How to Quantify the Impact of Lossy Transformations on Change Detection

Pavel Efros Erik Buchmann Adrian Englhardt Klemens Böhm

Karlsruhe Institute of Technology (KIT), Germany
{pavel.efros, erik.buchmann, klemens.boehm}@kit.edu
adrian.englhardt@student.kit.edu

ABSTRACT

To ease the proliferation of big data, it frequently is transformed, be it by compression, be it by anonymization. Such transformations however modify characteristics of the data, such as changes in the case of time series. Changes however are important for subsequent analyses. The impact of those modifications depends on the application scenario, and quantifying it is far from trivial. This is because a transformation can shift or modify existing changes or introduce new ones. In this paper, we propose MILTON, *a flexible and robust Measure for quantifying the Impact of Lossy Transformations on subsequent change detectiON*. MILTON is applicable to any lossy transformation technique on time-series data and to any general-purpose change-detection approach. We have evaluated it with three real-world use cases. Our evaluation shows that MILTON allows to quantify the impact of lossy transformations and to choose the best one from a class of transformation techniques for a given application scenario.

1. INTRODUCTION

Change detection on time series data is an important building block of many real-world applications [28, 17]. It converts a time series of measurements into one of events. Think of energy-consumption data from a smart meter, which serves as our running example. Change detection on such data allows to detect interesting events (turning on/off of a device, abnormal device activity). Such events are needed for demand side management, peak shifting, peak shaping, etc. – all basic techniques to integrate renewable energy sources into the Smart Grid. However, data transformation, e.g., lossy compression or anonymization, can modify the data considerably. This can significantly impact the subsequent detection of those events.

EXAMPLE 1. *An energy provider uses a lossy compression technique for time series from a smart meter, to reduce the data volume, before running a change-detection algorithm. Due to the compression loss, (a) some changes might be detected at different points in time, or (b) their significance might be altered, compared to the original time series. Next, (c) changes might go undetected at all, or (d) the compression might result in new changes. Using his domain knowledge, the provider can assess the importance of these impacts. Based on his assessment, he wants to select a concrete compression technique, together with a good parameter set.*

Due to the volume of the data, the complex semantics of changes to be detected and possible privacy violations [6], change detection on smart-meter data is a "Big Data" problem. Big data approaches like lossy compression [11], estimation [10] or perturbation/anonymization [29] lossily transform the time series before doing change detection: A lossy transformation can reduce the data volume, generate an optimized data model or remove personal information from a dataset. However, existing similarity measures for time series, applied to the original series and the compression result, cannot meaningfully quantify the impact of a lossy transformation on the result of a change-detection approach [3, 34, 41, 22]. Such a quantification however is needed to identify and parametrize a good compression algorithm or anonymization approach, given a certain dataset and quality requirements on the change-detection result. This quantification is difficult due to several open challenges: First, as shown in the example, the impact is manifold. One therefore needs to carve out possible effects of a lossy transformation on changes. Second, the definition of a measure for this impact is not obvious. It is necessary to thoroughly investigate application scenarios where one is working on the transformed data, in order to come up with respective requirements. Third, the measure envisioned should be customizable to the concrete application scenario. Think of the energy provider once again. For him, it will be more detrimental if compression eliminates changes from the data, as opposed to the insertion of new ones. Fourth, identifying the specific effect of a transformation on a change (e.g., shift in time vs. disappearance) is an application-dependent procedure, which must take all changes into account. This is because ascribing an effect to a certain change may cascade and influence the ascription of effects to other changes. Having defined a measure does not make finding a good algorithm computing it obsolete.

In this paper, we design and evaluate MILTON, *a practical and flexible Measure which quantifies the Impact of various Lossy Transformation methods for time series on subsequent change detectiON*. This measure is applicable to any use case where one wants to know how much a certain transformation approach for time series reduces the result quality of a change-detection technique, as compared to change detection on the original data. This lets an operator decide how much he can compress the data without affecting change detection considerably, or if the anonymization technique he intends to deploy does indeed conceal certain changes, as he had planned. To ensure flexibility, we do not impose restrictions on the change detection or the transformation ap-

SSDBM'15 San Diego, California USA

proach used, and we allow to flexibly weight effects on changes. We have carried out extensive experiments, which have revealed interesting insights on the relationship between the transformation technique in use and change-detection quality. For instance, different anonymization techniques may have a significantly different impact on changes, although they protect privacy equally well.

At first sight, an alternative to MILTON would have been to derive a model of the loss of data quality due to a transformation. It is however very demanding to build such a model that is generally applicable. The main reason is that it is difficult to impossible to integrate each of the many existing lossy transformation techniques and change-detection approaches into one model.

In this article, we now make the following contributions:

- We investigate application scenarios in detail that do change detection on lossily transformed time-series data.
- We propose a measure of the impact of time-series transformation methods on subsequent change detection.
- We carry out an evaluation of our measure using three different use cases, namely data compression in the Smart Grid, data-center energy management and data privacy in the Smart Grid.

MILTON is suitable with any general-purpose lossy transformation and change-detection approaches. In addition, its design enables a flexible parametrization. Finally, it is applicable in many application areas in a straightforward manner.

Paper structure: Section 2 describes three application scenarios for our measure. Section 3 introduces and explains MILTON, which Section 4 evaluates. Section 5 reviews related work, and Section 6 concludes.

2. APPLICATION SCENARIOS

In this section, we describe three scenarios which motivate our measure and derive the requirements on it. We have consciously decided to describe these scenarios in much detail, in order to reveal the subtle differences between them, which then give way to the requirements.

2.1 Data Compression in the Smart Grid

2.1.1 Description

The Smart Grid gives way to the collection of ever-increasing volumes of time-series data [12]. This data is useful for analysis purposes such as energy-consumption forecasts [33] or energy disaggregation [21]. To store this data, recent research has produced numerous model-based lossy compression techniques [11, 18, 23, 13]. In contrast to lossless ones, they can obtain significantly higher compression ratios. These methods typically make use of correlations in the time series and produce a piecewise approximation of the original data within an error threshold ϵ. Thus, they not only modify the original data, but also the changes present in it. The effects of such approximations on the changes have not been investigated yet.

An energy provider intending to use the compressed data for analytics needs to take these effects into account. For instance, by detecting changes in data streams and integrating them in the learning model, he can improve forecasting [35] or enhance stream mining [40]. We refer to this scenario as the "compression scenario".

