Detecting Anomaly in Smart homes based on Mahalanobis Distance

Hoang M. Ngo*†, Do H. Ha*, Quan D. Pham*, Thinh V. Le*‡, and Son H. Nguyen*§
*Faculty of Information Technology, VNU-University of Engineering and Technology, Hanoi 100000, Vietnam
†Department of Computer and Information Science and Engineering, University of Florida, Gainesville, UF 32611, USA
‡Department of Computer Science, Kennesaw State University, Marietta, GA 30060, USA
§Corresponding author: sonnh@vnu.edu.vn

***Abstract*—The services provided in smart homes depends heavily on the operation of smart devices which may occasionally have abnormal behaviours due to hardware-failure or improper use. Hence, accurate and quick anomaly detection of devices in smart homes is essential. Due to the explosion of the number of smart devices in smart homes including sensors and actuators, a new anomaly detection method which can deal with a large number of data obtained from smart devices is necessary. Furthermore, the line between normal events and abnormal events is really sensitive in some cases, an accurate assessment method is required to reduce the false positive rate in detecting anomaly. In this paper, we propose an anomaly detection method based on Mahalanobis distance to completely solve the above problems. Based on an assumption that complex faults can be detected when actuators are triggered, we group devices which appear together frequently by a key actuator, and then calculate the Mahalanobis distances between states of these frequent groups by KNN models. We also propose a controlling algorithm to judge the failing proportion of devices in order to reduce the false positive rate. The experiment results show that our proposed methods can achieve high detection rates with low false positive rates and small detection time.** [1]

***Index Terms*—anomaly, smart home, KNN, frequent group.**

I. Introduction

The Internet of Things (IoT) has revolutionized residential living, leading to the emergence of smart homes, a technological evolution that interconnects home appliances and devices via the Internet, facilitating remote control, automation, and a heightened level of intelligent living. The movement towards the automated home began with the advent of Wireless Sensor Networks (WSN) and Radio frequency identification (RFID) devices [1]. Concurrently, advancements in material science and microchip technologies have led to the miniaturization and amplification of smart device capabilities [2]. These devices can be very sophisticated Raspberry Pi or a simple sensing device equipped with an integrated circuit and an antenna for communication. The extensive range of applications these devices provide residents includes, but is not limited to, the control of home appliances and the automation of various home features, collectively contributing to a more connected and efficient living environment.

For recent decades, one-person households (OPH) have become increasingly common across the world [3]. However, living alone, particularly for the elderly and individuals with disabilities, is associated with heightened risks of experiencing physical and mental health disorders, which can have severe consequences if not detected in time. Addressing this challenge necessitates the development of an advanced monitoring system capable of detecting critical and potentially dangerous activities such as falls or fainting. Smart home devices play a crucial role in this context, offering a viable solution to mitigate these risks. In addition, the failure of smart devices in smart homes may result in an inconvenient and insecure living environment. For instance, the failure of a ventilation machine may exert negative impacts on asthma patients. As a result, besides detecting abnormal activities of home users, another duty of the said system is to detect abnormal behaviors of smart devices in order to replace or fix them in a timely manner. In general, we will refer to both abnormal activities of home users and abnormal behaviors of smart devices collectively as abnormal events.

Along with the population of smart homes, an increasing amount of research for anomaly detection in smart homes has been conducted [4], [5], [6]. In [7], the authors used Bayesian Belief Networks to calculate the probability that a sensor receives a certain value, through data of nearby sensors and historical data of the sensor itself. In another study [8], the sensor network is segmented into defined regions, each collecting a distribution of multidimensional data points. These data points are gathered using a window sliding technique and analyzed through the Kernel Density Estimators (KDE) method. S.Rost et al. [9] proposed a solution for detecting errors in sensor networks by having sensors monitor each other's periodically transmitted packets and send the collected information to each other. In [10], the authors proposed DICE, which consists of two phases: the pre-computing phase and the real-time phase. In the pre-computing phase, binary state sets are formed by extracting states of devices in length-fixed time windows and transition probabilities between state sets are computed. In the real-time phase, faulty devices can be detected by checking binary state sets and transition probabilities in the past. However, there are some drawbacks to this method. First of all, when extracting devices' states in time windows into state sets, numeric states have to be converted

[1]This research has been done under the research project QG.21.30 "Anomaly detection for IoT devices in smart home environment" of Vietnam National University, Hanoi.

to binary forms, so part of the information represented by numeric states may be lost. Secondly, because state sets consist of every device state in a house, so there is no dependency between some dimensions in state sets. As a result, the noise from independent dimensions in state sets can cause a high false positive rate when these state sets are utilized to identify anomalies.

