Resilient Edge Data Management Framework

Ivan Lujic, *Student Member, IEEE,* Vincenzo De Maio, *Member, IEEE,* and Ivona Brandic, *Member, IEEE*

Abstract—Transferring and processing huge amounts of data in the cloud can violate the low latency requirements of modern IoT applications, considering underlying network infrastructure limitations. Edge data analytics is a promising solution. However, edge resources have usually less computational capabilities than cloud nodes, resulting in a higher failure rate of IoT systems. Consequently, near-real-time decisions are often based on limited and incomplete data. State-of-the-art solutions, such as operational/workload flows, data reduction, reconstruction, focus mostly on resource and network optimization, while approaches for incomplete data recovery employ a single specific method, despite diverse data characteristics. Data quality impact on accuracy of the decision-making processes is often neglected. We propose EDMFrame, a framework featuring a generic mechanism for recovery of multiple gaps in incomplete datasets, using single-technique recovery (STR) and multiple-technique recovery (MTR) involving projection recovery maps (PRMs). We further devise an adaptive storage management mechanism for reducing data stored at the edge, keeping only the data necessary for predictive analytics. We conduct experiments using time series from smart buildings, (i) automatically recovering various multiple gaps and reducing errors up to 65.48% with MTR compared to STR; (ii) reducing amounts of data stored to 39.9% on average, keeping prediction accuracy around 98.83%.

Index Terms—Edge computing, Internet of Things, data, storage, forecasting, solution reference architectures, data flow architecture.

1 Introduction

The Internet of Things (IoT) has attracted attention from both academia and industry. Billions of devices will be connected to the Internet by 2020 [1], generating huge amounts of data. Today, IoT sensors are used in many applications, like eHealth [2], smart manufacturing [3], smart home and building systems [4], and smart cities [5]. These systems require sensors data collection, data analysis and acting based on the results of analysis. Usually, IoT data are processed in geographically distributed and distant cloud data centers [6], [7]. However, cloud data processing performance is affected by the increased size of data and due to the limited scalability of current network infrastructures [8]. Also, respecting critical predictive analytics in modern IoT systems, meeting the strict latency and accuracy requirements [9] of decision-making processes imposes new issues.

Edge analytics is a promising solution to latency and network size challenges by employing edge nodes, i.e., smaller scale cloud data centers deployed closer to IoT sensors [10]. Performing data processing in edge nodes allows near real-time decisions for IoT systems [11]. However, edge analytics has many open challenges. **First**, missing or invalid data may appear, due to monitoring system failures; data packet loss; sensor aging; or changes in external conditions [12]. Performing analytics on incomplete data can lead to inaccurate decisions [4], [13]. **Second**, compared to cloud data centers, edge nodes have limited storage and scalability affecting the accuracy of predictive analytics and, consequently, decisions for critical applications such as smart buildings [4] or manufacturing systems [3].

1.1 Edge Data Management Solutions and Limitations

There are different data management strategies for IoT and edge systems, considering IoT requests offloading [14], IoT resource management [2] and IoT security mechanism [15]. However, mentioned solutions focusing on QoS for distributed edge data processing, workload management and ensuring security of IoT sensitive data rather than focusing on data reconstruction and storage management. Although some works propose various reconstruction methods of incomplete datasets [16], [17], they do not distinguish recovery of various gaps, despite diverse data characteristics. We argue that for timely and accurate data recovery in modern IoT systems, it is necessary to combine different recovery techniques, even within the same datasets. Predictive analytics has a huge potential to revolutionise critical and proactive IoT applications, such as accurate diagnosis of patients in eHealth, maintenance services and failures prevention in smart manufacturing and building systems. For decision-making processes, Sensor-Cloud Infrastructure [18] is a promising solution. Other works like [19], [20] discuss IoT sensor data reduction and dynamic compression techniques, focusing on network optimizations. Still, there is a lack of solutions for accurate predictive analytics while dealing with incomplete data and limited edge storage capabilities. Traditional approaches for IoT and cloud data management address the challenges related to incomplete datasets and storage limitations [6], [7], [21], without considering the impact of data quality and data resilience on decision-making processes, which is of paramount importance to ensure efficient and accurate analytics in IoT systems [22].

1.2 Main Contributions

We bridge these gaps by introducing Edge Data Management Framework (EDMFrame[1]), a three-layer architecture model for resilient edge data management. The main contributions of this work are:

- **a novel, generic mechanism for adaptive recovery** of incomplete time series, incorporating a recovery cycle

I. Lujic, V. De Maio and I. Brandic are with the Institute of Information Systems Engineering, Vienna University of Technology, A-1040 Vienna, Austria. (e-mail: {ivan, vincenzo, ivona}@ec.tuwien.ac.at)

[1] https://github.com/lujic/EDMFrame

that ensures outliers removal, detection, and forecasting of each gap, using single-technique recovery (STR);
- **an edge storage management mechanism** that achieves a trade-off between the amount of data stored at the edge and high accuracy for predictive analytics;
- **a mediator component** featuring Projection Recovery Maps (PRMs) that detect the necessary range of historical data to recover different gaps for each dataset, as well as the recommended recovery technique, enabling multiple-technique recovery (MTR).

Preliminary results for the first and second contributions are shown in [23], [24], respectively. In this paper, we describe the complete framework, introduce the novel mediator component and show an entire data path through EDMFrame. We evaluate EDMFrame experimentally using six sensor-based time series from smart buildings and homes. Results show that EDMFrame can: (i) automatically recover gaps of various lengths with STR and achieving up to 65.48% less error with PRM-based MTR compared to STR; and (ii) reduce the amount of data stored to 39.9% on average per cycle, while obtaining prediction accuracy around 98.83%, thus storing only data relevant for predictive analytics at the network edge. In contrast to existing frameworks, such as [2], [4], [20], we enable (i) a generic data recovery, adaptable to different IoT data sources, and (ii) reliable decision-making for crucial predictive analytics in the context of storage-limited edge nodes.

We describe EDMFrame model in Section 2. Section 3 presents data recovery mechanism, while Section 4 shows the edge storage management algorithms. Section 5 describes the novel mediator component. Experimental evaluation and discussion are shown in Section 6. Related work is outlined in Section 7, and Section 8 concludes the paper.

2 Architecture Model Overview

Figure 1 shows an overview of EDMFrame architecture model. At the time we write, several frameworks for IoT data processing have been proposed, such as Eclipse Kura (https://www.eclipse.org/kura/), Node-RED (https://nodered.org/) and Flogo (https://www.flogo.io/). most of these frameworks focus on integrating heterogeneous IoT devices and on the interplay between the edge and cloud layers. They either do not provide methods for data recovery and predictive analytics, or they focus only on a specific technique for performing these tasks. Complementary to these works, we design EDMFrame as a service built on top of these or similar IoT data processing services, to enhance the data recovery and analytics features they offer. The aim is to devise a process to deliver accurate near real-time decisions while coping with (i) incomplete data, (ii) big volume of data and (iii) limited storage resources at the network edge. The architecture includes three software layers, namely: gathering layer, edge layer, and cloud layer. Even though the main focus of this paper is the edge layer, we describe all of them for completeness.

Gathering layer transmits IoT measurements to the edge layer to reduce communication costs, save bandwidth and meet latency requirements in distributed sensor networks. Gateways at this layer can aggregate sensor data sending them in an appropriate format and size to the monitoring

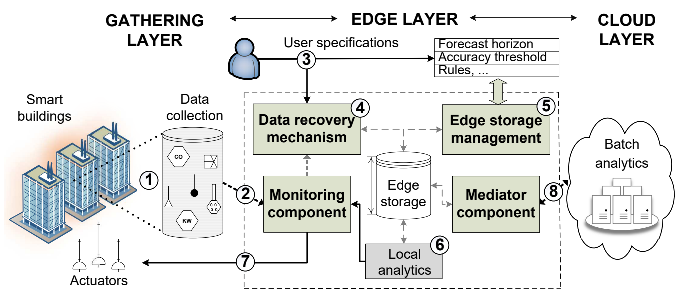


Fig. 1. EDMFrame overview, based on smart buildings example.

component. In step (1) data are collected from smart buildings and then in step (2) transferred to the edge layer.

Edge layer manages data through different stages of EDMFrame, to perform accurate and timely analytics. It is composed of edge nodes, e.g., edge servers and micro data centers [10], aiming to perform data processing closer to data sources. EDMFrame includes the following elements:

Monitoring component. This component (i) receives and analyses data to detect outliers and missing values, (ii) notifies mediator component about incomplete data, (iii) prepares data for the data recovery mechanism and (iv) triggers IoT actuators based on local edge analytics. It can also extrapolate data characteristics for further analytics.