2.1.2 Problem Domain

The result of lossy compression methods depends on the models they use (e.g., constants, straight lines, polynomials) and how they use them. To evaluate their impact on changes in the data, one thus needs to consider different classes of models. Another important parameter here is the error threshold ϵ. Compression results, and consequently their impact on changes, strongly depend on this parameter. One therefore needs to evaluate the impact of compression methods for different values of ϵ.

2.1.3 Setting

The energy provider employs a forecasting application that uses the compressed time series to predict the energy consumption. Detecting a change in the time series triggers an update of the underlying model of the forecasting algorithm, to improve predictions. In such a case, it makes sense to penalize changes which disappear ("missed") more than those which emerge ("false positives") as a result of the transformation. This is because a missed change prevents the forecasting algorithm from updating its model when necessary. This may impact its accuracy significantly. A false-positive change in turn will trigger an unnecessary update of the model, which may cause additional effort, but should not affect forecasting accuracy considerably. Regarding shifts of changes in time, the provider deems them important for forecasting, as they will delay or vice-versa advance the update of the underlying model. On the other hand, modifications of the importance of changes are not crucial in this case, so he chooses to ignore them altogether. This makes sense here, because, once a change is detected, the model is updated regardless of that importance.

2.2 Data-Center Energy Management

2.2.1 Description

The share of computer-energy consumption has been estimated at 7.15% of the total electricity consumption and will increase to 14.6% by 2020 [42]. Quantifying this type of consumption reliably is thus important for many business cases. Deploying a smart meter for each computer system to measure the energy consumption directly is however expensive. Instead of measuring it, some recent approaches estimate the consumption of computers [10], based on, say, specific information on the hardware [31] or based on a profile of the computer power usage [19]. This is another lossy transformation technique, which we refer to as estimation.

Estimates of computer-energy consumption are useful in many use cases. For instance, a data-center manager can use such data in the design phase of the data center or to keep track of the energy consumption of the IT infrastructure when operating the center [10]. He can additionally use the data for other use cases which employ change-detection methods including consumption-event characterization or detection of abnormal consumption. As with compression, estimates are an approximation of the real consumption data. A data-center manager thus needs to quantify the effect of estimation methods on changes and to choose an estimation method appropriate for subsequent change detection. We refer to this scenario as the "estimation scenario".

2.2.2 Problem Domain

Estimation methods function differently in order to obtain approximations. We are aware of several classes: A first class performs a sophisticated calibration process [14], while another one relies on specific models of components [27]. They thus obtain estimates of different accuracy. In an evaluation, it would be interesting to study the impact of estimation methods from different classes on the changes. Another parameter here is the time granularity of the estimates. There is a trade-off between this granularity and accuracy [10]. We therefore need to evaluate estimation methods with different values for that parameter.

2.2.3 Setting

Here, the data-center manager will use estimates to balance energy demand and supply with the following application: A significant change in consumption will trigger an alarm, so that the energy supply adjusts to the new level. This means that shifts and modifications of the importance of changes are critical to the subsequent application. Regarding missed and false-positive changes, the manager uses a similar logic as in the previous scenario. A missed change is critical here because it prevents from balancing energy consumption and supply. A false positive however only implies an unnecessary readjustment. Even though this indicates additional costs, it is not critical to the subsequent application.

2.3 Data Privacy in the Smart Grid

2.3.1 Description

Smart meters can accurately and frequently measure and communicate the energy consumption of households. While these measurements are useful for analytical purposes, they unintentionally allow to infer personal information, such as the daily routine of residents [6, 26]. The pseudonymization of the data is not sufficient. This is because an easy re-identification of consumers using simple statistical measures is possible [6]. Adding noise (e.g., white noise) to the data does not enhance privacy either, as one can easily filter it out [29]. One way to prevent filtering out noise is to first transform the data to another basis (e.g., apply a Wavelet transform), then to add noise to the data in that basis, and finally to re-transform the data to the original basis [29]. Although they do not guarantee anonymization, such methods give way to some extent of anonymization in many cases. We refer to these methods as anonymization methods in the following.

Energy providers can apply such methods to protect user privacy. However, using them has a significant impact on different use cases, as the functioning of local energy markets may have additional costs [7]. The impact of anonymization on changes in the data is not yet known. A provider intending to anonymize the energy consumption of users nevertheless needs to quantify this effect. This is because the data should remain useful for subsequent analyses. We refer to this scenario as the "anonymization scenario".

2.3.2 Problem Domain

The result of anonymization using the above-mentioned methods [29] depends on the basis chosen for the transformation of the data (e.g., Fourier or Wavelet). We thus need to determine how the choice of the basis affects the changes in the data. The other important parameter in this case is the magnitude of the noise σ added to the original data. We conjecture that, the larger the noise added to the data, the larger is the potential impact on the changes.

2.3.3 Setting

Here, the provider considers the general case of data publishing, implying that he has little or no information on the subsequent use of the data. He does not differentiate between a shift in time or importance of changes. He does the same for missed and false-positive changes.

2.4 Measure Requirements

Based on these application scenarios, we have compiled the following requirements for our measure:

R1: Generalizability The measure should be independent of the change-detection algorithm and should provide meaningful results for any combination of general-purpose change-detection approach and lossy transformation.

R2: Flexibility The user should be able to configure the measure to distinguish and weight four cases according to the subsequent application: shifts of changes in time, modifications of their importance, disappearance of changes and emergence of new ones.

R3: Robustness The measure should be robust. Here, robustness means that computation should return meaningful results for any parametrization of the measure.

3. MILTON

We first describe the basic functioning of MILTON. We then present MILTON and say how we have parameterized it.

3.1 Problem Definition

A change-detection algorithm *CD* transforms a time series of measurements X into one of events (changes) $CD(X) = \{(t_1, s_1), (t_2, s_2), \ldots, (t_m, s_m)\}$. Here, t_i with $i = 1, \ldots, m$ denotes the time the change occurred at, while s_i denotes its score. Many state-of-the-art change-detection approaches associate with each change a score [25], which characterizes its significance. Without loss of generality, we assume that a change has a score of 1 if a change-detection approach does not provide scores. A lossy transformation T on X produces a modified time series of measurements *T(X)*. Applying *CD* on *T(X)* thus produces a time series of events *CD(T(X))*, which is possibly different from *CD(X)*. Table 1 sums up our notation introduced so far.

