Pattern Mining Based Compression of IoT Data

Dusan Ramljak, Amitangshu Pal and Krishna Kant
Temple University, Philadelphia, PA 19122

ABSTRACT

The increasing proliferation of the Internet of Things (IoT) devices and systems result in large amounts of highly heterogeneous data to be collected. Although at least some of the collected sensor data is often consumed by the real-time decision making and control of the IoT system, that is not the only use of such data. Invariably, the collected data is stored, perhaps in some filtered or downselected fashion, so that it can be used for a variety of lower-frequency operations. It is expected that in a smart city environment with numerous IoT deployments, the volume of such data can become enormous. Therefore, mechanisms for lossy data compression that provide a trade-off between compression ratio and data usefulness for offline statistical analysis becomes necessary. In this paper, we discuss several simple pattern mining based compression strategies for multi-attribute IoT data streams. For each method, we evaluate the compressibility of the method vs. the level of similarity between original and compressed time series in the context of the home energy management system.

CCS CONCEPTS

• **Computing methodologies → Model development and analysis**;

KEYWORDS

Data Provenance; Internet of Things; IoT; Compression; Time series representation

ACM Reference Format:
Dusan Ramljak, Amitangshu Pal and Krishna Kant. 2018. Pattern Mining Based Compression of IoT Data. In *ICDCN '18: 19th International Conference on Distributed Computing and Networking, January 4–7, 2018, Varanasi, India.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3154273.3154294

1 INTRODUCTION

The vision of the Internet of Things is that individual objects of everyday life can be equipped with sensors which can track useful information about these objects, and allow for their intelligent collective control to enable rich services to the society at a low cost and energy footprint. The variety of such "smart" devices continues to expand, and includes wirelessly connected cameras, medical implants for healthcare, smart household appliances, personal/wearable devices, autonomous cars, sophisticated safety-critical systems such as nuclear plants, etc. [9, 13]. With smart city initiatives in full swing across the world, it is expected that IoT devices will pervade all major systems in a city. Gartner says that there were 6.4 billion connected devices in 2016 [17], while the predicted increase is 30% in 2017 that will result in 8.4 billion connected devices at the end of this year.

The key to the power of the IoT paradigm is the ability to provide real-time operational data from many different distributed sources to other machines, smart entities, and people for a variety of services. One major challenge is that the underlying data from different resources are extremely heterogeneous, can be very noisy, and are usually very large scale and distributed. Furthermore, it is hard for other entities to use the data effectively, without a clear description of what is available for processing. In order to enable effective use of this very heterogeneous and distributed data, *provenance* information is required to describe the data in a sufficiently intuitive way so that it becomes more easily usable. Provenance is the metadata that describes the origin and history of data use, derivation, and updates. Such information can be highly valuable for assessing the relevance, granularity, quality, and trustworthiness of the data or operations on it. Provenance as a subject has gained high visibility, and a number of provenance-related aspects have been explored, including capturing and managing provenance, building efficient queries, provenance storage, and security [3, 15]

By using the provenance information we could substantially reduce the data analytics overhead and improve the resulting quality. In the context of IoT systems, the provenance data can be thought of as the time and location stamped history of all events including reporting of sensor data and all actuation events. Recently, there have been attempts at developing an architectural model to overcome the rising challenges resulting from implementation of Data Provenance in the IoT [5] and addressing security and privacy issues of Data Provenance in the IoT [2]. The data that includes both the raw data and its provenance is of interest to us in this paper. There might be additional provenance or other information regarding the devices, actuators, and controllers themselves, but that is not of direct interest here.

In addition to the real-time insights, it is desirable to glean lower frequency or even historical insights from the underlying provenance data. Take, for example, the data collected from various busses in a power grid. At the extreme end of the time scale, the sensed voltage, frequency or power factor can be used immediately for initiating protection action if the values drift outside the acceptable range. At the next slower time scale, the data from all busses must be collected in a central place to enable state estimation and for correction of power flows. However, the collected data does not become useless at this point. The variation or drift in power flows over hours, days, and even months is crucial both for operating the power markets and for detecting degradation in the power network

This research was supported by NSF grant IIP-330295.

