Adaptive Recovery of Incomplete Datasets for Edge Analytics

Ivan Lujic, Vincenzo De Maio, Ivona Brandic
Institute of Information Systems Engineering, Vienna University of Technology
Favoritenstrasse 9-11/194, A-1040 Vienna, Austria
{ivan, vincenzo, ivona}@ec.tuwien.ac.at

***Abstract*—The Internet of Things (IoT) has attracted significant attention from both academia and industry, thanks to applications such as smart cities, smart buildings and intelligent traffic management. These systems rely on data, collected from IoT devices, that are sent to the cloud for analytics. Data are either used for near real-time decisions or stored for long-term analysis. However, in highly distributed IoT systems, missing or invalid data may appear because of different reasons including sensor failures, monitoring system failures and network failures. Analyzing incomplete datasets can lead to inaccurate results and imprecise decisions, with negative effects on the target systems. Also, due to the increasing size of such systems and the consequently increasing amount of data generated from sensors, recovery of incomplete datasets for analytics on the cloud is often infeasible, due to the limited bandwidth available and the strict latency constraints of IoT applications.**

We propose a novel semi-automatic recursive mechanism for recovery of incomplete datasets on the edge that is closer to the source of data. This mechanism enables efficient recovery of incomplete datasets employing different forecasting techniques for multiple gaps, based on user specifications. We evaluate our approach on datasets coming from the context of smart buildings and smart homes. The experimental results show that our approach is able to identify multiple gaps, then recover incomplete datasets, decreasing forecasting error by up to 82.68%, and reducing running time by up to 52.38%.

***Index Terms*—Edge Computing; Internet of Things; time series; incomplete data; data recovery; forecasting techniques.**

I. Introduction

In recent years, we have seen the proliferation of the Internet of Things (IoT) systems [1], such as smart agriculture [2], smart buildings [3], smart cities [4] or intelligent traffic management systems [5]. Such applications rely on monitoring of parameters such as temperature, movement, heart rate, electricity consumption, radiation and air quality, coming from plenty of sensors. Collected data have to be analyzed in order to allow applications to perform timely actions. Due to the limited storage and computing capabilities of sensors, IoT systems often rely on cloud computing services to perform needed analysis [6], [7].

However, performing data analytics in the cloud requires the transfer of big amount of data from sensors to geographically distributed data centers, often very far from the data source. This raises a number of issues, since critical applications like eHealth, smart grids or intelligent traffic management have to process a big amount of data with strict accuracy and latency requirements [8], [9]. Also smart building systems have such requirements for automatic management of heating and cooling systems, either to foster energy efficiency or to support energy demand management systems integrated with smart grids [10]. Moreover, performing data analytics in the cloud may be infeasible, since current bandwidth capabilities and network infrastructures cannot easily scale with the growing amount of data generated by sensors [11].

Edge computing has been proposed as a solution to these issues. Use of edge nodes allows exploitation of storage, compute and network resources across cloud boundaries [12] bringing processing closer to data sources, and thus helping to perform near real-time decisions for low-latency IoT requirements [13]. Nonetheless, errors, missing values and outliers may appear in data collected by sensors, due to (1) the highly distributed nature of IoT systems; (2) monitoring system failures; (3) data packet loss in sensor networks; (4) aging of the sensor; (5) changes in external conditions; or (6) periodic failures of some of the sensors [14].

Performing analytics on incomplete/invalid datasets can cause problems in different contexts, leading to inaccurate results and imprecise decisions [15], [8]. For example, many smart buildings manage renewable energy sources employing automation systems that control and improve energy efficiency [3]. Taking decisions based on incomplete/invalid datasets may affect such systems introducing imbalances in overall building management system and connected smart grid systems [16]. Also, in intelligent traffic management systems, traffic situation is constantly monitored by different types of IoT environmental traffic sensors [5] to optimize/control traffic flow and avoid collisions and congestions. Inaccurate analysis due to incomplete data may affect traffic monitoring, with negative effects on collision and congestion avoidance systems. Therefore, it is important to efficiently remove outliers and recover missing values in the collected datasets before processing them.

While dealing with the accuracy of near real-time decisions, we proposed in [17] an architecture model and corresponding algorithms for efficient edge storage management. However, we did not consider how to deal with datasets with missing and/or invalid values, which impact the process of predicting critical events in the future. Several papers discussed the reconstruction of incomplete datasets [18], [19]. However, these works do not consider the time criticality demands

in the context of IoT applications and improvement of data quality by using different forecasting techniques. Therefore, we bridge this gap by introducing user-defined and condition-based recovery in the choice of different forecasting techniques for adaptive recovery of incomplete data on the edge.

The paper's main contributions are summarized as follows:

- We propose a novel semi-automatic recursive mechanism for efficient recovery of incomplete time series datasets, incorporating a recovery cycle that ensures outliers removal, detection and forecasting of each gap separately;
- We introduce a generic approach for semi-automatic selection of forecasting techniques based on user specifications and algorithm repository. It allows selecting suitable forecasting techniques for gaps recovery;
- Finally, we evaluate the proposed approach by utilizing real-world time series data coming from the context of smart buildings and smart homes.

We perform evaluation of our work by recovering missing values caused by several monitoring system failures from Austria's largest Plus-Energy Office High-Rise Building. With the proposed approach we are able to recover all gaps in collected data, and improve accuracy of the required short and long term data analysis. We implement our approach using the R environment for statistical computing. We show that our mechanism is able to decrease forecasting error by up to 82.68%, while reducing overall running time by up to 52.38%.

The rest of the paper is structured as follows. In Section II we describe our motivational scenario and introduce time series data. In Section III, we propose our data recovering mechanism, describing each component and related algorithms. We also provide an analysis of the algorithms' complexity. In Section IV, we present the experimental evaluation and discuss our results. Section V presents related work, while Section VI concludes the paper and provides an outlook for further research.

II. Background

A. Motivational scenario

The use of IoT architectures is constantly increasing, and consequently also the amount of data collected by IoT smart devices [11]. Collected datasets can bring valuable information by performing data analytics. However, in order to extract meaningful information from the data, we need to ensure that datasets are complete and cleaned from outliers.

The use case we select is *smart home/building applications*, where energy efficient smart homes and buildings are equipped with automated energy management systems integrating renewable energy generation (solar and wind turbines), smart meters and smart sockets [10]. We consider impact of incomplete datasets in this scenario from two perspectives: (1) batch (long term) and (2) near real-time (short term) analytics. Incomplete datasets may affect batch analytics on historical data, affecting management system (for example, heating and cooling management), and thus decreasing energy efficiency with increasing operational costs. Moreover, it can affect near real-time power management system in the case of power fluctuation caused by intermittent renewable sources of electricity. Analytics performed on incomplete datasets may affect also load balancing in smart grids, and therefore reliability of energy supplies [16].

