# How to Quantify the Impact of Lossy Transformations on Event Detection

Pavel Efros, Erik Buchmann, Adrian Englhardt, Klemens Böhm

*Karlsruhe Institute of Technology, Karlsruhe, 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. In the case of time series, important characteristics are the occurrence of certain changes or patterns in the data, also referred to as *events*. Clearly, the less transformations modify events, the better for subsequent analyses. More specifically, the severity of those modifications depends on the application scenario, and quantifying it is far from trivial. In this paper, we propose MILTON, *a flexible and robust Measure for quantifying the Impact of Lossy Transformations on subsequent event detectiON*. MILTON is applicable to any lossy transformation technique on time-series data and to any general-purpose event-detection approach. We have evaluated it with several 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.

*Keywords:* time series, lossy transformations, event detection, change detection

## 1. Introduction

Event detection on time-series data is an important building block of many real-world applications [1, 2]. It perceives a time series of measurements as one of events. Our notion of event encompasses changes [2, 3] and frequent time-series patterns (motifs) [4, 5, 6]. Changes are points of time when properties (e.g., mean, probability distribution) of a time series change. Frequent patterns are contiguous subsequences of a time series that occur frequently, indicating a structure or information with some regularity [4]. To illustrate the notion of event detection, think of energy consumption data from a smart meter, which serves as our running example. Event detection on such data allows to detect interesting patterns (turning a device on/off, abnormal device activity). Detecting such events is necessary for demand side management, peak shifting, peak shaping, etc. – all elementary 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 in turn can aggravate the subsequent detection of those events significantly.

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

Approaches like lossy compression [7], estimation [8] or perturbation/anonymization [9] lossily transform the time series before event detection takes place: A lossy transformation can reduce the data volume, generate approximate versions of the data, or remove personal information from a dataset. However, existing similarity measures for time series, applied to the original time series and the one after lossy compression and decompression, do not quantify the impact of a lossy transformation on event-detection quality in a way that is conclusive in general [10, 11, 12, 13]. Such a quantification however is needed to identify and parameterize a compression algorithm or anonymization approach, given a certain dataset and quality requirements on the event-detection result. This quantification sought is difficult, for several reasons: First, as shown in Example 1, the impact is manifold. One therefore needs to determine possible effects of a lossy transformation on events. Second, the definition of a measure for this impact is not obvious. It is necessary to 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 most likely be more detrimental if compression eliminates certain events from the data, as opposed to the insertion of new ones. In other contexts, the picture is different. Fourth, identifying the specific effect of a transformation on an event (e.g., shift in time vs. disappearance) is an application-dependent procedure, which must take all events into account. This is because assigning an effect to a certain event may cascade and influence the assignment of effects to other events. Put differently, even if the measure is

defined, algorithms for its efficient computation remain to be designed.

In this paper, we propose and evaluate MILTON, *a practical and flexible Measure which quantifies the Impact of various Lossy Transformation methods for time series on subsequent event detectiON*. MILTON is applicable whenever one wants to know how much a certain transformation approach for time series reduces the result quality of an event-detection technique, as compared to event detection on the original data. This lets an operator, say, decide how much he can compress or perturb data without affecting event detection considerably. Thus, MILTON is useful when choosing from several lossy transformation techniques, by quantifying their impacts on events. To ensure flexibility, we do not impose any restriction on the event detection or the transformation approach used, and we allow to flexibly weight effects on events. We have also investigated several cases that we deem recurrent and propose corresponding parameterizations of MILTON. In contrast to metrics of the quality of event detection methods (e.g., recall, precision, F-score), MILTON's purpose is to quantify the impact of lossy transformations on events. It is applicable to any event-detection algorithm that takes place subsequently.

At first sight, a complement of MILTON or even an alternative to it might be a model of the loss of data quality due to a transformation. However, such a model would have to be generally applicable. But it is difficult to impossible to integrate each of the many existing lossy transformation techniques and event-detection approaches into one model.

In this article, we now make the following contributions:

- We study characteristics of application scenarios that do event detection on lossily transformed time-series data.
- We propose a measure of the impact of time-series transformation methods on subsequent event detection.
- We carry out an evaluation of our measure using five different use cases, namely compression, estimation, anonymization, assisted living and activity hiding.

We have carried out extensive experiments, which have revealed interesting insights on the relationship between the transformation technique in use and event-detection quality. For instance, different anonymization techniques may have a very different impact on event detection, although they protect against noise filtering equally well. We also have found MILTON suitable with any combination of lossy transformation technique and event-detection approach we have encountered. In addition, the design of MILTON enables a flexible customization of the different effects a lossy transformation may have on events. Finally, it is applicable in many application areas in a straightforward manner.

**Paper structure:** Section 2 describes five application scenarios for our measure. Section 3 introduces and explains MILTON, which Section 4 evaluates. Section 5 reviews related work, and Section 6 concludes. – This article is an extended version of [14]. The extensions are the following: Our study of application scenarios that consider time-series patterns is broader, as is our respective evaluation. Next, we have extended all of our approach so that it now also subsumes time-series patterns, as opposed to only changes.

## 2. Application Scenarios

In this section, we describe five scenarios – this will serve as motivation behind our measure. We then 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. Compression Scenario*

#### *2.1.1. Description*

The growing number of smart meters as well as the increasing frequency at which data is collected make storing and transferring the data much more expensive. To illustrate, while smart meter readings often take place every 15 minutes, meters that collect and send data every second are now proliferating. Moreover, such meters now collect and send several values instead of just one, e.g., voltage, current, frequency, active power, etc. The collected data is useful for analyses such as energy-consumption forecasts [15] or energy disaggregation [16]. To store and communicate this data, recent research has produced numerous model-based lossy compression techniques [7, 17, 18, 19]. In contrast to lossless ones, they can obtain significantly higher compression ratios. Lossy methods typically produce a piecewise approximation of the original data within an error threshold $\epsilon$. Thus, they do not only modify the original data, but also the changes present in it. An energy provider intending to use the compressed data for analytics needs to take these effects into account.

#### *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 $\epsilon$. We expect compression results, and consequently their impact on changes, to strongly depend on this parameter.

#### *2.1.3. Setting*

The energy provider employs a forecasting application that uses the compressed time series to predict the energy consumption. By detecting changes in data streams and integrating them in the learning model, he can improve forecasting [20] or enhance stream mining [21]. Thus, here, detecting a change in the time series triggers an update of the model behind the forecasting algorithm, to improve predictions. In such a case, it makes sense to penalize changes which disappear ("missed", also referred to as "false negative" in the literature) more than those which emerge ("false positives") as an effect 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. Estimation Scenario

#### 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 [22]. Quantifying this type of consumption reliably is thus important for many business cases. The cost of deploying a smart meter for each computer that is currently operational however is daunting. Instead of measuring it, some recent approaches estimate the consumption of computers [8], based on, say, specific information on the hardware [23] or based on a profile of the computer-power usage [24]. This can be seen as 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 running the center [8]. He can also use the data for other use cases which employ change-detection methods including consumption-event characterization or detection of abnormal consumption. A data-center manager thus needs to to choose an estimation method appropriate for subsequent change detection.

#### 2.2.2. Problem Domain

Estimation methods function differently from each other in order to obtain approximations. We are aware of several classes: A first class performs a sophisticated calibration process [25], while another one relies on specific models of components [26]. 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 [8].

