An Evaluation of Combinations of Lossy Compression and Change-Detection Approaches for Time-Series Data

Gregor Hollmig, Matthias Horne, Simon Leimkühler, Frederik Schöll,
Carsten Strunk, Adrian Englhardt, Pavel Efros, Erik Buchmann, Klemens Böhm

Karlsruhe Institute of Technology, Karlsruhe, Germany

{*gregor.hollmig, matthias.horne, simon.leimkuehler, carsten.strunk, adrian.englhardt*}*@student.kit.edu, schoell@ira.uka.de,*
{*pavel.efros, erik.buchmann, klemens.boehm*}*@kit.edu*

Abstract

Today, time series of numerical data are ubiquitous, for instance in the Internet of Things. In such scenarios, it is often necessary to compress the data to, say, reduce data-transmission costs, and to detect changes on it. More specifically, both methods are used *in combination*, i.e., data is lossily compressed and later decompressed, and then change detection takes place. There exists a broad variety of compression as well as of change-detection techniques. This calls for a systematic comparison of different combinations of compression and change-detection techniques, for different data sets, together with recommendations on how the values of the various (typically non-linear) parameters should be chosen. This article is such an evaluation. Its design is not trivial, necessitating a number of decisions. We work out the details and the rationale behind our design choices. Next to other results, our study shows that the choice of combinations of change detection and compression algorithm and their parameterization does affect result quality significantly. Our evaluation also indicates that results are highly contingent on the nature of the data.

Keywords: time series, compression, change detection

1. Introduction

Nowadays, time-series data is ubiquitous. More and more applications like the Smart Grid or the Internet of Things that produce and/or process time-series data are proliferating. Such data is often used to detect certain events and to react to them as soon as possible. In other words, change-detection methods are indispensable. On the other hand, because of the many devices generating data, the huge amount of data and the high data-transfer rates, an efficient compression is essential. Lossless compression reduces the statistical redundancy of the data. However, compression rates are relatively low; as an example, rates of 4 have been reported for smart-meter readings from individual buildings using the bzip2 algorithm [1]. Lossy compression in turn yields significantly higher compression ratios than the lossless one. At the same time, data compressed in this manner is still useful for many applications. In this study, we focus on lossy compression. Putting things together, it often is necessary to combine lossy compression[1] and change-detection techniques.

Example 1. *Smart meters may deliver data to a central analysis system via a wireless network. To save bandwidth and to reduce costs, the data is compressed directly on the device. The central data-analysis system can then do change detection to react to events such as a sudden increase in overall power consumption.*

When combining lossy compression and change detection, several issues arise. First, lossy compression introduces errors. In particular, changes can be lost, or new false changes can occur. Therefore a lossy compression method must be chosen which preserves the change information as much as possible. Furthermore, different use cases generate different kinds of time-series data, as we will explain. Thus, it is necessary to choose a good combination of compression and change-detection technique *per use case*. This is difficult due to the large number of possible combinations. Next, compression as well as change-detection algorithms usually have several parameters, which often have non-obvious effects on the outcome. The expectation typically is that domain experts select the parameter values. This means that these experts must have a deep insight into the algorithms used. But even if they have selected the values, it is hard to determine whether their selection is a good one. To investigate how combinations of compression and change-detection algorithms perform on different datasets, a systematic comparison is necessary. This article is such a study.

Designing our study has been challenging, partly due to the issues just mentioned. To illustrate, one of the various design decisions is as follows: It is difficult to choose the parameterization of the compression and the change-detection algorithms such that the comparison is fair. Reusing the parameter values suggested in the original publications may not be the best option. This is because proper choices of parameter values depend on the data the algorithms are applied to. Thus, we have decided

[1] For improved readability we usually refer to compression and later decompression simply as compression.

to perform an optimization on each dataset that yields the parameter values that give way to change-detection results after compression that are closest to some carefully chosen reference point. This article lists the design questions encountered in the context of our comparison, together with explanations behind our choices.

In line with these design decisions, we have implemented a framework that can be used for the evaluation of virtually any combination of compression and change-detection methods. In our specific study, we examine five compression algorithms like APCA [2] and five change-detection algorithms like Online-Kernel Change Detection [3] on five datasets, resulting in 125 possible combinations. We focus on result quality and leave aside criteria such as runtime performance or total cost of ownership, which highly depend on specifics of the implementations and the runtime environment as well as on characteristics of the underlying optimization framework.

The study shows that, while the choice of the dataset does have a huge impact on which combination of compression and change-detection technique performs best, some algorithms like Chebyshev Approximation [4, 5, 6] and Bayesian Online Change Detection [7] yield good results in many settings. We also observe that a good change detection is possible even on strongly compressed data. Next, our results are particularly interesting because studying the algorithms in isolation (e.g., compression without subsequent change detection) may yield a different picture. In [8] for instance, competing algorithms have outperformed Chebyshev Approximation with regard to the compression ratio. In our context in turn, this algorithm has proven to be suitable in combination with many change-detection algorithms.

Paper outline: Section 2 describes some application scenarios. Section 3 explains our design decisions. Section 4 summarizes the algorithms evaluated. Section 5 describes the experimental setup and Section 6 presents the results. Section 7 concludes.

2. Application Scenarios

In this section we describe two scenarios, with slightly different perspectives on the importance of compression and change detection. In the first scenario, compression and change-detection quality are roughly equally important. In the second one, the benefits of a high compression ratio tend to exceed those of the change detection.

2.1. Smart Grid

The Smart Grid is an intelligent communication network which monitors and controls a power network. The integration into such networks of renewable energy producers alters the conventional power flow [9]. These producers are inconsistent and have performance peaks, which in turn demand intelligent power distribution systems.

Consider a company which has to manage a power-distribution network. The company collects, stores and analyzes the data delivered by the many devices (e.g., smart meters, power plants) in its network. The data needs to be analyzed in real-time, thus online change detection is indispensable. To significantly reduce communication and storage costs, the data must be lossily compressed. Now think of a sudden increase in power consumption. The company must react as soon as possible for example by powering up additional power plants. To this end, it must detect the change in the first place, which is not only the consumption measured by one single device, but an aggregate of the entire grid. As a takeaway, we observe that good compression and high-quality change detection are both very important in this scenario.

2.2. Internet of Things

Internet of Things (IoT) refers to large networks of small or embedded devices, which communicate wirelessly. For many IoT entities, energy optimization is a primary constraint, as they are powered by batteries or use energy harvesting methods like micro solar panels. Thus, wireless data transmission often is the biggest factor regarding energy consumption, as the power required to transmit data increases quadratically or even with the power of 4 with the distance between sender and receiver [10]. The power consumption of data compression in turn increases only linearly with the size of the data. Thus, it is reasonable to send data that is lossily compressed over a distance. Detecting changes is often computationally heavy (e.g., overall computational complexity Bayesian Online Change Point Detection is $O(n^5)$, where n is the length of the sequence under consideration [7]) and should be performed on the central unit; it therefore has to take place on compressed and later decompressed data [11]. Now consider a home automation system, where a central control unit can adapt the heating when several temperature or humidity sensors detect a change in the weather. Online change detection is needed to react in short time. This specific scenario benefits more from a high compression ratio than from better change detection, in contrast to the previous scenario.

3. Design Decisions

Designing the comparison study envisioned is challenging; in particular, there are various design decisions that one must address. Figure 1 is an overview of our study framework. The framework takes as input a dataset and ground-truth change points. It then uses these change points to derive optimal parameters for the change-detection methods used. It subsequently uses an adequate error measure to evaluate the impact of compression on the changes in the dataset. This is in order to obtain an optimal combination of compression ratio and change-detection quality. In the following, we describe the important design alternatives and the rationale behind our choices. Even though the subsequent subsections will introduce them explicitly, Table 1 lists the basic notions we use and their definitions.

3.1. Training Data

The evaluation envisioned can take place on the complete dataset or on a subsequence. We see two advantages in using

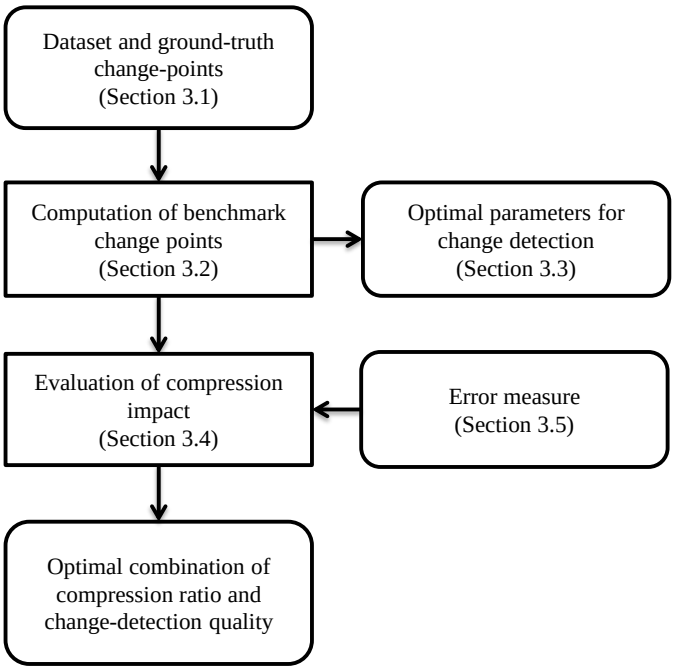

Figure 1: Overview of evaluation framework

Table 1: Basic notions

Notion	**Definition**
change point	point in time when a change occurs
ground-truth change point	real change point present in the data
benchmark change point	point in time determined as change by change-detection algorithm
compression ratio	ratio between the size of the compressed data and the size of the original data
change-detection quality	measure of the effectiveness of a change-detection algorithm

a subsequence. The first one is that the parameter optimization is quicker. Second, we can do so to validate the hypothesis that it is sufficient to run this optimization on a data subsequence, and the result also performs well on the complete dataset or on any other data of the same kind. This is important, because we focus on online algorithms that do not operate on complete datasets, but on streams of data.