Specification list. Once data are transmitted to the edge layer, user specifications are checked in step (3). Specification list contains application-dependent and user-defined parameters, useful for both data recovery and edge data management process (e.g., forecast horizon, monitoring frequency, forecast method, accuracy threshold, conditions).

Data recovery mechanism. The adaptive recovery process is performed in step (4). It receives data from the monitoring component and performs semi-automatic recovery of multiple gaps incorporating recovery cycles (see Section 3). The output is a dataset without gaps and cleaned from outliers.

Storage. Edge storage carries limited capacities. It stores data coming from the data recovery mechanism and communicates with the edge storage management, mediator component and local edge analytics processes.

Edge storage management. In step (5), edge storage management mechanism maintains limited storage keeping only data relevant for near real-time decisions. It checks available data, validates the specification list and implements the storage management phases (see Section 4). The available data are used in step (6) for local analytics, whose output is forwarded either to the storage or to the monitoring component sending commands to actuators in step (7).

Mediator component. The mediator manages PRMs to support data recovery mechanism (see Section 5). In step (8), mediator component communicates with the cloud data repository. It transfers the necessary data from/to the cloud. It can perform data filtration and data transformation to improve data transfer between edge and cloud layer.

Cloud layer contains the data repository, storing historical data collected from IoT systems. It performs compute-intensive big data analytics based on entire datasets.

Sections 3, 4 and 5, detail all edge layer components. We design an experimental implementation as a pipeline with the core algorithms: monitoring component (Algorithm 1),

TABLE 1
Main Notations and Definitions

Notation	Description
D_{in}	Matrix that represents incomplete input data.
D_{fr}	Matrix that represents framed (prepared) data to be stored.
ω	Vector that contains all indexes of missing values.
nom	Variable that counts number of missing values (based on ω).
$\hat{\omega}$	Vector that stores indexes of the current gap ($\hat{\omega} \subset \omega$).
γ	Vector containing forecast accuracies from the iteration phase.
v_γ	Standard deviation (volatility) of the entire vector γ.
s_f	Scaling factor dividing v_γ to set threshold for finding clusters.
CL_{th}	Threshold in identifying stable accuracy clusters.
Δ_v^γ	Set of standard deviations calculated from the sampled γ.
$\mathcal{C}$	Matrix containing detected stable accuracy clusters.
f_{th}^{ac}	Forecast accuracy threshold.
CL_{ap}	Appropriate cluster with stable forecast accuracy.
f_h	Forecast horizon - the amount of data as prediction length.
S_d	Array representing available dataset in storage.
d_f	Decrement factor that decreases available dataset S_d.
df_{pct}	Decrement factor percentage.

data recovery mechanism (Algorithm 2) and edge storage management (Algorithms 5 utilizing Algorithms 3 and 4). Table 1 lists the main notations used hereafter.

3 Data Recovery Mechanism

We present an adaptive mechanism for time series recovery. First, we define a *gap* as a sequence of one or more missing or invalid consecutive values, distributed in time series. Missing values occur due to sensor or monitoring failures. Invalid data represent outliers due to measurement errors.

Definition 1. *A gap G_n^k represents the n-th gap in an incomplete dataset with k missing/invalid values.*

For example, G_2^{17} refers to the second gap with 17 missing values. In Figure 2, we provide a flowchart of the proposed recovery mechanism. First, data are prepared in the monitoring component. To this end, data indexes from each gap in the dataset are detected and marked in the data preparation module. Then, the recovery cycle starts by detecting 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 current recovery cycle. The data processor component analyses data points preceding the current gap, that are important for the setup of the forecasting process, including user specifications. Then, necessary data and technique selected from the repository are forwarded to the forecasting process. Once the gap is recovered, conditions for the next recovering cycle are checked. The following subsections describe all components in detail.

3.1 Data Preparation

The goal of this component is to prepare an incomplete dataset for the recovery process. To this end, we apply a set of operations that detect each gap in the dataset. The data preparation process is described in Algorithm 1. First, line 1 creates an empty vector for indexes of missing values in the dataset. Outliers are identified according to minimum and maximum values, for particular sensors, that can be either application-dependent or predefined by the user. If a data

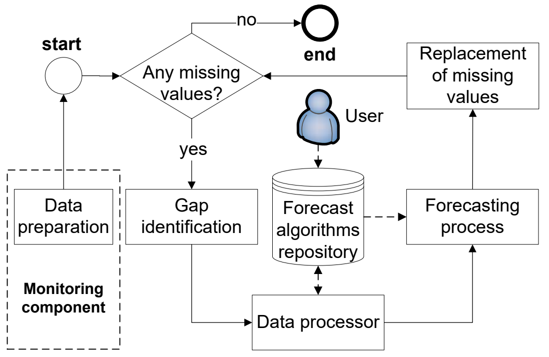


Fig. 2. Adaptive edge data recovery mechanism flowchart.

Algorithm 1 DataPreparation

```
Input: $D_{in}[timestamp, value]$, $D_{fr}[timestamp, \_]$
Output: vector $\omega$
 1: Create vector $\omega$
 2: Replace all outliers by a missing value indicator $NA$ (based on thresholds)
 3: $i \leftarrow 1; j \leftarrow 1$
 4: while $i \leq length(D_{in})$ do
 5:     if $D_{fr}[j,1] = D_{in}[i,1]$ then
 6:         $D_{fr}[j,2] \leftarrow D_{in}[i,2]$
 7:         $i \leftarrow i+1; j \leftarrow j+1$
 8:     else
 9:         $D_{fr}[j,2] \leftarrow NA$
10:         Add index $j$ in vector $\omega$
11:         $j \leftarrow j+1$
12:     end if
13: end while
14: $nom \leftarrow length(\omega)$
```

value is out of bounds, it is replaced by a missing value indicator such as *NA* (Not Available) (line 2), so that the correct value can be efficiently estimated in the recovering cycle. Missing values can occur for different reasons, like system or sensor failures. Once the system/sensor is recovered, the next received data point is stored right after the last generated timestamp. Therefore, to identify a gap, it is necessary to check timestamps. We propose a solution where the monitoring component receives data and stores either corresponding data value or *NA* for each created timestamp (lines 4-13). Counters i and j (line 3) count data from input and prepared D_{fr}, respectively. If timestamps from D_{fr} and D_{in} match (line 5), the data point is moved to D_{fr} beside the corresponding timestamp (line 6). Otherwise, *NA* is stored (line 9), and the index of a missing data point is moved to the created vector ω (line 10). Once the $while$ loop terminates, the vector ω contains all indexes of missing data, the amount of which is placed in the variable nom (line 14).

3.2 Gap Identification

The gap identification phase (see Algorithm 2) is responsible for detecting multiple gaps in a given dataset. It identifies the limits of each gap, using this information for the recovery process. Each gap is processed separately, to enable the selection of an appropriate forecasting technique based on characteristics of previous data or user predefined specifications. The counter i (line 1) is used to iterate over the vector ω, while data index of the first missing value is copied to the beginning of the vector $\hat{\omega}$ and stored also in the variable t (lines 2-3). As long as there are missing values in nom (line 4), the counter i looks for the next index of missing value, while the index stored in the variable t is

Algorithm 2 `GapIdentification`

```
Input: Vector $\omega$, variable $nom$
 1: $i \leftarrow 1$
 2: $\hat{\omega}[i] \leftarrow \omega[i]$   ▷ Create vector $\hat{\omega}[]$ storing missing indexes of current gap
 3: $t \leftarrow \omega[i]$   ▷ Create temporary variable $t$ and store first missing index
 4: while $nom > i$ do
 5:     $i \leftarrow i + 1; t \leftarrow t + 1$
 6:     if $t = \omega[i]$ then
 7:         $\hat{\omega}[i] \leftarrow \omega[i]$
 8:     else
 9:         Remove indexes of $\hat{\omega}$ from $\omega$
10:         break;
11:     end if
12: end while
```

incremented by 1 (line 5). It allows to check whether a gap of consecutive missing values exists (line 6). If indexes are consecutive, the corresponding index is copied to the vector $\hat{\omega}$ (line 7). Otherwise, all missing values from the current gap are detected, and the vector ω is updated in line 9.

3.3 Data Processor

This component performs extrapolation of data characteristics and parameters needed for the utilization of particular forecasting methods. Necessary characteristics are obtained during the analysis of available data preceding the first missing index of the current gap, that is identified in the previous component. To efficiently forecast NA marked gaps, predecessor data are analyzed to derive parameters necessary to the forecasting process. Parameter selection depends on the forecasting technique. Semi-automatic mechanism allows two scenarios: (i) *single-technique recovery* (STR) and (ii) *multiple-technique recovery* (MTR). In the first scenario, a single technique, that can be specified by users, is used to recover all gaps. In the latter scenario, a technique is selected for different gaps. Currently, we assume that techniques are predefined by users in the algorithm repository.