#### 2.2.3. Setting

Here, the data-center manager will use estimates to balance energy demand and supply with the following application: A significant event will trigger an alarm, so that the energy supply adjusts to the new level. This means that shifts and modifications of the importance of events are critical to the subsequent application. Regarding missed and false-positive events, the manager uses a similar logic as in the previous scenario. A missed event 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. Anonymization Scenario

#### 2.3.1. Description

Smart meters can measure the energy consumption of households with a high frequency and communicate it. While these measurements are useful for analytical purposes, they also allow to infer personal information, such as the daily routine of residents [27, 28]. The pseudonymization of the data is not sufficient. This is because an easy re-identification of consumers using simple statistical measures is possible [27]. Adding noise (e.g., white noise) to the data does not facilitate privacy either, as one can easily filter it out [9]. 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 [9]. Such methods ensure that certain pieces of information contained in the original data are perturbed. 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 an example, [29] has studied the impact of anonymization on local energy markets. In such a market, participants can place bids based on their consumption data, which contains private information. To enhance their privacy, participants can choose to anonymize their consumption data. However, this infers additional costs as the market now operates with perturbed data instead of the real one. The impact of anonymization on events in the data is not yet known. A participant in such a market intending to anonymize energy-consumption data nevertheless needs to quantify this effect. This is because the data should remain useful for subsequent analyses.

#### 2.3.2. Problem Domain

The result of anonymization using the above-mentioned methods depends on the basis chosen for the transformation of the data (e.g., Fourier or Wavelet). The other important parameter in this case is the magnitude of the noise $\sigma$ added to the original data. We conjecture that, the larger the noise added to the data, the larger is the impact on the changes.

#### 2.3.3. Setting

Here, the use case is the general one of data publishing, implying that the provider 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. Assisted Living Scenario

#### 2.4.1. Description

Time-series motifs are sequences of time series which are similar to each other [30]. They are useful in many domains, as they may hint at interesting structures. In energy-consumption time series, these may be activity patterns, i.e., consumption

patterns of devices. Detecting these can enable interesting applications: building an energy-consumption profile of a household or enabling an automatic surveillance of energy consumption [31]. Compressing this data piecewisely in a lossy manner [7, 17, 18, 19] may alter such patterns significantly and make this data less usable for those applications. Thus, an energy provider intending to offer such applications needs to take the effects of lossy transformations into consideration.

#### *2.4.2. Problem Domain*

The result of lossy compression methods depends on the compression model they use. Moreover, as compression methods depend on the error threshold $\epsilon$, we will study the impact of lossy compression on motifs for different values of this parameter.

#### *2.4.3. Setting*

The energy provider will use the compressed time series for an automatic surveillance in assisted home living. If the compression causes consumption patterns to disappear, an alarm will be set off although this is not adequate. Therefore, in this case, it makes sense to introduce penalties if patterns disappear. At the same time, if the compression introduces patterns which do not occur, this will prevent necessary alarms to be set off. Thus, we must also penalize the occurrence of false-positive patterns significantly. Additionally, we should penalize shifts of such usage patterns in time or score according to their magnitude, as they may hint to anomalous activity although there is none.

### *2.5. Activity Hiding Scenario*

#### *2.5.1. Description*

Smart meters facilitate inferring personal information. Such information may correspond to private daily activities, such as taking a shower or watching television. The methods described in the subsection on the anonymization scenario can perturb certain pieces of personal information. We consider here the extraction of personal information from energy consumption time-series. Thus, in the current scenario, an energy provider wants to investigate perturbation methods and determine which ones can conceal private information in the form of daily activities. Additionally, the energy provider wants to find adequate parameters, so that the necessary amount of perturbation is added to the time series.

#### *2.5.2. Problem Domain*

As in the case of the anonymization scenario, the result of the perturbation depends on the basis chosen for the corresponding transformation [9]. As the perturbation methods also depend on the extent $\sigma$ of noise added, we will also investigate how this parameter influences the detection of patterns in time series.

#### *2.5.3. Setting*

In this scenario, the energy provider will use an existing perturbation method to hide private activities. It is thus important to quantify to which extent events in the perturbed data can be matched to events in the original one. Perturbation methods typically are evaluated by counting how many events that are detected in the original data can be detected in the transformed data as well. However, we believe that the evaluation of these methods should be more elaborate than this conventional one. For some applications, it is important if events are shifted in time, while a modification of their significance is less important. Such an elaborate evaluation should be possible with our measure. In this scenario, we use the general case of data publishing, where no information on the subsequent use of the data is known.

### *2.6. Measure Requirements*

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

**R1: Generalizability** Given the huge number of scenarios, transformation techniques and event detection schemes currently in existence, the measure should be independent of the event-detection algorithm and should provide meaningful results for any combination of event-detection approach and lossy transformation.

**R2: Flexibility** A transformation can have various effects on events in the data. These effects may have different weights, depending on the subsequent application. Thus, the user should be able to configure the measure to distinguish and weight at least four cases according to the subsequent application: shifts of events in time, modifications of their importance, disappearance of events and emergence of new events.

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

Observe that a measure cannot fulfill all requirements in all situations. As an example, a user can assign certain weights to the different impacts of a transformation, i.e., flexibility is given. This however has an adverse impact on robustness, as the weighting is unlikely to lead to meaningful results in all situations.

## 3. MILTON

In this section, we first describe the main ideas behind MILTON. We then present MILTON in detail and say how we have parameterized it.

### *3.1. Problem Definition*

An event-detection algorithm *ED* transforms a time series of measurements $X$ into one of events (patterns, changes) $ED(X) = \{(t_1, s_1), (t_2, s_2), \ldots, (t_m, s_m)\}$. Here, $t_i$ with $i = 1, \ldots, m$ denotes the time the event occurred at, while $s_i$ denotes its score. Many state-of-the-art event-detection approaches associate with each event a score [2, 3], which characterizes its significance. As an example, in the case of a change, this corresponds to how abrupt it is, i.e., how big the difference in parameters of a time series before and after the change is. In the case

of time-series motifs, the significance could be defined as the similarity of a motif to a given occurrence. Without loss of generality, we assume that an event has a score of 1 if an event-detection approach does not provide scores. A change is given whenever the difference in a property (e.g., probability distribution) between two subsequences of a time series crosses a given threshold. A motif occurs in a time series once the distance between a subsequence of the time series and the motif is smaller than a given threshold. A lossy transformation $T$ on $X$ produces a modified time series of measurements *T(X)*. Applying *ED* on *T(X)* thus produces a time series of events *ED(T(X))*, which is possibly different from *ED(X)*. Table 1 sums up our notation introduced so far.

| Symbol | Definition |
|---|---|
| $X$ | original time series |
| $T$ | lossy transformation |
| *ED* | event detection algorithm |
| *ED(X)* | time series of events |

Table 1: Notation Summary

When comparing the events in *ED(X)* and *ED(T(X))*, we can assign each event to one of the following sets:

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

**MISS = ED(X) − PE:** As a result of the transformation, some events of *ED(X)* might not have a match in *ED(T(X))*. The set *MISS* contains such events, called "misses".

**FP = ED(T(X)) − PE:** In contrast, new events might appear in *ED(T(X))*, which do not have any match in *ED(X)*. We refer to such events as "false positives", which form the set *FP*.

From Requirement R2 (cf. Subsection 2.6), it follows that MILTON must consider each set defined above differently. In particular, we must determine the events the transformation has affected in a quantitative way (quantitative = shift in time or modification of score; *PE*), how many have disappeared (*MISS*), and how many have emerged as a result of the transformation (*FP*). Second, we should allow for application-dependent weights describing the importance of each set. For example, if missed events 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 event detection, MILTON quantifies how similar *ED(X)* and *ED(T(X))* are, i.e., we sum the weighted differences between the events *ED(X)* and *ED(T(X))* assigned to *PE*, and the weights of the events in *MISS* and *FP*.