ICDCN '18, January 4–7, 2018, Varanasi, India

ACM ISBN 978-1-4503-6372-3/18/01…$15.00
https://doi.org/10.1145/3154273.3154294

such as spurious power flows due to growing tree branches around power lines, the imperfect closing of aging relays, corona related insulation damage or capacitor degradation, etc. Often the regulations also require the data to be stored for a certain amount of time as evidence of operational integrity, resolving disputes, fixing blame, etc.

The rest of the paper is organized as follows. Section 2 elaborates how and why provenance information can be used in IoT. Section 3 introduces our proposed scheme to compactly represent and compress the provenance information. Section 4 discusses our experimental setup and results. Section 5 summarizes the paper and discusses future work.

2 IOT DATA USE CASES

A direct storage of all raw data along with its provenance information can easily fulfill all of the lower frequency and historical data analysis needs; however, this is difficult, particularly in the IoT environment because of limited storage, computing and communication power of the individual devices, and the continuous operation of a a large number of such devices. In this paper we are focused on addressing representation and compression of the data along with the provenance information. Looking at some prominent use cases, the specific issues to consider are (a) What information do we need to keep to handle the use case? (b) How do we represent this information?, and (c) What are the potential ways of compressing the information and the potential trade-off between accuracy and compression? In IoT systems, data provenance and historical data analysis can be useful for a variety of purposes as described below.

Conflict Investigation: When conflicts are observed in a complex IoT systems such as a smart environment, the reasons for it may not be immediately obvious. Various devices and subsystems in an IoT system may interact both across the *physical space* (i.e., energy management systems on a building floor and even across floors) and across *functional domains* in order to fulfill policy objectives and avoid conflicts and anomalies. The devices/subsystems in a shared space may conflict because they are being affected and affect the same or interdependent set of parameters. As an example of cross-subsystem interaction, the security subsystem on a floor may turn on lights as a deterrence even when no one is present (assuming that the latter violates the lighting policy). Similarly, if the emergency subsystem detects smoke, the security subsystem may be instructed to unlock all doors in the area even though this goes against the normal policy. It is clear that a system level coordination and conflict resolution are essential for proper functioning of the overall system. However, in a coordinated "system of systems", such malfunctioning or conflicts may arise, but the reason for them needs proper investigation. The collected provenance data will be extremely useful in this context to pinpoint exactly which events lead to a malfunction.

Attack detection: In addition to the coordination and conflict the resolution, large IoT systems must deal with a number of other difficult issues. These include misconfiguration, malfunctioning, and attacks. Manual handling of these issues in a large IoT system is impractical, and it is important to devise automated (or largely automated) mechanisms to collaboratively detect misconfigurations/malfunctions, and reconfigure the system to minimize impact. Most IoT environments are expected to evolve dynamically for at least four reasons: (a) Changes/evolution in control/operational policies, (b) Adaptations to malfunctions in devices/subsystems, (c) Evolution of resource availability and demands (e.g., more energy demands as the company using the building grows), and (d) Evolution in the devices themselves (e.g., new sensors/actuators added, old ones retired). Therefore it is important to verify operation, detect malfunctions and attacks, and harden the system over time.

If a subsystem/device (sensor, actuator, or controller) is compromised by a hacker, we need to find out the tracks of the attacker getting into the device/subsystem so that we can analyze it further to determine security weaknesses of the system. In this case, the most important thing to keep is an abnormal activity where the definition of "abnormal" could be loose (i.e., almost everything) or very specific (a specific type of accesses) and everything in between. This tracking can be done with various degrees of sophistication, and more sophisticated tracking can enable more advanced applications. In such situations, the collected provenance data are essential to trace back the root of malfunctions or attack points.

Proof of correct functioning: Smart systems are cyber-physical-human systems, i.e. an interaction between humans and machines, and indirectly between humans. In such a system provenance information is essential for attribution/liability, i.e. who did what and in what order. For example, in case of a road accident, an autonomous car owner claims that certain functionality didn't work correctly and it caused the unsafe situation or real problem (e.g., collision). In such a scenario, the manufacturer needs a way to establish that car did everything that it was designed to do. Malfunctioning detection is related to proof of correct functioning and conflict investigation. It may be triggered either by observed malfunction or passively as an assurance activity. In the first case, one could potentially enumerate classes of malfunctions and store provenance data for it; the second one perhaps stores all potential anomalies and related data.