3.4 Forecasting Process

In this component, a forecasting technique is selected from the repository and applied to the current gap. Figure 3 shows the adaptive recovery process including results of aforementioned components. After corresponding missing indexes are stored by the preparation component and the first gap identified by the gap identification component, the data processor 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 [25]. Here, we select two techniques, according to different dataset characteristics: (i) 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 [26]; (ii) the Exponential Smoothing method (ETS), although overlapping in some cases with ARIMA model, can be used for short-term seasonal series or with multiple complex seasonality [27]. If seasonality occurs in time series, by checking periodicity, the data processor can forward that information to the next component. Users can also specify additional information about the data, such as a monitoring

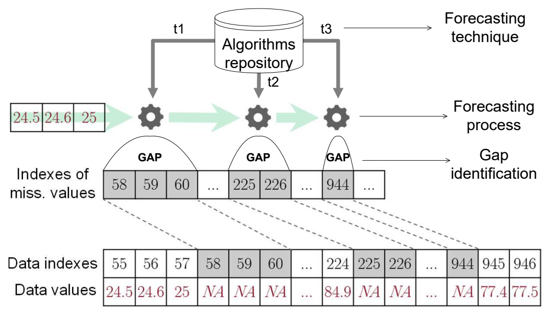


Fig. 3. Forecasting process of adaptive data recovery for multiple gaps.

frequency, e.g., if temperature data are collected every five minutes, then the seasonal parameter value 288, representing the expected daily seasonality ($12 \cdot 24$), is included in the forecasting procedure. Once 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 ω. Once the current gap is recovered, the next gap (if exists) is considered in a new cycle. The recovering process stops when no more missing values are left in nom.

3.5 Algorithm Complexity

By looking at the *while* loop in line 4 of Algorithm 1, we see it iterates over available dataset making it *O(n)*, where *n* represents a number of data points in the acquired dataset. Further, entering the *while* loop of Algorithm 2 (line 4), it iterates the vector of indexes of missing values that are always less than the amount of data in S_d. The other lines require $O(1)$. In case the forecasting process uses ARIMA based models, the time complexity is $O(n^2)$, where n is the size of data, resulting in the overall complexity $O(n^2)$. Running time is affected by different factors such as the size of the gap that has to be recovered, the amount of available historical data (finding an optimal trade-off between the gap size and necessary amount of historical data is given by the mediator component in Section 5) or seasonal complexity of time series. Since the proposed mechanism targets resource-limited edge nodes and analysis for near real-time decisions, we expect that the input size and dimensionality of incompleteness will not cause a violation of latency requirements.

4 Edge Storage Management

Effectiveness of a limited edge storage depends on the ability to determine the amount of necessary data to perform accurate near real-time decisions. Hence, an edge node should keep only relevant data for local data analytics, discarding or transferring the rest if they are irrelevant. Based on the architectural model (see Figure 1), we describe edge storage management phases, as shown in Figure 4, namely:

Learning phase. This phase derives information about data, such as time series pattern recognition, used to determine the most appropriate method for that specific pattern [28], or seasonality over a certain period, used to set up a forecast method [29]. This phase is executed only once and provides information used by all the other phases.

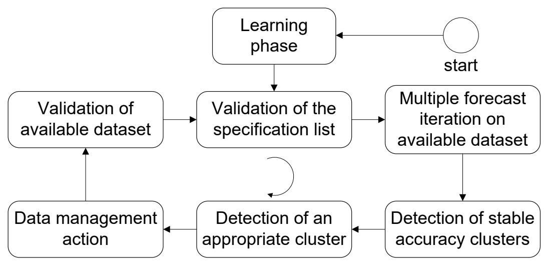

Learning phase
start
Validation of available dataset
Validation of the specification list
Multiple forecast iteration on available dataset
Data management action
Detection of an appropriate cluster
Detection of stable accuracy clusters

Fig. 4. Edge storage management flowchart and design principles.

Validation of the specification list. This phase checks the user-defined specification list. During the execution of the proposed algorithm, users can update the specification list anytime, e.g., setting forecast accuracy threshold, a new forecast horizon or different forecast methods. This list has to be checked each time a cycle starts since any changes made to it can affect the whole edge storage management.

Multiple forecast iteration on the available dataset. This phase takes one of the forecasting methods (in our case ETS or ARIMA) with accuracy threshold (f_{th}^{ac}) and forecast horizon (f_h) from the specification list. The available dataset is divided into training and test data. Test data are equal to the number of data points specified by the user in the specification list (i.e., f_h). The amount of training data is reduced in each iteration by a certain amount of data to find parts of the dataset resulting in required forecast accuracy. At the end of each iteration, forecast accuracy measures are added in the vector γ to be used in the next phase.

Detection of stable accuracy clusters. Here, stable clusters of accuracy values have to be found in the vector γ.

Definition 2. *We define a stable cluster CL_{st} as a set of subsequent data points in the vector γ whose standard deviation for contained values is less than a given percentage CL_{pct} of the standard deviation of entire vector γ, that is,*

$$CL_{st} \subset \gamma \quad \text{AND} \quad sd(CL_{st}) < CL_{pct} * (sd(\gamma)) \qquad (1)$$

We define that a cluster contains at least three members. To provide reliable information regarding future system behaviors, our predictions must be stable. When the multiple forecast iteration process is finished, cluster detection is applied on the vector, which consists in measuring forecast accuracy from each of forecast iterations. The method finds stable clusters of forecast accuracies close to the threshold defined in the specification list (see Section 4.1).

Detection of an appropriate cluster. Previous step can return more than one stable cluster, therefore we define how to select the most appropriate one. Stable clusters can differ in mean value and the amount of used data. Therefore, selection priorities have to be set. We propose the twofold priority for cluster selection and a corresponding algorithm. First priority is to satisfy user-specified threshold. Formally:

Definition 3. *A stable cluster $\mathcal{C}[i]$ of forecast accuracies is appropriate if its mean value ($\mathcal{C}[i]^{m\_v}$) is the closest to the accuracy threshold f_{th}^{ac} from the specification list, namely,*

$$CL_{ap} = \underset{\mathcal{C}[i]}{\arg\min} \left(|\mathcal{C}[i]^{m\_v} - f_{th}^{ac}| \right) \qquad (2)$$

We select, as appropriate cluster CL_{ap}, the one with minimum absolute difference between f_{th}^{ac} and $\mathcal{C}[i]^{m\_v}$. If there are clusters with higher accuracy, we select the one using less data, regarding second priority (see Section 4.2).

Data management action. This step releases irrelevant data from the storage. According to the appropriate cluster, we define that the central data index of this cluster indicates a border between relevant and irrelevant data. There are three possible cases: (i) We can reach an appropriate cluster among stable clusters respecting the desired accuracy threshold with fewer amount of data. In this case, all data not needed to obtain the observed accuracy cluster are released; (ii) None of the resulting stable clusters meets the specified accuracy threshold by the client. In this case, data management action will select the one with fewer data points; (iii) Forecast accuracy of stable clusters is higher with an increased amount of training data, e.g., forecast based on all available data from the storage. In that case, the mediator component can retrieve more data from the cloud repository.

Validation of available dataset. The adaptive algorithm is continuously repeated and in each cycle checks storage for newly collected data. Depending on application and data generation rate, the next cycle of edge storage management can act as a triggered event. In the next cycle, data received from the recovery mechanism and, potentially, from the mediator component, are included. The relevance of old data can be lower unless the prediction accuracy level shows that some stable clusters occur based on these data. In that case, if algorithm feedback shows that accuracy increases with historical data and the amount of currently stored data exceeds edge storage limitations, this data processing can be performed in the cloud. Hence, the approach requires to monitor accuracy rate variations for performed forecasts.