### *3.2. Calculating PE, MISS and FP*

We first explain how the function **pairing**() can be implemented to obtain the set *PE*. Obtaining sets *FP* and *MISS* follows in a straightforward manner.

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

Due to Requirement R2, our measure must take into account both the delay and difference in score (importance) between two events. 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 an event of *ED(X)* and $y = (t_y, s_y)$ one of *ED(T(X))*. We define $f_{TIME} : \mathbb{R} \mapsto \mathbb{R}^+$ as a function of the normalized difference in time ($\Delta_t$) between $x$ and $y$ and $f_{SCORE} : \mathbb{R} \mapsto \mathbb{R}^+$ as a function of the normalized difference in score ($\Delta_s$) between $x$ and $y$. The normalization can be performed differently. We have chosen to normalize shifts in time using the length of the original time series. This makes sense as the longer the time series is, the smaller the shifts are relative to its length. We have normalized the score using the maximum score of an event in the original time series. This is meaningful since scores with some event-detection techniques are unbounded. The functions $f_{TIME}$ and $f_{SCORE}$ are application-dependent, as explained above. The distance between two events then is a function $g$ of $f_{TIME}$ and $f_{SCORE}$:

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

In the remainder of this paper, 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 events *ED(X)* and *ED(T(X))* is to calculate all possible one-to-one combinations of events which maintain the original succession of events 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 events in *ED(X)* and *ED(T(X))*. Next, at first sight, the setting may resemble the one of the well-known Hungarian Algorithm. However, this is not a solution either: The difference is that we in our case must maintain

the original order of events for the matching. Several publications however have studied this specific problem or closely related ones [13, 32]. We use the Optimal Subsequence Bijection (OSB) algorithm introduced in [13] as starting point, which fulfills the matching prerequisites:

- the matching should consider all distances between matched events
- the matching should allow for events remaining unmatched (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 events into account. OSB matches two sequences *ED(X)* and *ED(T(X))* of (possibly different) lengths $m$ and $n$:

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

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

Its goal is to find best-matching subsequences $ED(X)'$ of *ED(X)* and $ED(T(X))'$ of *ED(T(X))*. Thus, it may skip events. The authors of OSB motivate having unmatched events with the fact that the sequences may contain outliers that should be skipped. In our case, these outliers correspond to false-positive and missed events. 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 events as input. To find the optimal matching, OSB minimizes the sum of the distances between matched events and the penalties for events skipped. It thus considers all distances between matched events, as required. OSB finds the optimal matching by performing a shortest path algorithm on a directed acyclic graph (DAG). The nodes of the DAG are index pairs $(i, j) \in \{1, \ldots, m\} \times \{1, \ldots, n\}$ of *ED(X)* and *ED(T(X))*, and the cost $w$ of an edge between nodes $(i, j)$ and $(k, l)$ is:

$$w((i,j),(k,l)) = \begin{cases} \sqrt{(k-i-1)^2 + (l-j-1)^2} \cdot C + dist((t_{x_k}, s_{x_k}), (t_{y_l}, s_{y_l})) & \\ \qquad \text{if } i < k \wedge j < l & \\ \infty \qquad \text{otherwise} & \end{cases}$$

where $dist((t_{x_k}, s_{x_k}), (t_{y_l}, s_{y_l}))$ is the distance between elements (events) $(t_{x_k}, s_{x_k}) \in ED(X)$ and $(t_{y_l}, s_{y_l}) \in ED(T(X))$. [13] uses the Euclidean distance. We adapt OSB to our case by using the distance from Equation (2), next to some other adaptations, described next.

There are many possibilities to set the penalty $C$. From our experiments, we have found that the standard penalty recommended in [13] produces correct matchings and rarely results in mismatches between events. We therefore used the standard penalty, which is defined as follows:

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

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

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

```
Let MISS = {}
Let FP = {}
PE = OSB(ED(X), ED(T(X)), C)
for x ∈ ED(X) do
    if x ∉ PE then
        MISS = MISS ∪ x
    end if
end for
for x ∈ ED(T(X)) do
    if x ∉ PE then
        FP = FP ∪ x
    end if
end for
```

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

**Example 2.** *Consider the time series ED(X) of original events, each represented as X, and the one of events after applying a transformation T, ED(T(X)), each represented as +, in Figure 1. Here, applying the matching algorithm provides the set PE containing the pairs of events within circles. The set MISS contains the third event from the left from ED(X), while the set FP contains the fourth event from the left from ED(T(X)).*

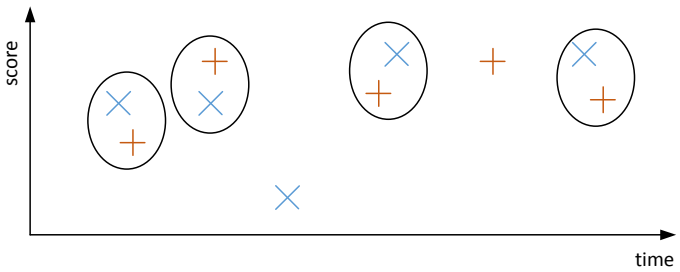


Figure 1: Paired events

### 3.3. Measure Definition

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

#### 3.3.1. Paired Events

Paired events 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 total impact of such events, we sum up the distances between paired events. This yields the following term, which is part of our measure:

$$errPE = \sum_{(x,y)\in PE} dist(x, y) =$$

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

The intuition is that, the more events are shifted in time and score, the bigger the difference between the original time series and the transformed one. In case separate information on the impact of shifts in time and the one of the score was necessary, $errPE$ could be split into two terms calculated separately:

$$errPE = errTIME + errSCORE$$

where

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

and

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

#### 3.3.2. 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 events with a low score and conversely assign a bigger weight to ones with a high score. We therefore introduce a weighting function on missed events $f_{MISS}$ which depends on their score. We normalize the score of events by dividing them by the maximum score of the original events. We discuss how we define this function in the following subsection. To quantify the total impact of missed events, we sum up their individual impacts weighted by $f_{MISS}$, yielding the second term of our measure:

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

#### 3.3.3. False Positives

As in the case of missed events, 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 arrive at 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 events |
| $f_{FP}$ | false-positive events |

Table 2: Notation Summary

#### 3.3.4. 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 events in the original time series $|ED(X)| = |PE| + |MISS|$. We explain the rationale using an example:

**Example 3.** *Suppose that, for a time series $X_1$, ED detects 2 events, 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 of the original events $ED(X_1)$ 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 events intact and introduces a shift in time and score of the two events resulting in the same impact of 0.5. Intuitive, the total impact between the two cases should be significantly different. This is because in the case of $X_2$, $T$ leaves 98% of the events intact, in contrast to 0% of the events in the case of $X_1$. We therefore need to normalize the total impact and divide it by the number of events ED detects in the original time series.*

Using the above results, we define MILTON as follows:

$$MILTON(X, T, ED) = \frac{errPE + errMISS + errFP}{|PE| + |MISS| + 1} \quad (5)$$

We add 1 to the denominator to cope with the case when $PE$ 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 $f_{FP}$. The first two are crucial for the matching process, while the latter two do not influence it. $f_{MISS}$ and $f_{FP}$ facilitate a weighting of missed and false-positive events according to the requirements of the application scenario. The definition of all parameters is up to the user of MILTON. To discuss the right choice of parameter values, we present two cases in the following, which we deem useful for the interpretation and definition of the parameters. The first one corresponds to a general use of MILTON with a basic parametrization, while the second one features parameter tuning for recurrent situations.