Symbol	Definition
X	original time series
T	lossy transformation
CD	change detection algorithm
CD(X)	time series of changes

Table 1: Notation Summary

When comparing the changes in *CD(X)* and *CD(T(X))*, we can assign each change to one of the following sets:

$\mathbf{PC} = \mathbf{pairing}(\mathbf{CD}(\mathbf{X}), \mathbf{CD}(\mathbf{T}(\mathbf{X}))$: As a result of the lossy transformation of X, changes of *CD(X)* might have been shifted in time or have their score altered. The set *PC* ("paired changes") contains pairs of changes of the form ($x \in$ *CD(X)*, $y \in$ *CD(T(X))*). Here, x is a change of *CD(X)*, and y is its corresponding change in *CD(T(X))*, eventually shifted or of altered score. The **pairing** function identifies and pairs such changes from *CD(X)* and *CD(T(X))*.

$\mathbf{MISS} = \mathbf{CD}(\mathbf{X}) - \mathbf{PC}$: As a result of the transformation, some changes of *CD(X)* might not have a match in *CD(T(X))*. The set *MISS* contains such changes, which we call "misses".

$\mathbf{FP} = \mathbf{CD}(\mathbf{T}(\mathbf{X})) - \mathbf{PC}$: In contrast, new changes might appear in *CD(T(X))*, which do not have any match in *CD(X)*. We refer to such changes as "false positives", which we add to the set *FP*.

From requirement R2 (cf. Subsection 2.4), it follows that MILTON must consider each set defined above differently. In particular, we must determine the changes the transformation has affected in a minor way (minor = shift in time or modification of score; *PC*), how many have disappeared (*MISS*), and how many have emerged as a result of the transformation (*FP*). Second, we should allow setting application-dependent weights on the impact of changes in each set. For example, if missed changes are critical to the subsequent application, our measure must attribute larger weights to such cases than to false positives or vice-versa. To evaluate the impact of a lossy transformation on subsequent change detection, MILTON quantifies how similar *CD(X)* and *CD(T(X))* are, i.e., we sum

the weighted differences between the changes *CD(X)* and *CD(T(X))* assigned to *PC*, and the weights of the changes in *MISS* and *FP*.

3.2 Calculating PC, MISS and FP

We first explain how the function $\mathbf{pairing}()$ can be implemented to obtain the set *PC*. Obtaining sets *FP* and *MISS* follows in a straightforward manner.

Finding the optimal matching between changes from *CD(X)* and *CD(T(X))* is not trivial. One reason is that matching two changes can affect the matching of other changes. As an example, suppose that we match two changes $x \in$ *CD(X)* and $y \in$ *CD(T(X))* incorrectly. This means that the correct match y' for x may now be matched with another change incorrectly. This holds for y and its match x' and may cascade. The incorrect matching of two changes may thus impact the entire matching process. The matching process therefore needs to consider all possible matching combinations of changes. Another reason is that the optimal matching may not include all changes in *CD(X)* or *CD(T(X))*, as there may be new changes (false positives) and removed ones (misses). The matching process may therefore need to skip changes. However, it is not clear how many changes it should skip.

Due to Requirement R2, our measure must take into account both the difference in time and score (importance) between two changes. Moreover, it should be possible to weight these differences depending on the application scenario. For example, in the compression scenario (Section 2.1), the difference in time has a higher weight than the difference in score. We therefore first define these weights: Let $x = (t_x, s_x)$ be a change of *CD(X)* and $y = (t_y, s_y)$ one of *CD(T(X))*. We define $f_{TIME} : \mathbb{R} \mapsto \mathbb{R}^+$ as a function of the normalized difference in time (Δ_t) between x and y and $f_{SCORE} : \mathbb{R} \mapsto \mathbb{R}^+$ as a function of the normalized difference in score (Δ_s) between x and y. These functions are application-dependent, as explained above. The distance between two changes then is a function g of f_{TIME} and f_{SCORE}:

$$dist(x, y) = g(f_{TIME}(\Delta_t(x, y)), f_{SCORE}(\Delta_s(x, y))) \quad (1)$$

We fix g as the sum of the contributions f_{MISS} and f_{SCORE}:

$$dist(x, y) = f_{TIME}(\Delta_t(x, y)) + f_{SCORE}(\Delta_s(x, y)) \quad (2)$$

In our experiments, we have tested other distances, such as the maximum between the two contributions: $max(f_{TIME}(\Delta_t(x, y)), f_{SCORE}(\Delta_s(x, y)))$. This has not lead to substantially different results.

Based on the above-defined distance, one trivial way to find the correspondence between changes *CD(X)* and *CD(T(X))* is to calculate all possible one-to-one combinations of events which maintain the original succession of changes and choose the one with the smallest total distance. However, this is computationally expensive; the number of such combinations grows exponentially with the number of changes in *CD(X)* and *CD(T(X))*. At first sight, the setting may resemble the one of the well-known Hungarian Algorithm. The difference however is the need to maintain the original order of changes for the matching. Several publications however have studied this specific problem or closely-related ones [22, 43]. We use the Optimal Subsequence Bijection (OSB) algorithm introduced in [22], which fulfills the matching prerequisites:

- the matching should consider all distances between matched changes
- the matching should allow leaving unmatched changes (misses and false-positives)

However, OSB as is does not solve our problem. This is because, after performing the matching, it does not take unmatched changes into account. OSB matches two sequences *CD(X)* and *CD(T(X))* of (possibly different) lengths m and n:

$$CD(X) = \{(t_{x_1}, s_{x_1}), (t_{x_2}, s_{x_2}), \ldots, (t_{x_1}, s_{x_m})\}$$

$$CD(T(X)) = \{(t_{y_1}, s_{y_1}), (t_{y_2}, s_{y_2}), \ldots, (t_{y_n}, s_{y_n})\}$$

Its goal is to find best-matching subsequences *CD(X)*$'$ of *CD(X)* and *CD(T(X))*$'$ of *CD(T(X))*. Thus, it may skip changes. The authors of OSB motivate having unmatched changes by the fact that the sequences may contain outliers that should be skipped. In our case, these correspond to false-positive and missed changes. However, skipping too much may result in random matches. To avoid this, OSB uses a penalty C for skipping.

The algorithm requires the two sequences, a distance measure and a penalty for skipping changes as input. To find the optimal matching, OSB minimizes the sum of the distances between matched changes and the penalties for changes skipped. It thus considers all distances between matched changes, as required. [22] uses the Euclidean distance. We adapt OSB to our case by using the distance from Equation (2), next to some other adaptations:

There are many possibilities to set the penalty C. From our experiments, we have found that the standard penalty recommended in [22] produces correct matchings and rarely results in mismatches between changes. We therefore used the standard penalty, which is defined as follows:

$$C(CD(X), CD(T(X))) = \underset{i}{\text{mean}}(\min_j(dist(x_i, y_j)) + \underset{i}{\text{std}}(\min_j(dist(x_i, y_j))$$

where $x_i \in$ *CD(X)*, $i = 1, \ldots, m$ and $y_j \in$ *CD(T(X))*, $j = 1, \ldots, n$.