3.2. Benchmark Change Points

To assess the quality of change-detection methods, it is very common to compare the detected change points with a ground truth. This however raises at least two issues. First, ground-truth metadata can diverge from the detectable changes. To illustrate, the heart-rate dataset PAMAP[2] comes together with the information when exactly a test person has changed his activity. The heart naturally takes some time to adapt to new activities. Second, most change-detection algorithms can only detect specific kinds of changes. ADWIN for instance is specialized on changes of mean values. In other words, comparing to a ground truth evaluates the suitability of the change-detection algorithm for the dataset. Thus, rather than comparing to a ground truth, we let the change-detection method identify change points on the specific dataset without any compression, and we use these change points as our benchmark, dubbed *benchmark change points*. Compression is only used in the actual comparison study, i.e., when looking for change points on the compressed data. We call the change points identified on the compressed data *comparison change points*.

[2]http://www.pamap.org/demo.html, May 18, 2015

3.3. Parametrization

The result quality and performance of change-detection and compression algorithms depend on their input parameters. In particular, setting the parameters of the change-detection algorithm when comparing alternatives is intricate. One option is to use the parameters recommended in the underlying publications. But this ignores characteristics of the data the algorithms run on. An alternative is to use the parameter values that give way to good results on the data currently examined. If so, these values obviously need to be found, and this is not trivial. We for our part pursue this option nevertheless, as follows. We use an optimization technique to find those parameter values. This requires a reference point. Despite the limits mentioned in the previous paragraph this reference point is the ground truth. I.e., that optimization minimizes the distance between it and the result of the change-detection algorithm without compression.

3.4. Multi-objective Optimization

With a focus on result quality, optimizing change-detection and compression algorithms in combination has two objectives: low error rate of change detection and good compression ratio. In general, there are several kinds of methods to perform optimization with multiple objectives. One approach is to derive a single value, using for example a weighted sum. This is easy to implement, but finding appropriate weights highly depends on the specific use case (see Section 2) and is notoriously difficult. A more sophisticated, but at the same time more costly approach is multi-objective optimization, resulting in a Pareto frontier. We have chosen this second option because it is more informative.

3.5. Error Measure

Finding good parameter values requires a measure for the change-detection error. One can use a relatively simple measure, such as the number of correctly detected change points. An alternative is to calculate individual errors for paired changes, misses and false positives, and one can further refine this using application-specific weights. While this is markedly more complex, it also provides more insight. Because we aim to compare change points in detail, we choose the latter option. We use a framework providing that functionality, the MILTON

distance measure [12]. We note that [12] does not feature a systematic comparison of combinations of change detection and lossy compression techniques nor a discussion of respective setups.

4. Fundamentals

In this section we review the compression and change-detection algorithms covered in our study (Table 2). We also review two evolutionary algorithms. We then say how we quantify the deviation of change points. To make the distinction between the different categories of algorithms more obvious, we prefix the original acronyms accordingly: C for compression, D for change detection and O for optimization. Here we can only provide informal summary descriptions of those algorithms, but the publications describing them contain detailed explanations. They also specify the respective parameters. Further information as well as code is available on the project website[3].

Table 2: Overview of algorithms and their abbreviations

Compression Algorithms	
C_APCA	Adaptive Piecewise Constant Approx. [2]
C_SF	Slide Filter [13]
C_CHEB	Chebyshev Approximation [4, 5, 6]
C_WAVE	Wavelet Approximation [14]
C_PPA	Piecewise Compression Algorithm [15]
Change Detection Algorithms	
D_ADWIN	Adaptive Windowing [16]
D_ED	Event Detection from time series data [17]
D_CF	ChangeFinder [18]
D_OKCD	Online Kernel Change Detection [3]
D_BOCD	Bayesian Online Changepoint Detection [7]
Optimization Algorithms	
O_SOEA	Single Objective Evolutionary
O_NSGA-II	Non-Dominated Sorting Genetic [19]

4.1. Compression Methods

A broad review of the literature has resulted in the following categories of model-based compression algorithms: constant, straight-line or polynomial model compression. For each category we have chosen at least one representative, typically one which has received a lot of coverage in the scientific literature. Other points behind our choices of compression methods are as follows: First, we concentrate on methods that use the uniform norm (L_∞-norm), which, in contrast to, say, L_2, enables an error guarantee on each value compressed. Second, we are interested in the best compression (not segmentation) of the time series. We thus leave aside several well-known data-segmentation algorithms.

Adaptive Piecewise Constant Approximation (C_APCA) [2] adds data points to an adaptive window until the difference between the maximal and minimal value within this window exceeds a given threshold. Each window then is compressed by summarizing the data points as the arithmetic mean of its maximal and minimal value. Although C_APCA is a data representation, the underlying algorithm can be regarded as a compression method [8].

Slide Filter (C_SF) [13] makes use of several straight-line functions which approximate a set of data points. It starts by computing the values of these functions for a window consisting of two data points. Then more points are added to the window, while the functions which do not fulfill the error threshold anymore are left aside. This is continued until only one function remains. This remaining function then is the approximation of the window.

Chebyshev Approximation (C_CHEB) [4, 5, 6] tries to represent fixed size windows by a linear combination of Chebyshev polynomials up to a given dimension. If the approximation deviates more than the given error threshold, it stores the original data instead.

Wavelet (C_WAVE) [14] uses a discrete wavelet transform (DWT) to compress time series. The data goes through a low-pass filter and a bandpass filter to construct the corresponding continuous wavelet function. We deploy the existing MATLAB implementation that handles all lengths of time series, including those that are not powers of two.

Piecewise Polynomial Algorithm (C_PPA) [15] proposes a method which combines several compression methods. The goal of the algorithm is to approximate the data in a piecewise manner so that the compression ratio is maximized. It keeps adding data points to the current window until the error threshold is not met anymore. It then compresses this window using the best compression algorithm out of several ones.

4.2. Change Detection Techniques

Our study covers change-detection techniques of the following important categories: sequential analysis, maximum-likelihood estimation, kernel based techniques and Bayesian analysis techniques. Again, we have chosen one representative for each category.

Adaptive Windowing (D_ADWIN)[4] [16] uses a sliding window which is partitioned into buckets. Each bucket can contain several data points; it does so by storing their number and an aggregate of their values. Each time a data point is added to the window, it is put into a new bucket. When a certain number of buckets is reached, the two oldest buckets are merged. If the difference of the average values of two neighboring buckets exceeds a dynamic threshold, a change is reported and the last bucket is dropped. This dynamic threshold is computed for each comparison of two buckets. It depends on the difference of the numbers of data points of the two buckets.

Event Detection (D_ED) [17] is based on maximum likelihood estimation. It examines a data window to which data points are added step by step. In each step, it determines if the window can be split into two significantly different segments. Each segment then is approximated by fitting a model to it, and

[3] https://dbis.ipd.kit.edu/2434.php

[4] More specifically, we use ADWIN2, which is often referred to as ADWIN.

the error between the model and the data is determined. The point which minimizes this error for both segments is reported as change point. The models used are derived from base classes such as algebraic polynomials, radial, wavelet or Fourier. We for our part have chosen algebraic polynomials, just as in [17].

ChangeFinder (D_CF) [18] describes a two-stage algorithm which combines outlier detection and change detection. In a first stage, the algorithm learns an auto regressive (AR) model from a given time series. For each data point of the time series, a score is obtained by calculating the loss, be it the logarithmic one or the quadratic one. An outlier results in an isolated high score, while changes manifest themselves as series of high scores. Smoothing the scores removes the outliers. The smoothed values from the first AR model are then used to learn another AR model in the second stage of the algorithm. The scores of the second model describe the probability for data points being change points.

Online Kernel Change Detection (D_OKCD) [3] uses one-class support vector machines for change detection. For each data point of the time series, the immediate past subset $x_{t,1}$ and the immediate future subset $x_{t,2}$ are mapped into a feature space. A kernel method is used; it ensures that the mapped input space is a subset of a hypersphere with radius one, centered at the origin of the feature space. Support vector classification then finds hyperplanes in the feature space which separate the training vectors $\Phi(x_{t,1})$ and $\Phi(x_{t,2})$ from the center of the hypersphere. To decide whether a change point is present, the authors introduce a dissimilarity measure in feature space:

$$D_H = \frac{\overset{\frown}{c_{t,1}c_{t,2}}}{\overset{\frown}{c_{t,1}p_{t,1}} + \overset{\frown}{c_{t,2}p_{t,2}}}, \quad (1)$$

where $c_{t,1}$ and $c_{t,2}$ are the centers of the hypersphere sections intersected by the hyperplanes, and $p_{t,1}$ and $p_{t,2}$ are two points where the hyperplanes intersect the hypersphere. The arc represents the arc distance between the two points. If the dissimilarity measure exceeds a given threshold, a change point is reported.

Bayesian Online Changepoint Detection (D_BOCD) [7] uses a Bayesian approach. It divides a time series into partitions and assumes that for each partition there is an i.i.d. probability distribution of the data values. Thus, the change points are the boundaries between the partitions. For each new data point, the algorithm estimates the probability distribution since the last change point and then computes the probability that the new point belongs to this distribution. When this probability drops suddenly, a change is reported.