*Basic case:* In the so-called basic case, $f_{TIME}$ and $f_{SCORE}$ are set equal to the normalized absolute values of the difference between the time of occurrence and, respectively, the score of matched events:

$$f_{TIME}(\Delta_t(x, y)) = |\Delta_t(x, y)|$$
$$f_{SCORE}(\Delta_s(x, y)) = |\Delta_s(x, y)|$$

where $x$ and $y$ are matched events. Thus, in this case the matching algorithm is the OSB algorithm, which we do not modify otherwise. In this case *errTIME* and *errSCORE* will calculate the sum of shifts in time and score between original and transformed events. $f_{MISS}$ and $f_{FP}$ can be set equal to the scores of the missed and, respectively, false-positive events:

$$f_{MISS}(s) = s$$
$$f_{FP}(s) = s$$

Given this, *errMISS* and *errFP* are the sums of the scores of the missed and false-positive events. We also see another meaningful instantiation of these functions, in which $f_{MISS}$ and $f_{FP}$ are set equal to 1. In this case, *errMISS* and *errFP* are the numbers of missed and false-positive events occurring due to the transformation.

*Tuned case:* This case includes additional refinements of the parameters for situations that we deem recurring. For instance, this case covers the following situations:

1. The score of an event often is secondary, and only the fact that the event has occurred is crucial. To account for this, $f_{SCORE}$ can be set equal to zero, and MILTON will thus ignore alterations of the score.

2. In many use cases, measurements are approximate. Thus, only estimates of the time an event has occurred and its score are given. In such cases, small variations in time and score can be ignored by setting $f_{TIME}$ and $f_{SCORE}$ equal to zero for small values:

$$f_{TIME}(\Delta_t(x,y)) = \begin{cases} 0 & \text{if } |\Delta_t(x,y)| < \delta \\ |\Delta_t(x,y)| & \text{otherwise} \end{cases}$$

   where $\delta$ corresponds to the maximum measurement error.

3. In some situations, if a shift in time of an event is bigger than a given threshold, it should not be matched, but treated as a missed or a false-positive event. An example of such a situation is when an action should take place as a response to an event immediately after it or within a given period of time $R$. Here, $f_{TIME}$ can be defined as follows:

$$f_{TIME}(\Delta_t(x,y)) = \begin{cases} |\Delta_t(x,y)| & \text{if } |\Delta_t(x,y)| < R \\ N & \text{otherwise} \end{cases}$$

   where $N$ is a value big enough to prevent the matching of the events.

We have used $f_{TIME}$ and $f_{SCORE}$ to illustrate the situations above. $f_{MISS}$ and $f_{FP}$ can be defined in a similar manner. As an example, they can be set to 0 if missed or false-positive events should be ignored if their score is small.

In the following we say how we set the parameters for each application scenario described in Section 2. We start with the compression scenario. Here, we must penalize missed changes significantly 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 $f_{SCORE}$ to zero. This is in order to ignore alterations of the importance of changes.

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

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 let $f_{TIME}$ and $f_{SCORE}$ grow exponentially with increasing shifts in time and score (Table 3). Regarding $f_{MISS}$ and $f_{FP}$, we use a similar reasoning as for the previous scenario where we penalize missed changes substantially more than false-positive ones.

Next, we consider the anonymization scenario. As stated, this is 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 types of shift (Table 3). We thus set *errFP* and *errMISS* equal to the sum of the scores of changes in the respective sets (*MISS* and *FP*).

For the assisted living scenario, as explained earlier, we let $f_{MISS}$ and $f_{FP}$ grow with the magnitude of the shifts. This is because the more events shift in time or score, the more the corresponding pattern may seem anomalous. We also penalize missed and false-positive events accordingly.

Lastly, for the activity hiding scenario we use the parameters of the basic case described above. This is because we assume that the energy provider does not have any initial requirement regarding the impact of perturbation/anonymization on patterns.

## 4. Evaluation

MILTON operates as intended if it fulfills the requirements from Subsection 2.6. We have covered the flexibility and robustness requirements by design, as explained in Subsection 2.6. In addition, we experimentally evaluate the robustness of our measure with our five different parametrizations (Table 3). To cope with generalizability, we evaluate MILTON using our five scenarios. For the compression scenario, we have performed a further experiment with different change-detection methods on one dataset, in combination with compression (end of Section 4.3.1). We have made the code of MILTON available on our website[1]. In the following, we describe the datasets used, the setup of our experiments and their results.

[1] https://dbis.ipd.kit.edu/2411.php

### 4.1. Datasets

To evaluate MILTON, we use five datasets, as follows. We use the first two for the evaluation of the compression, anonymization, assisted living and activity hiding 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 [16]. 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 [33]. 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 carried out at our institute. They contain real and estimated energy-consumption data from three computer systems. We had used a digital multimeter Wattsup PRO [34] (accuracy: 1.5%) to record the reference energy consumption. We have used a dynamic estimator and a calibration-based one [8] 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 [8] 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 [8]. 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 involving change detection, we have used *CUSUM* [1], 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. For the assisted living and activity hiding scenarios we used the method from [30] to identify motifs in time series. An example of such a motif would be the consumption pattern of an electrical device. Given these motifs, we localize their occurrences in the time series. 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.

**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 [35].

b) Piecewise Linear Histogram (*PWLH*) compresses the data in a similar manner as APCA, using straight-line functions instead of constant ones [36].

c) Adaptive Polynomial Piecewise Compression (*APP*) combines polynomial functions of different degrees to approximate the data piecewisely in an incremental manner [7].

All these methods compress the data so that the maximum deviation between the original and decompressed data under the uniform norm is smaller than an error threshold $\epsilon$.

**Estimation Scenario:** As mentioned, we use a dynamic and a calibration-based estimator to obtain the energy-consumption estimates, cf. [8].

**Anonymization Scenario:** We use two data perturbation/anonymization methods: one using the Fourier transform and one using the Wavelet transform [9].

**Assisted Living Scenario:** For this scenario we use the compression techniques from the compression scenario.

**Activity Hiding Scenario:** For this scenario we use the data perturbation techniques from the anonymization scenario.

### 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 with a good parameter set, while the maximal impact on the changes in the data is given. To this end, we have first computed MILTON for all three compression techniques for different values of the threshold $\epsilon$, from 0.2% to 5% of the range of the time series used. Figure 2 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 $\epsilon$, compression using *APCA* has a worse impact on the subsequent change detection than the other two methods.

To understand the reasons behind the results in Figure 2, we list and inspect the number of changes in sets *PE*, *MISS* and *FP*, as well as MILTON components *errPE*, *errMISS* and *errFP* for one time series of the Smart Dataset (Table 4). 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 have set

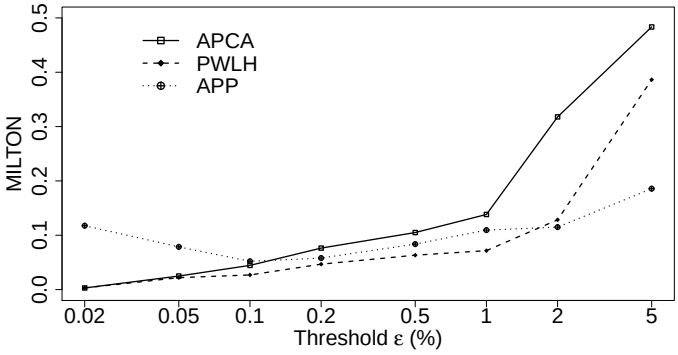


Figure 2: MILTON vs. Threshold ($\epsilon$) - Smart Dataset

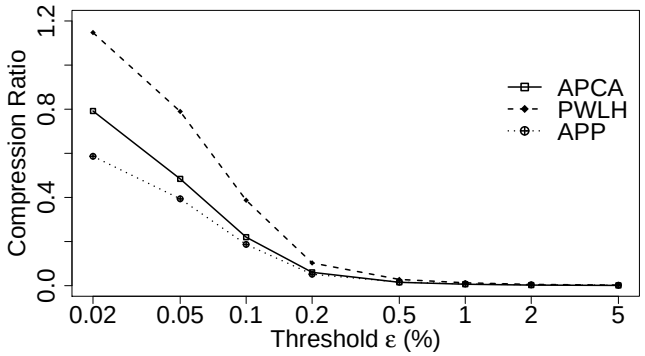


Figure 3: Compression Ratio vs. Threshold ($\epsilon$) – Smart Dataset

the weight $f_{FP}$ for false-positive changes significantly smaller than for misses ($f_{MISS}$).