Figure 1 depicts the proposed edge analytics model for smart buildings system. In step (1), data are collected from smart buildings. Currently, these data are mostly processed in cloud data centers [6]. However, due to the increasing amount of data generated by smart buildings and homes, in step (2) analytics are performed on edge layer to save bandwidth. Edge layer is composed of edge nodes, such as edge gateways and edge micro data centers, aiming to perform data processing closer to the data sources. Once certain amount of data is transmitted to the edge layer, user specifications are checked in step (3). The specifications consist of application dependent information and potential preferences for data recovery process. Then, adaptive recovery process is performed in step

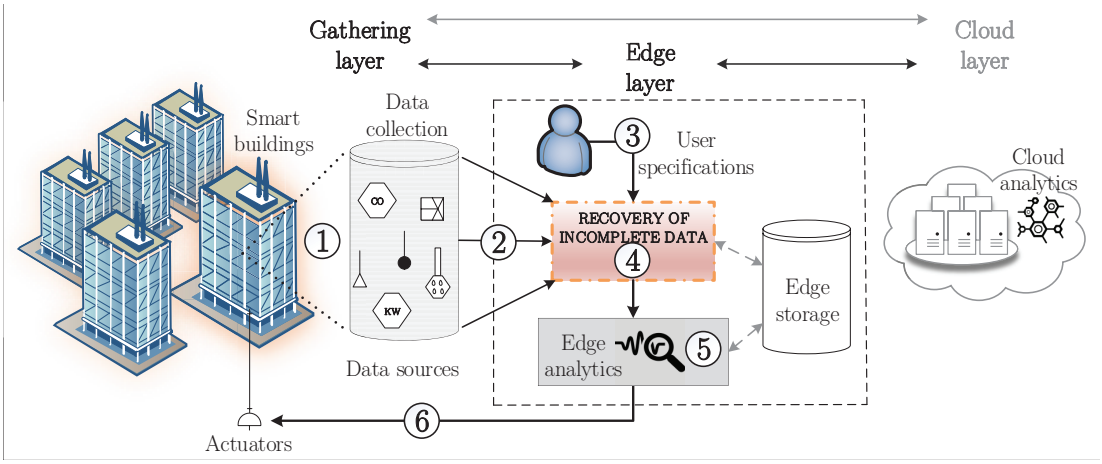


Fig. 1: Edge analytics model with adaptive recovery of incomplete datasets.

(4), whose output is the dataset without gaps and cleaned from outliers. This output is then forwarded either to edge storage or to edge analytics operations in step (5). The analytic results are either stored or used to take operative decisions (for example, commands sent to actuators) in step (6).

Data analytics is performed on the edge nodes to deal with the increasing scale of smart buildings systems. However, edge layer has limited resources in comparison with cloud layer, where resource demanding batch analytics can be performed. Solutions such as resource management mechanisms [20] and optimized service placement [21] have been proposed to deal with the resource constraints of the edge. However, such approaches do not propose efficient and adaptive solutions for data quality improvement. In this work we focus on edge layer, leaving the interplay between cloud and edge for future work.

B. Time series

There are different types of sensor-based data [22], such as text, video, audio or social media. However, we target sensor-based time series data that are very common in IoT data sources for applications like smart buildings and homes [3]. Time series can be collected from systems having either equal or unequal monitoring time intervals between data points. The first case is more common, for example, periodically reading weather conditions in weather stations: such data are called *evenly-spaced* or *regular time series*. In the second case, they are called *unevenly-spaced* or *irregular time series*. They can occur when data collection is triggered by certain events or whilst dynamically changing the static monitoring frequency to avoid redundant data during steady runs of the system [23]. Since in our scenarios data analytics rely particularly on regularly time-stamped measurements, we focus on regular time series that incorporate potential missing/invalid values. For the sake of clarity and consistency, we use term incomplete data in the rest of the paper.

III. Data recovery mechanism

We present an adaptive mechanism for recovery of time series. First, we define gaps in time series data.

*Definition 1: A **gap** is sequence of one or more missing or invalid consecutive values, irregularly distributed in time series.*

By missing values we mean data that are missing due to sensors/monitoring failures and by invalid data we mean outliers due to errors in the measurement. In Figure 2, we provide a flowchart of the proposed recursive mechanism for recovery of incomplete datasets. First, in data preparation component, data are prepared for the recovering process by marking indexes of each gap in the time series. Then, the recovery cycle starts by using the information of data preparation component to detect the amount of missing/invalid values. The cycle terminates when there are no more missing values. Otherwise, the gap identification component detects the size of the current gap, selecting it for the separate recovering process.

Further, the data processor component analyses data points preceding the current gap that are important for the right setup

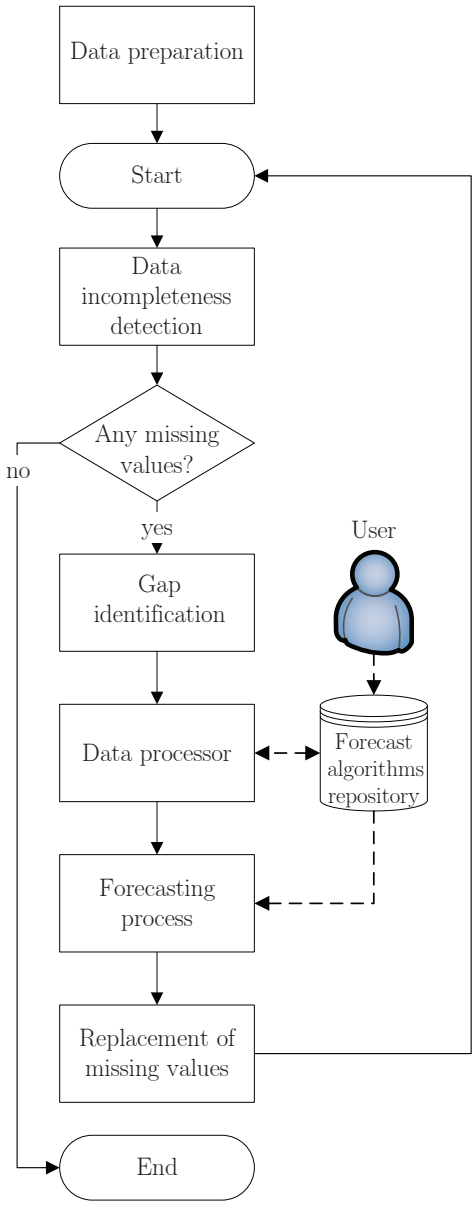


Fig. 2: Algorithm flowchart for the recovering process of incomplete datasets. Solid and dashed lines represent data flow and control flow, respectively.

of forecasting technique, according to user specifications. Then, data needed and the selected technique are forwarded from the repository to the forecasting process component, where the gap is replaced by predicted values. Afterwards, conditions for the next recovering cycle are checked. Table I describes symbols used in given algorithms. The following subsections describe all components in detail.

