An Outlier Detection Method to Improve Gathered Datasets for Network Behavior Analysis in IoT

Amin Shahraki and Øystein Haugen
Faculy of Computer Science, Østfold University College, Halden 1783, Norway
Email: {amin.shahraki, oystein.haugen}@hiof.no

Abstract —Outlier detection is a subfield of data mining to determine data points that notably deviate from the rest of a dataset. Their deviation can indicate that these data points are generated by errors and should therefore be removed or repaired. There are many reasons for outliers in a network dataset such as human or instrument errors, noise or system behavior changes. On the other side, Network Behavior Analysis (NBA) is a way to monitor traffic and recognize unusual actions in a network. Analyzing data trends in NBA methods is a common way to interpret network situation. Outliers can deviate and produce erroneous trends that influence the results of the NBA methods. This paper presents an approach that based on a method for trend detection divides the data set into subsets where contextual outliers are discovered. The outliers can then be removed to have a clear dataset that better shows the network behavior when using NBA methods. Increasing the accuracy and reliability are the goals of our method. We compare the proposed method with the Hampel method on simulated IoT network data.

Index Terms—Contextual outlier detection, trend change analysis, network behavior analysis, poisson distribution datasets, internet of things

I. Introduction

Network Behavior Analysis(NBA) has always been a significant subject to monitor the network [1]. It tries to discover unusual actions as operational problems. To determine the network behavior, data should be gathered from the network and useful information should be extracted. Although deep packet analyzing is a fruitful method to monitor the network, it is a type of privacy violation [2]. Data streams can be manipulated by different external and internal agents such as security attacks, noise, mobile nodes, etc. The Poisson distribution is one of the most common behavior models of data streams in computer networks [3]. In general, Poisson distribution has been known to be characteristic for arrival times of packets in computer networks for many years. Internet of Things (IoT) [4] as an important new generation of computer networks generates different types of data streams like Poisson based data streams. There are a lot of applications in IoT which can generate Poisson distribution datasets [1], [5]-[7]. Generally, Poisson distribution is one of the most common methods to explain data generation in different applications of IoT. As an important type of data stream, analyzing Poisson distribution datasets can help to recognize network situation. Even though the network behavior in general can be modeled by a Poisson distribution, the datasets will include several small non-periodic jitters and short-term insignificant trend changes that make them hard to analyze. Each time series dataset includes a sequence of trends. A trend is a pattern of gradual change in a time series dataset that follows the same pattern for a while. A pattern can be uptrend, downtrend or sideways. As a result, the proposed method tries to detect outliers in Poisson distribution-based datasets which assumed to represent IoT networks. By removing the outliers, the dataset will be more suitable to use as the input of NBA methods. In other words, the method is used as a preprocessing step of analyzing a dataset to determine the network behavior.

NBA methods often find trends that can explain the overall behavior at that specific time. The method is used to remove outliers by finding the trend and detecting outliers in each one. To assess the network behavior by a dataset, our approach uses two steps. The first step is recognizing the boundary of predominant trends to divide a big dataset into smaller datasets with disjunct predominant trends. A predominant trend is a pattern that the data of a trend period corresponds to. A trend period is a part of a time-series dataset that follows a same pattern. The second step is determining the behavior of each subset. Between the first and second step, detecting outlier data points can help to improve the second step. An outlier must be a data point that is caused by special underlying reasons and not only by chance. By removing or rectifying outliers, the second step can extract more significant and accurate information. The result of removing outliers would be a dataset that is clear and easier to analyze and visualize. To detect outliers, the dataset must be processed and this requires processing resources. In many cases such as Wireless Sensor Networks [8] and IoT [4], there is a scarcity of resources to allocate. Consequently, methods consuming little processing power are needed. In our proposed method, we use LSTCP [9] to divide Poisson distribution based datasets into several subsets with different predominant trends. After dividing, each trend period will be considered as a subset to detect contextual outliers.