3 BACKGROUND AND PROPOSED METHODOLOGY

In an IoT environment, the typical scenario is that of multiple sensors, each of which provides a "reading" at successive time points, such that a continuous stream of data is being generated. The sensors could be heterogeneous and measure a variety of quantities such as temperature, light intensity, noise, etc. Thus, with n sensors, each sample will be an n-element vector, but each sensor could emit a value of different type (including integers, real numbers, enumerated types, etc.). Figure 1 shows an office layout equipped with multiple IoT-enabled sensors. Thus, our primary interest is in compressing a stream of vectors of different attributes accumulated from different such IoT devices.

We are looking into use cases when it is acceptable to scale the received values such that the value of the i-th sensor can be discretized into b_i bins, for a suitable value of b_i. We could then represent the value by the bin index. This gives us a vector of integers for each sample. The integer vector stream can be compressed in many ways.

Scalable Compression Mechanism Requirements: There are two aspects that need to be considered in order to devise a scalable compression mechanism. The first aspect is suitable compression

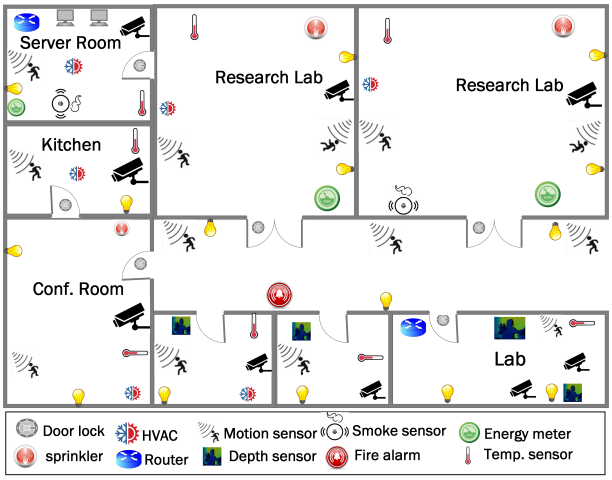


Figure 1: Office building - Illustrative IoT Layout

mechanism for a segment of data that we want to keep, where a segment could be defined as a relatively long period over which the same amount of compression would be desirable. Such techniques are discussed in the literature under the area of time series approximation and representation. Several techniques like Fourier transform [7], wavelet transform [12], piecewise polynomials [20], singular value decomposition [6] etc. fall in this category. However, these techniques don't preserve the information about the sequence of events, which is crucial in IoT context.

The second aspect is how we vary compression over successive data segments. For example, each segment may represent one year's worth of data, and as we go into the past, we use more compression. In such techniques, the compression is not homogeneous, rather heterogeneous across time. In [8, 11] the authors have modeled such aspects as "amnesic" functions, that represent recent data with better precision than the past.

Lossy compression: Straightforward lossless compression such as LZ is rarely appropriate in this scenario, because a lossless compression does not provide any opportunity to increase compression as the data ages, and depending on the type of data, the compression ratio may be rather poor. Thus, our interest is primarily in *lossy compression*, where the key issue is what we are willing to lose and that in turn depends on the usage model. A simple lossy compression technique is the aggregation of data, which could be done on many levels. Lossy compression we envisioned could be regarded as similar to summarization of provenance information.

Similar approaches: Provenance can be represented by a graph, with a standardized representation called Open Provenance Model (OPM) [10]. In OPM, the nodes are artifacts (states), processes (actions), or agents (enablers) and arcs represent dependencies such as "derived from", "generated by", "used by", "controlled by", and "triggered by". One approach, discussed in [1] defines a distance function between provenance expressions based on the intended use, and optimizing this distance while still obtaining small expressions guide the summarization. Given the provenance graph form, techniques similar to ones used for web graph compression can be applied to compressing provenance [18, 19]. The steps are: (a) encode the successor list of one node by using similar successors of another node as a reference, thus efficiently avoiding encoding the duplicate data, (b) encode consecutive numbers by only recording the start number and length, reducing the number of successors that need to be encoded, (c) encode the gap between the successors of a node rather than the successors themselves, which typically requires fewer bits to be encoded, and (d) add dictionary compression on top of compressing provenance graphs.