A. Data preparation

The goal of data preparation is to apply a set of operations that ensure detection and forecasting of each gap separately. To this end, the algorithm has to identify all missing values and

TABLE I: Summary of Notations.

Notation	Description
$input_{data}$	2D array that represents incomplete data coming from IoT sources.
$storage_{data}$	2D array that represents data in storage.
$index_{miss}$	Vector that contains all indexes of missing values.
no_{miss}	This variable counts number of missing values based on $index_{miss}$.
gap_{first}	Vector that stores all missing indexes for the current gap.

outliers in the collected dataset making an incomplete dataset ready for the adaptive data recovery process.

The data preparation process is described by Algorithm 1. First, *i* and *j* (lines 1-2) count data from input and storage, respectively, while line 3 creates an empty vector for indexes of missing values in a dataset. Outliers are identified according to the minimum and maximum values, for particular sensors, that can be either application-dependent or predefined by user. If a data value is out of bounds (line 5), it is replaced by a missing value indicator (line 6) represented by a symbol such as NA (Not Available), so that the correct value can be efficiently estimated in the recovering cycle based on previous data points. In lines 4-9, all outliers are replaced with NA.

Missing values can occur due to different reasons, like a system failure. Once the system is recovered, the next value collected by the sensors is stored immediately after the last generated time stamp, thus making it difficult to identify a gap in the time series. Therefore, we propose a solution where the monitoring system stores either corresponding data value or a missing value indicator, for each time stamp generated before the last measured value (lines 11-21). If time stamps from storage and input match (line 12), the corresponding value is moved to the storage beside the time stamp (line 13). A missing value indicator is stored in case the system does not receive data value for corresponding time stamp (line 17). In this case, the index of missing data is stored in the created vector $index_{miss}$ (line 18). Once the while loop is finished, the vector $index_{miss}$ contains all indexes of missing data.

Algorithm 1: `DataPreparation`

Input: 2D array $input_{data}[timestamp, value]$, 2D array $storage_{data}[timestamp,]$, variable $lower_{bound}$, variable $upper_{bound}$
Output: vector $index_{miss}$ that contains indexes of missing values in a dataset

```
1  Set counter i ← 1
2  Set counter j ← 1
3  Create vector index_miss
4  for each value i ∈ input_data do
5      if input_data[i,2] < lower_bound OR
         input_data[i,2] > upper_bound then
6          input_data[i,2] ← NA
7          Increment counter i
8      end
9  end
10 Set counter i ← 1
11 while i <= length(input_data) do
12     if storage_data[j,1] == input_data[i,1] then
13         storage_data[j,2] ← input_data[i,2]
14         Increment counter i
15         Increment counter j
16     else
17         storage_data[j,2] ← NA
18         Add index j in vector index_miss
19         Increment counter j
20     end
21 end
```

B. Data incompleteness detection

All indexes of corresponding missing values (indicated by NA symbols) are stored in the $index_{miss}$ vector. The number of elements in this vector represents the number of remaining missing values. Therefore, this number is checked at the beginning of each cycle, and the algorithm terminates in case there are no more missing values left (see Figure 2). Otherwise, it continues with identification of the next gap to be recovered. After forecasting process is finished, indexes of recovered values are removed from the vector $index_{miss}$.

C. Gap identification

The gap identification phase is responsible for detecting multiple gaps in a given dataset. It identifies the beginning and the end of each gap, using this information for the separate recovering process. Each gap is then processed separately, to enable selection of appropriate forecasting technique according to characteristics of previous data or user predefined specification. This component is described by Algorithm 2. The counter i, that is used to iterate over the vector of indexes of missing values $index_{miss}$, is set to 1, while data index of the first missing value is copied to the first place of the vector gap_{first} and stored in the temporal variable $temp$ (lines 2-3). Until total number of missing values is below a value in the counter i (line 4), the counter i is updated with the next index of missing value, while the index, stored in the variable $temp$, is incremented (lines 5-6) by 1. This allows to check whether indexes of missing values are consecutive (line 7). In that case, the corresponding index is copied to the vector gap_{first} (line 8). Otherwise, if there are no more consecutive indexes (line 9), all missing values from the current gap are detected and the vector $index_{miss}$ is updated in line 10.

D. Data processor

This component performs the extrapolation of data characteristics or parameters needed for application of particular forecasting technique. Additionally, it can perform also data classification or aggregation. The obtained data are forwarded to the forecast algorithm repository that contains multiple data

Algorithm 2: `GapIdentification`

Input: Vector of indexes of missing values $index_{miss}$, variable no_{miss} that contains number of missing values

1 **Set** counter $i \leftarrow 1$
2 **Create** vector $gap_{first}[i] \leftarrow index_{miss}[i]$
3 **Create** temporal variable $temp \leftarrow index_{miss}[i]$
4 **while** $no_{miss} > i$ **do**
5 **Increment** counter i
6 **Increment** variable $temp$
7 **if** $temp == index_{miss}[i]$ **then**
8 $gap_{first}[i] \leftarrow index_{miss}[i]$
9 **else**
10 **Remove** indexes of gap_{first} from $index_{miss}$
11 break;
12 **end**
13 **end**

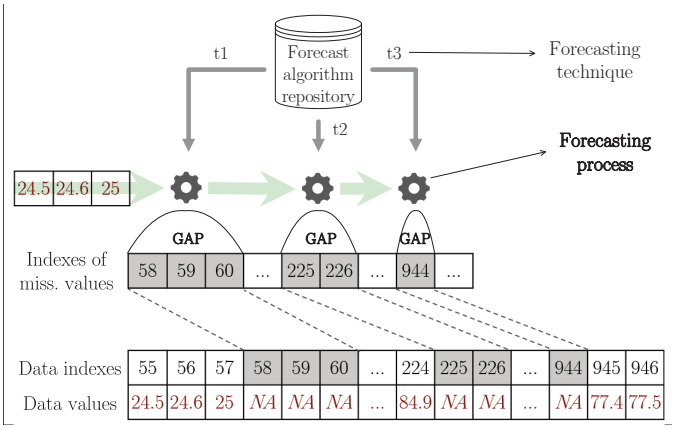


Fig. 3: Adaptive data recovery process for multiple gaps.

recovery techniques. Necessary characteristics are obtained during analysis of available data up to the first missing index k of a potential gap. After the current gap is identified (in the last component), in order to efficiently forecast missing values, predecessor data are analyzed to derive parameters that are necessary for the forecasting process. Parameter selection depends on the forecasting technique. Semi-automatic mechanism allows two possible scenarios: (1) *single-technique recovery* and (2) *condition-based recovery*. In the first scenario, users can either select an existing technique, or adding a new one to the algorithm repository. The selected technique is used for recovering all gaps. In the latter scenario, technique selection is performed at runtime, either according to user-specified conditions or automatically. In this work, we focus on selection based on user-specified and predefined conditions. Full-automatic selection is left for further research.