Manuscript received September 10, 2018; revised April 25, 2019.
Corresponding author email: amin.shahraki@hiof.no
doi:10.12720/jcm.14.6.455-462

Contextual outliers are data elements that deviate from the rest of the dataset in a specific context. In other words, they are found to deviate from the other data of the context subset but may not be recognized as outliers when seeing the whole dataset. Separating trends in a dataset would make it easy to find outliers in each subset. In a subset, each data point will be evaluated and get a score to classify as an outlier or not. By having some small subset rather than a big one, time complexity would be lower. Also, the accuracy will be improved based on analyzing contexts rather than the whole dataset. There are a lot of factors in IoT that can provide information about the network such as delay, speed, input rate, etc. Based on preparing data to monitor IoT networks, we use One-Way Delay (OWD) to recognize abnormal situations [10], [11]. In section II, some related methods will be introduced. The proposed method will be explained in section III and will be evaluated in section IV. Section V includes the future works and conclusion.

II. Related Work

During the last decade, data assimilation and their processing methods have been changed [12]. Now, data streams are one of the most important datasets to process. There are a lot of agents that generate data streams like MANET based IoT or Wireless Sensor Networks (WSN) [13]. Generally, in a computer network, a data stream is a sequence of data or packets that are transferred in a period of time [14]. There are a lot of applications that need to process data stream such as data mining [15] optimization [16] resource and storage management [17], [18] big data [19]. Outlier detection is a helpful method to provide a clear and processable data stream. Guta et al.[20], Thakkar et al. [21] recently provide extensive reviews of outlier detection methods. In [22], Knox and Ng introduce an outlier detection method for data streams which is based on distances of data points. Kuncheva et al. [23] use a sliding window to capture subsets of data points and evaluate them to find outliers. In [24], authors use a regression technique based on the distance between each data point and predicted data points. Also some methods such as [25] use clustering methods like k-means [26] to detect outliers. Some methods like [27] use data mining and static based methods. They use the density of data to approximate data distribution and find regions that have low density as outliers. In [28] Rebbapragada et al. use k-means clustering algorithm to find anomalies in periodic time series. Eamonn Keogh *et al.* [29] Introduce a technique that determines a pattern surprising in a time series database, if the frequency of these patterns differs from the expected by chance. In [30] a method has been introduced that uses the median to detect unusual values from time series data which are unable to be modeled. Their use case is a sensor data on an airplane. Also, in [31] Hill and Minsker present an autoregressive data-driven model that is used for environmental data streams in heterogeneous sensor networks to find outliers. DJ Hill *et al.* in [32] use Dynamic Bayesian Networks to determine the anomalies and outliers in real time datasets which are generated by WSNs. In [33], authors use a similarity measure based method to determine the degree of change within a network topology which is based on time series dataset. Also, authors in [34] represent median graph concept and analyze large data networks by using that to determine abnormal network change detection. Hampel method which is introduced in [35] uses to determine outliers in a time series based on median and standard deviation. We use the Hampel method to compare and evaluate the proposed method as a fundamental outlier detection method for time series.

III. Proposed Method

There are a lot of circumstances that outlier detection methods can help improve the accuracy of different applications. One of the most important outlier detection applications is computer network monitoring. There are different parameters to evaluate the network situation such as delay, energy consumption, jitter, etc. Delay is a factor that can show the network behavior, especially in ad hoc networks. Gathering data about the delay in a network can help to recognize any abnormal situation. The proposed method uses *OWD* as the main parameter to be analyzed. To find any disturber, the overall network behavior should be recognized. One way is to compare current network behavior with previous situations to determine significant changes. Consequently, gathered dataset should divide to subsets with different behaviors. Each trend period should analyze to recognize the behavior and extract useful information. In other words, there are two steps. First dividing dataset, second, analyzing each trend period to determine the behavior. In second step data should be explicit to extract accurate information especially in applications which need to be very accurate data like predictions or real-time applications. The proposed method can help to prepare each trend period to analyze before the second step. There are some data points that are boundaries of different subsets called change points (CP). Determining the change points to distinguish subsets is the goal of step one. Also, by considering that each trend period follows the same pattern, analyzing the dataset in step two has better time complexity in terms of resource efficiency. Each node in IoT networks can help other nodes as a hop to transfer data. Each node has a queue that schedules the packets to process. There are different types of queue processing methods. *M/M/1* [36] is one of the most common and prevalent processing methods in computer networks. The input rate of the model is based on Poisson distribution. *M/M/1* is used to represent the memoryless occurrences of events which randomly appear with a specific rate. As a consequence of the Poisson distribution, the distribution of the intervals between the events is modeled by the negative exponential distribution. To write the Poisson distribution based on the negative exponential distribution we have (1).