4.1 Detection of Stable Clusters

Detection of smooth behaviors for consecutive forecast accuracies, calculated in the forecast iteration phase, presents the cornerstone of our algorithm. There are many clustering techniques [30] such as partitioning, hierarchical or density-based, but they are not suitable for our case, because often they require specification of a certain number of clusters beforehand as well as separating the entire dataset based on similarity. Our case requires a dynamic approach which discovers as few clusters as possible based on Definition 2, and considering only corresponding parts of the entire dataset. The process consists of three steps: first, we calculate the overall standard deviation for all forecast accuracies and mark it as a baseline. Second, forecast accuracies are grouped into clusters of fixed length and standard deviation is calculated per cluster. Third, obtained deviations are compared to the baseline considering the previously calculated threshold. Consequently, stable clusters show where the forecast accuracies are stable. Pseudo-code for detecting stable accuracy clusters is presented in Algorithm 3. The method requires vector γ consisting of forecast accuracy measures (MAPA) from the forecast iteration process and scaling factor s_f that is used for the threshold calculation. The threshold CL_{th} differs between different datasets, because each measurement has its scale of values with unpredicted volatility. Based on experiments, by default the scaling factor is always equal to 5 in the first attempt of stable clusters detection. This means that only stable clusters with CL_{pct} equal to 20% (see Definition 2) of the baseline standard deviation will be selected. However, even with a fixed threshold it is possible to have no clusters. In case it is impossible to meet any stable clusters for the specified threshold, i.e., since forecast accuracies show greater dispersion, the threshold is increased and the process is repeated. For example, decreasing the scaling factor from 5 to 4, the CL_{pct} becomes 25% of the baseline for detecting stable clusters, by setting in Algorithm 5. In line 1, the algorithm calculates standard deviation of the entire vector γ and divides the result (line 2) by scaling factor s_f to set a threshold CL_{th} for finding clusters. In line 3, standard deviations will be calculated for each of grouped iteration results in a sliding window in vector γ and then stored to vector Δ_v^γ. Before searching for stable clusters, algorithm initializes two counters and creates an empty matrix in lines 4-5, i.e., counter i will count clusters in Δ_v^γ and counter j will denote recognized stable clusters in matrix $\mathcal{C}$ including attributes: mean value of forecast accuracies, and corresponding ranges of cluster indexes. To detect stable clusters, the algorithm starts from the beginning of Δ_v^γ (line 6) and checks if a standard deviation of the first cluster is below threshold CL_{th} (line 7). If a cluster is recognized as stable, corresponding data will be added to $\mathcal{C}$ (lines 8-11). In some cases, stable clusters can be wider, so it is necessary to check the neighbor cluster (line 12), and if the new one is recognized as stable (line 13), it will continue to check other neighbors (line 14). For each next cluster recognized as stable, existing cluster is extended updating its corresponding mean value and end index (lines 15-16) and checks the next one (line 17). When there are no more stable clusters in a row, a place for a new stable cluster is prepared increasing counter j in line 19. In case the next cluster is not recognized as stable (line 20), the algorithm will simply close the existing cluster and check the next one (line 21). For each non-stable cluster (line 23), the algorithm will increment counter i (line 24) and loop back to line 6. Finally, matrix $\mathcal{C}$, whose rows represent stable clusters with the attributes, is returned.

4.2 Detection of the Appropriate Cluster

The cluster selection process is described in Algorithm 4. It starts by checking the amount of stable clusters. The *else* branch in line 11 is executed only if one stable cluster is recognized and it will become the appropriate cluster (line 12), otherwise, the algorithm will find the appropriate cluster (lines 2-10). Considering the first priority, the appropriate cluster becomes the one that is closest to the specified accuracy threshold (line 2). Further, all stable clusters (line 3) that have better accuracy than selected CL_{ap}, i.e., higher mean value, and whose start index begins after end index of selected CL_{ap} (line 4), become potential appropriate clusters (line 5). If there are such clusters (lines 8-10), the one including less data, i.e., which has the lowest start index (line 9), is selected as a new appropriate cluster CL_{ap}. Finally, appropriate cluster CL_{ap} is returned in line 14.

4.3 Adaptive Algorithm

The adaptive algorithm integrates all design principles shown in Figure 4, and includes calls on described Algorithms 3 and 4. Algorithm 5 requires a forecast horizon

Algorithm 3 `DetectionOfStableClusters`

```
Input: Vector of iteration results $\gamma$, scaling factor $s_f$
Output: Matrix $\mathcal{C}$
 1: $v_\gamma \leftarrow sd(\gamma)$   ▷ Calculate standard deviation (volatility) of entire vector $\gamma$
 2: $CL_{th} \leftarrow \frac{v_\gamma}{s_f}$   ▷ Calculate threshold $CL_{th}$
 3: Create vector $\Delta_v^\gamma$ storing STDs of sliding windows (length 3) on vector $\gamma$
 4: $i \leftarrow 1; j \leftarrow 1$   ▷ Initialize counters $i$ and $j$
 5: Create matrix $\mathcal{C}$ forming mean value, start and end index of stable clusters
 6: while $i < length(\Delta_v^\gamma)$ do
 7:     if $\Delta_v^\gamma[i] < CL_{th}$ then   ▷ Satisfying conditions in Equation 1
 8:         Add cluster in $\mathcal{C}$ such that
 9:         $\mathcal{C}[j,1] \leftarrow$ mean value of corresponding range in $\gamma$
10:         $\mathcal{C}[j,2] \leftarrow$ start data index of corresponding range
11:         $\mathcal{C}[j,3] \leftarrow$ end date index of corresponding range
12:         $i \leftarrow i+1$   ▷ Incrementing $i$ to check next potential cluster member
13:         if $\Delta_v^\gamma < CL_{th}$ then   ▷ Satisfying conditions in Equation 1
14:             while $\Delta_v^\gamma[i] < CL_{th}$ do
15:                 Update mean value $\mathcal{C}[j,1]$
16:                 Update end index $\mathcal{C}[j,3]$
17:                 $i \leftarrow i+1$
18:             end while
19:             $j \leftarrow j+1$
20:         else
21:             $i \leftarrow i+1; j \leftarrow j+1$
22:         end if
23:     else
24:         $i \leftarrow i+1$
25:     end if
26: end while
27: Return $\mathcal{C}$
```

Algorithm 4 `DetectionOfTheAppropriateCluster`

```
Input: Matrix $\mathcal{C}$, accuracy threshold $f_{th}^{ac}$
Output: Appropriate cluster $CL_{ap}$
 1: if $\mathcal{C}$ has more than 1 cluster then
 2:     Compute $CL_{ap}$ using Equation 2
 3:     for each cluster $i \in \mathcal{C}$ do
 4:         if $\mathcal{C}[i]^{m\_v} > CL_{ap}^{m\_v}$ AND $\mathcal{C}[i]^{start\_index} > CL_{ap}^{end\_index}$ then
 5:             Add $\mathcal{C}[i]$ to temporary matrix $\mathcal{A}$
 6:         end if
 7:     end for
 8:     if $\mathcal{A}$ is not empty then
 9:         $CL_{ap} \leftarrow \mathcal{A}_i$ with minimum starting index
10:     end if
11: else
12:     $CL_{ap} \leftarrow \mathcal{C}[0]$
13: end if
14: Return $CL_{ap}$
```

Algorithm 5 `AdaptiveAlgorithm`

```
Input: forecast horizon $f_h$, accuracy threshold $f_{th}^{ac}$, storage data $S_d$
 1: Calculate $d_f$ using Equations 4 and 5
 2: while $length(S_d) > 2 * f_h$ do
 3:     Perform method $(S_d, f_h)$
 4:     Calculate MAPA (See Equation 7)
 5:     Add MAPA to vector $\gamma$
 6:     $S_d \leftarrow S_d$ decreased by $d_f$
 7: end while
 8: $s_f \leftarrow 5$   ▷ Set threshold on 20% of overall standard dev., i.e., $\frac{1}{5}$)
 9: $\mathcal{C} \leftarrow$ DetectionOfStableClusters$(\gamma, s_f)$
10: while $\mathcal{C}$ is empty do
11:     $s_f \leftarrow s_f - 1$   ▷ Decrease scaling factor $s_f$
12:     $\mathcal{C} \leftarrow$ DetectionOfStableClusters$(\gamma, s_f)$
13: end while
14: $CL_{ap} \leftarrow$ DetectionOfTheAppropriateCluster$(\mathcal{C}, f_{th}^{ac})$   ▷ Find $CL_{ap}$
15: Release data from $S$ in range between the oldest and the central index of the
    appropriate cluster $CL_{ap}$, retaining only relevant data in the storage.
```

f_h and accuracy threshold f_{th}^{ac} that are specified in the specification list, and array S_d that denotes data available in storage. As shown in Figure 4, the learning phase helps to select and set up the appropriate method. One of the possibilities is to find periodicity as a necessity to determine the seasonality and thereby to make a better forecast, as described in [31]. Next, at the beginning of the algorithm, the decrement factor d_f is calculated utilizing Equations 4 and 5. The calculated d_f will be decreasing storage data S_d

in the multiple forecast iteration phase. Forecast iterations (lines 2-7) will continue until the amount of data in S_d becomes less than the two lengths of a forecast horizon (more details in Section 4.3.1). Appropriate forecast method uses storage data S_d and other attributes (e.g., periodicity), to make prediction for defined forecast horizon f_h and calculates Mean Absolute Percentage Accuracy (MAPA) (see Equation 7) in lines 3-4. At the end of each iteration, the MAPA is stored in vector γ (line 5) and a certain amount of old data is removed (line 6) based on decrement factor d_f. Next, for the detection of stable accuracy clusters phase (lines 9-13), scaling factor s_f is set to number 5 representing the impact of 20% in determining the threshold for finding stable clusters in Algorithm 4. If any stable cluster is recognized, the matrix $\mathcal{C}$ gets corresponding information (line 9): mean value, start and end index of the cluster. Otherwise, if stable clusters cannot be found (line 10), the algorithm will decrease the s_f and keep looking for the clusters (lines 11-12). Line 14 finds the appropriate cluster CL_{ap}. Finally, data in array S_d are released in range between the oldest index and the central index of the appropriate cluster CL_{ap} (line 15). Adaptive algorithm repeats itself based on demands in the specification list.