Algorithm 1 Algorithm computing *PC*, *MISS* and *FP*

```
1: Let MISS = {}
2: Let FP = {}
3: PC = OSB(CD(X), CD(T(X)), C)
4: for x ∈ CD(X) do
5:     if x ∉ PC then
6:         MISS = MISS ∪ x
7:     end if
8: end for
9: for x ∈ CD(T(X)) do
10:    if x ∉ PC then
11:        FP = FP ∪ x
12:    end if
13: end for
```

In our case, OSB outputs a one-to-one pairing between changes in *CD(X)* and *CD(T(X))*, which makes up the set *PC*. To identify the changes which disappeared as a result of the transformation (*MISS*), we loop over changes in *CD(X)* and select those which do not have a match in *CD(T(X))*, i.e., are not in *PC*. Similarly, to obtain new changes (*FP*) we loop over the changes in *CD(T(X))* and select those without a match in *CD(X)*. See Algorithm 1.

3.3 Measure Definition

As explained in the previous subsection, having sets *PC*, *MISS* and *FP*, we can construct a general-purpose measure, which satisfies Requirements R1, R2 and R3. We first consider the impact of each set separately, followed by the total impact.

Paired Changes.

Changes in such a couple may differ in the time when they occur and in their score. As just explained, we quantify this difference using the distance defined in Equation (2). To quantify the global impact of such changes, we sum up the distances between "paired changes". This creates the following term, which is part of our measure:

$$errPC = \sum_{(x,y)\in PC} dist(x, y) =$$

$$\sum_{(x,y)\in PC} f_{TIME}(\Delta_t(x,y)) + f_{SCORE}(\Delta_s(x,y))$$

The intuition is that, the more changes are shifted in time and score, the bigger the impact of the transformation on them and vice versa. In case information on the particular impact of shifts in time and score was necessary, $errPC$ could be split into two terms calculated separately:

$$errPC = errTIME + errSCORE$$

where

$$errTIME = \sum_{(x,y)\in PC} f_{TIME}(\Delta_t(x,y))$$

and

$$errSCORE = \sum_{(x,y)\in PC} f_{SCORE}(\Delta_s(x,y))$$

Misses.

Depending on the application scenario considered, we may want to deal with misses in a differentiated manner according to their score. For example, we may choose to completely ignore missed changes with a low score and conversely assign a bigger weight to ones with a high score. We therefore introduce a weighting function on missed changes f_{MISS} which depends on their score. We discuss how we define this function in the following subsection. To quantify the total impact of missed changes, we sum up their individual impacts weighted by f_{MISS}, yielding the second term of our measure:

$$errMISS = \sum_{(t,s)\in MISS} f_{MISS}(s) \quad (3)$$

False Positives.

As in the case of missed changes, we may want to handle false positives in a differentiated manner depending on their score. For this, we introduce a weighting function on false positives f_{FP}. As in the previous cases, we sum up the individual impacts weighted by f_{FP} and create the last term of our measure:

$$errFP = \sum_{(t,s)\in FP} f_{FP}(s) \quad (4)$$

Weight function	Argument
f_{TIME}	shift in time
f_{SCORE}	shift in score
f_{MISS}	missed changes
f_{FP}	false-positive changes

Table 2: Notation Summary

Total Impact.

To quantify the total impact of the different terms introduced above, MILTON sums them up. However, there is another issue, which MILTON should take into account, namely, the number of changes in the original time series $|CD(X)| = |PC| + |MISS|$. We explain the rationale using an example:

EXAMPLE 2. *Suppose that, for a time series X_1, CD detects 2 changes, while for another time series X_2, it detects 100. Let us further assume that applying transformation T on X_1 introduces a shift in time and score to the original changes $CD(X_1)$ and the summed-up impact is equal to 0.5. We also assume that applying the same transformation T on X_2 leaves all but two changes intact and introduces a shift in time and score to the two changes resulting in an equivalent impact equal to 0.5. Logically, the global impact between the two cases should be significantly different. This is because in the case of X_2, T leaves 98% of the changes intact, while it affects 100% of the changes in the case of X_1. We therefore need to normalize the total impact and divide it by the number of changes CD detects in the original time series.*

Using the above results, we define MILTON as follows:

$$MILTON(X, T, CD) = \frac{errPC + errMISS + errFP}{|PC| + |MISS| + 1} \quad (5)$$

We add 1 to the denominator to account for the case when *PC* and *MISS* are both empty.

Table 2 lists the functions presented above. We have explained the need to use weights within MILTON, and we have provided some intuition on how to set them.

3.4 Parametrization

MILTON has four parameters: f_{SCORE}, f_{TIME}, f_{MISS} and f_{FP}. In the following we say how we set these parameters for each application scenario described in Section 2. Observe that the domain of these functions is normalized to the interval $[0, 1]$.

We first consider the compression scenario. Here, we must penalize missed changes substantially more than false positives. We therefore assign a larger weight to f_{MISS} than to f_{FP}. See Table 3. Concerning shifts in time and score, we set f_{TIME} proportional to the size of the shift and we set f_{SCORE} equal to zero. This is because we ignore alterations of the importance of changes.

	Compression	Estimation	Anonymization
$f_{TIME}(\Delta_t)$	$\lvert\Delta_t\rvert$	$e^{\lvert\Delta_t\rvert} - 1$	$\frac{1}{2} \cdot \lvert\Delta_t\rvert$
$f_{SCORE}(\Delta_s)$	0	$e^{\lvert\Delta_s\rvert} - 1$	$\frac{1}{2} \cdot \lvert\Delta_s\rvert$
$f_{MISS}(s)$	$s^2 + 1$	$e^s - 1$	s
$f_{FP}(s)$	s	s	s

Table 3: Measure parametrization

We now turn to the estimation scenario. As mentioned, shifts and modifications of the scores are critical to the subsequent application. Thus, we decide to set f_{TIME} and f_{SCORE} to grow exponentially with increasing shifts in time and score (Table 3). Regarding f_{MISS} and f_{FP}, we use a similar logic as for the previous scenario where we penalize missed changes substantially more than false-positive ones.

Lastly, we consider the anonymization scenario. As stated, we are in the general case of data publishing. This means that little or no information on the subsequent use of the data is available. We therefore use the average of shifts in time and score (importance) between two changes as distance, with no differentiation between

the type of shift (Table 3). We let the terms *errFP* and *errMISS* due to missed and false positive changes correspond to the sum of the scores of changes in the respective sets (*MISS* and *FP*).