To summarize our selection of change-detection algorithms, we first repeat that they are based on different models. See Table 3 for key characteristics. Second, the algorithms have different runtimes and memory requirements. For instance, with W being the length of the current window, D_ADWIN has $O(log(W))$ runtime and memory requirements, while D_ED has $O(W^2)$ runtime and $O(W)$ memory requirements.

At first sight, it might be interesting to study the influence of the parameter values of the change-detection methods as well.

Table 3: Key characteristics of change-detection algorithms

Algorithms	Key characteristics
D_ADWIN	• uses adaptive windowing • fast and memory efficient
D_ED	• uses maximum likelihood estimation • fits models (polynomials) to data
D_CF	• works in two stages using autoregressive models • combines outlier detection and change detection
D_OKCD	• partitions and maps data in feature space • uses support vector machines clasification
D_BOCD	• online algorithm • estimates probability distributions

However, with five change-detection methods, five compression schemes and (as we will explain later) five different data sets, the space of configurations to examine has 125 elements already. Adding several other dimensions to it would exceed the scope of this study. Finding meaningful ways to narrow down this extended space is future work.

4.3. Optimization Techniques

On the technical level, some decisions like choosing an optimization algorithm have been necessary as well. We for our part use evolutionary algorithms. NSGA-II [19] is our choice for multi-objective optimization. Calculating benchmark change points needs only single-objective optimization (see *Change Point Baseline* in Section 3), which leads us to the faster O_SOEA algorithm.

Single Objective Evolutionary Algorithms (O_SOEA) start with a random set of problem solutions, referred to as initial population. The objective is to identify individuals, i.e., solutions, with low fitness. In each generation step, the individuals are sorted by their fitness, and two parents are randomly selected from among the top τ percent. They create two children by crossing over, and these children are mutated with a certain probability. Additionally a new random individual is introduced in each generation. The three newly created individuals replace the three individuals with the worst fitness. The algorithm terminates after a certain number of generations, or when the fitness falls below a certain threshold.

Non-dominated Sorting Genetic Algorithm [19] is an evolutionary optimization algorithm with multi-objective support (O_NSGA-II). It approximates a Pareto-optimal frontier over several generations. It starts with an initial, random population, and each generation categorizes the individuals into fronts, sorts the individuals within these fronts and uses the best individuals to create a new population, which then are added to the population of the next generation. More specifically:

1. An individual belongs to a front if there does not exist another individual in this current or in any previous front dominating it. An individual x dominates another individual y if and only if x is never inferior to y in any objective and x is superior to y in at least one objective.

2. For the sorting within the fronts, a so-called density value is assigned to each individual. It quantifies the density of solutions surrounding this individual.
3. The best individuals are chosen based on front and density. They are used to create a new population by means of recombination and mutation.

After several generations, the first front typically is a nearly Pareto-optimal frontier.

4.4. Measure for Quantifying the Impact of Lossy Transformations on Subsequent Change Detection (MILTON)

An important constituent needed for a study such as ours is a measure quantifying the difference of two time series containing change points cp and $\hat{cp}$. d_{MILTON} is such a measure [12]. It categorizes the changes as paired changes (PC), false positives (FP) and misses ($MISS$). Paired changes are changes which occur in both time series and are mapped to each other. False Positives are change points which occur in $\hat{cp}$ but not in cp, while misses are change points occurring in cp but not in $\hat{cp}$. For each of these categories an error is calculated ($errPC$, $errFP$, $errMISS$). These errors then are combined into a total one:

$$d_{MILTON}(cp, \hat{cp}) = \frac{errPC + errMISS + errFP}{|PC| + |MISS| + 1} \quad (2)$$

We explain our parametrization of MILTON in Section 5.

5. Experiment Setup and Initialization

In the following, we first present our framework (Section 5.1), followed by a summary of our notation (Section 5.2). We then describe the datasets used (Section 5.3). Our evaluation consists of three phases which build on each other, as shown in Figure 2. Phase 0 finds optimal parameters for a change-detection algorithm on a subsequence of an uncompressed dataset, to provide benchmark-change points (Section 5.4). Phase 1 (Section 5.5) uses the benchmark-change points to find good parameters of the compression and change-detection algorithms for any combination of dataset, compression algorithm and change-detection algorithm. This brings up the question under which circumstances the parameters found on a subsequence are also well suited for the complete dataset. We study this question, i.e., the validity and applicability of good parameters on complete datasets, in Phase 2 (Section 5.6).

We have additionally performed an experiment with a time series orders of magnitude larger than those we used in the phases described above. The goal of this experiment has been to show that our framework can handle long time series as well.

5.1. Framework

For the experimental evaluation we have designed and implemented a flexible generic framework which supports the different algorithms and is extensible to test further algorithms. We have integrated existing implementations whenever available.

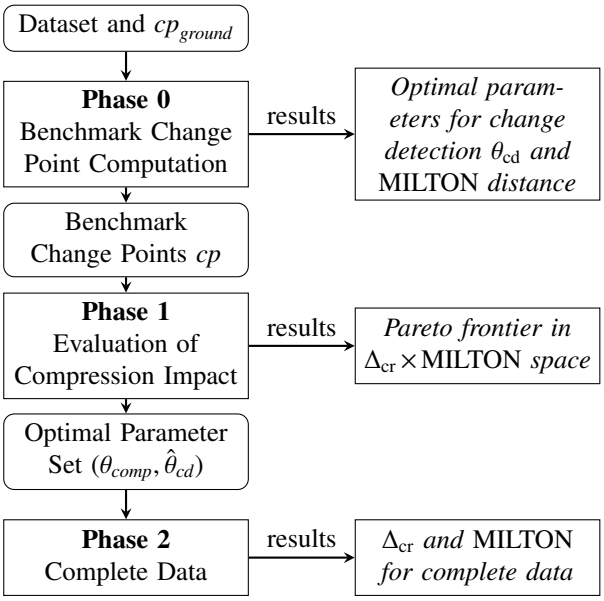

Figure 2: Overview of the experiments

For C_APCA, C_SF and C_CHEB we have used the implementations of [8][5]. The source code for D_ADWIN[6] and D_BOCD[7] is publicly available as well. For the wavelet compression we use a method from [14], which is part of the MATLAB libraries. We also reuse existing implementations of C_PPA and *MILTON*. We have implemented the remaining algorithms (D_ED, D_CF and D_OKCD) in MATLAB following the original publications, and they can be downloaded from our web page. Our framework handles algorithm implementations in C, C++, C#, R, MATLAB or Java.

The framework allows to define jobs for each experiment. A job consists of the algorithms chosen for compression, change detection and optimization, together with their parameters. It also includes the dataset and reference change points. Jobs cover the workflow of the experiments depicted in Figure 2. To distribute the work among several machines, the jobs are stored in a database where any free node can poll an open job. The results are then stored in the database as well.

5.2. Notation

In this paper $x(t) \in \mathbb{R}, t = 1, 2, \ldots, n$ is a real-valued one-dimensional time series. $cd(\cdot|\theta_{cd})$ stands for a change-detection algorithm with parameters θ_{cd}. By applying it to a time series x, we get $cp(t) \in \{0, 1\}, t = 1, 2, \ldots, n$ where

$$cp(t) = \begin{cases} 1 & \text{if } t \text{ is a change point} \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

We further define a transformation $trans(\cdot|\theta_{trans})$ with parameters θ_{trans}. This method takes the input time series x and creates

[5] lsirwww.epfl.ch/benchmark/, May 18, 2015
[6] https://github.com/abifet/adwin, May 18, 2015
[7] hips.seas.harvard.edu/content/bayesian-online-changepoint-detection, May 18, 2015

the time series $\hat{x}(t) \in \mathbb{R}, t = 1, 2, \ldots, n$, where each value $\hat{x}(t)$ is the result of a compression and subsequent decompression step:

$$\hat{x} = trans\,(x|\theta_{trans}) \tag{4}$$

Δ_{cr} is the compression ratio:

$$\Delta_{cr} = \frac{size\_of\_compressed\_data}{size\_of\_original\_data} \tag{5}$$

Note that *size_of_compressed_data* cannot be derived from $\hat{x}$. This is because it does not represent the compressed data. We summarize our notation in Table 4.

Table 4: Symbols used and their meaning

Symbol	Meaning
x	Original time series
$\hat{x}$	Compressed time series
cp_{ground}	Gound truth change points
cp	Benchmark change points on x
$\hat{cp}$	Comparison change points on $\hat{x}$
Δ_{cr}	Compression ratio
PC	Number of paired changes
FP	Number of false positives
$MISS$	Number of missed changes
$errPC$	total error of paired changes
$errFP$	total error of false positives
$errMISS$	total error of missed changes

5.3. Datasets

For the experiments we use artificial datasets as well as real world datasets, see Figures 3 and 4.

5.3.1. Synthetic datasets

We have generated the artificial datasets in line with earlier work [18, 20]. Our rationale here is to have well-defined change points of the types that are most frequent in reality, namely changes in mean or variance. We use an autoregressive function similar to the one in [20] which generates change points at every 200th point of time. This is in contrast to the real-world datasets below where changes occur irregularly. In the *Rising Mean* dataset we increase the mean of normally distributed noise by 1 at every change point. The *Variance Change* dataset alters the variance of the noise between 1 and 3 at a change point. The rationale is to study the behavior of the algorithms under another kind of change. For the additional experiment regarding long time series (Section 6.4), we used a time-series (*Long*) of one million points, which is much longer than the other time-series we used (e.g., 100 times larger than REDD). The time series we generated contains changes of random size at random points in time. Algorithms 1, 2, 3 contain the pseudo code generating this data, the companion web page contains it as MATLAB code.