4.3.1 Optimal Parameter Settings

Our goal is to set up necessary parameters enabling continuous operation of the edge storage management mechanism. To allow proper performance of proposed algorithms, storage must always contain enough data for training, plus additional test data (amount of which is equal to defined forecast horizon), and there must be enough number of forecast iterations in each cycle to find stable clusters. This is ensured by keeping training data as

$$T = 2 * f_h + k, \quad k \geq 3 \tag{3}$$

where T denotes the amount of training data points, f_h denotes a forecast horizon that is application dependent and given in the specification list (see Figure 1), and k is a natural number. Based on Equation 3, there will always be at least 4 forecast iterations resulting in 4 MAPA measures as a basis for finding at least one stable cluster (see Definition 2).

Also, running time depends on the number of multiple forecast iterations, which is affected by the decrement factor d_f, calculated using Equations 4 and 5:

$$\psi = round(df_{pct} * (T - 2 * f_h)) \tag{4}$$

$$d_f = \begin{cases} \psi, & \text{if } \psi >= 1. \\ 1, & \text{otherwise,} \end{cases} \tag{5}$$

where *round()* is a function that rounds the result half away from zero to integer and df_{pct} is decrement factor percentage. In order to set optimal value for df_{pct}, we performed experiments using all possible ranges of df_{pct} on different datasets. The evaluation is done on 144 data points, representing data collected every 5min over 12h with 1h forecast horizon, satisfying Equation 3 to have enough iterations to find stable clusters. Thus, maximum df_{pct} of 30%, that is a representative from a range of 26%-33%, gives the necessary 4 forecast iterations. Results showed that as df_{pct} becomes very low (1%-2%) and very high (8%-30%), the algorithm cannot always find stable clusters in the first run of calculation (Algorithm 5, line 8) while at the same time having a number of clustered MAPA measures near 100%. Among rest df_{pct} values, to find a setting that releases more data without significantly decreasing the accuracy of the appropriate cluster, we set df_{pct} at 3% resulting in 34 iterations on average. Further, we assume that the prediction of data, potentially containing seasonality, requires at least twice as many data points compared to the forecast horizon. This assumption is derived from the constraint that the prediction of one period of seasonal time series requires at least two periods of prior data. Therefore, to calculate the d_f, the training dataset is reduced by two lengths of the f_h.

4.4 Complexity Analysis

Considering Algorithm 5, its complexity is $O(n^2)$, where n represents the size of data. ARIMA method (line 4) has $O(n^2)$ complexity and the outer *while* loop (lines 2-7) iterates the entire dataset until n is equal to the two lengths of the specified f_h. In the worst case, it is decreased by 1 at each iteration, leading to a $O(n)$. Further, both Algorithms 3 and 4 have the complexity of $O(n)$. The *while* loop (line 6) from the Algorithm 3 iterates over the size of the vector Δ_v^γ. The inner *while* loop (line 14) uses the same counter as the outer loop resulting in the same $O(n)$ and decreasing the space complexity by not creating new objects in line 8. Algorithm 4 instead iterates over each cluster in the *for* loop in line 3, whose number is always less than n. Other operations have either a $O(1)$ or a $O(n)$, resulting in a $O(n^2)$ time complexity. Such complexity can be reduced by using less accurate forecasting methods. Even though $O(n^2)$ is not suitable for big datasets, it provides acceptable response time in this context since we target edge storage.

5 Mediator Component

Storage space limitations on edge nodes prevent keeping all measurements. Many IoT systems require both local edge and global batch data analytics [11], [32]. Consequently, an edge node can keep only relevant data, and send all acquired data to be integrated into the cloud data repository. Once historical data are available in the cloud, batch analytics can be applied. By using historical data it is possible to calculate the Projection Recovery Map (PRM), which recommends ranges of data for recovering gaps of various lengths, as well as the appropriate recovery technique for each dataset. In this context, the proposed mediator component can either employ cloud-based PRMs to improve recovery of gaps detected by the monitoring component (see Figure 1), or transfer data considering local analytics requirements. In both cases the mediator can retrieve necessary data from the cloud, storing them locally when needed. Here, we describe the first case. Regarding PRMs, recommended ranges of data for certain lengths of detected gaps can be found by slightly modifying edge storage management mechanism (Algorithm 4). Selecting the appropriate cluster CL_{ap_med} means selecting a cluster with the highest accuracy:

$$CL_{ap\_med} = \underset{\mathcal{C}[i]}{\arg\max} \, (\mathcal{C}[i]^{mean\_value}) \tag{6}$$

TABLE 2
Main Characteristics of Datasets

Source	Dataset	Sensor type	Range of values	Volatility [SD]
1	h_1	heat index [F]	52.11-88.79	8.74
	h_2	room temp. [F]	68.90-79.70	2.51
	h_3	RH [%]	30.2-92.7	18.52
2	b_1	el. meter [kWh]	19.86-19.97	0.04
	b_2	flow temp. [C]	41.3-48.1	1.32
	b_3	IAQ [ppm]	458.11-707.88	62.53

In this case, we employ a priority criteria different from Section 4.2. Once the CL_{ap_med} is detected, bounds (upper and lower) for the amount of used data points and corresponding MAPA measures are stored. This process is repeated with multiple-recovery techniques (MTR case) over consecutive gap lengths and then merged in a PRM for each dataset. In extreme cases where big gap lengths are expected, PRM can be calculated over non-consecutive number of missing values and, if required, interpolate ranges that are not calculated. To check the accuracy of PRM-based MTR, we test it on available historical data (Section 6.4).

6 Performance Evaluation

EDMFrame is simulated using R on a 64-bit Windows 10 PC, with a 2.70-2.90 GHz Intel i7-7500U CPU and 16 GB RAM.

6.1 Data and Measures

We evaluate our approach on sensor-based time series data, typical in IoT applications like smart buildings and homes [4]. The main characteristics of datasets are presented in Table 2. Datasets h_1-3 contain traces from the Smart* project [33] for optimization of energy consumption in smart homes, obtained by UMass Trace Repository [34]. They represent various environmental and electrical data, such as temperature, relative humidity, wind information, and heat index. Datasets b_1-3 come from the monitoring system of Austria's largest Plus-Energy Office High-Rise Building. These datasets contain various measurements used for automatic heating, cooling, ventilation, energy management (e.g., temperature, indoor air quality, electricity consumption, and production). We use Mean Absolute Percentage Error (MAPE) for forecast accuracy evaluation among different datasets, due to its scale independence. We also define Mean Absolute Percentage Accuracy (MAPA) as in Equation 7:

$$\mathtt{MAPA}(Y,\hat{Y}) = 100 - \mathtt{MAPE}(Y,\hat{Y}) \qquad (7)$$

where $\mathtt{MAPE}(Y,\hat{Y})$ is equal to $\frac{100}{n}\sum_{i=1}^{n}|\frac{(y_i-\hat{y}_i)}{y_i}|$, Y and $\hat{Y}$ are respectively the sets of actual values and their forecasts, n is the number of data points, $y_i - \hat{y}_i$ is the forecast error, y_i and $\hat{y}_i$ are respectively the i-th value of y and its forecast.

6.2 Data Recovery Process

We evaluate the applicability of the proposed approach (see Section 3) by recovering multiple gaps in different datasets. We see an example in Figure 5. The upper graph shows an incomplete subset of dataset *h_1* before the recovery, while the lower graph shows a complete dataset after recovery.