Approximate Vector Stream Compression (AVSC): In this paper, we propose to use a *sequence pattern mining* based compression. The purpose of this is to ensure that the sequence of events can be derived (or inferred) from the compressed data, which are crucial for conflict resolution, intrusion or malfunction detection in a complex IoT system as discussed in section 2. While performing compression we want to keep some events intact and approximate other events to improve the compressibility. Our proposed method runs in two stages.

In the **first** stage, we use the SQS (Summarizing event seQuenceS) proposed in [16] to generate a summarization of the data. SQS uses a single parameter heuristics to find a good set of patterns. That parameter is a threshold, which is the number of times a pattern shows up in a sequence, with the default number being one. Varying the threshold gives us the opportunity to vary the value of the patterns. The method formalizes how to encode a sequence dataset given a set of patterns. It uses the Minimum Description Length (MDL) principle to identify the best set where the encoded length is used as a quality score. The complexity of the method is linear in a number of events, objects, and patterns. In effect, the SQS method efficiently discovers high-quality patterns that summarize data well and correctly identify key patterns. However, the SQS method does inexact matching, i.e. patterns are allowed to have gaps, but the objective function heavily penalizes long gaps. Thus the summarization is necessary, but it is not sufficient.

In the **second** stage, using that summary as a basis for a dictionary we try to approximately match the remaining events with the patterns in our dictionary. Events that can't be matched to any of the patterns are added to the dictionary. We denote the resultant scheme as *AVSC* i.e. Approximate Vector Stream Compression. We consider the following two ways to apply AVSC:

(Correlated) Directly compress each vector, i.e., look for similarities across vectors. This retains the correlation across streams perfectly, but it may show poor tradeoff of compressibility and accuracy because of lack of too many similar vectors. We denote correlated version as AVSC-C.

(Uncorrelated) Compress i-th stream separately, but with a possibility that the information about corresponding streams is captured approximately due to different approximations across vectors. This method is expected to yield much better compression but the correlation across attributes may be somewhat compromised. We denote uncorrelated version as AVSC-NC.

The overall scheme is depicted in Algorithm 1. In the first stage, we discretize the time series and apply the SQS method in order to determine the patterns (with gaps). After the original SQS method determines the patterns, we preserve the identified patterns in the dictionary $\mathbb{P}$ and identify set of remaining subsequences $\mathbb{U}$. In the second stage, we try to approximately match the remaining subsequences with the patterns in our dictionary. To determine how close a subsequence matches with the existing patterns, we

define a threshold τ which is the Euclidean distance between the subsequence and the existing pattern from the dictionary. If the tested subsequence doesn't *closely* match with any of the identified patterns (i.e. their Euclidean distance is more than τ), we break the subsequences with length ℓ and try to find a match with the existing patterns. If no matches are found, the subsequence is added in the dictionary. We keep doing this until the entire time series is covered. This yields a compressed stream, which can be inverted to get an approximate representation of the original stream.

Algorithm 1 Approximate Vector Stream Compression (AVSC)

global variables
OT: Original time series ▷ INPUT
$\mathbb{P}$: Ordered set of patterns found ▷ SQS Output
$\mathbb{U}$: Ordered set of remaining subsequences ▷ SQS Output
$\mathbb{I}$: List of tuples $\langle$ time series index (ind), element of $\{\mathbb{P} \cup \mathbb{U}\}\rangle$ ▷ SQS Output
$d(s, p)$: Euclidean distance between vectors s and p
s: Top element of $\mathbb{U}$
p_{best}: Element of $\mathbb{P}$ that has a closest match with s
B: Number of bins for discretization ▷ Input Parameter
τ: Max Euclidean distance ▷ Input Parameter
ℓ: Pre-defined sequence length ▷ Input Parameter
COT: Compressed version of OT ▷ OUTPUT
end global variables