$$f(x) = \Pr[T > t] = \Pr[X = 0] = \left(\frac{(\partial t)^0}{0!}\right) e^{-\delta t} = e^{-\delta t} \qquad (1)$$

If arrival rate is *t*, and *T* is the time of a random event, consequently F(t) will be $1 - e^{-\delta t}$ which is negative exponential distribution. *D* refers to the one-way delay between source and destination of a packet. Eq. (2) is based on calculating *D* in a MANET-IoT multi-hop network.

$$rd_D = \sum_{s=1}^{N} (rd_{transmission} + rd_{propagation} + rd_{process} + rd_{queueing})_s \qquad (2)$$

N is number of hops between source and destination and "*rd*" is rad raw-data delay. Eq. (2) is used in different Multi-hop networks such as MANET-IoT networks. In some applications that use data streams such as multimedia, datasets which are used to monitor network is so big. Sampling methods are used to reduce the amount of data which is so effective in NBA methods and their performance. Eq. (3) is used for sampling of raw-data. *r* is a parameter to show the sampling rate.

$$x_n = \sum_{n-r}^{n} rd \Big/ r \qquad (3)$$

Based on the proposed method, after gathering a dataset, it should be divided into new subsets with different trends. To make it simple, we find the last CP in a dataset. After that, the dataset will be divided into two new subsets. From beginning to last-determined CP, the dataset will be evaluated again0 by a recursive method until dataset is finished. In a Poisson based dataset, many small jitters disguise main trends. Using smoothing method can help to reduce the effect of small jitters on predominant trend in each context. Although smoothing method will remove some contextual outliers accidentally, after the first step the unsmoothed dataset will be used again to determine outliers. In other words, smoothing method is just used for dividing procedure, not outlier detection. Running median is an efficient method to smooth Poisson based datasets. By assuming $\{x_1, x_2 \ldots\ldots x_m\}$ is the raw dataset after sampling, x_i is replaced by d_i as a smoothed data based on (4) with a windows size equal to $2*k+1$.

$$d_i = median[x_{i-k}, x_{i-k+1}, \ldots, x_{i+k}, x_{i+k}] \qquad (4)$$

The proposed method is able to find the last predominant trend by connecting each data point to previous one, creating a new dataset and giving a weight to the finite difference of the new dataset. The finite difference is used to detrend the dataset by (5)

$$w_j = d_{j+1} - d_j \qquad (j = 1, \ldots, n-1) \qquad (5)$$

As a method to find last CP, it needs to evaluate each trend period to check it contains the last CP or not. We need to connect each part of data to other parts to compare. For comparing a matrix model is used. Giving a weight to each part of data can help to compare dataset from the beginning to end of the connected dataset based on changing the importance of each part. When weights give to a first data points of the dataset, they will be more important than the end of the dataset. Matrix "*A*" is used to give weight. Also, in each row of the matrix, we use 1 percent more from the beginning of dataset to connect to each other and giving weight. As an example, in the 10th row, 10 percent of data is connected to each other from the beginning. Changing the amount of data in each row can balance the accuracy and resource consumption. If we consider every 5 percent of data in each row, we have 20 rows totally, but the location of the CH is in a part of the dataset which is five times bigger than the 1 percent that has no same accuracy. There are 2 levels in determining the location of last predominant CP. level 1 is based on detrending and giving weight. 2nd level is based removing the effect of negative and first-order finite difference (detrended data). Matrix "*A*" is used to apply level 1 based on (6).

$$A = \left[W1 \middle| (w_j - a_{i(j-1)})\lambda_i\right]_{m\times(n-1)} \quad i = 1, \ldots, m \quad j = 2, \ldots, n-1 \qquad (6)$$

$\lambda_i \in [0,1]$ is used to get weight by ith row based on (7).

$$\lambda_i = \frac{i}{m} \qquad (i = 1, 2, \ldots, m) \qquad (7)$$

When λ is 1, the importance of all data points is equal. When λ is approaching to 0, the importance of data points from the beginning of dataset will be more than the tail. Also, W1 is constant. The reason is that there is no previous data to subtract from a_{i1}. Eq. (8) is used to explain how (6) is calculating matrix "A".

$$\begin{cases} a_{i1} = w_1 & i = 1, \ldots, m \\ a_{i(j-1)} = (w_{j-1} - a_{i(j-2)})\lambda_i & i = 1, \ldots, m \\ \quad j = 3, \ldots, n & \end{cases} \qquad (8)$$