In this paper, our proposed solution can deal completely with these above problems. The contributions of this paper are summarized as follows.

- We propose the novel concept of the *Frequent ACT Group*, consisting of a key actuator and devices that are frequently triggered simultaneously with the key actuator. This concept aims to group devices with a strong correlation, which is an important factor in detecting anomalies.
- We apply unsupervised learning model KNN with the Mahalanobis metric to learn historical data points of frequent groups.
- We propose a controlling algorithm to decide whether a state of a frequent group is abnormal or not.
- We implement and test our solution with real-world datasets to compare the results with baseline methods.

Organization. The rest of this paper is organized as follows. In §II, we introduce kinds of anomaly in smart homes and classify them. §III illustrates our proposed approach to detect anomaly. Following that, we give the evaluation of the results of experiments in §IV. §V concludes and highlights our contribution and mentions the future works.

II. Preliminaries

In this section, we illustrate the definition of abnormal behaviors and anomalies in the context of smart homes. Then, we state the anomaly detection problem in smart homes.

A. Anomalies classification in smart home dataset

When considering smart homes' environment, there are numerous factors that are involved. These factors can be intrinsic factors such as homeowners, smart devices, or extrinsic factors such as temperature and humidity. As a result, any change in the state of a factor can lead to a shift in the state of the smart home. States that occurred frequently in the past are considered as *normal states*, while states that never or rarely occurred are *abnormal states*. Some obvious abnormal behaviors can be perceived by human senses. However, homeowners cannot monitor their smart homes' environment continuously, nor can they detect all abnormal behaviors. Therefore, IoT systems have been developed to monitor and detect abnormal behaviors for homeowners. Formally, we define an anomaly in a smart home's environment at a time t as a set of recorded states from one or many smart devices that is inconsistent with data points obtained before the time t and persists for a sufficiently long duration. With the above definition, we categorize common types of anomaly as follows:

Interference. This refers to anomalies caused by outside factors. For example, some tapes or dirt accidentally cover the light sensors so they report values that are lower than usual, or in the case of a fire hazard, temperature sensors will report values much higher than average.

Location. In some cases, sensors can be moved to the wrong location or motion sensors just fire off without a user's presence causing a location anomaly. During sensor maintenance where the user is forced to uninstall the device and reinstall it again later, they can potentially install it in the wrong location. Another case is in a house that does not have regular clean-up, spider webs can cover the motion sensor causing it to go off constantly even though nobody is around. All of these will generate an abnormal location pattern in the reported data.

Stuck at. The sensor itself can have some problems either with its embedded software being too outdated or its hardware getting damaged which can lead to readings that just fluctuate around a certain value over a long period of time. Other devices that rely on this sensor will also behave abnormally as a result.

Malfunction. Sensors are not the only devices that can get faulty. These can include the electronic devices in our home as well, which is what this type of anomaly is about. Air conditioners or fans can get broken down leading to an abnormally low temperature reading even though the actuator controlling these devices has turned on. The same kind of scenario can happen to other sensors such as light sensors, humidity sensors, etc.

B. Anomaly classification in smart home dataset

Detecting abnormal behaviors automatically is crucial in protecting and facilitating homeowners in their own houses. As we mentioned in the previous part, abnormal behaviors may cause anomalies in the data set collected from smart devices. Therefore, detecting an abnormal behavior is equivalent to detecting its corresponding anomaly in the data set. Formally, we define the anomaly detection problem as follows:

Definition 2.1: **(Anomaly detection)** Consider the environment of a smart home, consisting of a set of smart devices D. Let $P = \{(t_i, d_i, v_i) \mid t_i \in \mathbb{R}, d_i \in D, v_i \in \mathbb{R}^*\}$ denote the collected data set, where t_i indicates the time at which the data point was collected, d_i indicates the device reporting the data point, and v_i indicates the state (value) of device d_i at time t_i. Given that there are abnormal behaviors occurring non-simultaneously in the environment, we need to detect anomalies corresponding to these abnormal behaviors in the data set P.