| | $\epsilon$ | *PE* | *MISS* | *FP* | *errPE* | *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: MILTON Components by Threshold $\epsilon$ – homeB – Smart Dataset

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

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

Figure 3 shows the average results for the Smart Dataset. We notice that the results of the different compression methods diverge significantly for small values of $\epsilon$ ($\epsilon < 0.2\%$) and converge to similar ones when $\epsilon$ grows ($\epsilon > 1\%$). For small values of $\epsilon$, compression with *PWLH* is worst, followed by *APCA* and *APP*. For very small values of $\epsilon$, *PWLH* does not achieve any compression at all. This is because it needs more space for the piecewise segments that approximate the data than for the original data.

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 $\epsilon$ 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 compression method is generally superior to the other ones.*

*Evaluation with Further Change-Detection Schemes.* We have performed one further experiment regarding the compression scenario. We have fixed the dataset (REDD) and the compression method (APCA) and have calculated MILTON with the same parametrization as above for two additional change-detection schemes. In the following we first describe these schemes and then present the results of this experiment.

We have used two established change-detection schemes provided by the *cpm* R-package [37]. The first uses the Student-t test statistic to determine if two sets of data are significantly different from each other. We call this scheme "Student" in the following. The second one uses the Cramer-von-Mises test statistic to detect arbitrary changes in a stream of data. We refer to this scheme as "Cramer" in the following.

Table 5 shows the results for one random day of House 3 of the REDD dataset. Results are similar in the case of the other houses. We make the following observations: First, for both schemes, the number of changes missed, together with the value of MILTON, increases with $\epsilon$. This is in line with the results of the compression scenario above. Second, the Student scheme detects more changes overall than the Cramer scheme. Third, the Student scheme is less sensitive to compression than Cramer for small values of $\epsilon$. All this means that MILTON can be used to analyze how compression affects different change-detection schemes.

| | $\epsilon$ | *PE* | *MISS* | *FP* | *errPE* | *errMISS* | *errFP* |
|---|---|---|---|---|---|---|---|
| Student | 0.02 | 92 | 0 | 0 | 0.000 | 0.000 | 0.000 |
| | 0.05 | 92 | 0 | 0 | 0.000 | 0.000 | 0.000 |
| | 0.1 | 92 | 0 | 0 | 0.000 | 0.000 | 0.000 |
| | 0.2 | 91 | 1 | 1 | 0.000 | 1.000 | 0.000 |
| | 0.5 | 91 | 1 | 1 | 0.000 | 1.000 | 0.000 |
| | 1 | 90 | 2 | 0 | 0.003 | 2.000 | 0.000 |
| | 2 | 86 | 6 | 12 | 0.155 | 6.000 | 0.002 |
| | 5 | 79 | 13 | 14 | 0.129 | 13.000 | 0.002 |
| Cramer | 0.02 | 81 | 3 | 3 | 0.004 | 3.000 | 0.000 |
| | 0.05 | 81 | 3 | 3 | 0.010 | 3.000 | 0.000 |
| | 0.1 | 79 | 5 | 8 | 0.010 | 5.000 | 0.001 |
| | 0.2 | 81 | 3 | 3 | 0.029 | 3.000 | 0.000 |
| | 0.5 | 82 | 2 | 3 | 0.046 | 2.000 | 0.000 |
| | 1 | 80 | 4 | 2 | 0.061 | 4.000 | 0.000 |
| | 2 | 78 | 6 | 5 | 0.103 | 6.000 | 0.001 |
| | 5 | 78 | 6 | 4 | 0.143 | 6.000 | 0.001 |

Table 5: MILTON Components by Threshold $\epsilon$ – Student-t and Cramer-Von-Mises Change Detection Schemes – house 3 – REDD Dataset

#### *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. To this end, 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 6 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 [8], 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 6: 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 *PE*, *MISS* and *FP*, as well as *errPE*, *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 lets the values of MILTON grow significantly due to the definition of $f_{MISS}$ and $f_{FP}$. Moreover, the pairing matches changes better for small interval lengths than for longer ones. This accounts for the big impact of aggregation on changes for long intervals.

| | Interval length (min) | *PE* | *MISS* | *FP* | *errPE* | *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 7: Number of Changes and Measure Components by Interval Length – Desktop Dataset

Summary: *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.*

#### *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 analyse. To study this, we computed MILTON for both anonymization methods. We vary the value of the perturbation $\sigma$ these methods add, as described in [9].

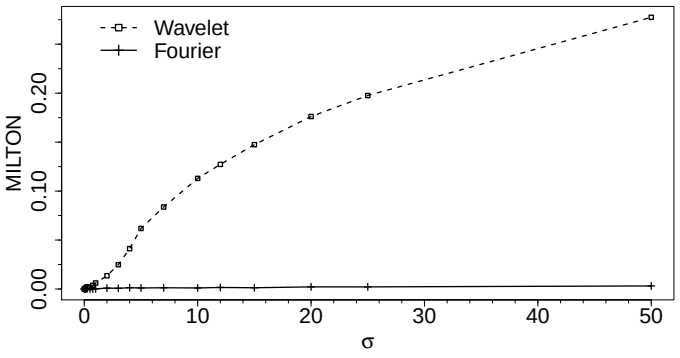


Figure 4: Milton vs. perturbation added $\sigma$ – House 4 – REDD Dataset

Figure 4 shows the average values of MILTON for House 4

of the REDD dataset when the perturbation $\sigma$ goes from 0.001 to 50. In this case, these values of $\sigma$ 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 *PE*, *MISS* and *FP*, as well as MILTON components *errPE*, *errMISS* and *errFP* for one time series of House 4 in Table 8. 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.

| | $\sigma$ | *PE* | *MISS* | *FP* | *errPE* | *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 8: Number of Changes and Measure Components by Perturbation $\sigma$ – 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 [9].*

#### *4.3.4. Assisted Living Scenario*

In this scenario, the energy provider wants to identify the method which delivers the best compression ratio for a given impact on the consumption patterns together with a good parameter set for a household-surveillance application. Before we present results, we exemplarily illustrate the impact of a compression method on a consumption pattern. Figure 5 shows the original (top) and decompressed sequence of House 1 of the REDD dataset using *APCA* with two error thresholds (middle and bottom). The repeated pattern present in the figure corresponds to the consumption of the refrigerator. As we can see, for $\epsilon = 1\%$ the compression smoothen out details of the data but does not alter the occurrences of the pattern significantly. On the other hand, for $\epsilon = 10\%$, the compression does have a big impact on them.