Eq. (9) is used to create matrix "*B*" as 2nd level. By using power 2 the effect of negative data will be removed. The whole equation can remove the effect of first-order finite difference.

$$B = \left[\left(w_{j-1} - a_{ij}\right)^2\right]_{m\times(n-2)} \quad i = 1, \ldots, m \quad j = 2, \ldots, n-1 \qquad (9)$$

Now, in Matrix "*B*", there are some rows that each of them has a weight. Also, in comparison with the upper row, each row has 1 percent more connected data. After the last predominant CP, all data points follow the same trend. Detrending data can help to change the axis of all dataset to the x-axis. So, all the data will be around the x-axis. By summing up all the data in each row, there is one row that is more near to zero in comparison with others. It shows that after giving weight and rotating data around the x-axis, all data are in the same trend in comparison with other rows. By considering that giving weight and connecting data in each row is a bit more than the upper row, there will be a row that is completely near to zero because the given weight is in a same trend period with the CP. Eq. (10) is used to sum up all data in a row.

$$c_i = \sum_{j=2}^{n-1} b_{ij} \qquad (i = 1, \ldots, m) \tag{10}$$

The result of the last equation shows the position of last CP. If k= 90, the amount of dataset is 1000 and each row contain one percent of data, the position of last CP will be between data 890-900. Using a recursive method can help to find all CPs in a dataset. After finding each CP from the end of the dataset, the last trend period will be removed, and the rest of dataset will be processed recursively. A threshold can finish the recursive method based on the amount of data.

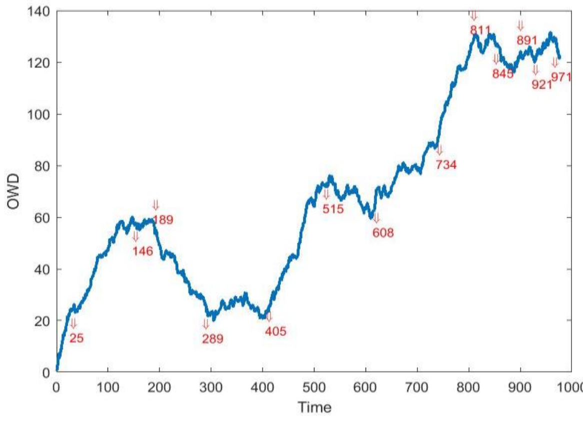

Fig 1. An example for the results of step 1

TABLE I: DATA GENERATORS FOR FIG. 1

Time	Data Generator
0-100	Expo (0.0098)
100-200	Expo (0.0099)
200-300	Expo (0.0103)
300-400	Expo (0.01)
400-500	Expo (0.0095)
500-600	Expo (0.0101)
600-700	Expo (0.0097)
700-800	Expo (0.0095)
800-900	Expo (0.01)
900-1000	Expo (0.0101)

Fig. 1 shows the result of step 1 and red arrows illustrate the exact time of each determined CP by the proposed method. We use Riverbed modeler (OPNET modeler) and MATLAB to simulate the proposed method. The data generators of node1 send data to node3 by using a middle node. All nodes have processing power to process 100 packets in each second. The dataset in Fig. 1 is gathered by node3. The packet size is Exponential (1024) as well. Also, Table I shows the data generators of Fig. 1. As you can see, most of CPs are not in the exact starting time of data generators which is based on the effect of *M/M/1* queue processing delay. As mentioned above, there are 2 steps to detect contextual outliers. Based on (3) to (11) a dataset will be divided into smaller subsets which are ready for outlier detection and outlier correction. The proposed method gives a score to each data point. The score shows whether the data point is an outlier or not. To find the data point's score, each subset should detrend separately and compare with raw data. Least Squares linear regression equation is used to detrend subsets. Since we are to gain a linear approximation of available time series, there are two parameters that should be determined. The equation is (12).

$$y = mx + b \tag{12}$$

where *m* is the slope of desired line and b represents its intercept. Assuming that the line slope is known, one can claim that the new line equation with x replaced by the mean of x's $(\bar{x})$ should result in mean of y's $(\bar{y})$. So, *b* can be calculated as (13)

$$b = \bar{y} - m\bar{x} \tag{13}$$

Slope calculation can be found in (14) which poses:

$$slope = m = \frac{\sum_{i=1}^{x} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{x} (x_i - \bar{x})^2} \tag{14}$$