III. Method

Here, we present our proposed method to detect anomalies. Our method consists of four steps: Classifying, Grouping, Learning and Detecting.

A. Classifying devices and splitting dataset

Based on the function of devices in smart homes, we categorize these devices into two main groups: actuators and sensors. Actuators operate and send data to home gateways only when triggered. For example, doors or light switches send their states of 'ON' and 'OFF' to gateways only when they are activated by the user. In contrast, sensors periodically send data to home gateways. For instance, temperature sensors and humidity sensors report real-time temperature and humidity levels to the gateway every 5 minutes. Changes in the states of devices over time are recorded in the dataset in chronological order. In addition, we segment the entire dataset into time windows with an equal size. Each time window contains changes in the states of devices over a fixed duration.

B. Grouping devices

In smart homes, user activities usually trigger multiple devices simultaneously. Different activities lead to the activation of different sets of devices. Thus, in the dataset, certain sets of devices will appear more frequently in the same time windows than others. We refer to such sets of devices as *frequent groups*. More specifically, given a threshold ω, we provide the definition of frequent groups as follows:

Definition 3.1: Frequent groups are collections containing devices in a smart home such that these devices report the change of their states simultaneously at a rate no smaller than a given threshold ω.

Based on Definition 3.1, devices on the same frequent groups are supposed to have strong correlations to each other. However, in smart homes, sensors, which monitor properties such as temperature or humidity, are very sensitive to environmental changes and will frequently fire events reporting those changes. Thus, there is a case that sensors, despite lacking a meaningful correlation, report state changes within the same time windows. This defeats the intended purpose of the frequent groups. On the other hand, in smart homes, the frequency of reporting states' change from actuators is typically low, as these devices are triggered only when activated by the user's actions. In addition, activation or deactivation of actuators usually triggers state changes for specific devices that closely relate to it. Hence, in order to mitigate the impact of high-frequency state changes in sensors, we introduce a novel concept of *frequent ACT groups* defined as follows:

Definition 3.2: Frequent ACT groups are frequent groups that associate with a designated key actuator.

Frequent ACT groups can be discovered by association rule mining algorithms such as FP-Max or FP-Growth [11]. For each frequent ACT group, we construct a collection of states of devices in the group in observable time windows, denoted as ACT points. In the case that there is a device missing in a time window, we assume that the state of the device do not change in that time window and replace the missing value of that device in the ACT point by its most recent state. As a result, each frequent ACT group is associated with a set of ACT points.

C. Learning

In order to determine whether a data point observed from a frequent ACT group is abnormal or normal, we base it on the Mahalanobis distance between the ACT points and their k neighbors in the same set. Although there are other distance metrics such as Euclidean, Manhattan, Cosine, and Canberra,... most of them do not consider the distribution of data like Mahalanobis. Since features of vectors in the space that represent frequent ACT groups have a strong correlation or in other words, they tend to form groups, we would like to capture this characteristic. Compared to other metrics, Mahalanobis metric is superior in detecting anomalies in ACT groups because Mahalanobis metric also measures the correlation between features in ACT points. For reference, the *Mahalanobis distance* between two ACT points $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ is given by $\Delta^2 = (\boldsymbol{x}^i - \boldsymbol{x}^j)^\top \Sigma^{-1} (\boldsymbol{x}^i - \boldsymbol{x}^j)$, where Σ is a $d \times d$ covariance matrix.

We construct a set of KNN models using the Mahalanobis metric associated with frequent ACT groups. More specifically, for a given frequent ACT group G, the corresponding KNN model is trained using a set of ACT points from G collected in the historical dataset. Subsequently, when presented with a new ACT point, the trained KNN model can determine the nearest Mahalanobis distance between the new point and the points in the training set. This distance is used to determine whether a state corresponding to a frequent ACT group in a time window is potentially abnormal or not.