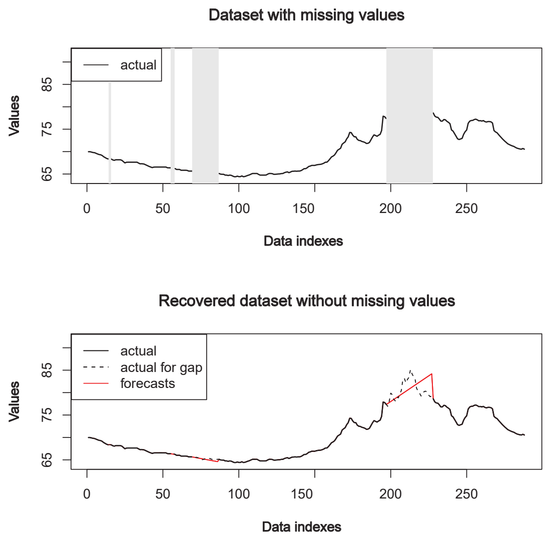


Fig. 5. Adaptive data recovery of multiple gaps (G_1^1, G_2^2, G_3^{17} and G_4^{30}) on dataset *h_1*, employing ARIMA, as STR approach.

Gray shaded areas indicate four gaps. In all datasets by source 2, we observe several gaps in collected data affecting data analytics. Therefore, we identify these gaps and artificially make several gaps with same sizes in the dataset *h_1* (precisely, gaps with 1 (G_1^1), 2 (G_2^2), 17 (G_3^{17}) and 30 (G_4^{30}) consecutive missing/invalid data values). Then, these gaps are recovered using the proposed mechanism, and forecast accuracy is evaluated using MAPE. The black solid line represents the actual state of a dataset. The black dashed line shows actual data for corresponding gaps, while the red solid line represents predicted values of missing/invalid data, calculated using forecasting techniques. In this case, multiple gaps are automatically recovered using ARIMA, representing the STR scenario. Sensor-based time series, as shown in the first sub-figure, can contain different behaviors from stationary to trend and volatile patterns. For this reason, the MAPE of reconstructed gaps G_1^1, G_2^2 and G_3^{17} are significantly low, 0.1843%, 0.1317% and 0.3366% respectively, while the MAPE for the G_4^{30} is 2.6797%. We also notice a direct relationship between gap length and forecasting error. Results show that our mechanism is able to recover all gaps with running time of $0.68s$. Moreover, based on the proposed mechanism, multiple techniques can be involved in the recovery process of each gap separately. MTR is employed by the mediator component in Section 6.4.

6.3 Storage Management Process

We simulate edge storage management with a fixed amount of data. We use Algorithm 4 for appropriate cluster selection. Dataset *h_1* is shown in Figure 6 and represents 3 cycles of edge storage management. In the first cycle, the same data from the data recovery process are used. Further, in the second and third cycles, we set, in addition to data coming from the previous cycle, the upcoming amount of data on 144 (5min interval for 12h) data points per turn. Also, we define some rules to simulate the specification list. Forecast horizon f_h is set to 12 data points, representing one hour ($12 \cdot 5$min) for which forecasts are calculated. f_h is fixed in the current cycle, while the user can change

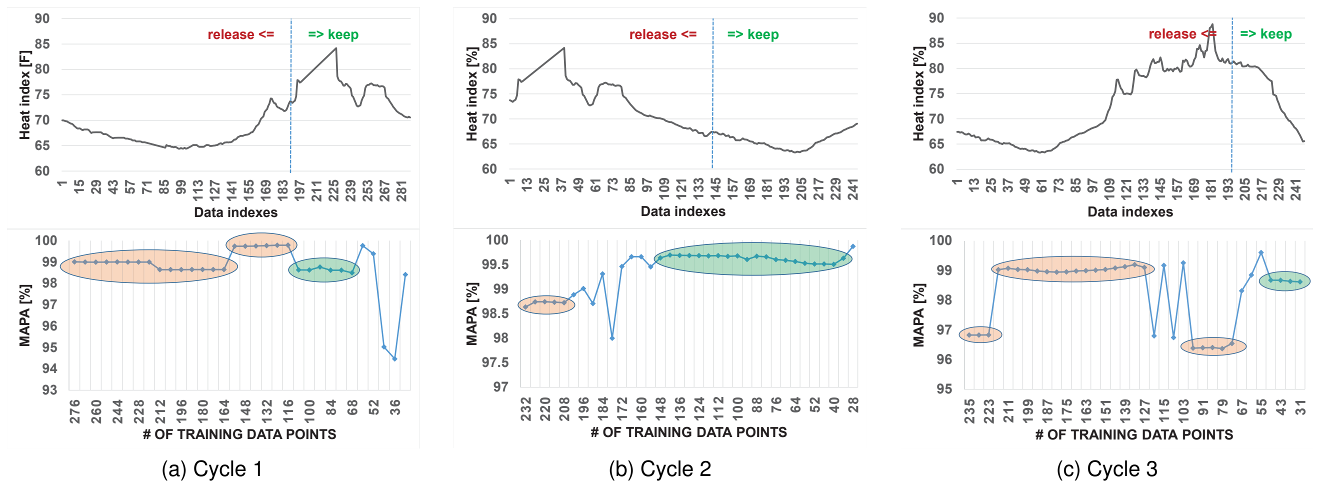


Fig. 6. Evaluation of edge storage management on *h_1* dataset - cycles 1-3 showing available dataset (upper graphs) and stable clusters of forecast accuracies (lower graphs). Vertical blue dotted line represents retained data points.

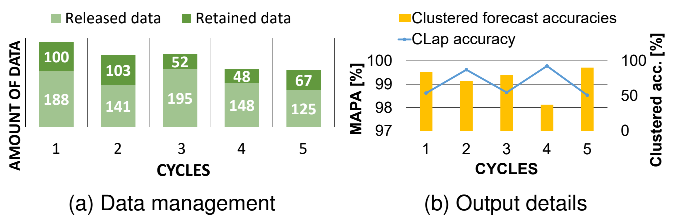


Fig. 7. Released/retained data, appropriate cluster accuracy, clustered forecast accuracies percentage of Algorithm 5 after 5 cycles.

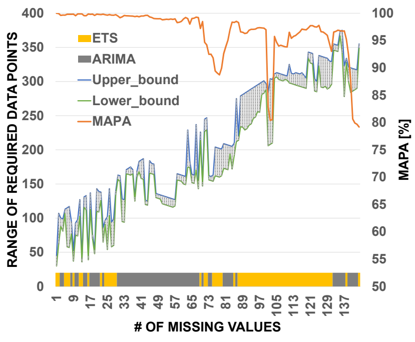
TABLE 3
Mean Results of 5 Storage Management Cycles per Dataset

Dataset	Retain.	Retain. [%]	Clusters	Clust. [%]	Iterations	CL_{ap}	Time
h_1	74	31.4	3.2	72.73	33	99.04	2.48
h_2	63.4	33.4	3.8	65.06	33.8	99.77	2.53
h_3	94.6	42.8	4	59.89	34.6	96.90	3.09
b_1	69.8	35.2	3.2	73.88	33.8	99.99	2.26
b_2	97.2	44.6	3.6	81.24	34.4	99.20	3.10
b_3	137.2	52	3.4	61.43	33.6	98.05	2.62

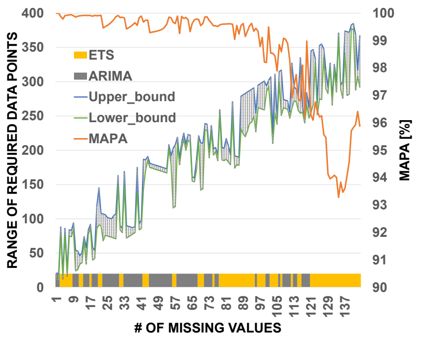
the desired f_h in the specification list. Then, we consider the threshold CL_{th} for identifying stable accuracy clusters of 90%, since the proposed framework targets near real-time decision-making in IoT applications and we expect accurate MAPA measures due to the short f_h comparing to the available dataset size. Regarding Figure 6, the upper graph represents the original dataset. The vertical blue dotted line represents the data management decision after the procedures in lower graphs. The lower graphs represent the result after applying the forecast method and validating the principle for multiple forecast iterations on the available dataset. In MAPA measurements, orange areas are stable clusters, while green areas are selected appropriate clusters. The last 12 data points are test data, while the rest is used in different variations to predict f_h. In the first cycle, the algorithm finds an appropriate cluster in a range between 68 and 108 available data points, corresponding to the cluster between data indexes 168 and 208 in our original dataset (upper graph). Central index of that cluster indicates that data management will release data points in range 1-188, respectively, indexes in range 189-288 will retain in the edge storage. The process repeats for each next cycle. Figure 7 summarizes all 5 cycles. Figure 7a shows both, released and retained data per each cycle, while Figure 7b shows the accuracy of the selected appropriate cluster CL_{ap} and the percentage of clustered MAPA values (lower graphs in Figure 6) per cycle. In Table 3, all 5 cycles are averaged and compared among datasets in Table 2. The results show that on average 39.9% of data points can be retained, while keeping the appropriate cluster CL_{ap} accuracy, depending on algorithms' parameters. Generally, we can find appropriate clusters with accuracy around 98.83% and clustered MAPA measurements above 50%, based on approximately 34 multiple forecast iterations. Results show that our mechanism can achieve user-defined accuracy using less data.