Eq. (14) guarantees least square error for line regression. At this point, there is a line which represents the trend in the best way and it will be used for detrending. In order to detrend data in specified trend, data series is subtracted based on (15).

$$Scores = |d - y| \tag{15}$$

where *Scores* stands for detrended data, *d* represents subset in specific trend, and *y* is approximated line with Least Mean Squares (LMS) algorithm. To adapt results with outlier scoring, all scores should be positive. Consequently, the absolute operation is applied to inverse negative results. To find the final results, a threshold that can separate outliers from original data is needed. Eq. (16) is used to calculate the threshold to find the outliers and separate them from other data points. 1 mean that the data point is an outlier.

$$f(x) = \begin{cases} 0 & \forall x \,\big|\, Score(x) \le Avg(Scores) + SQRT(Var(Scores)) \\ 1 & \forall x \,\big|\, Score(x) > Avg(Scores) + SQRT(Var(Scores)) \end{cases} \tag{16}$$

IV. PEFRORMANCE EVALUATION

In order to evaluate the efficiency of the proposed method, we compare our results with a fundamental method which is based on Hampel filtering [35] which is used to find outliers in Time-series. The method will calculate the median of 7 data for each *'x'*, as a data in a time series, with 3 data per each side. After that it will calculate the standard deviation for each *'x'* by using median absolute deviation. If there is a difference between the median and more than 3 standard deviations, it will be replaced with the median. General dataset to evaluate the proposed method is based on Fig. 1. Also, the proposed method is evaluated based on 3 different scenarios. We have injected outliers in the scenarios randomly. Table II shows times of outliers for each scenario. In each time, there are 2 outliers near each other. To generate values of injected outliers, the following equation has been done. Data points in odd seconds will be multiplied by 0.8 and even seconds will be multiplied by 1.2 to have different outlier values. Outliers have been placed after generating the dataset to evaluate the proposed method. By considering that *OWD* in each time is different, we have different values as outliers based on the basic equation multiplying values. When the *OWD* is big, the outlier will be bigger than *OWD*s which are small. So, there are different outliers which are completely different in values. Also, minimum value between outliers and raw data is 10.

TABLE II: OUTLIERS FOR DIFFERENT SCENARIOS

Scenario #	Outlier times
1	72, 141, 193, 242, 310, 381, 452, 471, 512, 573, 620, 691, 732, 790, 821, 840
2	82, 171, 182, 231, 290, 330, 371, 402, 432, 477, 502, 543, 592, 630, 681, 722, 770, 791, 815, 840
3	20, 82, 171, 179, 182, 231, 242, 290, 301, 330, 371, 402, 432, 477, 502, 543, 592, 630, 681, 722, 733, 770, 791, 815, 840

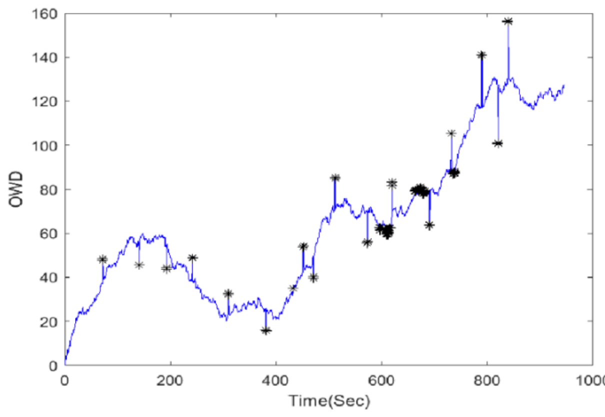

Fig. 2. The results of Scenario 1(True: 32, False Negative:40, False Positive:0)

The evaluated dataset is based on Table I. Fig. 2 shows the results of the first scenario. Stars show the points were found by using the proposed method. Each peak is an outlier. Each peak without any star shows two false positive results. Also, each star that is not at a peak shows a false negative result. To have a better understanding, all data points have been connected in figures. Although the proposed method finds all outliers, the number of false negative results are high. But there is a special phenomenon that most of them are centralized in 2 areas. After comparing the proposed method with Hampel, the reason for the phenomenon will be explained. In Fig. 3, results of scenario 2 with 40 outliers have been illustrated. By increasing the number of outliers, the number of false negative results will decrease. It is based on that the number of outliers can change the results of step 1. If we have several outliers, especially when they are near each other, the first step will assess outliers as change points which can be cause of more subsets with different trends. When there are smaller subsets, the accuracy can be increased by changing the size of the matrix A and B, so the number of false positive will decrease.