D. Detecting

Here, we propose the detecting algorithm (Algorithm 1) to detect the device that causes the anomaly and the time at which it happens. Our algorithm takes the collected dataset P, the set of frequent ACT groups $\mathcal{A}$, the set of trained KNN models corresponding to these groups $\mathcal{K}$, the starting time T_{start}, the ending time T_{end}, the time window length t_w, the distance threshold γ and the anomaly threshold ψ as inputs. The algorithm returns the abnormal device $\bar{d}$ and the detecting time $\bar{t}$ as outputs.

For the first step, Algorithm 1 initializes anomaly scores for frequent ACT groups (line 1). These scores are used to decide whether a group is abnormal or not. Then, we repeatedly consider the time slot t in range of (T_{start}, T_{end}) (line 2). For each t, we can construct a time window from $t - t_w$ to t, and the set of events happened during that time window (line 3). In each time window, we consider each frequent ACT group $G \in \mathcal{G}$ by forming an ACT point S corresponding to the group (line 4-5). Then, after defining the KNN model corresponding to the group G as $K^{(G)}$, we calculate the Mahalanobis distance l from the ACT point S to the model $K^{(G)}$ (line 6-7). If the distance l exceeds the distance threshold γ, we increase the anomaly score for the group G (line 8-9). In case the score of G exceeds the anomaly threshold ψ, the time window from $t - t_w$ to t is considered an abnormal time window. Next, we find the abnormal device in this time window (line 10-21). Specifically, in order to find this device, we examine each device $d \in G$ and try to replace the current value of d in S with another value v' that occurred in the past (line 12-15). If the replacement of the value of d eliminates the anomaly, we detect d as the abnormal device (line 16-21). The total runtime of Algorithm 1 is heavily dependent on the number of frequent ACT groups, and the maximum size of frequent ACT groups. Because the number of actuators in smart homes,

```
Algorithm 1: Detecting algorithm
Input: The collected dataset
       $P = \{(t, d, v) \mid t \in \mathbb{R}, d \in D, v \in \mathbb{R}^*\}$, the set
       of frequent ACT groups $\mathcal{G}$, the set of trained
       KNN models $\mathcal{K}$, the starting time $T_{start}$, the
       ending time $T_{end}$, the time window length $t_w$,
       the distance threshold $\gamma$, and the anomaly
       threshold $\psi$
Output: The abnormal device $\bar{d}$, and detecting time $\bar{t}$
1  Initialize $c : G \to \mathcal{R}$ as the anomaly scores with
     $c[G] = 0 \forall G \in \mathcal{G}$
2  for $t = T_{start}; t \leq T_{end}; t = t + t_w$ do
3      Let $P' = \{p = (t, d, v) | p \in P, t - t_w <= p.t <= t\}$
         as the set of events happened in the time window
         from $t - t_w$ to $t$.
4      for $G$ in $\mathcal{G}$ do
5          Let $S$ as the ACT point constructed by the set
             of datapoints $\{p.v | p \in P', p.d \in G\}$.
6          Let $K^{(G)} \in \mathcal{K}$ as the KNN model
             corresponding to the group $G$.
7          $l \leftarrow K^{(G)}(S)$
8          if $l > \gamma$ then
9              $c[G] \leftarrow c[G] + 1$
10         if $c[G] > \psi$ then
11             for $d$ in $G$ do
12                 Let $V^{(d)}$ as the set of values of device
                     $d$ that happen frequently in the history.
13                 $is\_anomaly \leftarrow False$
14                 for $v' \in V^{(d)}$ do
15                     Let $S'$ as the ACT point obtained
                         from $S$ by replacing the value of
                         device $d$ in $S$ to $v'$.
16                     $l' \leftarrow K^{(G)}(S')$
17                     if $l' \leq \gamma$ then
18                         $is\_anomaly \leftarrow True$
19                         break
20                 if $is\_anomaly$ then
21                     return $\bar{d} = d, \bar{t} = t$
```

which corresponds to the number of frequent ACT groups, is usually small, our detecting algorithm can be efficient in practice.

IV. Experiment and Evaluation

In this section, we aim to evaluate the efficiency of our proposed method. First, we explain the setting for our experiments. Then, we present the experimental results.