6.4 Mediator Process

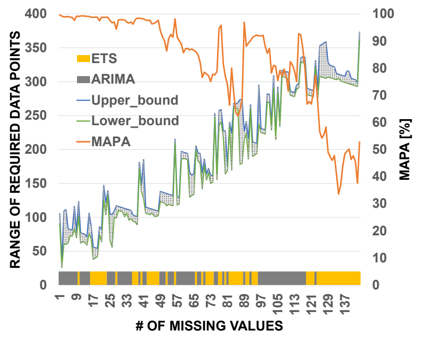
In Section 4.3, we define two methods for data recovery: STR and MTR. Section 6.2 describes STR scenario, where each gap is recovered with a single technique, that uses data points preceding each gap as input. For the second method, we employ PRMs, used by the recovery mechanism to select the recommended range of required data points and recommended recovery method. In Figure 8, for each number of missing values from 1 to 144, the algorithm finds recommended range of required data points by finding stable clusters (Algorithm 3) and calculating the most accurate cluster. The original algorithm is modified by considering stable clusters with the highest forecast accuracy (see Definition 6). The blue line shows the upper border of the cluster, while the green line its lower. We apply ETS and ARIMA, and the method with the highest accuracy is selected. Figure 8(a-f) shows PRMs for all datasets. ETS results are in yellow, while ARIMA in grey. For each selected range of required data points, we show MAPA value in orange. The mediator component stores completed PRMs, and once the monitoring component detects a gap, the mediator recommends a



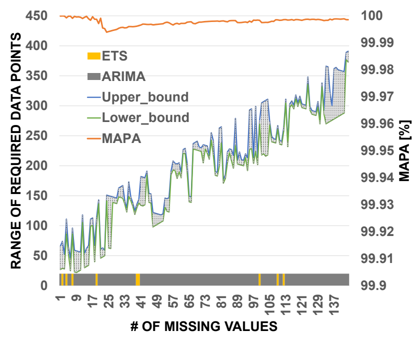
(a) h_1



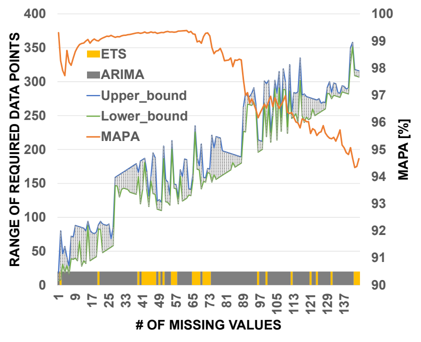
(b) h_2



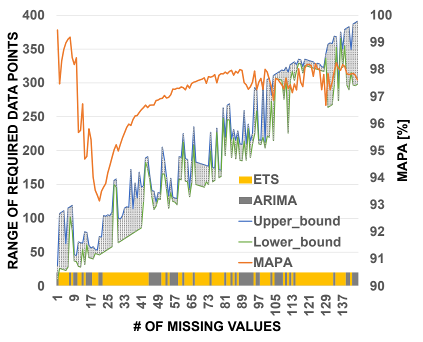
(c) h_3



(d) b_1



(e) b_2



(f) b_3

Fig. 8. Projection recovery maps for six evaluated datasets.

TABLE 4
Comparison of PRM-based Multiple Technique Recovery (MTR) vs Single Technique Recovery (STR) for four gaps on six datasets

Datasets	h_1-STR	h_1-PRM	h_2-STR	h_2-PRM	h_3-STR	h_3-PRM	b_1-STR	b_1-PRM	b_2-STR	b_2-PRM	b_3-STR	b_3-PRM
Runtime [s]	0.68	2.47	0.98	0.83	0.97	1.62	0.89	0.75	1.2	1.1	1.08	0.92
MAPE	0.8331	0.8061	0.4919	0.3476	13.3229	11.789	0.0084	0.0029	0.9377	0.9321	0.767	0.7836

range of required data points and a forecasting method for recovery. In case there is not enough data in edge storage, mediator component retrieves data from the cloud, storing them for the time of data recovery. We test data recovery with 4 defined gaps (see Section 6.2) using STR as a baseline. In Table 4, we compare running time and MAPE between STR and PRM-based MTR. STR achieves running time of $0.97s$ on average, while PRM-based recovery can achieve up to 65.48% less error (e.g., dataset b_1) and only 31.96% more time on average compared to STR. Using PRMs, we can improve at least one objective or both in some cases.

7 Related Work

Data management. Authors in [35] discuss an IoT data management framework, focusing on data collection, storage, and processing. [36] targets the resilience and privacy of sensitive data in delay tolerant networks. Other solutions target resilient edge systems for IoT data management and cloud, focusing on system resource management [2], location-based energy control [4], network congestion [14], security [15] or data integrity [37]. Others [7], [21], [38], describe the interplay and communication models for cloud and IoT resources due to growing data streaming and increasing latency issues of smart sensors. These works do not discuss critical decision-making processes at the edge.

Data science. Typical data recovery methods are based on cubic interpolation [39], Singular Spectrum Analysis [40] or Lomb-Scargle method [41]. [42] uses univariate imputation for air pollution data, considering fixed size gaps. All these works either do not cover IoT scenario or employ a single specific method for recovery of various gap lengths, despite diverse data characteristics. Time series forecasting has wide application in predictive data analytics. Authors in [25] describe time series forecast methods, including a self-adaptive approach for method selection based on users' forecasting objectives. The use of different methods is motivated by different data characteristics before each gap. Methods like ARIMA and ETS [27], [43], are very suitable for near real-time decisions in IoT, since they do not require much user interaction. Further, despite scenarios where data generation is triggered by certain events, here we focus on regularly time-stamped measurements. However, our EDM-Frame is designed in a generic way, such that depending on the application context and sensor data characteristics, different methods can be utilized for both data recovery mechanism and adaptive edge storage management.

Storage management. Micro data centers are described in [44]. The problem of reducing data transmission at the edge nodes, such as micro data centers, has been discussed by several works. In [45], a solution for network-edge data reduction for IoT devices is presented, without considering latency requirements of IoT applications and improvement of data quality by using different forecasting techniques. Paper [46] proposes a dynamic compression-based technique for sensor data. Works like [47], focus on data storage structure, memory allocation strategy, and data compression to improve storage capacity. According to Sensor-Cloud Infrastructure [18], it is possible to employ service innovation to accelerate decision-making. These challenges have been considered from traditional IoT and cloud perspectives [6], [12], but cannot be used due to edge limited storage.

Industrial frameworks. Collection and data analysis at the edge is the basis of industrial cloud platforms such as AWS IoT Greengrass (https://aws.amazon.com/greengrass/), which performs data storage on the cloud, rather than on the edge; Azure IoT Edge (https://azure.microsoft.com/en-us/services/iot-edge/) employs containers to package modules and custom logic at the edge. AWS IoT Analytics (https://docs.aws.amazon.com/iotanalytics/) offers remote device management, optimized IoT data storage, and time series analytics, enabling end-to-end workflow automation for large amounts of data and connecting IoT devices with cloud applications. Eclipse Kura (https://www.eclipse.org/kura/) represents a reference IoT Edge framework for building IoT gateways, incorporating networking protocols, and data services, allowing connectivity of IoT devices to their cloud platform. These approaches focus on fully managed workload services instead of edge data management on limited storage and adaptive data recovery with multiple techniques. However, they offer a possibility to incorporate custom logic for it, such as EDMFrame.