4. EVALUATION

MILTON operates as intended if it fulfills the requirements from Subsection 2.4. We have covered the flexibility and robustness requirements by design. To cope with generalizability, we have evaluated MILTON using our three scenarios. In the following, we present the datasets used, the setup of our experiments and their results.

4.1 Datasets

To evaluate MILTON, we use five datasets, as follows. We use the first two for the evaluation of the compression and anonymization scenarios. We use the other three for the estimation scenario.

The **Reference Energy Disaggregation Dataset (REDD)** comes from the field of energy disaggregation and is publicly available [21]. It contains measurements of smart meters from several buildings. For our experiments, we use a part of it, namely data measured second-wise from four individual houses.

The **Smart Home Dataset (Smart)** includes data collected from real homes and is publicly available [2]. Its goal is to facilitate further research on home-energy consumption. We use the second-wise measurements of aggregate electricity consumption from one building for our experiments.

The following three datasets consist of measurements we had performed at our institute. These contain real and estimated energy-consumption data from three computer systems. We had used a digital multimeter Wattsup PRO [44] (accuracy: 1.5%) to record the reference energy consumption. We have used a dynamic estimator and a calibration-based one [10] to obtain estimates of energy consumption.

The **Desktop Dataset** contains measurements of three weeks of real and estimated energy consumption from an office computer with a sampling frequency of one second. Its workload is the result of typical secretarial tasks, e.g., MS Office, Internet Explorer and a number of custom-made administrative applications. The workload rarely reaches the maximal computing capacity, and the computer is active only during office hours.

For the **Laptop Dataset** we have measured the real and estimated energy consumption of a laptop over a period of two weeks with a sampling frequency of one second. We had used the laptop [10] for research purposes, i.e., in contrast to the desktop computer, the system load does not follow any regular pattern and shows both idle periods and maximum load conditions.

The **Server Dataset** is about a mail server filtering spam by constantly executing SpamAssassin. Its workload is a daily pattern with low usage during the night and high usage in the morning and afternoon hours [10]. Load peaks occur when the server checks bulks of e-mails sent to large mailing lists. We had measured and estimated the energy consumption every minute over a period of three weeks.

4.2 Setup

For the evaluation of all scenarios, we have used $CUSUM$ [28], an established change-detection method. We have configured it to detect changes of the mean of a data sequence of at least 5% of its range. We set the parameters (weights) of MILTON according to each application scenario (Table 3), cf. Subsection 3.4. In the following we present the lossy transformation methods used for each scenario.

4.2.1 Compression Scenario

We use compression techniques based on different classes of models:

a) Adaptive Piecewise Constant Approximation (*APCA*) uses constant functions to approximate segments of data of varying length [20].

b) Piecewise Linear Histogram (*PWLH*) compresses the data in a similar manner as APCA using straight-line functions instead of constant ones [8].

c) Adaptive Polynomial Piecewise Compression (*APP*) combines polynomial functions of different degrees to approximate the data piecewisely in an incremental manner [11].

All of these methods compress the data, such that the maximum deviation between the original and decompressed data under the uniform norm is smaller than an error threshold ϵ.

4.2.2 Estimation Scenario

As mentioned, we use a dynamic and a calibration-based estimator to obtain the energy-consumption estimates, cf. [10].

4.2.3 Anonymization Scenario

We use two data perturbation/anonymization methods: one using the Fourier transform and one using the Wavelet one [29].

4.3 Scenario Evaluation

We now present our evaluation of MILTON.

4.3.1 Compression Scenario

In this scenario, the energy provider wants to identify the method which delivers the best compression ratio for a given impact on the changes in the data together with a good parameter set. For this, we have first computed MILTON for all three compression techniques for different values of the threshold ϵ going from 0.2% to 5% of the range of the time series used. Figure 1 shows the average results for the Smart Dataset. They are similar to those we obtained with the REDD Dataset. We observe that the measure generally increases with a growing threshold value. This is because some changes disappear as a result of the rougher compression. We also observe that, for large values of ϵ, compression using *APCA* has a worse impact on the subsequent change detection than the other two methods.

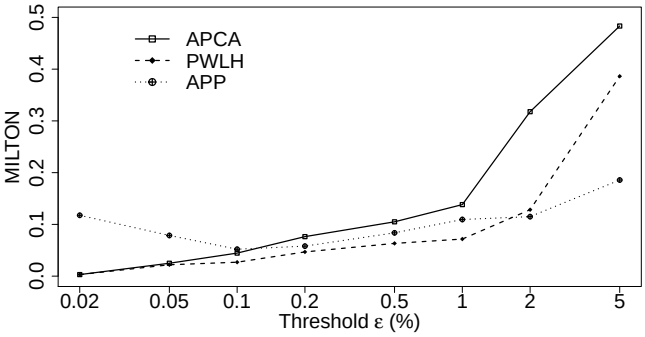

Figure 1: MILTON vs. Threshold (ϵ) - Smart Dataset

To understand the reasons behind the results in Figure 1, we list and inspect the number of changes in sets *PC*, *MISS* and *FP*, as well as MILTON components *errPC*, *errMISS* and *errFP* for one time series of the Smart dataset. We first notice that *PWLH* and *APP* preserve existing changes better than *APCA* when $\epsilon > 0.2\%$. This

accounts for their lower values of MILTON. Second, in comparison to both *PWLH* and *APCA*, *APP* introduces considerably more false-positive changes. We assume that this is due to the use of polynomials of degree higher than 1. These may introduce "bumps" in the time series, which *CUSUM* interprets as changes. However, they do not impact the value of MILTON significantly, as we set the weight f_{FP} for false-positive changes significantly smaller than for misses (f_{MISS}).