A. Setting

1) Datasets: Our method is evaluated on five real-world public datasets namely hh102, hh106, hh108, hh113 and hh114 from the CASAS smart home projects[12]. All of these came from the same data group titled Human Activity Recognition from Continuous Ambient Sensor Data and were collected in multiple smart home testbeds hosted at Washington State University in 2021, so it is relatively new compared to other data from CASAS. Each smart home in these datasets was instrumented with various types of sensors. The layout of these devices can be checked in the sensor map that came with each dataset like Figure 1. Sensor data are collected continuously while residents perform their normal routines. Each dataset with an average of 115 devices generates around 3 to 12 million sensor events. A congested environment like this is the reason why we chose these specific datasets. It will help us explore a wide range of closely related device groups, thereby demonstrating the capabilities of the association rule mining algorithm.

Dataset	Sensors	Actuators	Events
hh102	95	10	6472396
hh106	155	23	6364167
hh108	138	18	12550848
hh113	95	16	3213217
hh114	92	16	11688878

TABLE I
Number of devices and events in each dataset

2) Methodology and Measurement: The goal of the experiment is to assess the effectiveness of detecting anomalies mentioned in section II-A. This assessment will be based on three measurements: *precision* - the fraction of actual noises among the detected abnormal events, *recall* - the percentage of injected abnormal events being detected and *det_time* - the time distance between the timestamp when anomalous events start appearing and the first time an abnormal event is detected.

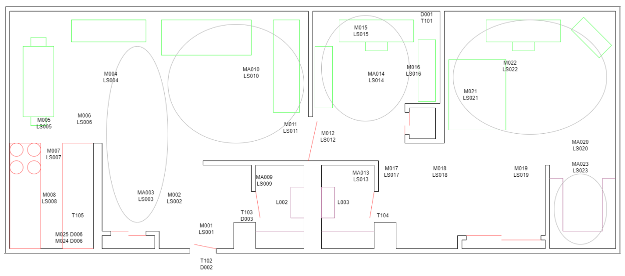

Fig. 1. Smart home sensor map in the hh102 dataset.

Since there are little to no anomalies in the original CASAS datasets, the time window that we can test our detection method is very limited. Thus, to broaden it, we will generate and insert faulty events into the dataset. About 15 weeks of data that is independent of the training data will be extracted from each original dataset to act as a base for test data. With each type of anomaly, we specify a context or test scenario that is suitable with the current dataset and simulate it with the aforementioned anomalous events:

Interference. This anomaly can be represented in many ways but in our experiment, we would like to simulate an attack from a third party that tampered with the sensor data

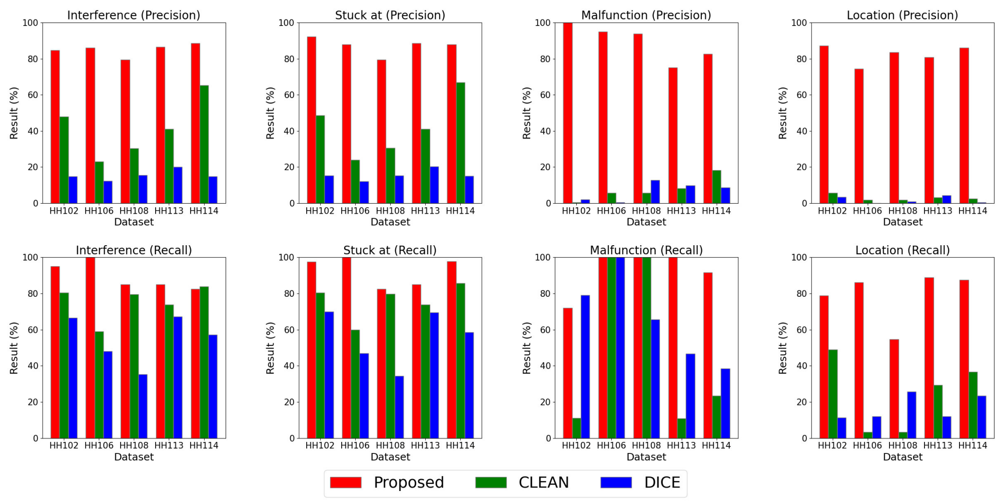

Fig. 2. Anomaly detection results collected after running through all 200 test data. In each graph, the y-axis is the collected result in percentage, the x-axis is the dataset used. The first graph row is the precision while the second one is the recall. Each graph column represents an anomaly type that we want to detect.

because in our opinion, this is the most dangerous scenario that can happen. After selecting a set of faulty devices, we perform some statistics methods to identify the value range that these devices usually report. We then select a value that is completely outside of this range and insert it into every event that belongs to the abnormal devices.