The data processor is present in each cycle, because values of forecasting parameters may be different when analyzing predecessor data of next possible gaps. The information about the forecasting technique is forwarded to the next component by the algorithm repository.

E. Forecasting process

In this component, a forecasting technique is selected from the repository and applied on current gap. Figure 3 illustrates the adaptive recovery process including aforementioned processes. After corresponding missing indexes are stored by data incompleteness detection component and the first gap identified by gap identification component, data processor component analyzes predecessor data before the gap. Selected forecasting technique is then applied for the recovering process. The figure shows our approach, where forecasting process component applies different techniques (*t1*, *t2* and *t3*) for different gaps. The choice of suitable techniques depends on data characteristics and forecast objectives as described in [24]. In this work we select three techniques, according to different dataset characteristics: (1) the Autoregressive Integrated Moving Average (ARIMA) method can be used if data contain stationary characteristics, such as trend stationarity, that can be explored by methods proposed in [25]; (2) the Exponential Smoothing method (ETS), although overlapping in same cases with ARIMA models, can be used for short-term seasonal series; (3) the TBATS [26] forecasting model can be used for time series with multiple complex seasonality or with long seasonal periods. If seasonality occurs in time series, by checking periodicity, the data processor component can forward that information to the next component. Users can also specify additional information about the data, such as a monitoring frequency for seasonality: for example, if temperature data are collected every five minutes, then the seasonal parameter value 288 (12*24), representing the expected daily seasonality, is included in the forecasting procedure. Further, users can specify that for each gap containing less than 10 missing values, n-point average method is selected.

When all necessary parameters are forwarded from the data processor, the forecasting process can start. Missing values are replaced in the original dataset, and their indexes are removed from the vector $index_{miss}$. When the current gap is recovered, the next gap (if exists) will be considered in a new recovering cycle. The recovering process stops when no more missing values in no_{miss} are left.

F. Algorithm complexity

The computational complexity of the proposed mechanism depends on the complexity of each algorithm. By analyzing Algorithm 1, looking at the *for* loop (lines 4-9) and the *while* loop (lines 11-21), the algorithm iterates over available dataset making it *O(n)*, where *n* represents number of data points in the dataset. Further, entering in the *while* loop of Algorithm 2 in line 11, iterates the vector of indexes of missing values that are always less than the number of data in the available dataset. For all other lines, complexity is *O(1)*. In case forecasting process uses the ARIMA technique, the time complexity to forecast certain amount of data is *O(n)*, where *n* is the number of data points in the training data used for forecasting, and

thereby it would result in the overall complexity *O(n)*. Running time is affected by different factors, such as the size of the gap that has to be recovered and seasonal complexity of time series. Time complexity also highly depends on the complexity of selected forecasting technique. However, since the proposed mechanism targets resource-limited edge nodes and analysis for near-real time decisions, the size of input and dimensionality of incompleteness are not expected to be big enough to violate the time requirements.

IV. Experimental evaluation

We implement the proposed mechanism by utilizing the R language using the forecast package [27]. We evaluate the applicability of the proposed approach by recovering multiple gaps in different datasets and measuring accuracy and runtime for three cases: (1) ARIMA, (2) ETS and (3) AdaptOpt (Adaptive Option), respectively. In first two cases, the same technique is used for all gaps, that is, single-technique recovery, while in the last case, condition-based recovery, based on combination of different forecasting techniques on different gaps, is employed. All the simulations and the running time are obtained on a 64-bit Windows 10 machine, configured with a 2.70-2.90 GHz Intel i7-7500U CPU and 16 GB memory.

A. Data

We evaluate our approach on data from real-world IoT systems that are the main target of this work:

1) Data collection from **smart buildings**. Datasets from this source are obtained by the monitoring system of Austria's largest Plus-Energy Office High-Rise Building [28]. Sensor-based data collection contains various measurements such as power usage, electricity consumption and production, temperature and air quality, that are used for operative decisions such as automatic heating and cooling, ventilation, efficient energy consumption;
2) Data collection from **smart homes**. Datasets from this source are obtained by UMass Trace Repository [29] containing traces from the Smart* project [30] for the purpose of optimizing energy consumption in designing smart homes. It represents variety of environmental and electrical type of data, such as temperature, humidity, wind information and heat index.

After applying proposed mechanism on collected datasets, we select two representative from each source, targeting characteristics such as sensor type and range of values, as it is shown in Table II. Each dataset contains around 14600 data points for experiments.

In both datasets by source 1, we observe several gaps in collected data that affected data analytics. Therefore, to evaluate proposed approach, after the analysis of datasets we identify those gaps and artificially made several gaps with approximately same sizes, precisely, gaps with 238, 3 and 5016 consecutive missing/invalid data values. Then, these gaps are recovered using the proposed mechanism, and forecast accuracy is evaluated by comparing predicted data with actual data from each gap. This choice has been made to ensure a fair comparison between all techniques, comparing them according to the forecast error they achieve on the same gap sizes and on the same data.

For the forecast accuracy evaluation, we use Mean Absolute Percentage Error (MAPE) measure that expresses accuracy of prediction as a percentage based on forecast error. As a scale-independent measure, it allows us to compare effectiveness of recovering process among different types of datasets.