	ϵ	*PC*	*MISS*	*FP*	*errPC*	*errMISS*	*errFP*
APCA	0.02	170	0	0	0.000	0.000	0.000
	0.05	167	3	3	0.006	3.000	0.015
	0.1	160	10	7	0.007	10.000	0.036
	0.2	156	14	7	0.010	14.000	0.038
	0.5	153	17	2	0.007	17.000	0.013
	1	145	25	1	0.010	25.001	0.000
	2	110	60	0	0.008	60.002	0.000
	5	89	81	0	0.000	81.003	0.000
PWLH	0.02	170	0	0	0.000	0.000	0.000
	0.05	164	6	6	0.002	6.000	0.031
	0.1	164	6	6	0.005	6.000	0.031
	0.2	162	8	8	0.006	8.000	0.041
	0.5	160	10	9	0.014	10.000	0.046
	1	157	13	10	0.019	13.000	0.053
	2	148	22	6	0.031	22.001	0.032
	5	110	60	2	0.091	60.002	0.012
APP	0.02	148	22	15	0.013	22.001	0.345
	0.05	147	23	10	0.011	23.001	0.310
	0.1	165	5	4	0.006	5.000	0.021
	0.2	165	5	3	0.022	5.000	0.015
	0.5	159	11	8	0.020	11.000	0.041
	1	154	16	7	0.022	16.000	0.040
	2	154	16	6	0.026	16.000	0.032
	5	138	32	52	0.226	32.001	0.373

Table 4: Number of Changes and Measure Components in Compressed Data for Threshold ϵ – homeB – Smart Dataset

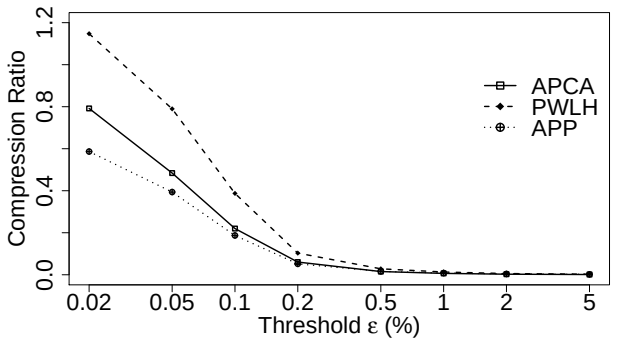

Figure 2: Compression Ratio vs. Threshold (ϵ) – Smart Dataset

We next compute the compression ratio for all three methods for the same values of ϵ as in our previous experiment. We use the following formula:

$$compression\ ratio = \frac{\text{size of compressed data}}{\text{size of initial data}} \quad (6)$$

Figure 2 shows the average results for the Smart Dataset. We notice that the techniques diverge significantly for small values of ϵ ($\epsilon < 0.2\%$) and converge to similar ones when ϵ grows ($\epsilon > 1\%$). For small values of ϵ, compression with *PWLH* is worst, followed by *APCA* and *APP*.

Using the results above, the provider can identify a good compression method with a good parameter set if a bound on the impact on the changes is given. To this end, he first needs to identify, for each method, the value of ϵ which gives way to an impact within the bound. Then, using the results on compression ratios, he can select the best method.

Summary: *We have shown that MILTON can help the provider decide which compression method suits his needs best. In this use case, for the specific settings used here, no method is generally superior to the other ones.*

4.3.2 Estimation Scenario

In this scenario, a data-center manager wants to identify which estimation method at which aggregation level (e.g., per minute or hourly) has the smallest impact on changes in the data. For this, we aggregate the energy-consumption time series to time intervals from one minute to 60 minutes. We then calculate our measure for both estimators on all datasets for these intervals. Table 5 shows the average value of MILTON for time series in each dataset tested. We first notice that the calibration-based estimator has a smaller impact on changes than the dynamic one for almost all datasets and interval lengths. For the laptop dataset, for instance, MILTON is 30% smaller on average. Furthermore, the value of MILTON generally increases with the interval length. This means that, while using a longer time interval for aggregation may improve accuracy as shown in [10], it has a bigger impact on changes in the data.

		Time interval length (min.)					
		1	2	5	15	30	60
ATIS	Dynamic	0.01	0.01	0.03	0.00	0.00	0.04
	Calibr.	0.00	0.01	0.02	0.00	0.00	0.05
Desktop	Dynamic	0.01	0.03	0.04	0.06	0.11	0.33
	Calibr.	0.00	0.00	0.01	0.01	0.03	0.27
Laptop	Dynamic	0.01	0.01	0.01	0.04	0.06	0.17
	Calibr.	0.00	0.01	0.01	0.02	0.04	0.09

Table 5: MILTON by Time Interval Length

To find out why MILTON behaves in this way in this case, we list the number of changes in the sets *PC*, *MISS* and *FP*, as well as MILTON components *errPC*, *errMISS* and *errFP* for one time series of the Desktop Dataset. We see that the dynamic estimator usually produces more missed and false positive changes than the calibration-based one. This makes MILTON grow significantly due the definition of f_{MISS} and f_{FP}. Moreover, for small interval lengths the pairing matches changes better than for longer ones. This accounts for the big impact of aggregation on changes for long intervals .

Summary: *For this scenario MILTON lets a data-center manager assess the impact of an estimation method on change detection. The calibration-based estimator impacts changes significantly less than the dynamic one.*

	Interval length (min)	*PC*	*MISS*	*FP*	*errPC*	*errMISS*	*errFP*
Dynamic	1	76	37	75	0.614	0.057	0.351
	2	43	1	33	1.191	0.022	0.284
	5	19	1	20	0.472	0.053	0.208
	15	8	0	6	0.431	0.000	0.142
	30	3	0	4	0.312	0.000	0.129
	60	1	2	0	1.055	0.000	0.000
Calibr.-based	1	86	27	0	0.152	0.032	0.012
	2	34	10	1	0.123	0.021	0.003
	5	11	9	0	0.031	0.033	0.000
	15	5	3	0	0.065	0.033	0.000
	30	3	0	0	0.189	0.000	0.000
	60	1	2	0	0.061	0.134	0.000

Table 6: Number of Changes and Measure Components by Interval Length – Desktop Dataset

4.3.3 Anonymization Scenario

In this scenario, an energy provider needs to quantify the impact of anonymization methods on the changes in the data. The goal is to identify the method which protects privacy best while keeping the data useful for subsequent analytics. For this, we computed MILTON for both anonymization methods. We vary the value of the perturbation σ these methods add, as described in [29].

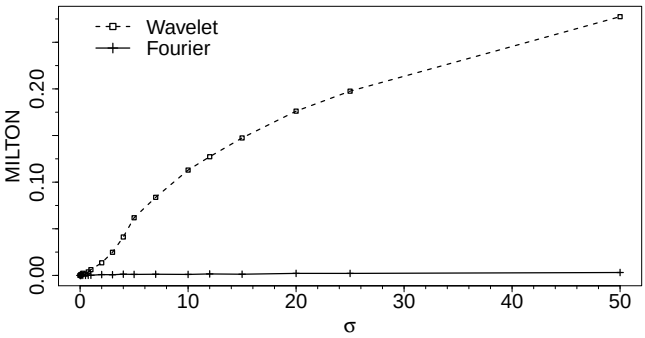

Figure 3: Milton vs. perturbation added σ – House 4 – REDD Dataset

Figure 3 shows the average values of MILTON for House 4 of the REDD dataset when the perturbation σ goes from 0.001 to 50. In this case, these values of σ correspond to the interval $[0.01\%, 58\%]$ of the standard deviation of the energy-consumption time series of House 4. We obtained similar results for all other REDD and Smart time series.