Location. We generate these anomalies by selecting pairs of motion sensors that belong in different rooms and have their events fire simultaneously.

Stuck at. We simulate this by selecting a set of sensor devices and setting a stuck-at value for each of them. Then we insert multiple events that belong to these sensors with values close to the stuck-at value into the dataset. These events will also be inserted at a randomized time frequency to see how our method performs under such uncertainty.

Malfunction. Since there is little information about the electronic devices in the dataset we are using, generating this anomaly can be quite limited. One way we achieve this is by assuming some light bulbs in the smart home get damaged, which leads to abnormal light sensor values when the actuators or light switches get turned on. So we first need to know which light sensors are directly affected by a light switch, this can easily be done by checking the sensor map like Figure 1 that came with each dataset. Afterward, we just need to change their value to a significantly lower one compared to when the light switch is on.

All of these simulation methods are based on our observation of real-life anomalies. The value of the anomalous events as well as their placement inside the original dataset are selected so that they can reflect reality as much as possible. A total of 200 test data files are generated with 10 for each pair of anomaly type and dataset. Aside from evaluating our method, we will also run two other anomaly detection methods that also use a heterogeneous approach, CLEAN[13] and DICE[10] on these test data and compare their effectiveness to our own. Both DICE and our method require a learning phase so to ensure fairness, both will be receiving 6 months' worth of data from each original dataset as training data. The third measurement *det_time* will be calculated in a separate experiment mainly because we notice that the time taken to detect anomaly depends on how frequently the actuator event is fired. So to assess *det_time*, we generate a different set of malfunction anomalies test data where we manually insert actuator events as well as abnormal sensor events related to it at different time frequencies into the data set.

B. Anomaly Detection Test Results

After running every method on each test data, we calculate the average precision and recall. As shown in Figure 2, both measures produced by our method are consistently high across all test data with most of them reaching above 80%, outperforming CLEAN and DICE. This shows that the frequent ACT groups our method learned do indeed contain devices that are closely related to each other and that the algorithm can detect abnormal data points. In terms of precision, in some test data, such as location and malfunction anomalies, the two methods that we want to compare barely detect anything. This may contradict the overall better results collected in their original work. After inspecting the generated results, we notice that these numbers are not caused by the lack of detection but rather an overabundance of false positives, evidence in the high recall and low precision in most test cases.

For CLEAN, even though detecting location anomalies is its strong point thanks to its hierarchy-based algorithm, it still produces low precision. This is due to the fact that CLEAN can not detect "missing" data. In our experiments, specifically the location anomalies, this "missing" data refers to events belong to a motion sensor that has been moved to a different area so

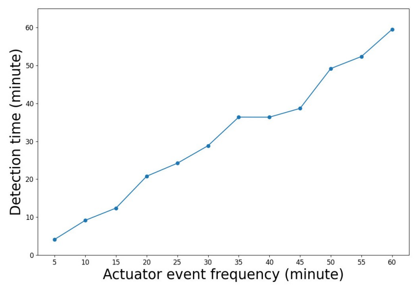

Fig. 3. Detection time results. This graph showcases the relationship between how frequently the actuator is used and the speed at which our method can detect anomalies, both properties are measured in minutes.

it does not fire with the rest of the devices in its old area. In other words, CLEAN can only detect location anomalies if it shows up in an area that it is not supposed to, not when it is missing from its original area. When trying to detect the latter, which occurs quite frequently in the test data, CLEAN will produce a great number of false positives, lowering the overall precision. This is not the case for our method since whether a device is missing or not adds to the penalty score during the detection step. Thus, it does not suffer from the same weakness as CLEAN.

In DICE's case, the algorithm resolves around how the sensors' state changes over time, which in order to be effective requires it to learn a lot of transactions. So when the training data is too short, there is a high chance that most of the state changes in the testing data have not been learned yet and will be classified as anomalies. In our experiments, with the same training data, our method outperforms DICE in most test data since we do not rely on transactions or state changes but rather on whether the current data point deviates greatly or not from past data points. Clearly, from the test results, it does not require as much training data as the prior.