B. Experimental results

We apply the proposed mechanism and show two representative examples among used datasets from each of data sources as it is illustrated in Figure 4. Each example is represented in a separated sub-figure. Each sub-figure consists of two graphs: the upper graph indicates incomplete dataset before recovering process, while the lower graph indicates complete dataset after applying recovering procedures. Gray shaded areas indicate three gaps, precisely, 238, 3 and 5016 missing values in a sequence. The black solid line represents existing values, that is, the actual state of a collected dataset on the edge. The black dashed line shows actual data for corresponding gaps, while the red solid line represents predicted values of missing data points as an output of applied forecasting techniques. In this case, all gaps are recovered using ARIMA technique from R forecast package, representing single-technique recovery scenario. It can be seen that time series in the first sub-figure contains deterministic linear trend pattern that makes ARIMA forecasting appropriate and more accurate in comparison to the behavior of time series shown in the second sub-figure. This dataset shows an irregular pattern making forecasting process more difficult. For this type of time series, where no trend or seasonality can be captured, the algorithm calculates best suitable straightforward prediction line with 95% confidence interval. For this reason, the MAPE of reconstructed gap 3 by *bsmart2* is around 1.68315%, while the MAPE for the same gap by *bsmart1* is around 0.11999%. It can be also seen that, as the gap increases, the forecasting error increases too. Finally, the graph confirms that the proposed automated recursive mechanism is able to efficiently cope with multiple gaps by forecasting all gaps with running time of $2.27s$ and $4.18s$ (see Figure 6), for *bsmart1* and *bsmart2*, respectively. These scenarios simulate users that specify a particular forecasting technique for the whole recovering process of incomplete dataset (see Subsection III-D). On the other hand, based on the proposed mechanism, it is possible to specify different forecasting techniques involved in recovering process of each gap separately, that is shown in the following section.

TABLE II: Dataset Information.

Source	Dataset	Sensor type	Range of values
1	bsmart1	el. meter [kWh]	21.71-23.37
	bsmart2	room temp. [C]	21.06-23.78
2	hsmart1	room temp. [F]	65.89-83.30
	hsmart2	heat index [F]	27.86-107.64

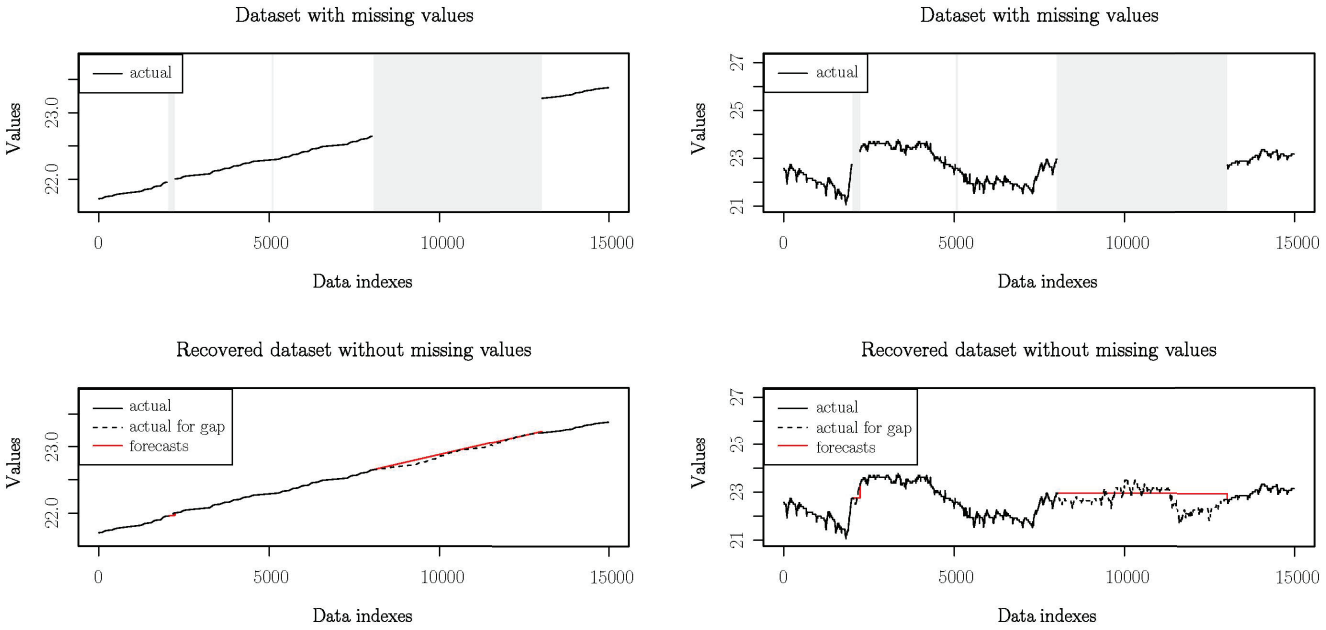


Fig. 4: Results of semi-automatic recursive recovery of multiple gaps. Sub-figures (a) and (b) are representative examples of recovered gaps for datasets *bsmart1* and *bsmart2*, respectively. It shows results of *single-technique recovery*. In this case, ARIMA is used for all gaps, instead of selecting the most appropriate recovery method for each of three gaps, as in *condition-based recovery*. Therefore, flat forecasts for the third gap (Figure 4b) depict the same possible mean value for missing data, since the selected method is not able to obtain significant characteristics such as trend or seasonal components of prior data.

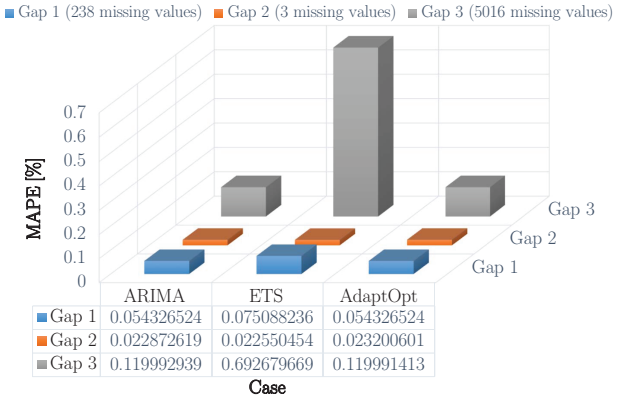
C. Discussion

We compare forecast accuracy measures (Figure 5) and perform code running time analysis (Figure 6), showing the efficiency of the proposed mechanism with selection of different forecasting techniques from the repository. Beside possibility of using different techniques for each time series, it is also possible to select different techniques for each gap. This allows to avoid computation expensive techniques for small gaps or individual missing values and thus improving performance.