8 Conclusions and Future Work

Edge analytics faces us with the problems of making accurate and near real-time decisions based on limited and often incomplete data. Such issues require an efficient edge data management solution considering the impact on quality of decision-making processes for IoT systems. We introduce EDMFrame, a framework for edge data management featuring a mechanism for recovery of multiple gaps using both STR and PRM-based MTR recovery, and a novel approach for accurate decision-making using limited storage resources. Results show that EDMFrame is able (i) to recover gaps of various lengths by incorporating recovery cycles, achieving running time of $0.97s$ on average, while PRM-based recovery achieves up to 65.48% less error than with STR; (ii) to retain data points by 39.9% on average, while obtaining prediction accuracy around 98.83%. However, utilized methods such as ARIMA expect historical and regularly time-stamped data, and they are not often appropriate for some IoT streaming cases in which data generation and collection are triggered by certain events. Also, current PRM calculation requires a predefined number of consecutive gap lengths, posing obstacles in extreme cases of big gap lengths expected. In future work, we will extend EDMFrame addressing these limitations by investigating Recurrent Neural Networks for time series with irregular timestamps, and considering interpolation of ranges allowing PRM calculation for dynamic IoT cases. For future IoT services, it is crucial to make fast decisions as in smart cities requiring distributed ML at the edge, e.g., in the case of consistent ML models that must be updated when data streams evolve over time, posing critical issues to observe correct data at the right time [32]. Hence, we also plan to explore using EDMFrame for reliable distributed ML at the edge.

Acknowledgments

The work described in this article has been funded through Rucon project (Runtime Control in Multi Clouds), FWF Y 904 START-Programm 2015 and supported with Ivan Lujic's netidee scholarship by the Internet Foundation Austria. The authors would like to thank staff at TU Wien's Plus-Energy Office High-Rise Building, especially to Prof. Thomas Bednar and Alexander David on useful discussions and the Federal Real Estate Company (BIG) for providing data sources.

References

[1] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, 2011.

[2] S. Oueida, Y. Kotb, M. Aloqaily, Y. Jararweh, and T. Baker, "An edge computing based smart healthcare framework for resource management," *Sensors*, vol. 18, no. 12, p. 4307, 2018.

[3] F. Tao, Q. Qi, A. Liu, and A. Kusiak, "Data-driven smart manufacturing," *Journal of Manufacturing Systems*, vol. 48, pp. 157–169, 2018.

[4] 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.

[5] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by internet of things," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp. 81–93, 2014.

[6] B. P. Rao, P. Saluia, N. Sharma, A. Mittal, and S. V. Sharma, "Cloud computing for internet of things & sensing based applications," in *International Conference on Sensing Technology*. IEEE, 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," 2016.

[8] 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.

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

[10] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through fog micro datacenter," in *International conference on pervasive computing and communication workshops*. IEEE, 2015, pp. 105–110.

[11] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[12] C. C. Aggarwal, N. Ashish, and A. Sheth, "The internet of things: A survey from the data-centric perspective," in *Managing and mining sensor data*. Springer, 2013, pp. 383–428.

[13] 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.

[14] M. Al-Khafajiy, T. Baker, A. Waraich, D. Al-Jumeily, and A. Hussain, "Iot-fog optimal workload via fog offloading," in *International Conference on Utility and Cloud Computing Companion*. IEEE, 2018, pp. 359–364.

[15] N. Abbas, M. Asim, N. Tariq, T. Baker, and S. Abbas, "A mechanism for securing iot-enabled applications at the fog layer," *Journal of Sensor and Actuator Networks*, vol. 8, no. 1, p. 16, 2019.

[16] 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.

[17] 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 *EDBT*, 2017, pp. 330–341.
[18] M. Yuriyama, T. Kushida, and M. Itakura, "A new model of accelerating service innovation with sensor-cloud infrastructure," in *Annual SRII Global Conference*. IEEE, 2011, pp. 308–314.
[19] A. Papageorgiou, B. Cheng, and E. Kovacs, "Real-time data reduction at the network edge of internet-of-things systems," in *International Conference on Network and Service Management*. IEEE, 2015, pp. 284–291.
[20] A. Ukil, S. Bandyopadhyay, and A. Pal, "Iot data compression: Sensor-agnostic approach," in *Data Compression Conference (DCC), 2015*. IEEE, 2015, pp. 303–312.
[21] F. Van den Abeele, J. Hoebeke, I. Moerman, and P. Demeester, "Integration of heterogeneous devices and communication models via the cloud in the constrained internet of things," *International Journal of Distributed Sensor Networks*, vol. 11, no. 10, pp. 1–16, 2015.
[22] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: a survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161 – 175, 2018.
[23] I. Lujic, V. D. Maio, and I. Brandic, "Adaptive recovery of incomplete datasets for edge analytics," in *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, May 2018, pp. 1–10.
[24] ——, "Efficient edge storage management based on near real-time forecasts," in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, May 2017, pp. 21–30.
[25] N. R. Herbst, N. Huber, S. Kounev, and E. Amrehn, "Self-adaptive workload classification and forecasting for proactive resource provisioning," in *International Conference on Performance Engineering*. ACM, 2013, pp. 187–198.
[26] 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.
[27] 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.
[28] J. Lin, S. Williamson, K. Borne, and D. De Barr, *Pattern Recognition in Time Series, Advances in Machine Learning and Data Mining for Astronomy*. Chapman and Hall, 2012.
[29] X. Wang, K. Smith-Miles, and R. Hyndman, "Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series," *Neurocomputing*, vol. 72, no. 10-12, pp. 2581–2594, 2009.
[30] T. W. Liao, "Clustering of time series dataa survey," *Pattern recognition*, pp. 1857–1874, 2005.
[31] X. Wang, K. Smith, and R. Hyndman, "Characteristic-based clustering for time series data," *Data mining and knowledge Discovery*, vol. 13, no. 3, pp. 335–364, 2006.
[32] A. Aral and I. Brandic, "Consistency of the fittest: Towards dynamic staleness control for edge data analytics," in *Second International Workshop on Autonomic Solutions for Parallel and Distributed Data Stream Processing (Auto-DaSP)*. Springer, 2018, pp. 40–52.
[33] 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.
[34] "Umass trace repository," http://traces.cs.umass.edu/, 2017, [Online; accessed 22-September-2018].
[35] 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.
[36] R. Montella, M. Ruggieri, and S. Kosta, "A fast, secure, reliable, and resilient data transfer framework for pervasive iot applications," in *Conference on Computer Communications Workshops*. IEEE, 2018, pp. 710–715.
[37] T. Baker, M. Asim, Á. MacDermott, F. Iqbal, F. Kamoun, B. Shah, O. Alfandi, and M. Hammoudeh, "A secure fog-based platform for scada-based iot critical infrastructure," *Software: Practice and Experience*, 2019.
[38] Z. Maamar, T. Baker, M. Sellami, M. Asim, E. Ugljanin, and N. Faci, "Cloud vs edge: Who serves the internet-of-things better?" *Internet Technology Letters*, vol. 1, no. 5, p. e66, 2018.
[39] 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.
[40] 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.
[41] 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.
[42] 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.
[43] 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.
[44] A. Mehta, W. Trneberg, C. Klein, J. Tordsson, M. Kihl, and E. Elmroth, "How beneficial are intermediate layer data centers in mobile edge networks?" in *International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2016, pp. 222–229.
[45] A. Papageorgiou, B. Cheng, and E. Kovacs, "Real-time data reduction at the network edge of internet-of-things systems," in *International Conference on Network and Service Management*, 2015, pp. 284–291.
[46] A. Ukil, S. Bandyopadhyay, and A. Pal, "Iot data compression: Sensor-agnostic approach," in *Data Compression Conference*, 2015, pp. 303–312.
[47] Q. Yan and Y. Y. Wang, "A kind of efficient data archiving method for historical sensor data," in *ICISCE '16*, July 2016, pp. 44–48.

Ivan Lujic is a PhD candidate at the Institute of Information Systems Engineering of the Vienna University of Technology, Austria. In 2018 he received a scholarship by netidee, Austria's largest Internet funding initiative, by the Internet Foundation Austria (IPA). He received his master's degree in Computer Science in 2016 from the University of Split, Croatia. His main research interests are data management strategies for near real-time Cloud/Edge analytics.

Vincenzo De Maio received his PhD in 2016 at the University of Innsbruck, Austria. His research in the area of parallel and distributed systems comprises energy-aware Cloud computing and scheduling. Since 2017, he is a postdoctoral researcher at the Institute of Information Systems Engineering of the Vienna University of Technology. He authored different conference and journal publications on the topic of energy efficiency and modeling for Cloud and Edge computing.

Ivona Brandic is a Professor at Vienna University of Technology. In 2015 she was awarded FWF START prize, the highest Austrian award for young researchers. She received her PhD degree in 2007 from Vienna University of Technology. In 2011 she received the Distinguished Young Scientist Award from the Vienna University of Technology for her project on the Holistic Energy Efficient Hybrid Clouds. Her main research interests are cloud computing, large scale distributed systems, energy efficiency, QoS and autonomic computing.