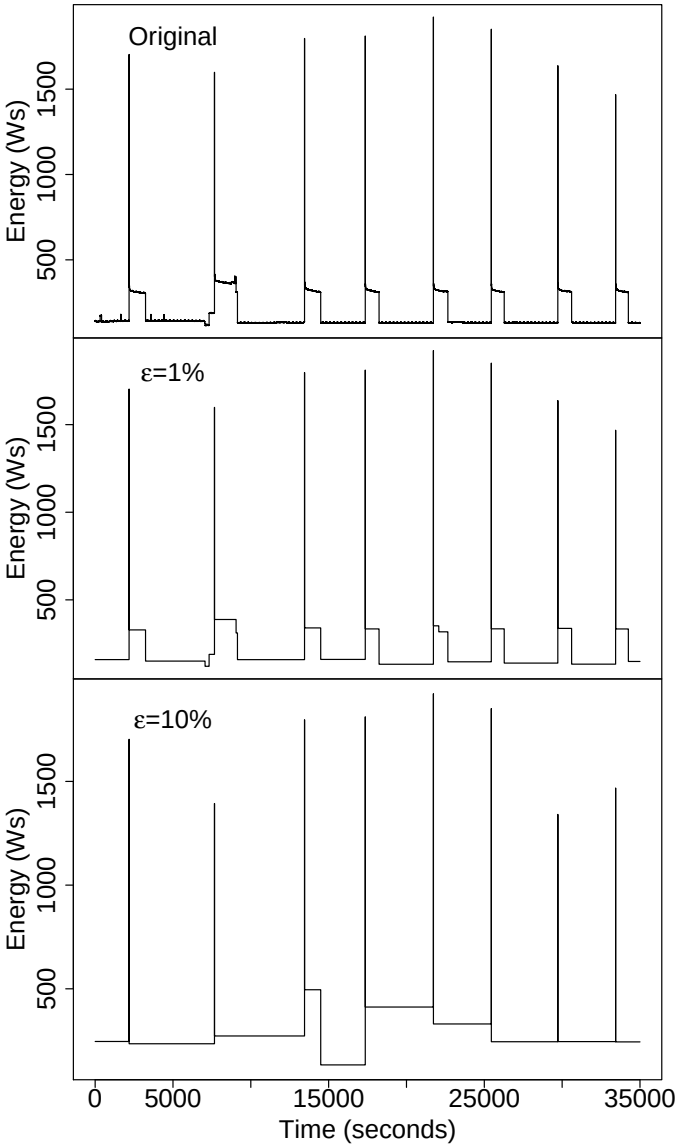


Figure 5: Original and decompressed data for different values of $\epsilon$ using APCA

As a next step, we computed MILTON for all three compression techniques for different values of the threshold $\epsilon$, from 0.1% to 5% of the range of the time series used. Figure 6 shows the average results for the Smart Dataset. These are similar to those we obtained with the REDD Dataset. As expected in line with the illustration above, we also actually observe that MILTON *generally* increases with a growing value of $\epsilon$. The reason is that patterns are altered significantly and cannot be matched as an effect of the rougher compression. We notice that, in some cases, MILTON decreases when the compression becomes rougher (e.g., when going from 0.5% to 2%). This means that a rougher compression (increase in error threshold $\epsilon$) does not always correspond to a larger distance between the original and decompressed pattern.

Interestingly, *APP* has an almost overall worse impact on pat-

tern detection than the other two methods. Given the results from Figure 3 regarding compression ratios, we see that there is a trade-off in this case. Using APP, we achieve better compression ratios. This however comes at the cost of a bigger impact on patterns in the time series.

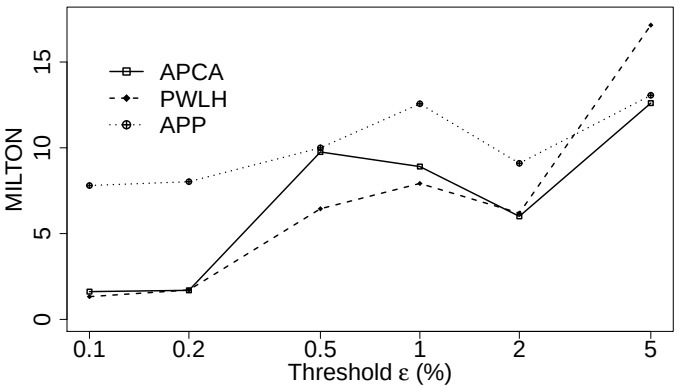


Figure 6: MILTON vs. Threshold ($\epsilon$) - Smart Dataset Motif

To understand why the effects above occur, we study the number of changes in sets *PE*, *MISS* and *FP* as well as *errPE*, *errMISS* and *errFP* for one time series of the Smart dataset (Table 9). We see that the rougher compression causes more patterns to be unmatched and thus increases the number of misses. Another interesting fact is that compressing the data does not tend to cause false positives. Using the results above, the provider can identify the compression method with the best compression ratio given a bound on the impact it may cause.

Summary: *We have shown that MILTON can help the provider decide which compression method suits his needs best. In this use case, the results show a trade-off between compression performance and impact on patterns.*

| | $\epsilon$ | *PE* | *MISS* | *FP* | *errPE* | *errMISS* | *errFP* |
|---|---|---|---|---|---|---|---|
| APCA | 0.1 | 16 | 0 | 0 | 0.084 | 0 | 0 |
| | 0.2 | 16 | 0 | 0 | 0.252 | 0 | 0 |
| | 0.5 | 13 | 3 | 0 | 0.304 | 87.792 | 0 |
| | 1 | 13 | 3 | 0 | 0.529 | 87.792 | 0 |
| | 2 | 13 | 3 | 0 | 0.427 | 87.792 | 0 |
| | 5 | 4 | 12 | 0 | 0.577 | 344.774 | 0 |
| PWLH | 0.1 | 16 | 0 | 0 | 0.084 | 0 | 0 |
| | 0.2 | 16 | 0 | 0 | 0.374 | 0 | 0 |
| | 0.5 | 13 | 3 | 0 | 0.541 | 87.792 | 0 |
| | 1 | 15 | 1 | 0 | 1.257 | 29.179 | 0 |
| | 2 | 13 | 3 | 0 | 0.445 | 87.792 | 0 |
| | 5 | 2 | 14 | 0 | 0.841 | 402.747 | 0 |
| APP | 0.1 | 16 | 0 | 0 | 0.004 | 0 | 0 |
| | 0.2 | 16 | 0 | 0 | 0.461 | 0 | 0 |
| | 0.5 | 16 | 0 | 0 | 0.484 | 0 | 0 |
| | 1 | 15 | 1 | 0 | 1.441 | 29.46 | 0 |
| | 2 | 13 | 3 | 0 | 0.320 | 87.792 | 0 |
| | 5 | 4 | 12 | 0 | 0.919 | 344.585 | 0 |

Table 9: Number of Patterns and Measure Components in Compressed Data for Threshold $\epsilon$ – homeB – Smart Dataset

*4.3.5. Activity Hiding Scenario*

In this scenario the energy provider wants to investigate data perturbation methods and identify which methods with which parameter set can better conceal private daily activities. As with the previous scenario, we present an illustrative example of how the perturbation methods affect patterns. Figure 7 shows the perturbed sequences of House 1 of the REDD dataset using the Fourier (left) and Wavelet (right) transforms for two values of the noise added $\sigma$. For illustration purposes, we again choose the pattern corresponding to the consumption of the refrigerator. As expected, the perturbation is bigger and visually observable for growing values of $\sigma$. However, for the Wavelet transform, we can visually notice a more significant impact on patterns than with the Fourier transform. For $\sigma = 50$, patterns are visually difficult to detect for the Wavelet transform, while for the Fourier transform they can be matched with the original ones in a straightforward manner.