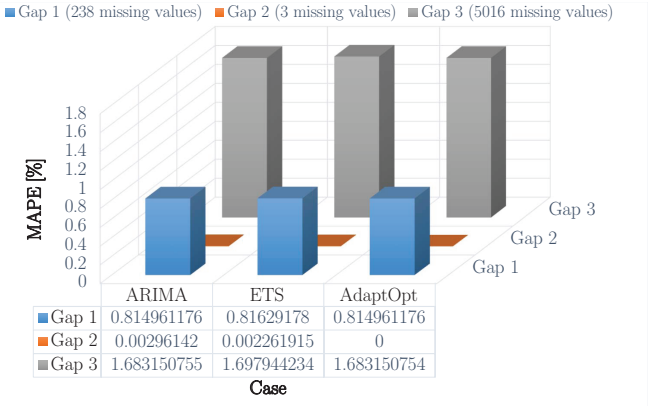
Figure 5 depicts the proposed mechanism applied on each of four different datasets containing same sizes of multiple gaps. Each of four sub-graphs represents one dataset. The recovering process is tested by utilizing different forecasting techniques in three cases shown on horizontal axis, (1) ARIMA, (2) ETS and (3) AdaptOpt, respectively. In first two cases, the same technique is used for recovering all gaps. The latter (Adaptive Option) depicts the second scenario, where we utilize a combination of different forecasting techniques, showing the benefits of recovering procedure. Vertical axis shows MAPE for corresponding gaps. Comparing the MAPE between same gaps within one particular dataset, the error increases as size of gap increases in *bsmart1* and *bsmart2*, while for *hsmart1* and *hsmart2* is the opposite. The reason lies in wider range of values and more volatile behaviour for data from the source two, where, e.g., for *bsmart2*, the error increases by 107.02% on average, while for *hsmart2* it decreases by 46.47% on average, between gaps 1 and 3.

For the case 3, in each dataset, three gaps are recovered, respectively, by applying techniques ARIMA, ETS, ARIMA for *bsmart1*, ARIMA, n-point average, ARIMA, for *bsmart2*, TBATS, n-point average, ETS, for both, *hsmart1* and *hsmart2*. With this setup, we are able to reduce overall running time by 25.11% and 52.38% for *bsmart1* comparing case 3 to cases 1 and 2, respectively, and in the same time having 82.68% less forecasting error for gap 3 comparing with case 2. For *bsmart2*, we have a slightly better forecast accuracy, but reduced overall running time by up to 48.09% in comparison to cases 1 and 2. Additionally, by performing n-point average for the gap 2, we were able to completely recover that gap, having 0 for the MAPE value, because of the presence of redundant predecessor values. Although the running time for the case 3 is not the lowest for *hsmart1* and *hsmart2*, the error decreases up to 41.2% for the first gap of dataset *hsmart2*. By performing the proposed *AdaptOpt* in case 3, depending on type of datasets, we are able to reduce either only forecast error or both, the error and the running time.

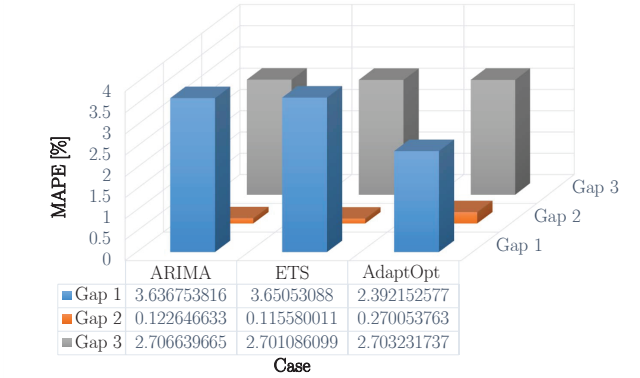
Figure 7 shows absolute percentage error behavior for two datasets (*bsmart1* and *hsmart2*) along two bigger gaps (gaps 1 and 3) of the incomplete data. It illustrates that application of proposed *AdaptOpt* approach, in overall, results



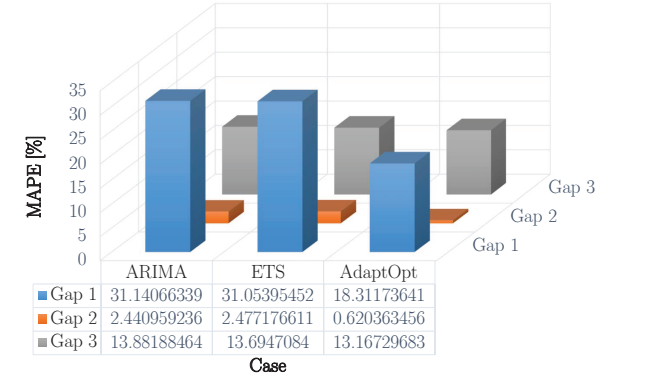
(a) Data recovery in dataset *bsmart1*.



(b) Data recovery in dataset *bsmart2*.



(c) Data recovery in dataset *hsmart1*.



(d) Data recovery in dataset *hsmart2*.

Fig. 5: MAPE accuracy measure for three recovered gaps of missing values among 4 datasets. All three gaps (that is, 238, 3 and 5016, respectively), are recovered by utilizing different forecasting models in three cases: ARIMA, ETS and AdaptOpt.

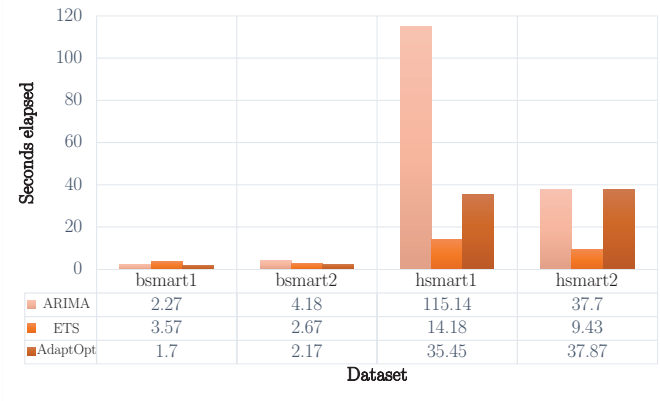
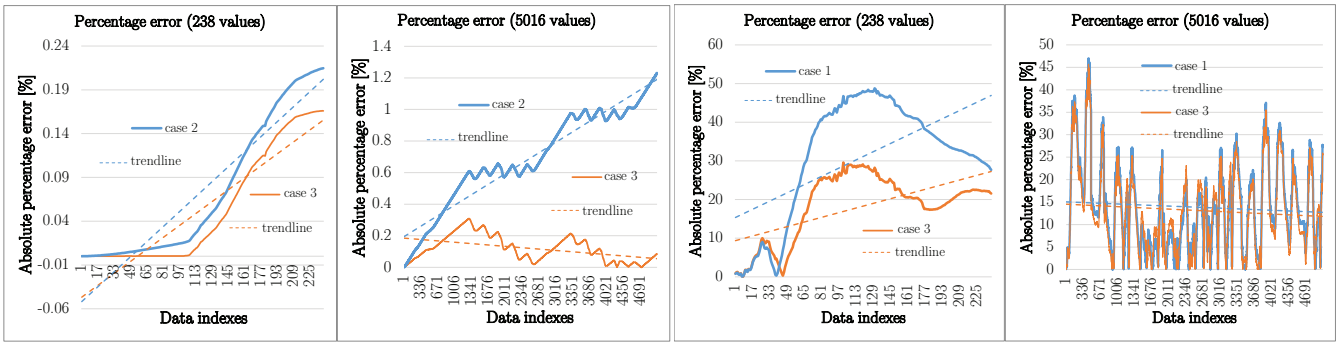