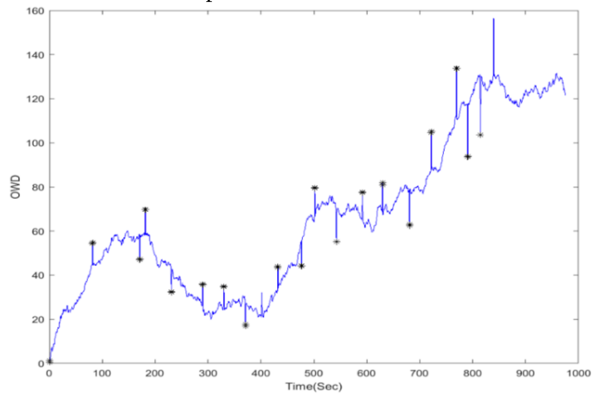

Fig. 3. The results of Scenario 2(True: 38, False Negative: 2, False Positive:2)

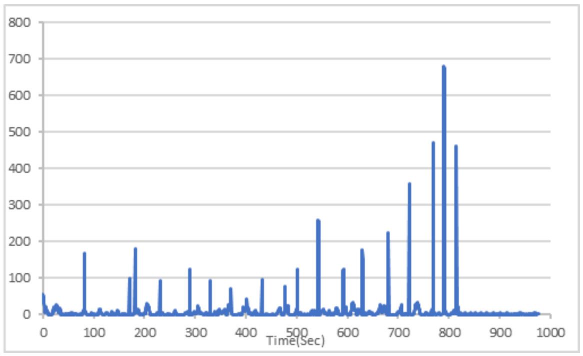

Fig. 4. Scores for Scenario 2

Fig. 4 illustrates the scores based on Scenario 2. Based on the scores, different methods such as machine learning can be used to determine the threshold. Basically, the proposed method is used to determine a score for each data point. The threshold can be calculated based on different other methods. This figure shows that when we have big outliers, scores will be big for outliers too. In Fig. 5, the results of 50 outliers have been shown. The proposed method is able to find almost all outliers. The number of false negative point is not high in comparison with 32 outliers. In addition, most of the false negatives are gathered in some areas in the dataset that can be considered as small trends in a subset or collective outliers. Table II shows the results of the proposed method and Hampel. (h) shows the results of Hampel method. Also, there are some values in parentheses that show the number of false negative which are gathered in a point as collective outliers. For instance, (8) in scenario 3 with 50 outliers is related to the time about 790 in Fig. 5.

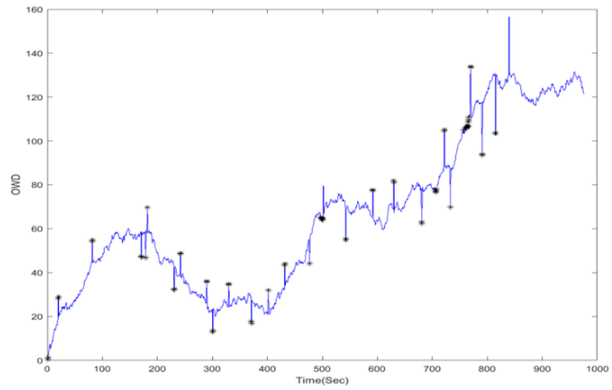

Fig. 5. The results for Scenario 3 (True:48, False Negative 18, False Positive: 2)

As shown in Table III, the number of false negative results is lower than Hampel. In addition, most of the false negative results of the proposed method are gathered in some specific areas that are local trends. Based on step 1, the proposed method divides the dataset into several subsets that each one is following a period trend. There are some jitters and small trends that change the predominant trend for a period of time. Based on determining several outliers in an area of small jitter like 620 in Fig. 2, we can say that our method is able to find collective outliers as well. Determining and eliminating collective outliers is considered as future works.

In Table III the number of false positives is about same. Results of Hampel in scenario 1 shows that it is not trustworthy in some cases like this scenario. Although both of methods are able to find most of the outliers, the accuracy and reliability of the proposed method are much higher than Hampel according to Table III.