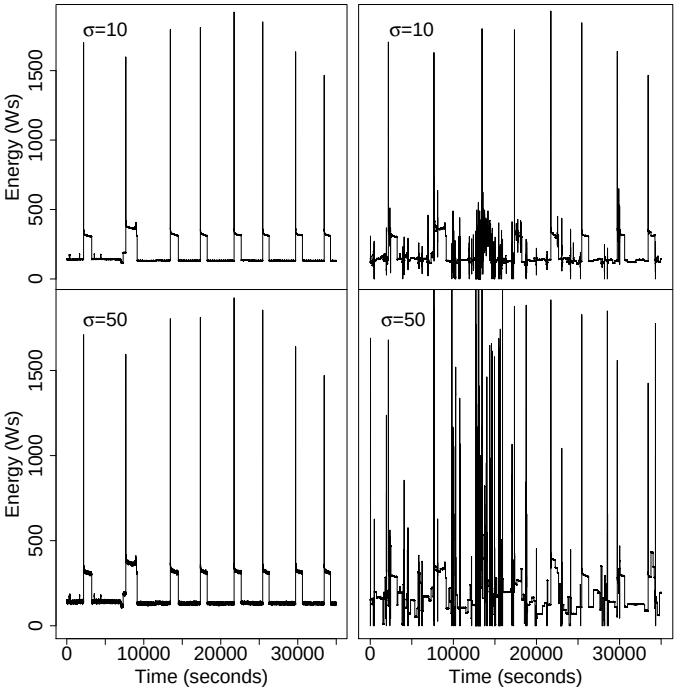


Figure 7: Comparison of anonymized data using Fourier (left) and Wavelet (right)

We now present more detailed results. We first calculated the average values of MILTON for the energy-consumption time-series of each individual household of the REDD and Smart datasets. Figure 8 shows the average values of MILTON for House 4 of the REDD dataset when the perturbation $\sigma$ goes from 1 to 50. The results are similar to the other REDD and Smart time series. We notice that, as in the case of the anonymization scenario, the impact of the Wavelet-based perturbation on patterns grows significantly with the amount of noise added in comparison to the Fourier-based method.

To understand the reasons behind this, we investigated the number of changes in the sets *PE*, *MISS* and *FP*, as well as *errPE*, *errMISS* and *errFP* for different values of $\sigma$. We show the results for one time series of House 4 in Table 10. Here, we

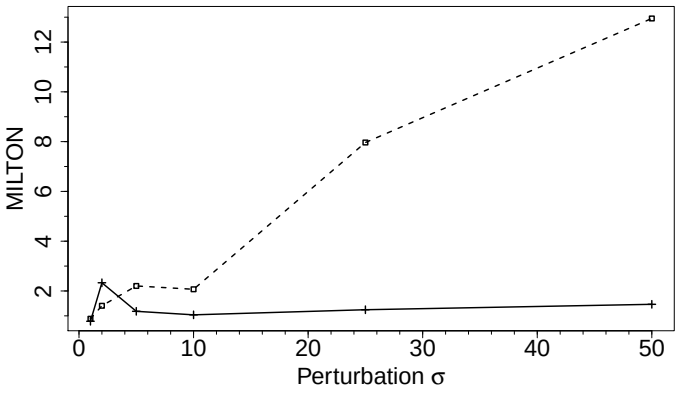


Figure 8: Milton vs. perturbation added $\sigma$ – House 4 – REDD Dataset

see that for the Wavelet-based method, the number of missed pattern occurrences is significant for values of $\sigma$ above 25, while it stays almost constant for the Fourier-based method. We also see a result that may seem counter-intuitive at first sight – in some cases, the impact is smaller for a bigger $\sigma$. This happens due to the random nature of the noise added. It may happen that for a bigger $\sigma$ a bigger part of the noise has been added to parts of the time series not containing the patterns.

| | $\sigma$ | *PE* | *MISS* | *FP* | *errPE* | *errMISS* | *errFP* |
|---|---|---|---|---|---|---|---|
| Fourier | 1 | 17 | 0 | 0 | 0.001 | 0.000 | 0.000 |
| | 2 | 16 | 1 | 1 | 0.06 | 28.211 | 28.217 |
| | 5 | 16 | 1 | 1 | 0.06 | 28.211 | 28.217 |
| | 10 | 17 | 0 | 0 | 0.006 | 0.000 | 0.000 |
| | 25 | 16 | 1 | 1 | 0.612 | 27.940 | 28.043 |
| | 50 | 17 | 0 | 1 | 0.637 | 0.000 | 28.231 |
| Wavelet | 1 | 17 | 0 | 0 | 0.513 | 0.000 | 0.000 |
| | 2 | 17 | 0 | 1 | 2.248 | 0.000 | 27.778 |
| | 5 | 17 | 0 | 1 | 2.248 | 0.000 | 27.778 |
| | 10 | 17 | 0 | 1 | 2.282 | 0.000 | 27.587 |
| | 25 | 15 | 2 | 0 | 2.105 | 56.058 | 0.000 |
| | 50 | 8 | 9 | 0 | 1.085 | 252.306 | 0.000 |

Table 10: Number of Changes and Measure Components by Perturbation $\sigma$ – House 4 – REDD Dataset

Summary: *Using our measure, the energy provider can quantify the impact of anonymization methods on consumption patterns. In this case, the methods (Fourier-based and Wavelet-based) have a significantly different impact on the patterns. Thus the energy provider can choose the one that better conceals consumption patterns with an adequate set of parameters.*

## 5. Related Work

We now review modern change-detection methods, time-series similarity measures and lossy transformation techniques. We are aware of three classes of lossy transformation techniques, 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. [3] 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 [1]. It calculates a cumulative sum for the segment currently considered and issues a change alert once this sum exceeds a given threshold. [39] 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. [2] 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, [40] uses an auto-regressive model to detect and remove outliers before detecting changes in the data. [41] 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 [10, 11]. 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 [12]. LCSS determines the longest common subsequence between two sequences. The Optimal Subsequence Bijection (OSB) [13] 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 [17]. Thus, [18] uses constants to approximate fixed-length intervals. [19] presents two methods which produce connected and disconnected piecewise straight-line segments of variable length. [42] proposes using several models in parallel and choosing the one which best compresses the current segment. [7] 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. To this end, the *uniform norm* ($L_\infty$*-norm*) is commonly used [17]. To additionally evaluate the quality of approximation, [17] uses the root mean square error (RMSE). None of the proposals otherwise considers 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 [43]. 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 [44]. Such models are not very 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. [25] for instance estimates the power consumption of a large group of servers by matching it with CPU utilization. [45] 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 [44, 45]. Recent work has identified application scenarios where estimation methods are useful. The authors of [26] use their model to detect energy hotspots in software. [8] presents several use cases which make use of computer energy-consumption data, e.g., energy-aware management of data centers. We are not aware of 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 [46] for an overview. *Differential Privacy* is an intuitive measure of the risk of one's privacy when having personal data in a database [47]. [48] is an example of a privacy-preserving system compliant with differential privacy. Other work has addressed time-series anonymization. [9] proposes several schemes for time-series anonymization in a streaming context. [49] 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 [29] allows to quantify the economic and environmental effects of anonymization on local energy markets. [50] argues that the quality of anonymized data should be measured based on the workload the data would be used for. We however are not aware of any work which explicitly investigates this effect.

### *5.6. Time-Series Pattern Detection*

Time-series pattern detection is a well-researched area with many applications, e.g., clustering, summarization or rule discovery in time series. Some of the work has focused on finding recurring subsequences (motifs) in one dimensional time-series [30, 51]. Such approaches usually discretize the time series and use the Euclidean Distance to detect similar subsequences of equal length. Several approaches have addressed the detection of motifs of variable length [52, 53]. [54] proposes an extension to mining multidimensional time-series motifs. Another part of recent research has focused on the detection of anomalous or surprising patterns [6]. We have not found any work which investigates the effect of lossy transformations on time-series patterns.