Fig. 6: Running time of recovering all gaps.

in smaller error which is shown also by calculated trendlines. Although the gap to recover can be significantly bigger (5016 values), we see that the MAPE increases only by up to 0.31% for *bsmart1* and to the peak value of 46.57% for *hsmart2*. Because of the volatile behavior of data from *hsmart2*, the error can come to the higher peak values. However, with the appropriate methods, it can be greatly improved, for example, using ARIMA instead of ETS and TBATS instead of ARIMA for *bsmart1* and *hsmart2*, respectively. Considering unlikely cases with big gaps at the edge, the experimental results confirm benefits of our approach, and especially of selection of different techniques for recovering different gaps within the same incomplete dataset.

V. Related Work

In the last few decades, time series forecasting gained much attention due to necessary predictive analytics that rely on increasing amount of time-stamped measurements. Authors in [24] provide an extensive overview of existing time series forecast methods, including a self-adaptive approach for optimized forecasting method selection based on users' forecasting objectives. The use of different forecasting methods is motivated by the fact that forecast accuracy depends on characteristics of data before each gap, therefore we chose ARIMA, ETS and TBATS, described also in [31], [26]. Also, these methods do not require constant user interactions. Thus, we use them for adaptive recovery of multiple gaps, due to



(a) Recovered gaps from *bsmart1*.

(b) Recovered gaps from *hsmart2*.

Fig. 7: Absolute percentage error behavior along recovered gaps comparing with proposed *AdaptOpt* (case 3).

the near real-time requirements of systems, running on self-contained edge nodes, such as IoT applications.

Wellenzohn et al. [19] propose Top-k Case Matching (TKCM) for imputing missing values in time series. Based on principles of correlation between time series, it is possible to impute missing values in streams of data by comparing incomplete series with a set of reference time series, and additionally allowing phase shifts. In [18] a program for imputing missing data in multivariate time series is proposed for handling missing data from stationary processes. Since most of the IoT measured processes are non-stationary, the proposed approach cannot be easily applied and can be inefficient for univariate time series. Also, none of the proposed approaches allows recovering of multiple gaps separately giving the possibility to select different models for recovery. In [32], the authors design an adaptive model selection based on Hidden Markov models aiming to constantly validate mean percentage error in a prediction algorithm, resulting in a higher accuracy in time series of stock price prediction. The approach constantly validates mean percentage error to find best model. However, machine learning based models for prediction of missing data are time consuming and require training, making these approaches difficult for data recovery on the edge.

Further, authors in [33] focus on data management solutions, proposing a comprehensive description of components of IoT data management framework. The authors mostly target design elements for efficient data handling. Furthermore, in [34] univariate imputation is used for air pollution data, but this approach target gaps of fixed size. The challenge of recovering missing data has been investigated by many researchers, providing methods relying on cubic interpolation [35], Singular Spectrum Analysis [36] or Lomb-Scargle method [37]. However, these works either have not been validated on IoT sources or propose approaches targeting only specific cases of time series. In this work, we show how to combine different forecasting techniques for the recovery of different gaps affecting both, the accuracy and performance of recovering processes. Additionally, the proposed approach allows users to specify preferred conditions and define other algorithms for recovering of incomplete data.

VI. Conclusions and Future Work

Analysis of data including missing or invalid measurements, may affect quality of decisions for many IoT applications such as energy efficient smart buildings or smart homes. To increase accuracy of near real-time data analytics, particularly for time-sensitive IoT applications, it is necessary to efficiently recover incomplete datasets beforehand. We propose a mechanism for semi-automatic recovery of incomplete time series coming from IoT sources, relying on edge nodes that are closer to the source of data, instead of utilizing only cloud resources. Applying proposed concepts, we are able to remove outliers, detect and recursively recover all existing gaps in incomplete datasets. Also, we introduce a two approaches for the selection of forecasting techniques from the algorithm repository, namely, single-technique recovery and condition-based recovery. Experimental results show that using the proposed approach, it is possible to have up to 82.68% less forecasting error in comparison to recovering with the same method for all gaps. Concurrently, the overall running time of recovering process can be reduced by up to 52.38%.

Nonetheless, we are aware that recovering process depends on multiple elements including number of missing/invalid data values, gap sizes, time series data behavior and the performance of used techniques on that elements. In case of full-automatic mechanism, that is, based on the preliminary characteristics of prior data without user interaction, we plan to extend our work by exploring how to automatically adapt trigger conditions for using different forecasting techniques. We also consider involving machine learning predictive analytics, by partially employing cloud resources and thereby crossing over expensive and time-consuming learning processes. Moreover, it would be interesting to explore possibility of recovering missing data using information from correlated time series. Finally, we plan to extend our approach to scenarios with strict latency requirements, such as eHealth and intelligent traffic management systems. In the first case, we plan to use our approach for recovering of datasets used for real-time analytics in monitoring illnesses such as diabetes and heart diseases, to help such systems to timely react to the change of patients'

condition. In the latter case, our mechanism can be used to recover datasets coming from road sensors, helping the system to accurately and timely react before collisions happen.

ACKNOWLEDGMENT

The work described in this article has been funded through the Haley project (Holistic Energy Efficient Hybrid Clouds) as part of the TU Vienna Distinguished Young Scientist Award 2011 and Rucon project (Runtime Control in Multi Clouds), FWF Y 904 START-Programm 2015. The authors would like to thank all stuff members at TU Wien's Plus-Energy Office High-Rise Building, especially to Thomas Bednar and Alexander David on valuable discussions and the Federal Real Estate Company (BIG) for providing data sources.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] K. A. Patil and N. R. Kale, "A model for smart agriculture using iot," in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, Dec 2016, pp. 543–545.

[3] J. Pan, R. Jain, S. Paul, T. Vu, A. Saifullah, and M. Sha, "An internet of things framework for smart energy in buildings: designs, prototype, and experiments," *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 527–537, 2015.