$\mu \leftarrow 0, \sigma \leftarrow 1$
$x(0) \leftarrow 0, x(1) \leftarrow 0$
for $t \leftarrow 2$ *to* $t = length\_of\_dataset$ **do**
  $x(t) \leftarrow 0.6 \cdot x(t-1) - 0.5 \cdot x(t-2) + N(\mu, \sigma^2)$
  if t *mod* $200 = 0$ **then**
    $\mu \leftarrow \mu + 1$
  end
end

Algorithm 1: *Rising Mean*

$\mu \leftarrow 0, \sigma \leftarrow 1$
$x(0) \leftarrow 0, x(1) \leftarrow 0$
for $t \leftarrow 2$ *to* $t = length\_of\_dataset$ **do**
  $x(t) \leftarrow 0.6 \cdot x(t-1) - 0.5 \cdot x(t-2) + N(\mu, \sigma^2)$
  if t *mod* $200 = 0$ **then**
    $\sigma \leftarrow 4 - \sigma$
  end
end

Algorithm 2: *Variance Change*

5.3.2. Real-world datasets

Unfortunately, we are not aware of any large real-world dataset that is labeled so that it can be used in quality experiments on change detection. One reason is that, in real-world datasets, defining change points unambiguously is not possible in general. Data containing change-point annotations by hand as metadata exists nevertheless, and we use such data from different fields: EEG data, heart rate monitoring and electricity data. To account for real-world datasets with complex change points, we have included two additional datasets from the electricity domain. Here, instead of metadata annotated by hand, we have used an established change point detection method to obtain ground-truth change points. This is because changes are more diverse and irregular than with the simpler real-world data sets, so we deem intellectural annotations less reliable. Next, for the same reason, we expect subsequences of the complex data sets to differ from one another by much. Hence, we only use the complete complex datasets in our evaluation. We thus differentiate between simple real-world datasets and complex ones, as follows.

$nb\_changes \leftarrow 100$
$\sigma = 1$
$change\_times \leftarrow generate\_random\_values(nb\_changes)$
$change\_sizes \leftarrow generate\_random\_means(nb\_changes)$
for $t \leftarrow 1$ *to* $t = nb\_changes$ **do**
  for $k \leftarrow t$ *to* $k = change\_times(t+1)$ **do**
    $x(k) \leftarrow N(\mu = change\_sizes(t), \sigma^2)$
  end
end

Algorithm 3: *Long*

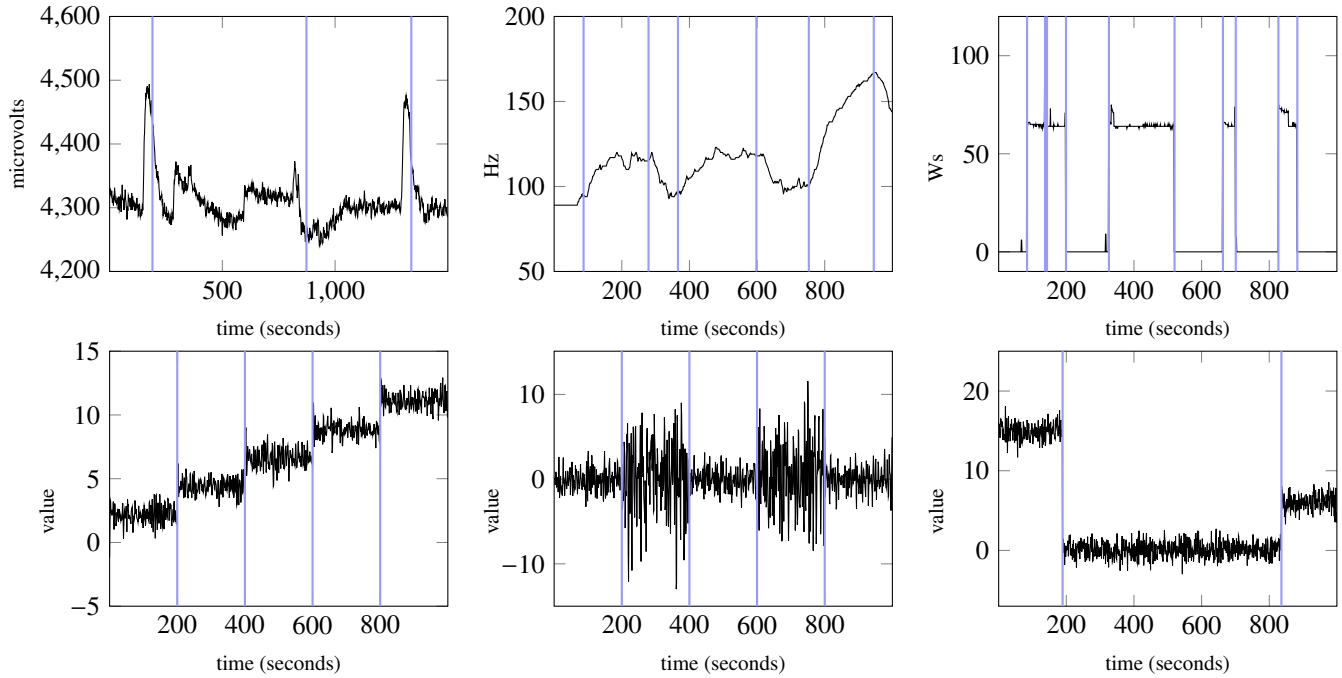

Figure 3: Plots of excerpts of all datasets including ground truth. Top row shows real world datasets: EEG, PAMAP and REDD (from left to right); bottom row shows the artificial datasets: Rising Mean, Variance Change and Long.

Table 5: Overview of datasets with their corresponding MILTON distance of initial parameter calculation, where $|CP|$ is the number of change points in the whole dataset and $|CP_{Seg}|$ in the segment

					d_{MILTON}				
Name	**Length**	$\|CP\|$	**Segment**	$\|CP_{Seg}\|$	**D_ADWIN**	**D_ED**	**D_CF**	**OKCD**	**D_BOCD**
Rising Mean	10000	49	1000	4	2.006	1.049	1.009	0.202	0.003
Variance Change	10000	49	1000	4	-	2.822	0.833	0.422	0.217
REDD	10000	144	1000	10	2.365	-	0.104	-	0.464
PAMAP	2280	13	1000	6	1.018	-	-	1.041	0.791
EEG	14979	23	1500	3	0.27	0.272	-	1.287	0.017

Simple real-world datasets. The EEG dataset[8] has been captured while the subject was opening and closing his eyes; this leads to a noticeable peak. We have removed a one-value outlier at 898 by interpolating the neighboring values to get more stable change-detection results.

The heart-rate dataset comes from the (PAMAP) project. More specifically, we use the outdoor dataset of Subject 2. It contains activities like sitting, walking, running or playing soccer. Since the data has been captured with 100 Hz, which is way above the resolution of the heart-rate monitor, we have reduced the dataset by using every hundredth data point, in order to reduce the data volume and the huge runtime of some of the change-detection algorithms.

We also use the REDD energy-consumption data[9]. It records the power consumption of a house broken down into different electrical consumers. For our evaluation we have selected Channel 17 of House 1, which is the lighting in one of the rooms. This is because the lighting is relatively independent of other household appliances.

Table 5 shows the lengths of the subsequences we have selected according to the *Training Data* design decision. For the EEG dataset the segment is slightly larger than for the other data. This is because the ground-truth change points are farther apart.

Complex real-world datasets. The REDD_ALL dataset is a complex version of the REDD dataset (Figure 4). Here, we have selected the per-minute aggregated power consumption of one of the households of the REDD dataset. As motivated earlier, we have annotated this data using the Student change-detection algorithm from the *cpm* R-package [21].

We have obtained the CREST dataset by using the high-resolution energy demand model from [22] (Figure 4). The model simulates appliance use and home occupancy in order to obtain data with characteristics similar to real-world household power consumption. As in the case of the REDD_ALL dataset, we have used the Student change-detection algorithm from the *cpm* R-package to annotate ground-truth change points.