## 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 events contained in the data, 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 event detection. Our evaluation shows that it is useful for various application scenarios. It can be used to identify adequate parameters of a lossy transformation so that its advantages are maximized while the impact on subsequent event detection is bounded.

## 7. References


[1] E. S. Page, Continuous inspection schemes, Biometrika.

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

[3] S. Liu, M. Yamada, N. Collier, M. Sugiyama, Change-point detection in time-series data by relative density-ratio estimation, Neural Networks.

[4] P. Patel, E. Keogh, J. Lin, S. Lonardi, Mining motifs in massive time series databases, in: International Conference on Data Mining (ICDM), 2002.

[5] E. Keogh, Efficiently finding arbitrarily scaled patterns in massive time series databases, in: Knowledge Discovery in Databases: PKDD 2003, Springer Berlin Heidelberg, 2003.

[6] E. Keogh, S. Lonardi, B. Y.-c. Chiu, Finding surprising patterns in a time series database in linear time and space, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002.

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

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

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

[10] D. J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series., in: KDD workshop, 1994.

[11] C. A. Ratanamahatana, E. Keogh, Three myths about dynamic time warping data mining, in: SIAM International Conference on Data Mining, 2005.

[12] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, 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.

[13] L. Latecki, Q. Wang, S. Koknar-Tezel, V. Megalooikonomou, Optimal subsequence bijection, in: IEEE International Conference on Data Mining, 2007.
[14] P. Efros, E. Buchmann, A. Englhardt, K. Böhm, How to quantify the impact of lossy transformations on change detection, in: International Conference on Scientific and Statistical Database Management, 2015.
[15] R. Ramanathan, R. Engle, C. W. Granger, F. Vahid-Araghi, C. Brace, Short-run forecasts of electricity loads and peaks, Cambridge University Press, 2001.
[16] J. Z. Kolter, M. J. Johnson, Redd: A public data set for energy disaggregation research, in: SIGKDD Workshop on Data Mining Applications in Sustainability, 2011.
[17] 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.
[18] I. Lazaridis, S. Mehrotra, Capturing sensor-generated time series with quality guarantees, in: International Conference on Data Engineering, 2003.
[19] H. Elmeleegy, A. K. Elmagarmid, E. Cecchet, W. G. Aref, W. Zwaenepoel, Online piece-wise linear approximation of numerical streams with precision guarantees, VLDB Endowment 2.
[20] G. Ristanoski, W. Liu, 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.
[21] Y. Tao, M. T. Ozsu, Mining data streams with periodically changing distributions, in: ACM Conference on Information and Knowledge Management, 2009.
[22] W. Vereecken et al., Overall ICT Footprint and Green Communication Technologies, in: International Symposium on Communications, Control and Signal Processing, 2010.
[23] M. Poess, R. O. Nambiar, Power based performance and capacity estimation models for enterprise information systems., IEEE Data Engineering Bulletin.
[24] A. Kansal, F. Zhao, J. Liu, N. Kothari, A. A. Bhattacharya, Virtual machine power metering and provisioning, in: ACM Symposium on Cloud Computing, 2010.
[25] X. Fan, W.-D. Weber, L. A. Barroso, Power provisioning for a warehouse-sized computer, in: Annual International Symposium on Computer Architecture, 2007.
[26] A. Noureddine, A. Bourdon, R. Rouvoy, L. Seinturier, Runtime monitoring of software energy hotspots, in: IEEE/ACM International Conference on Automated Software Engineering, 2012.
[27] E. Buchmann, K. Böhm, T. Burghardt, S. Kessler, Re-identification of smart meter data, Personal and Ubiquitous Computing.
[28] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, D. Irwin, Private memoirs of a smart meter, in: ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building, 2010.
[29] E. Buchmann, S. Kessler, P. Jochem, K. Böhm, The costs of privacy in local energy markets, in: IEEE Conference on Business Informatics, 2013.
[30] A. Mueen, E. Keogh, Q. Zhu, S. Cash, B. Westover, Exact discovery of time series motifs, in: SIAM International Conference on Data Mining, 2009.
[31] F. Eichinger, D. Pathmaperuma, H. Vogt, E. Müller, Data analysis challenges in the future energy domain, Computational Intelligent Data Analysis for Sustainable Development.
[32] Y. Wang, T. Pavlidis, Optimal correspondence of string subsequences, IEEE Transactions on Pattern Analysis and Machine Intelligence.
[33] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, J. Albrecht, Smart*: An open data set and tools for enabling research in sustainable homes, SustKDD Workshop on Data Mining Applications in Sustainability.
[34] Watts up? Meters, https://www.wattsupmeters.com (Accessed 2 June 2016).
URL https://www.wattsupmeters.com/secure/products.php?pn=0&wai=638&spec=8
[35] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, Locally adaptive dimensionality reduction for indexing large time series databases, ACM SIGMOD Record.
[36] C. Buragohain, N. Shrivastava, S. Suri, Space efficient streaming algorithms for the maximum error histogram, in: IEEE International Conference on Data Engineering, 2007.
[37] R. Killick, I. Eckley, changepoint: An r package for changepoint analysis, Journal of Statistical Software 58 (2014) 1–19.
[38] X. Song, M. Wu, C. Jermaine, S. Ranka, Statistical change detection for multi-dimensional data, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007.
[39] A. Bifet, R. Gavaldà, Learning from Time-Changing Data with Adaptive Windowing, Ch. 42.
[40] J. Takeuchi, K. Yamanishi, A unifying framework for detecting outliers and change points from time series, IEEE Transactions on Knowledge and Data Engineering.
[41] Y. Saatci, R. D. Turner, C. E. Rasmussen, Gaussian process change point models, in: International Conference on Machine Learning, 2010.
[42] T. G. Papaioannou, M. Riahi, K. Aberer, Towards online multi-model approximation of time series, in: IEEE International Conference on Mobile Data Management, 2011.
[43] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, K. Cameron, Powerpack: Energy profiling and analysis of high-performance systems and applications, IEEE Transactions on Parallel and Distributed Systems.
[44] W. L. Bircher, L. K. John, Complete system power estimation using processor performance events, IEEE Transactions on Computers.
[45] S. Rivoire, P. Ranganathan, C. Kozyrakis, A comparison of high-level full-system power models., HotPower.
[46] B. C. M. Fung, K. Wang, R. Chen, P. S. Yu, Privacy-preserving data publishing: A survey of recent developments, ACM Computing Surveys.
[47] C. Dwork, Differential privacy, in: Automata, languages and programming, Springer, 2006.
[48] G. Acs, C. Castelluccia, I have a dream!(differentially private smart metering), in: Information Hiding, 2011.
[49] S. Rajagopalan, L. Sankar, S. Mohajer, H. Poor, Smart meter privacy: A utility-privacy framework, in: IEEE International Conference on Smart Grid Communications, 2011.
[50] K. LeFevre, D. J. DeWitt, R. Ramakrishnan, Workload-aware anonymization, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006.
[51] B. Chiu, E. Keogh, S. Lonardi, Probabilistic discovery of time series motifs, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003.
[52] Y. Li, J. Lin, T. Oates, Visualizing variable-length time series motifs., in: SIAM International Conference on Data Mining, 2012.
[53] A. Mueen, Enumeration of time series motifs of all lengths, in: IEEE International Conference on Data Mining, 2013.
[54] A. McGovern, D. Rosendahl, R. Brown, K. Droegemeier, Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction, Data Mining and Knowledge Discovery.