We observe that MILTON increases if we add more perturbation when using the Wavelet transform, while it stays practically constant when using Fourier. To understand why this happens, we show the number of changes in the sets *PC*, *MISS* and *FP*, as well as MILTON components *errPC*, *errMISS* and *errFP* for one time series of House 4 in Table 7. The number of paired changes and missed ones varies only slightly in both cases. However, adding a bigger perturbation when using the Wavelet transform introduces more false-positive changes (*FP*), which makes MILTON grow. We believe that this is due to the nature of the Haar-Wavelet, which the Wavelet-based method uses. Adding perturbation in the form of Haar-Wavelets of significant magnitude introduces changes which have not been present in the data originally.

	σ	*PC*	*MISS*	*FP*	*errPC*	*errMISS*	*errFP*
Fourier	1	160	0	0	0.001	0.000	0.000
	2	160	0	0	0.002	0.000	0.000
	5	156	4	4	0.004	0.394	0.394
	10	159	1	1	0.011	0.209	0.209
	25	158	2	3	0.033	0.148	0.168
	50	157	3	5	0.071	0.213	0.257
Wavelet	1	153	7	9	0.025	1.109	1.164
	2	157	3	6	0.065	0.370	0.434
	5	149	11	120	0.044	1.965	4.931
	10	152	8	336	0.048	1.171	11.606
	25	148	12	469	0.038	2.158	26.093
	50	152	8	498	0.047	0.984	38.317

Table 7: Number of Changes and Measure Components by Perturbation σ – House 4 – REDD Dataset

Summary: *Using MILTON, an energy provider can quantify the impact of anonymization methods on subsequent change detection. In this case, the evaluation of the two methods (Fourier-based and Wavelet-based) shows that they have a significantly different impact on the changes. This is even though they protect the privacy on most of the datasets tested to a similar extent according to [29].*

5. RELATED WORK

We now review modern change-detection methods, time-series similarity measures and lossy transformation techniques. We are aware of three classes, namely lossy time-series compression, computer energy-consumption estimation and time-series anonymization.

5.1 Change Detection

The goal of change detection is identifying significant changes of the data or of its parameters. Research has produced numerous methods for different types of change to be detected.

Some methods compare the probability distributions of (subsequent) sequences of data. [38] features a test which checks if two datasets are sampled from the same underlying distribution using a Gaussian kernel density estimator. [25] uses a density estimator to instead calculate the ratio of the distributions of two consecutive subsequences of data and to detect if they come from different distributions. Another research direction is detecting changes in parameters of a data sequence (e.g., mean or variance). CUSUM is an established sequential analysis method for change detection of the parameters of a probability distribution [28]. It calculates a cumulative sum for the segment currently considered and issues a change alert once this exceeds a given threshold. [4] describes a method which adjusts the size of the sliding window once a change is detected, such that the parameter of the data in the current window (e.g., mean) is constant. A further research area is detecting changes using models describing the data. [17] uses polynomials to describe a time series piecewisely and to detect a change once the approximation error of the current model crosses a threshold. As another example, [39] uses an auto-regressive model to detect and remove outliers before detecting changes in the data. [37] uses Gaussian Processes to model and predict the current run length – the length of a time segment between two consecutive changes.

5.2 Time-Series Similarity Measures

Time-series similarity is a well-researched area. The Euclidean Distance is a frequently used distance. Another measure, Dynamic Time Warping (DTW), is commonly used to align sequences [3,

34]. The DTW between two sequences is the sum of distances of their corresponding elements. The classic DTW algorithm employs dynamic programming to identify corresponding elements so that this distance is minimal. The Longest Common Subsequence (LCSS) is another measure used to solve the alignment problem and to detect outliers in time series [41]. LCSS determines the longest common subsequence between two sequences. The Optimal Subsequence Bijection (OSB) [22] is yet another measure which, in contrast to DTW, creates a one-to-one correspondence between two subsequences. Another difference to DTW is that OSB allows skipping of elements.

5.3 Lossy Time-Series Compression

Many modern lossy compression methods divide time series into pieces which they approximate with mathematical models [18]. Thus, [23] uses constants to approximate fixed-length intervals. [13] presents two methods which produce connected and disconnected piecewise straight-line segments of variable length. [30] proposes using multiple models in parallel and choosing the one which best compresses the current segment. [11] uses the same idea and proposes an incremental approach using polynomials of different degrees for compressing energy-consumption time series.

To quantify the loss of data due to approximation and to guarantee a certain quality of the compressed data, the methods consider the error between the original and the approximated data. For this purpose, the *uniform norm* (L_∞*-norm*) is commonly used [18]. To additionally evaluate the quality of approximation, [18] uses the root mean square error (RMSE). None of the proposals otherwise consider how lossy compression might affect subsequent change detection.

5.4 Computer Energy-Consumption Estimation

Smart meters are commonly used to monitor the energy consumption of computers [16]. As monitoring a large number of computing systems in this way comes at high costs, recent research has developed methods to characterize and estimate computer energy consumption. Some of this work has built models based on collecting micro-architectural events using hardware registers [5]. Such models are less portable and not applicable to large heterogeneous deployments of computers, as they are hardware-specific. Other methods create black-box models using high-level statistical information obtained from the operating system. [14] for instance estimates the power consumption of a large group of servers by matching it with CPU utilization. [36] compares several models which use CPU and disk utilization and shows that these attain good accuracies over many workloads.

The accuracy of the above-mentioned estimation methods is measured using common measures, such as the mean absolute percentage error [5, 36]. Recent work has identified application scenarios where estimation methods are useful. Thus, the authors of [27] use their model to detect energy hotspots in software. [10] presents several use cases which make use of computer energy-consumption data, e.g., energy-aware management of data centers. We have not found any work which considers the impact of computer energy-consumption estimation on subsequent change detection.

5.5 Time-Series Anonymization

Research has produced a multitude of anonymization techniques. See [15] for an overview. *Differential Privacy* is an intuitive measure of the risk of one's privacy when having personal data in a database [9]. [1] is an example of a privacy-preserving system compliant with differential privacy. Other work has addressed time-series anonymization. [29] proposes several schemes for time-series anonymization in a streaming context. [32] studies the problem of smart-meter time-series anonymization by filtering out low-power frequency components.

Others have analyzed the effect of anonymization on subsequent use of the data. As an example, the framework developed in [7] allows to quantify the economic and environmental effects of anonymization on local energy markets. [24] argues that the quality of anonymized data should be measured based on the workload the data would be used for. We have however not found any work which explicitly investigates this effect.