[8] https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State, May 18, 2015
[9] http://redd.csail.mit.edu/, May 18, 2015

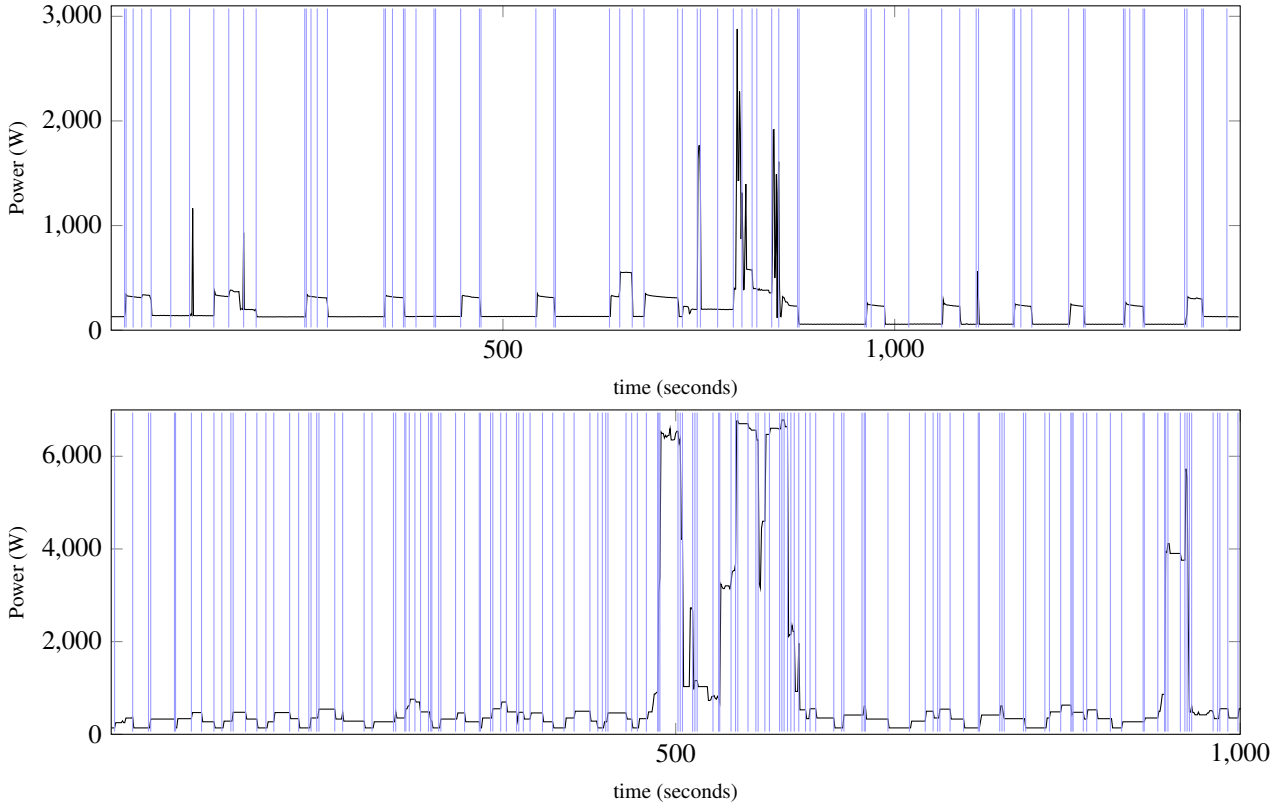

Figure 4: Plots of excerpts of real-world complex datasets including ground truths: above REDD_ALL, below CREST.

5.4. Phase 0: Benchmark Change Point Computation

Recall that Phase 0 is not an experiment in its own right, but an initialization step. It is described next. As stated under *Benchmark Change Points* in Section 3 on Design Decisions, we do not use the ground-truth change points as reference for our evaluation. Instead we use benchmark change points by minimizing the MILTON distance to the ground truth:

$$\arg\min_{\theta_{cd}} d_{MILTON}\left(cd\left(x|\theta_{cd}\right), cp_{ground}\right) \tag{6}$$

To this end, we use an O_SOEA. Table 6 shows its parameterization.

We have executed the further phases only for combinations of change-detection techniques and datasets which lead to acceptable results. We deem a result acceptable if the number of paired changes exceeds the one of false positives and the one of misses, so that most of the original change points are found. To facilitate a comparison, Table 5 lists the MILTON distance against the ground-truth change points. '-' stands for results that have not been accepted. Most algorithms have found a parametrization on Rising Mean and Variance Change. The D_ADWIN and Variance Change combination does not have a result, because D_ADWIN only finds changes of mean values [16]. As expected, some change-detection algorithms do not perform well on some real-world datasets. This is because their ground truth is based on secondary observations that do not necessarily cause a change in the data at exactly the same time.

Table 6: List of parameters for the optimization algorithms, fitness function and MILTON distance with their corresponding values used for our experiments.

Algorithm	**Parameter**	**Value**
O_SOEA	population size	100
	exit fitness	0.001
	max. generations	500
	mutation rate	0.2
	mutation change	0.4
	selection pressure τ	0.4
O_NSGA-II	population size	500
	exit fitness	0.001
	max. generations	10
	mutation rate	0.2
	mutation change	0.4
Fitness	weight α	0.5
MILTON	$f_{TIME}\left(\Delta_t\right)$	$\lvert\Delta_t\rvert$
	$f_{SCORE}\left(\Delta_s\right)$	0
	$f_{MISS}\left(s\right)$	$(s+1)^2$
	$f_{FP}\left(s\right)$	s

5.5. Initialization of Phase 1: Evaluation of Compression Impact

An important goal of our evaluation is to determine an optimal set of parameters which preserves the change points

cp found before compression as much as possible while maximizing the compression at the same time. This is a multi-objective optimization problem with the objectives $d_{MILTON}(cp, \hat{cp})$ and compression ratio Δ_{cr}, where $\hat{cp} = cd(\hat{x}|\hat{\theta}_{cd})$. The parameter space consists of the parameters for the compression and the change-detection algorithm: $\theta = (\theta_{trans}, \hat{\theta}_{cd})$. To evaluate the algorithms we study all possible combinations on each dataset.

To find optimal parameter sets we use an adaptation of NSGA-II described in the next paragraph. See Table 6 for the parameterization of NSGA-II. The weighting functions for the errors in the MILTON distance are important as well. See again Table 6, with functions similar to [12]. In a nutshell, these functions ($f_{TIME}, f_{SCORE}, f_{MISS}, f_{FP}$) determine the weight given to the different kinds of errors caused by compression. In our case, for example, a shift in time of a change (f_{TIME}) is weighted proportionally to its absolute value.

The parameters of the change-detection and compression algorithms have a range of validity which must be kept during the optimization. Therefore we modify NSGA-II to calculate its random values in the range $[\phi_{min}, \phi_{max})$ during the initialization of the population and for mutations. For some parameters we have reduced this range even further to reduce the search space and to speed up the optimization process. Tables 7 and 8 list the parameters and the ranges we have selected. An additional modification is the distinction between float and integer parameters. Random values for the float parameters are calculated as $\phi' = \phi_{min} + r \cdot (\phi_{max} - \phi_{min})$, where r is an equally distributed random number in [0; 1). For integer parameters this value is then rounded: $\phi' = \lfloor \phi' \rfloor$.

Table 7: List of parameters for all compression algorithms and the ranges of the optimization

Algorithm	Parameter	Min	Max	Type
All methods	threshold ϵ	0	0.3	float
C_CHEB	segment length	4	$\lvert x \rvert$	int
C_WAVE	max. level	1	10	int
C_PPA	max degree	2	5	int

The result of this phase is a Pareto frontier that represents the best possible trade-offs between compression ratio and the preservation of change points. Each individual consists of the MILTON distance, the compression ratio and the root-mean-square error (RMSE) calculated for the corresponding set of compression and change-detection parameters. To select an individual of this frontier suitable for a specific application scenario, a *fitness* is calculated:

$$fitness = \alpha \cdot d_{MILTON}(cp, \hat{cp}) + (1 - \alpha) \cdot \Delta_{cr} \quad (7)$$

where α is a parameter to weigh the addends. Note that α is used only to select an individual in the result set *after* the optimization is finished. This provides a lot of freedom and flexibility during the evaluation.

[10] In our case *MSet* specifies the maximum degree of the polynomials for the approximation (cf. Subsection 4.2).

Table 8: List of parameters for all change detection algorithms and the ranges of the optimization

Algorithm	Parameter	Min	Max	Type
D_ADWIN	M	2	10	int
	δ	0	1	float
D_ED	δ	0	1	float
	p	1	200	int
	MSet [10]	0	5	int
D_CF	T	3	10	int
	k	2	10	int
	r	0	0.4	float
D_OKCD	m_1	2	200	int
	m_2	2	200	int
	ν	0.2	0.8	float
	η	0	1	float
	σ	0	1	float
D_BOCD	μ_0	0	2	float
	κ_0	0	5	float
	α_0	0	5	float
	β_0	0	5	float
	λ	> 0	500	float

5.6. Initialization of Phase 2: Complete Datasets

Section 3 has explained the necessity to evaluate the best performing combination of compression and change detection on a subsequence but also on the complete dataset. The details of the experimental setup when it comes to the complete dataset are as follows: The parameters for a specific α of the Phase 1 experiment are applied to the complete dataset. For the Rising Mean dataset, we have divided the ϵ threshold by 10, because it depends on the global maximum that is 10 times higher on the complete Rising Mean dataset. As a reference, we use the parameters computed in Phase 0 (Section 5.4) on the complete dataset. Then the MILTON distance d_{MILTON}, compression ratio Δ_{cr} and RMSE are calculated.

6. Results

This section first describes and discusses our results of the Phase 1 experiments and then presents the results of Phase 2 on the complete data. As an initial, exemplary illustration, Figure 5 visualizes the data transformation in the different phases for the combination REDD, D_BOCD and APCA. The top plot, although not the focus of our study, shows the raw data with ground-truth change points as vertical straight lines. The middle plot shows the benchmark change points calculated in Phase 0. The bottom plot shows the change points on the compressed data with the best result parameters of Phase 1. Comparing the top plot to the middle plot, we can see that D_BOCD fails to detect two changes and incorrectly identifies two other ones. Taking benchmark change points as a reference when run on the compressed data, i.e., comparing the middle plot to the bottom plot, D_BOCD fails to detect two changes. Thus, the decrease in the performance of D_BOCD on compressed data is small in this case, compared to its performance on the original data.