TABLE III: COMPARING PROPOSED METHOD AND HAMPEL(H) BASED ON TABLE I SCENARIOS

Number of outliers	False Negative	True	False Positive
50(h)	27	48	2
50	18(4)(8)(3)	48	2
40(h)	32	38	2
40	2	38	2
32(h)	62	14	18
32	40(14)(4)(16)(5)	32	0

VI. CONCLUSION AND FUTURE WORKS

The proposed method can find outliers in time-series datasets. The method is able to improve datasets which gathered from the network as the input of NBA methods. NBA methods need reliable datasets to extract useful information and outliers are a hindrance to do it appropriately. After determining and removing outliers, NBA method can show more accurate information about the network. The proposed method can find contextual outliers with high accuracy based on dividing the full dataset into subsets with different trends. The number of false negative results are low which make the proposed method trustworthy. The proposed method is also able to find almost all outliers in a dataset with high accuracy and reliability as our goals. In IoT networks, monitoring is a critical task based on their nature, especially in ad hoc use cases. Although the method is based on finding contextual outliers in offline datasets, determining collective outliers in online datasets are considered as future work. The future work model would be helpful to analyze the situation of the network in a real-time manner.

ACKOWLEDGMENT

The work on this paper has been sponsored by the focus are for technology, energy and society at Østfold University College.

REFERENCES

[1] S. Y. Lim and A. Jones, “Network anomaly detection system: The state of art of network behaviour analysis,” in *Proc. International Conference on Convergence and Hybrid Information Technology*, Busan, 2008, pp. 459-465.

[2] F. Tegeler, X. Fu, G. Vigna, and C. Kruegel, “Botfinder: Finding bots in network traffic without deep packet inspection,” in *Proc. 8th International Conference on Emerging Networking Experiments and Technologies*, Nice, 2012, pp. 349-360.

[3] J. Hayes, *Modeling and Analysis of Computer Communications Networks*: Springer Science & Business Media, 2013.

[4] V. Gazis, “A survey of standards for Machine to Machine (M2M) and the Internet of Things (IoT),” *IEEE Communications Surveys & Tutorials*, 2016.

[5] W. X. Li, J. Xu, and H. Jiang, “Queuing states analysis on a hybrid scheduling strategies for heterogeneous traffics in iot,” in *Proc. International Conference on Computer Science & Service System*, Nanjing, 2012, pp. 1007-1008.

[6] A. Baz, A. A. Al-Naja, and M. Baz, “Statistical model for IoT/5G networks,” in *Proc. Seventh International Conference on Ubiquitous and Future Networks (ICUFN)*, Sapporo, 2015, pp. 109-111.

[7] W. H. Hsu, Q. Li, X. H. Han, and C. W. Huang, “A hybrid IoT traffic generator for mobile network performance assessment,” in *Proc. 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Valencia, 2017, pp. 441-445.

[8] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, “Wireless sensor networks: A survey on recent developments and potential synergies,” *The Journal of Supercomputing*, vol. 68, pp. 1-48, 2014.

[9] O. H. A. Shahraki and H. Taherzadeh, “Last significant trend change detection method for offline poisson distribution datasets,” presented at the IEEE International Symposium on Networks, Computers and Communications, Marrakech, 2017.

[10] Y. Wang, M. C. Vuran, and S. Goddard, “Cross-layer analysis of the end-to-end delay distribution in wireless

sensor networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, pp. 305-318, 2012.

[11] M. Dehmer, S. Pickl, and Z. Wang, "A short survey on statistical network analysis," in *Proc. International Conference on Foundations of Computer Science (FCS)*, Nevada, 2015, p. 110.

[12] M. P. Raj and C. G. Rao, "Data mining–past, present and future–a typical survey on data streams," *Procedia Technology*, vol. 12, pp. 255-263, 2014.

[13] S. Muthukrishnan, "Data streams: Algorithms and applications," *Foundations and Trends® in Theoretical Computer Science*, vol. 1, pp. 117-236, 2005.

[14] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, *et al.*, "Open challenges for data stream mining research," *ACM SIGKDD Explorations Newsletter*, vol. 16, pp. 1-10, 2014.

[15] S. Ram ŕez-Gallego, B. Krawczyk, S. Garc á, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39-57, 2017.

[16] C. Heinz, J. Kramer, T. Riemenschneider, and B. Seeger, "Toward simulation-based optimization in data stream management systems," in *Proc. IEEE 24th International Conference on Data Engineering*, Cancun, 2008, pp. 1580-1583.