Figure 3 presents the results of our detection time experiments. In this graph, the x-axis can be interpreted as the frequency at which we inserted the actuator events for each test case, the y-axis is the *det_time*. Although it is not linear, there is a clear trend that if the actuator device is used less frequently, it will take more time for our method to detect anomalies related to that device. This can become a vulnerability if sensors that are in the same frequent group as these actuators become faulty as they will not be detected as easily as other devices. But like these actuators, they rarely affect the user's daily activities, so this behavior is acceptable. Other than that, the detection time is quite reasonable as it is mostly lower than the actuator events frequency. In these cases, our method detects almost immediately when the user interacts with the actuator.

V. Conclusion and Future Work

In this paper, we proposed an anomaly detection method that is based on a heterogeneous approach. It first requires a learning phase that uses an association rule mining algorithm to learn closely related device groups known as frequent ACT groups. Afterward, it trains KNN models based on these groups's data points and Mahalanobis distance to detect abnormal device states. We tested our method's effectiveness on five real-world data sets with four different scenarios in which anomalies can occur. The experiments showed promising results in detecting both binary and non-binary sensor anomalies as well as performing under a reasonable amount of training data. It also demonstrated that it can overcome some of its predecessors' weaknesses such as CLEAN inability to detect missing data. Currently, our method can work well with devices that are in frequent ACT groups. But in some cases, a device that does not belong to any frequent groups can exist. This happens due to the fact that in every smart home or home in general, there are always some locations in the house that the user rarely visits or is visited less frequently than others. Our future work will tackle this problem either by improving the rule-mining algorithm or integrating a different type of data structure such as a hierarchy tree or a correlation matrix in order to better visualize the relationship between sensors.

References

[1] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: Challenges and solutions," *Journal of cleaner production*, vol. 140, pp. 1454–1464, 2017.

[2] S. S. I. Samuel, "A review of connectivity challenges in iot-smart home," in *2016 3rd MEC International conference on big data and smart city (ICBDSC)*. IEEE, 2016, pp. 1–4.

[3] J. B. . T. N. Desk, "New census data reveals nearly 30are single occupancy." [Online]. Available: https://shorturl.at/iHJN4

[4] M. Yamauchi, Y. Ohsita, M. Murata, K. Ueda, and Y. Kato, "Anomaly detection in smart home operation from user behaviors and home conditions," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 2, pp. 183–192, 2020.

[5] L. G. Fahad and M. Rajarajan, "Anomalies detection in smart-home activities," in *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*. IEEE, 2015, pp. 419–422.

[6] S. Ramapatruni, S. N. Narayanan, S. Mittal, A. Joshi, and K. Joshi, "Anomaly detection models for smart home security," in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. IEEE, 2019, pp. 19–24.

[7] D. Janakiram, A. Kumar, and A. M. Reddy V., "Outlier detection in wireless sensor networks using bayesian belief networks," in *2006 1st International Conference on Communication Systems Software Middleware*, 2006, pp. 1–6.

[8] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, "Online outlier detection in sensor data using non-parametric models," in *Proceedings of the 32nd International Conference on Very Large Data Bases*, ser. VLDB '06. VLDB Endowment, 2006, p. 187–198.

[9] S. Rost and H. Balakrishnan, "Memento: A health monitoring system for wireless sensor networks," in *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, vol. 2, 2006, pp. 575–584.

[10] J. Choi, H. Jeoung, J. Kim, Y. Ko, W. Jung, H. Kim, and J. Kim, "Detecting and identifying faulty iot devices in smart home with context extraction," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2018, pp. 610–621.

[11] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang, "Pfp: Parallel fp-growth for query recommendation," in *Proceedings of the 2008 ACM Conference on Recommender Systems*, ser. RecSys '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 107–114. [Online]. Available: https://doi.org/10.1145/1454008.1454027

[12] "MS Windows NT kernel description," https://casas.wsu.edu/datasets/.

[13] J. Ye, G. Stevenson, and S. Dobson, "Fault detection for binary sensors in smart home environments," in *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2015, pp. 20–28.