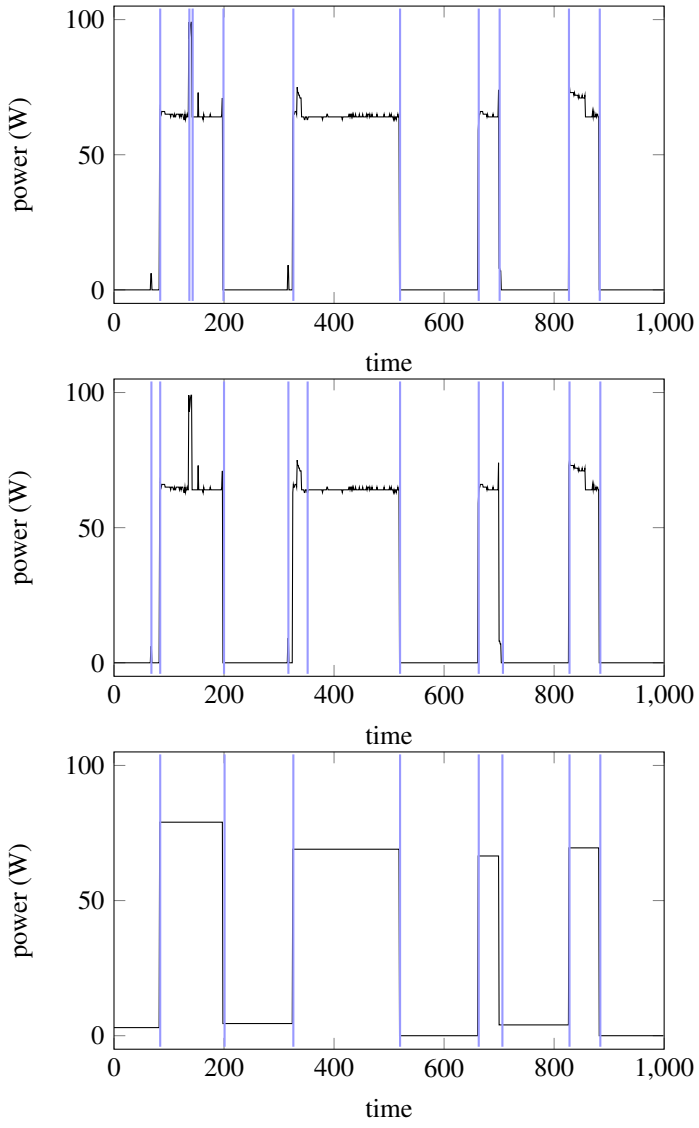

Figure 5: D_BOCD and C_APCA on REDD. Top down: raw data with ground-truth change points, result of benchmark-change point computation, best result of change detection with compression ($\alpha = 0.5$).

6.1. Phase 1 – Results

The results of the Phase 1 experiments are Pareto frontiers. To illustrate, Figure 6 shows a sample of the Pareto frontiers on the Variance Change dataset. Each plot contains all compression techniques for one change-detection algorithm, except for D_ADWIN, which is not applicable to this dataset (see Section 5.4). There is not any frontier dominating all other frontiers, therefore no single best solution exists. Dependent on the dataset and parameter α, different combinations of change-detection and compression algorithm yield the best result.

We observe that comparing the large number of Pareto frontiers produced by our experiments is difficult. Thus, we select the individuals with the lowest fitness (see Equation 7) of each Pareto frontier for different values of α and compare their MILTON distance and compression ratio in Figures 7 and 8.

Earlier we have described two scenarios (SmartGrid, IoT)

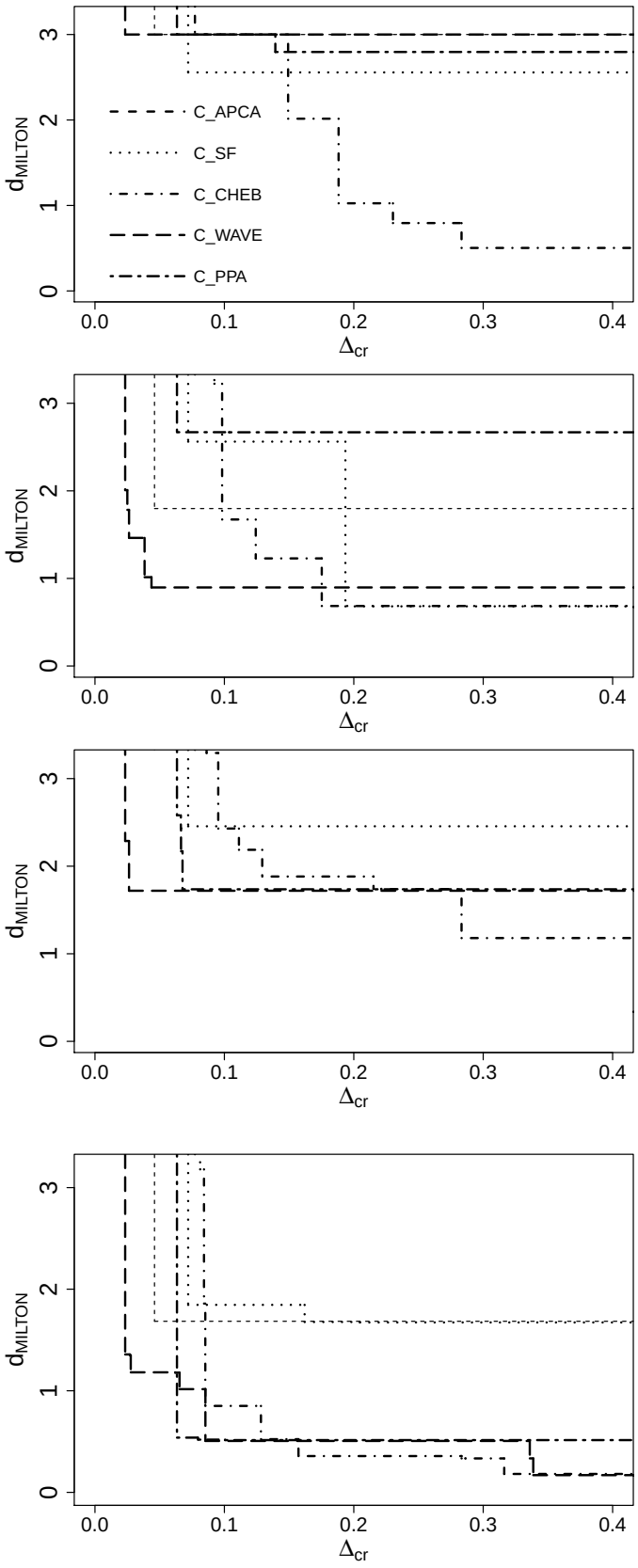

Figure 6: Pareto frontiers for D_ED,D_CF,D_OKCD and D_BOCD (from top to bottom) on the *Variance Change* dataset.

where an approach such as ours is indispensable if one wants to find a good combination of compression and change-detection algorithms. These scenarios have different requirements. We therefore examine the Pareto frontiers for two different α values, $\alpha = 0.5$ for the Smart Grid, and $\alpha = 0.05$ for IoT. For each solution on the Pareto frontier we calculate a fitness value using the respective α value. We have chosen the parametrization with the lowest fitness that indicates the best result for the scenario. We get a triple (*fitness*, MILTON distance, Δ_{cr}) for each experiment. See Figures 7 and 8 for the MILTON distance and Δ_{cr} values, for $\alpha = 0.5$ and $\alpha = 0.05$. For several datasets, some change-detection algorithms (e.g., D_ED on REDD) did not detect any changes or have given very poor results. We therefore did not include them in Figures 7 and 8. Figure 8 contains the results only on the Variance Change dataset. The reason is that for two datasets (EEG and PAMAP) the results for both values of α are identical, while for the other two they are very similar. The MILTON distance is the value above the horizontal axis, the compression ratio is below. (For both d_{MILTON} and Δ_{cr}, lower values are better.) Using the corresponding value of α, the value of the fitness can be retrieved. For instance, for Figure 7 ($\alpha = 0.5$), it is the average of d_{MILTON} and Δ_{cr}. For this value of α, the best combinations of compression and change-detection algorithm for each dataset are as follows.

Synthetic datasets:

- Rising Mean: D_BOCD with C_APCA clearly is the best solution. This is because it achieves a MILTON distance of almost zero and also has the best compression ratio.
- Variance Change: The best fitness is obtained with D_BOCD and C_CHEB, closely followed by D_BOCD and C_PPA, although the compression ratio is not optimal.

Simple real-world datasets:

- REDD: D_CF with C_SF is the best combination. This is because it has a close-to-zero MILTON distance and a very good compression ratio.
- PAMAP: The best algorithms are D_BOCD and C_CHEB, mainly because of the low MILTON distance. D_BOCD with C_WAVE also performs very well.
- EEG: D_BOCD with C_SF clearly is optimal.

Overall D_BOCD performs very well on all datasets, be they synthetic, be they real, and is only beaten once by D_CF. We have made further noteworthy observations:

- A MILTON distance larger than 3 means that no change points have been found after compression. Thus, D_CF on Rising Mean and D_ADWIN on the EEG data do not work at all.
- The Variance Change dataset has relatively high compression ratios. This is because the changes in variance are difficult to compress, especially on combinations D_OKCD with C_PPA and D_ED with C_CHEB.
- The REDD dataset is easy to compress while keeping the change points, because of its very sharp edges and low signal-to-noise ratio. It therefore has the lowest average fitness of all datasets.

Comparing the results for $\alpha = 0.05$ (Figure 8) to the ones above, the compression ratios are smaller. However, the MILTON distances are higher. Both effects are expected. On the Variance Change dataset, the results with the lowest MILTON distance for $\alpha = 0.5$ has a very high compression ratio. For $\alpha = 0.05$, those combinations yield a much lower compression ratio.