[17] M. Cammert, J. Kramer, B. Seeger, and S. Vaupel, "A cost-based approach to adaptive resource management in data stream systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, pp. 230-245, 2008.

[18] I. Botan, G. Alonso, P. M. Fischer, D. Kossmann, and N. Tatbul, "Flexible and scalable storage management for data-intensive stream processing," in *Proc. 12th International Conference on Extending Database Technology: Advances in Database Technology*, Saint-Petersburg, 2009, pp. 934-945.

[19] N. Marz and J. Warren, *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*: Manning Publications Co., 2015.

[20] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, pp. 2250-2267, 2014.

[21] P. Thakkar, J. Vala, and V. Prajapati, "Survey on outlier detection in data stream," *International Journal of Computer Applications (0975–8887) Volume*, 2016.

[22] E. M. Knox and R. T. Ng, "Algorithms for mining distancebased outliers in large datasets," in *Proc. International Conference on Very Large Data Bases*, New York, 1998, pp. 392-403.

[23] L. I. Kuncheva, "Change detection in streaming multivariate data using likelihood detectors," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, pp. 1175-1180, 2013.

[24] M. Shuai, K. Xie, G. Chen, X. Ma, and G. Song, "A Kalman filter based approach for outlier detection in sensor networks," in *Proc. International Conference on Computer Science and Software Engineering*, Wuhan, 2008, pp. 154-157.

[25] S. Ando and E. Suzuki, "Detection of unique temporal segments by information theoretic meta-clustering," in *Proc. 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, pp. 59-68.

[26] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, pp. 651-666, 2010.

[27] D. Agarwal, "An empirical bayes approach to detect anomalies in dynamic multidimensional arrays," in *Proc. Fifth IEEE International Conference on Data Mining*, Washington, 2005, p. 8.

[28] U. Rebbapragada, P. Protopapas, C. E. Brodley, and C. Alcock, "Finding anomalous periodic time series," *Machine Learning*, vol. 74, pp. 281-313, 2009.

[29] E. Keogh, S. Lonardi, and B. Y. C. Chiu, "Finding surprising patterns in a time series database in linear time and space," in *Proc. Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 550-556.

[30] S. Basu and M. Meckesheimer, "Automatic outlier detection for time series: an application to sensor data," *Knowledge and Information Systems*, vol. 11, pp. 137-154, 2007.

[31] D. J. Hill and B. S. Minsker, "Anomaly detection in streaming environmental sensor data: A data-driven modeling approach," *Environmental Modelling & Software*, vol. 25, pp. 1014-1022, 2010.

[32] D. J. Hill, B. S. Minsker, and E. Amir, "Real-time Bayesian anomaly detection for environmental sensor data," in *Proc. Congress-International Association for Hydraulic Research*, Venice, 2007, p. 503.

[33] P. Showbridge, M. Kraetzl, and D. Ray, "Detection of abnormal change in dynamic networks," in *Proc. Information, Decision and Control, IDC 99. Proceedings. 1999*, Sydney, 1999, pp. 557-562.

[34] P. Dickinson, H. Bunke, A. Dadej, and M. Kraetzl, "Median graphs and anomalous change detection in communication networks," in *Proc. Final Program and Abstracts Information, Decision and Control*, 2002, pp. 59-64.

[35] F. R. Hampel, "The influence curve and its role in robust estimation," *Journal of the American Statistical Association*, vol. 69, pp. 383-393, 1974.

[36] M. Schwarz, C. Sauer, H. Daduna, R. Kulik, and R. Szekli, "M/M/1 queueing systems with inventory," *Queueing Systems*, vol. 54, pp. 55-78, 2006.

Amin Shahraki was born in Mashhad, Iran, 1988. He received his bachelor's degree in computer software engineering in Iran, 2009, and Master degree from Islamic Azad University, QIAU branch, Iran in 2012. Now He is a Ph.D. student at university of Oslo, Norway, has a fully-funded research fellowship from Østfold University College and working on Smart Buildings and welfare project. He is a member of National Elite Foundation in Iran. Also, he is a member of IEEE since 2011. His current research interests are Internet of Things, Network Behavior Analysis, Time Series Analysis, Clustering and QoS.

Øystein Haugen is a professor at Østfold University College. His research focuses on cyber-physical systems, variability modeling and software product lines, object oriented languages and methods, software engineering, and practical formal methods.