Technology, Rzeszow, Poland, in 2010, the MA degree in economics from the University of Rzeszow, Rzeszow, in 2011, and the PhD and DSc degrees from the Warsaw University of Technology, Warsaw, Poland, in 2016 and 2021, respectively. She is associate professor with the Faculty of Mathematics and Information Science, Warsaw University of Technology. Her research interests include machine learning and fuzzy modeling. She serves as an associate editor of *Applied Soft Computing* journal.