On PAMAP and EEG the solutions for $\alpha = 0.5$ and $\alpha = 0.05$ are the same. The best results for $\alpha = 0.05$ from Rising Mean and REDD Data are identical to those from $\alpha = 0.5$. This is because their low compression ratios are not reduced further. Some of the other results have slightly lower compression ratios.

Our takeaway is that our experiments do indeed help to find solutions for specific use cases. The results of this phase also show that the quality differs a lot between different combinations in the same setting. Thus it is very important to be able to study different combinations using a setup as elaborate as ours.

We have carried out further experiments to illustrate the influence of α on the fitness function. As a reminder, α is not used in the optimization process. It is used to select an individual in the result after the optimization is finished. Figure 9 shows the value of the fitness function depending on α for the combination of D_BOCD and C_CHEB on the REDD dataset. The crosses in the figure represent the cases where a different individual is chosen, since its fitness now is the minimal one. We see that there are three such cases and four corresponding individuals overall. Thus, by choosing an appropriate α based on the requirements of the application scenario, we can flexibly adjust the choice of the individual for any values of d_{MILTON} and Δ_{cr} in order to, say, eliminate the possible dominance of one over the other.

Lessons Learned. Finally, we have found some general heuristics regarding the change-detection algorithms, as follows:

- D_ADWIN performs well on the Rising Mean and the REDD datasets. These datasets contain sharp and sudden changes. In contrast, D_ADWIN performs worst on PAMAP and EEG, where changes are of a different nature. Here, compressing the data makes changes practically undetectable for D_ADWIN.
- All in all, D_BOCD has a stable performance on all datasets and most compression algorithms. In the case where it performs worse than other algorithms, it does so only slightly. We therefore conclude D_BOCD to be a rather robust algorithm in compression scenarios.
- The performance of D_ED and D_CF strictly depend on the dataset, and it is hard to find characteristics of the datasets for which these algorithms perform well. For instance, D_CF gives the best results on the REDD dataset, while it performs the worst on the Rising Mean dataset, although we think that these datasets are rather similar.

Table 9: Comparison of dataset excerpts (left) and complete datasets (right) using the same parameters.

Best fitnesses							
Combination	d_{MILTON}	Δ_{cr}	**RMSE**	**Fitness**	**|PC|**	**|FP|**	**|MISS|**
Rising Mean: C_APCA, D_BOCD	0.000 | 0.674	0.010 | 0.011	0.827 | 0.690	0.010 | 0.342	4 | 48	0 | 3	0 | 9
Variance Change: C_CHEB, D_BOCD	0.181 | 1.716	0.316 | 1.031	1.559 | 0.000	0.249 | 1.374	5 | 75	1 | 39	0 | 40
REDD: C_SF, D_CF	0.001 | 0.957	0.041 | 0.048	4.449 | 4.479	0.021 | 0.503	11 | 126	0 | 9	0 | 37
PAMAP: C_CHEB, D_BOCD	0.177 | 0.476	0.039 | 0.039	4.585 | 4.873	0.108 | 0.258	6 | 13	1 | 3	0 | 1
EEG: C_SF, D_BOCD	0.038 | 3.104	0.030 | 0.017	14.95 | 17.305	0.034 | 1.560	3 | 9	0 | 30	0 | 1

Best generalization							
Combination	d_{MILTON}	Δ_{cr}	**RMSE**	**Fitness**	**|PC|**	**|FP|**	**|MISS|**
Rising Mean: C_APCA, D_ED	0.021 | 0.432	0.014 | 0.018	0.698 | 0.669	0.035 | 0.225	4 | 43	0 | 1	0 | 5
Variance Change: C_SF, D_CF	0.681 | 0.580	0.194 | 0.182	2.153 | 2.422	0.435 | 0.762	7 | 56	2 | 5	1 | 8
REDD: C_WAVE, D_CF	0.083 | 0.244	0.213 | 0.207	0.921 | 1.021	0.255 | 0.225	11 | 155	1 | 8	0 | 8
EEG: C_PPA, D_BOCD	0.772 | 1.919	0.005 | 0.002	36.01 | 43.65	0.333 | 0.960	3 | 10	3 | 21	0 | 0

6.2. Phase 2 – Results

As explained in Section 5.6, we apply the parameters of the results from Phase 1 to the complete data. For every set we choose the combination with the lowest fitness. We expect that those parameter sets will achieve the same quality of results, mainly in terms of stable ratios between PC, FP and MISS, as on the subsequence datasets. Since d_{MILTON} grows quadratically for the number of FP and linearly for the one of MISS (see Table 6), it is hardly comparable on the complete dataset. On the other hand, we expect that the compression ratio stays constant.

Table 9 shows our results. We do not discuss these specific results separately for synthetic and simple real-world datasets because results are similar regardless of the nature of the data. This means that the characteristics of simple real-world change points do not influence the results of Phase 2. We also do not include results for the PAMAP dataset because none of the results corresponds to a good generalisation on this dataset. In contrast to our expectation, there are disproportionally more FPs and misses than on small sets. From EEG we conclude that three change points are not sufficient to train the change detection properly. The O_NSGA-II overfits the parameters on the training data. To avoid this, we have also tested the other combinations which do not show the best but nevertheless good fitness. The lower part of Table 9 shows that these parameters yield results on the complete dataset which are as good as the ones on the excerpts. On REDD for instance, ChangeFinder detects eleven times as many change points but only eight times as many FPs. In contrast to Section 6.1, D_BOCD performs best in only two out of five cases. In every case the compression ratio differs only slightly. To sum up, we can say that the results from a short subsequence can be used on the complete dataset without losing quality. However, it is necessary to rely on several results from Phase 1 to find the ones adapting best.

6.3. Complex Real-world Datasets

Table 10 is an overview of the best solutions for $\alpha = 0.5$ on the combinations of all compression algorithms and D_ADWIN and D_OBCD for the complex real-world datasets. Results are in most cases similar for both datasets. The combinations with the best fitness for REDD_ALL and CREST are obtained with D_BOCD, and C_CHEB and C_WAVE, respectively. This confirms the result obtained with synthetic and simple real-world datasets, where D_BOCD has had a stable performance.

We also notice that, for certain combinations of compression and change-detection algorithms, results are much worse than for the previous datasets. As an example, D_ADWIN and C_APCA obtains a $MILTON$ distance of around 58 in the best case, which is around one order of magnitude higher than for all other combinations. Apparently, this is because compressing complex real-world changes using simple polynomials such as constant functions does alter changes significantly.

Table 10: Overview of the best solutions for $\alpha = 0.5$ on each combination of compression algorithm, change detection algorithm and complex real-world dataset.

	Compression algorithm	Change-detection algorithm	Δ_{cr}	d_{MILTON}
REDD_ALL	D_ADWIN	C_APCA	0.02083	58.2486
	D_ADWIN	C_CF	0.01875	6.4055
	D_ADWIN	C_CHEB	0.2861	3.8407
	D_ADWIN	C_PCA	0.1097	5.2491
	D_ADWIN	C_WAVE	0.0851	6.9779
	D_BOCD	C_APCA	0.01389	62.4420
	D_BOCD	C_CF	0.0141	16.3038
	D_BOCD	C_CHEB	0.1410	0.5881
	D_BOCD	C_PCA	0.671	1.9527
	D_BOCD	C_WAVE	0.0747	5.4674
CREST	D_ADWIN	C_APCA	0.0167	50.1308
	D_ADWIN	C_CF	0.0094	6.6849
	D_ADWIN	C_CHEB	0.2826	4.1020
	D_ADWIN	C_PCA	0.0051	20.3947
	D_ADWIN	C_WAVE	0.1114	5.1926
	D_BOCD	C_APCA	0.0375	52.0951
	D_BOCD	C_CF	0.0094	17.9635
	D_BOCD	C_CHEB	0.3770	2.9814
	D_BOCD	C_PCA	0.0088	39.5499
	D_BOCD	C_WAVE	0.0625	4.7573