[4] Y. Sun, H. Song, A. J. Jara, and R. Bie, "Internet of things and big data analytics for smart and connected communities," *IEEE Access*, vol. 4, pp. 766–773, 2016.

[5] P. Pyykönen, J. Laitinen, J. Viitanen, P. Eloranta, and T. Korhonen, "Iot for intelligent traffic system," in *Intelligent Computer Communication and Processing (ICCP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 175–179.

[6] B. B. P. Rao, P. Saluia, N. Sharma, A. Mittal, and S. V. Sharma, "Cloud computing for internet of things amp; sensing based applications," in *2012 Sixth International Conference on Sensing Technology (ICST)*, 2012, pp. 374–380.

[7] M. Villari, A. Al-Anbuky, A. Celesti, and K. Moessner, "Leveraging the internet of things: Integration of sensors and cloud computing systems," *International Journal of Distributed Sensor Networks*, vol. 12, no. 7, 2016.

[8] A. Artikis, C. Baber, P. Bizarro, C. C. de Wit, O. Etzion, F. Fournier, P. Goulart, A. Howes, J. Lygeros, G. Paliouras, A. Schuster, and I. Sharfman, "Scalable proactive event-driven decision making," *IEEE Technology and Society Magazine*, vol. 33, no. 3, pp. 35–41, Fall 2014.

[9] C. Doukas and I. Maglogiannis, "Bringing iot and cloud computing towards pervasive healthcare," in *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2012, pp. 922–926.

[10] C. Keles, A. Karabiber, M. Akcin, A. Kaygusuz, B. B. Alagoz, and O. Gul, "A smart building power management concept: Smart socket applications with dc distribution," *International Journal of Electrical Power & Energy Systems*, vol. 64, pp. 679–688, 2015.

[11] I. Lee and K. Lee, "The internet of things (iot): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015.

[12] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. ACM, 2012, pp. 13–16.

[13] L. F. Bittencourt, O. Rana, and I. Petri, "Cloud computing at the edges," in *International Conference on Cloud Computing and Services Science*. Springer, 2015, pp. 3–12.

[14] C. C. Aggarwal, N. Ashish, and A. Sheth, *The Internet of Things: A Survey from the Data-Centric Perspective*. Boston, MA: Springer US, 2013, pp. 383–428.

[15] L. Cai and Y. Zhu, "The challenges of data quality and data quality assessment in the big data era," *Data Science Journal*, vol. 14, 2015.

[16] A. Mikulec and V. Mikuličić, "Influence of renewable energy sources on distribution network availability," *International Journal of Electrical and Computer Engineering Systems*, vol. 2, no. 1, pp. 37–48, 2011.

[17] I. Lujic, V. D. Maio, and I. Brandic, "Efficient edge storage management based on near real-time forecasts," in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, 2017, pp. 21–30.

[18] S. Liu and P. C. Molenaar, "ivar: A program for imputing missing data in multivariate time series using vector autoregressive models," *Behavior research methods*, vol. 46, no. 4, pp. 1138–1148, 2014.

[19] K. Wellenzohn, M. H. Böhlen, A. Dignös, J. Gamper, and H. Mitterer, "Continuous imputation of missing values in streams of pattern-determining time series," in *Proceedings of the 20th International Conference on Extending Database Technology (EDBT)*, 2017, pp. 330–341.

[20] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.

[21] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, "Optimized iot service placement in the fog," *Service Oriented Computing and Applications*, vol. 11, no. 4, pp. 427–443, Dec 2017.

[22] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *International Journal of Information Management*, vol. 35, no. 2, pp. 137–144, 2015.

[23] T. Mastelic and I. Brandic, "Data velocity scaling via dynamic monitoring frequency on ultrascale infrastructures," in *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, Nov 2015, pp. 422–425.

[24] N. R. Herbst, N. Huber, S. Kounev, and E. Amrehn, "Self-adaptive workload classification and forecasting for proactive resource provisioning," in *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '13. ACM, 2013, pp. 187–198.

[25] L. Tang, L. Yu, F. Liu, and W. Xu, "An integrated data characteristic testing scheme for complex time series data exploration," *International Journal of Information Technology & Decision Making*, vol. 12, no. 03, pp. 491–521, 2013.

[26] A. M. De Livera, R. J. Hyndman, and R. D. Snyder, "Forecasting time series with complex seasonal patterns using exponential smoothing," *Journal of the American Statistical Association*, vol. 106, no. 496, pp. 1513–1527, 2011.

[27] R. J. Hyndman, *forecast: Forecasting functions for time series and linear models*, 2017, r package version 8.2. [Online]. Available: http://pkg.robjhyndman.com/forecast

[28] T. Bednar, A. David, H. Schöberl, and G. Kratochwil, "University plus-energy office high-rise building innovations for buildings in practice," 2016.

[29] "Umass trace repository," http://traces.cs.umass.edu/, 2017, [Online; accessed 17-November-2017].

[30] 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, August*, vol. 111, p. 112, 2012.

[31] R. Hyndman and Y. Khandakar, "Automatic time series forecasting: The forecast package for r," *Journal of Statistical Software, Articles*, vol. 27, no. 3, pp. 1–22, 2008.

[32] J. Duan, W. Wang, J. Zeng, D. Zhang, and B. Shi, "A prediction algorithm for time series based on adaptive model selection," *Expert Systems with Applications*, vol. 36, no. 2, pp. 1308–1314, 2009.

[33] M. Abu-Elkheir, M. Hayajneh, and N. A. Ali, "Data management for the internet of things: Design primitives and solution," *Sensors*, vol. 13, pp. 15 582–15 612, 2013.

[34] W. Junger and A. P. de Leon, "Imputation of missing data in time series for air pollutants," *Atmospheric Environment*, vol. 102, pp. 96–104, 2015.

[35] F. Cismondi, A. S. Fialho, S. M. Vieira, J. M. C. Sousa, S. R. Reti, M. D. Howell, and S. N. Finkelstein, "Computational intelligence methods for processing misaligned, unevenly sampled time series containing missing data," in *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, April 2011, pp. 224–231.

[36] J. P. Musial, M. M. Verstraete, and N. Gobron, "Comparing the effectiveness of recent algorithms to fill and smooth incomplete and noisy time series," *Atmospheric chemistry and physics*, vol. 11, no. 15, pp. 7905–7923, 2011.

[37] K. Hocke and N. Kämpfer, "Gap filling and noise reduction of unevenly sampled data by means of the lomb-scargle periodogram," *Atmospheric Chemistry and Physics*, vol. 9, no. 12, pp. 4197–4206, 2009.