6. CONCLUSIONS

Recent research has proposed numerous lossy transformation techniques for time-series data. While transforming the data, they may impact characteristics of the data, such as changes, which are important for further analyses. To address this issue, we have developed a generalizable and flexible measure which quantifies the impact of a lossy transformation on subsequent change detection. Our evaluation shows that it is useful for various application scenarios to identify adequate parameters of a lossy transformation so that its advantages are maximized while the impact on subsequent change detection is bounded.

7. REFERENCES

[1] G. Acs and C. Castelluccia. I have a dream!(differentially private smart metering). In *Information Hiding*, 2011.

[2] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht. Smart*: An open data set and tools for enabling research in sustainable homes. *SustKDD Workshop on Data Mining Applications in Sustainability*, 2012.

[3] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, 1994.

[4] A. Bifet and R. Gavaldà. *Learning from Time-Changing Data with Adaptive Windowing*, chapter 42.

[5] W. L. Bircher and L. K. John. Complete system power estimation using processor performance events. *IEEE Transactions on Computers*, 61:563–577, 2012.

[6] E. Buchmann, K. Böhm, T. Burghardt, and S. Kessler. Re-identification of smart meter data. *Personal and Ubiquitous Computing*, 17:653–662, 2013.

[7] E. Buchmann, S. Kessler, P. Jochem, and K. Böhm. The costs of privacy in local energy markets. In *IEEE Conference on Business Informatics*, 2013.

[8] C. Buragohain, N. Shrivastava, and S. Suri. Space efficient streaming algorithms for the maximum error histogram. In *IEEE International Conference on Data Engineering*, 2007.

[9] C. Dwork. Differential privacy. In *Automata, languages and programming*. Springer, 2006.

[10] P. Efros, E. Buchmann, and K. Böhm. FRESCO: A framework to estimate the energy consumption of computers. In *IEEE Conference on Business Informatics*, 2014.

[11] F. Eichinger, P. Efros, S. Karnouskos, and K. Böhm. A time-series compression technique and its application to the smart grid. *The VLDB Journal*, 24:193–218, 2015.

[12] F. Eichinger, D. Pathmaperuma, H. Vogt, and E. Müller. Data analysis challenges in the future energy domain. *Computational Intelligent Data Analysis for Sustainable Development*, 2012.

[13] H. Elmeleegy, A. K. Elmagarmid, E. Cecchet, W. G. Aref, and W. Zwaenepoel. Online piece-wise linear approximation

of numerical streams with precision guarantees. *VLDB Endowment*, 2:145–156, 2009.

[14] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *Annual International Symposium on Computer Architecture*, 2007.

[15] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42:14:1–14:53, 2010.

[16] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. Cameron. Powerpack: Energy profiling and analysis of high-performance systems and applications. *IEEE Transactions on Parallel and Distributed Systems*, 21:658–671, 2010.

[17] V. Guralnik and J. Srivastava. Event detection from time series data. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

[18] N. Q. V. Hung, H. Jeung, and K. Aberer. An evaluation of model-based approaches to sensor data compression. *IEEE Transactions on Knowledge and Data Engineering*, 25:2434–2447, 2013.

[19] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya. Virtual machine power metering and provisioning. In *ACM Symposium on Cloud Computing*, 2010.

[20] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM SIGMOD Record*, 30:151–162, 2001.

[21] J. Z. Kolter and M. J. Johnson. Redd: A public data set for energy disaggregation research. In *SIGKDD Workshop on Data Mining Applications in Sustainability*, 2011.

[22] L. Latecki, Q. Wang, S. Koknar-Tezel, and V. Megalooikonomou. Optimal subsequence bijection. In *IEEE International Conference on Data Mining*, 2007.

[23] I. Lazaridis and S. Mehrotra. Capturing sensor-generated time series with quality guarantees. In *International Conference on Data Engineering*, 2003.

[24] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Workload-aware anonymization. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

[25] S. Liu, M. Yamada, N. Collier, and M. Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72 – 83, 2013.

[26] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, 2010.

[27] A. Noureddine, A. Bourdon, R. Rouvoy, and L. Seinturier. Runtime monitoring of software energy hotspots. In *IEEE/ACM International Conference on Automated Software Engineering*, 2012.

[28] E. S. Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.

[29] S. Papadimitriou, F. Li, G. Kollios, and P. S. Yu. Time series compressibility and privacy. In *International Conference on Very Large Data Bases*, 2007.

[30] T. G. Papaioannou, M. Riahi, and K. Aberer. Towards online multi-model approximation of time series. In *IEEE International Conference on Mobile Data Management*, 2011.

[31] M. Poess and R. O. Nambiar. Power based performance and capacity estimation models for enterprise information systems. *IEEE Data Engineering Bulletin*, 34:34–49, 2011.

[32] S. Rajagopalan, L. Sankar, S. Mohajer, and H. Poor. Smart meter privacy: A utility-privacy framework. In *IEEE International Conference on Smart Grid Communications*, 2011.

[33] R. Ramanathan, R. Engle, C. W. Granger, F. Vahid-Araghi, and C. Brace. *Short-run forecasts of electricity loads and peaks*. Cambridge University Press, 2001.

[34] C. A. Ratanamahatana and E. Keogh. Three myths about dynamic time warping data mining. In *SIAM International Conference on Data Mining*, 2005.

[35] G. Ristanoski, W. Liu, and J. Bailey. A time-dependent enhanced support vector machine for time series regression. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013.

[36] S. Rivoire, P. Ranganathan, and C. Kozyrakis. A comparison of high-level full-system power models. *HotPower*, 8:3–3, 2008.

[37] Y. Saatçi, R. D. Turner, and C. E. Rasmussen. Gaussian process change point models. In *International Conference on Machine Learning*, 2010.

[38] X. Song, M. Wu, C. Jermaine, and S. Ranka. Statistical change detection for multi-dimensional data. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

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

[40] Y. Tao and M. T. Ozsu. Mining data streams with periodically changing distributions. In *ACM Conference on Information and Knowledge Management*, 2009.

[41] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. Indexing multi-dimensional time-series with support for multiple distance measures. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[42] W. Vereecken et al. Overall ICT Footprint and Green Communication Technologies. In *International Symposium on Communications, Control and Signal Processing*, 2010.

[43] Y. Wang and T. Pavlidis. Optimal correspondence of string subsequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:1080–1087, 1990.

[44] Watts up? Meters. https://www.wattsupmeters.com, Accessed 22 March 2015.