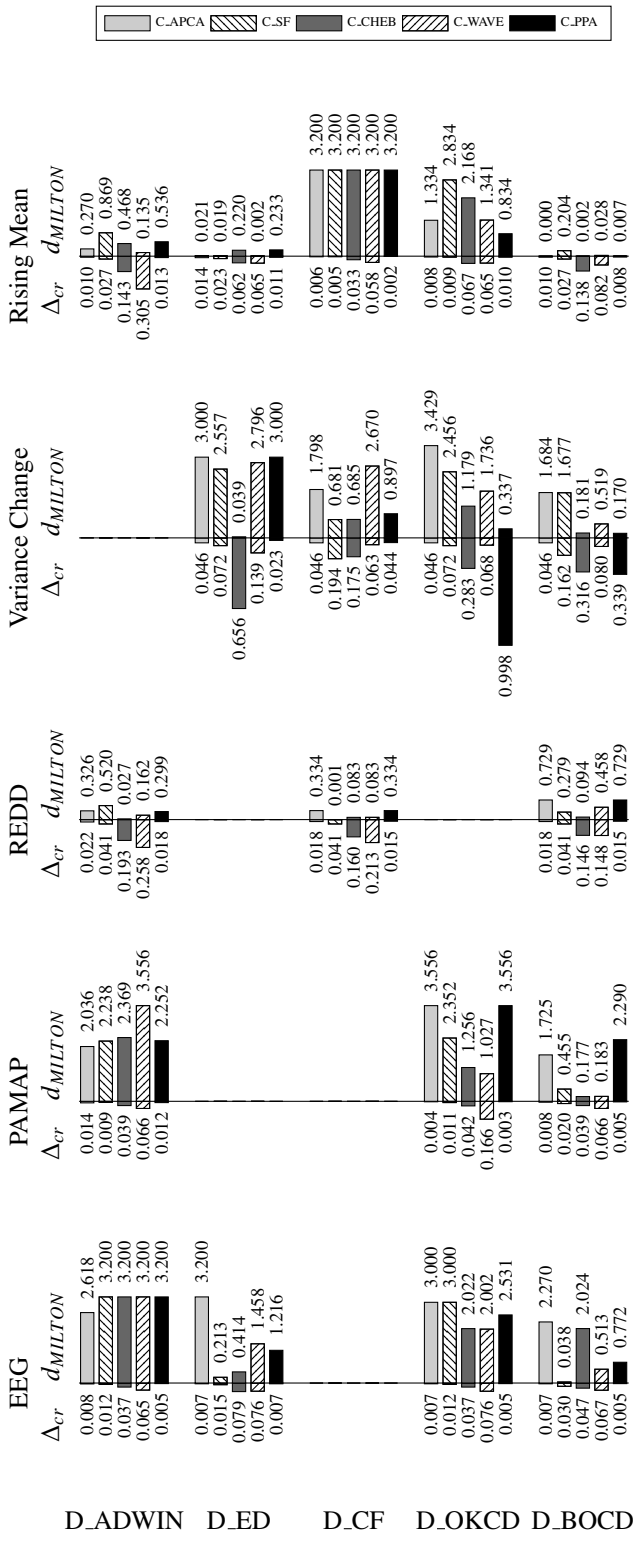

Figure 7: Overview of the best solutions for $\alpha = 0.5$ on each combination of compression algorithm, change detection algorithm and dataset.

6.4. Experiment with Long Time Series

The goal of this experiment is to use our framework with a time series significantly bigger than the ones we used in Phases 1 and 2 and show that our framework is applicable to such time series as well. We use one combination of a compression and change detection algorithm on the *Long* time series, resulting in the Pareto front in Figure 10. As expected, we obtain sets of trade-off solutions. These are useful to choose adequate parameters for the compression depending on the requirements of the application scenario.

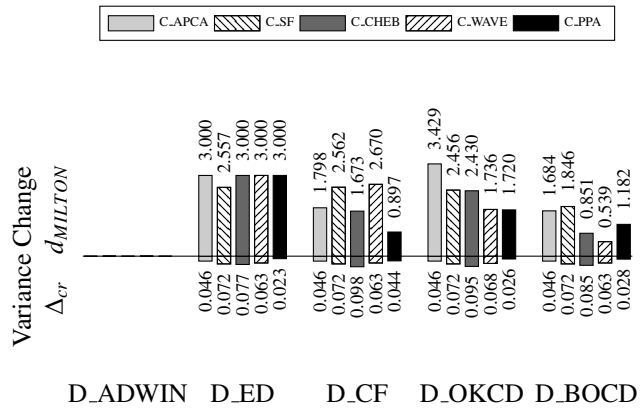

Figure 8: Solutions for $\alpha = 0.05$ on Variance Change.

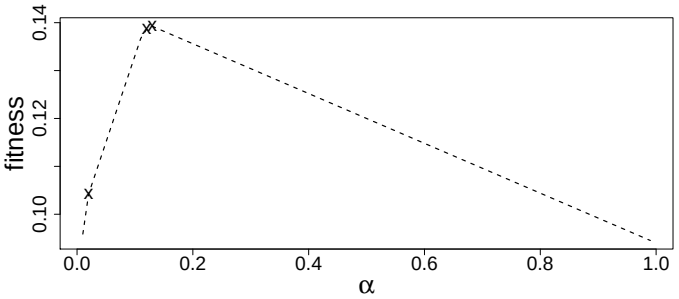

Figure 9: Fitness vs. α for D_BOCD and C_CHEB on the REDD dataset

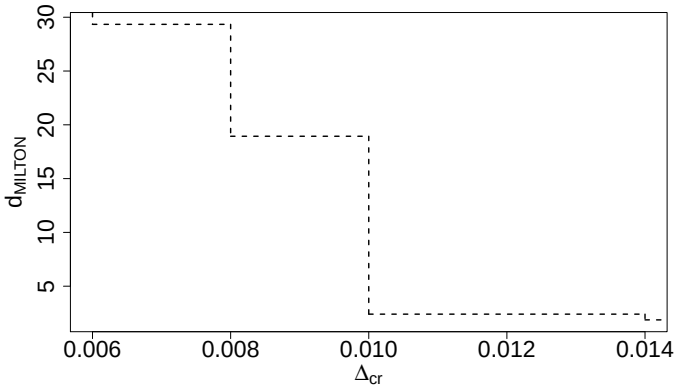

Figure 10: Pareto front for the *Long* time series

7. Conclusions

In many situations, compression and change-detection methods must be used in combination. In such a setting however, a number of questions are unclear, e.g.: Which combination is best for a given scenario? How to find a good parameter-

ization of compression and change-detection algorithms when these are used together? How well can we trade compression ratio against change-detection quality? This article has featured a comprehensive experimental evaluation that addresses these questions.

A study such as ours requires a number of non-trivial design decisions. This article has listed the important issues, together with the respective options and our rationale behind the 'winner' alternatives.

An important insight is that the overall picture is very differentiated. Result quality highly depends on the dataset. For instance, the change-detection method ChangeFinder is the best performing algorithm on REDD, but the worst performing one on the Rising Mean dataset. Our platform has turned out to be an appropriate tool to find good parameterizations, at least if the dataset inspected is sufficiently representative and large.

When data is compressed, the intention always is to decompress it later and use it in some way. Change detection is one kind of data usage, but other kinds of usage obviously abound and are important as well. Just think of the plethora of different stream-mining approaches which have been proposed in the recent past. Generalizing the work described here to other kinds of usage is important and is part of our future work.

8. References

[1] M. Ringwelski, C. Renner, A. Reinhardt, A. Weigel, V. Turau, The hitchhiker's guide to choosing the compression algorithm for your smart meter data, in: Energy Conference and Exhibition (ENERGYCON), 2012 IEEE International, 2012.

[2] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, Locally adaptive dimensionality reduction for indexing large time series databases, ACM SIGMOD Record, 2001, 30 (2).

[3] F. Desobry, M. Davy, C. Doncarli, An online kernel change detection algorithm, IEEE Transactions on Signal Processing, 2005, 53 (8).

[4] Y. Cai, R. Ng, Indexing spatio-temporal trajectories with chebyshev polynomials, in: Proceedings of the 2004 ACM SIGMOD.

[5] A. F. Cheng, Hawkins III, S Edward, L. Nguyen, C. A. Monaco, G. G. Seagrave, Data compression using chebyshev transform (2007).

[6] A. Arion, H. Jeung, K. Aberer, Efficiently maintaining distributed model-based views on real-time data streams, in: Global Telecommunications Conference (GLOBECOM 2011), IEEE.

[7] R. P. Adams, D. J. C. MacKay, Bayesian online changepoint detection, arXiv preprint arXiv:0710.3742, 2007.

[8] N. Q. V. Hung, H. Jeung, K. Aberer, An evaluation of model-based approaches to sensor data compression, IEEE Transactions on Knowledge and Data Engineering, 2013, 25 (11).

[9] Communication Networks for Smart Grids, Computer Communications and Networks, Springer London, 2014.

[10] I. F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Communications Magazine, 2002, 40 (8).

[11] D. Miorandi, S. Sicari, F. d. Pellegrini, I. Chlamtac, Internet of things: Vision, applications and research challenges, Ad Hoc Networks, 2012, 10 (7).

[12] P. Efros, E. Buchmann, A. Englhardt, K. Böhm, How to quantify the impact of lossy transformations on change detection, in: Proceedings of the 27th International Conference on Scientific and Statistical Database Management, 2015.

[13] H. Elmeleegy, A. K. Elmagarmid, E. Cecchet, W. G. Aref, W. Zwaenepoel, Online piece-wise linear approximation of numerical streams with precision guarantees, Proceedings of the VLDB Endowment, 2009, 2 (1).

[14] M. Vishwanath, The recursive pyramid algorithm for the discrete wavelet transform, IEEE Transactions on Signal Processing, 1994, 42 (3).

[15] F. Eichinger, P. Efros, S. Karnouskos, K. Böhm, A time-series compression technique and its application to the smart grid, The VLDB Journal, 2015, 24 (2).

[16] A. Bifet, R. Gavaldà, Learning from time-changing data with adaptive windowing, in: Proceedings of the 2007 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics.

[17] V. Guralnik, J. Srivastava, Event detection from time series data, in: SIGKDD, 1999.

[18] J. Takeuchi, K. Yamanishi, A unifying framework for detecting outliers and change points from time series, IEEE Transactions on Knowledge and Data Engineering, 2006 18 (4).

[19] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2).

[20] Y. Kawahara, M. Sugiyama, Change-point detection in time-series data by direct density-ratio estimation, in: Proceedings of the 2009 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics.

[21] R. Killick, I. Eckley, changepoint: An r package for changepoint analysis, Journal of Statistical Software 58 (2014) 1–19.

[22] I. Richardson, M. Thomson, D. Infield, C. Clifford, Domestic electricity use: A high-resolution energy demand model, Energy and Buildings 42 (2010) 1878–1887.