procedure AVSC
  Discretize the OT into B bins;
  Run SQS to identify $\mathbb{P}$, $\mathbb{U}$, and $\mathbb{I}$;
  while $\mathbb{U}$ not empty **do**
    p_{best} = NULL;
    Remove s from $\mathbb{U}$;
    Match s with elements of $\mathbb{P}$; ▷ p_{best}=NULL if no match found of same length
    if $p_{\text{best}} \neq$ NULL and $d(s, p_{\text{best}}) < \tau$ **then**
      $\mathbb{I}(ind, s) = \mathbb{I}(ind, p_{\text{best}})$;
    else
      if length(s) $> \ell$ **then**
        Break s to r subsequences of length ℓ;
        Add r subsequences to the top of $\mathbb{U}$ in proper order;
      else
        Add s to $\mathbb{P}$;
      end if
    end if
  end while
  LZ compress $\mathbb{I}$ to identify COT;
end procedure

Accuracy evaluation: Given the original stream and the approximated reconstructed stream, we can compare the two using time series similarity measures in order to determine how similar the two are. This provides us with a measure of accuracy, or fidelity of the compression. We could thus evaluate the compressibility vs. accuracy trade-off.

There are many methods in the literature to assess the similarity of two-time series. Reference [14] provides a survey of several methods. The main methods include similarity based on (a) Euclidean distance, (b) Fourier coefficients, (c) autoregressive models of the time series, (d) dynamic time warping (DTW), (e) Edit distances (ED), (f) Time warped edit distance (TWED), and (g) minimum jump cost dissimilarity (MJC). It turns out that the last four methods show almost the same performance. For our experiments we consider the Euclidean distance based metric explained in the evaluation section.

Algorithm discussion: Note that AVSC-NC maintains the correlations across vector elements approximately, since each series when reconstructed independently, will still yield approximately the right value for each attribute going across the attributes. However, the pattern driven approximation could also add some spurious correlation. AVSC-C, on the other hand, preserves the correlations exactly at the cost of lower accuracy. The accuracy of our method could be further improved if we run a second pass through subsequences that remained after running SQS. This is due to the fact that some of the sequences that are added to the dictionary might be a better match and reduce the loss of accuracy stemming from this approximation. The second pass could be particularly important for vector streams where the first pass does not find too many patterns. Both accuracy and compressibility might also be improved by a proper choice of parameter ℓ, which is currently set to number 3.

Approximate matching as an amnesic function? Notice that there is a trade-off between the accuracy and storage. We increase the accuracy at the cost of more storage. This trade-off can be extended to model amnesic function, that allows us to represent the recent past with greater precision. A simple way to integrate amnesic function in our representation is to require more accuracy for recent past and lesser for the older ones. A more sophisticated framework will allow more accuracy in the "area of interest" while requiring less accuracy elsewhere.

4 EVALUATION

For the evaluation, we consider the home energy usage data available from [4]. The dataset consists of a wide variety of data collected from three real homes (named as home-A, B, and C), including electrical (usage and generation), environmental (e.g., temperature and humidity), and operational (e.g., wall switch events). We mainly use the electricity usage data for these three homes as a function of time and weather. The data corresponding to home-B and home-C are obtained over a period of 3 months (from May to July 2012) from various sensors with readings every 5 minutes. We also use the data for home-A where readings are taken at every hour. Since there is a strong correlation between weather and energy usage (due to the use of HVAC), we consider a time series with 4 element vectors. The vector elements are, respectively, outdoor temperature, outdoor humidity, wind velocity, and power usage. Unless otherwise mentioned we use 128 bins for discretization of all attribute values.

Notice that while compressing the original time series (OT), we (a) first discretize the samples into bins, then (b) compress the bins using AVSC scheme, we call these two intermediate series of bins as S1 and S2. From S2 we can approximate (or reconstruct) the original time series which we denote as reconstructed time series or RT. To compare the accuracy and compressibility of the original time series (x_t) with the reconstructed one ($\hat{x}_t$), we use two metrics named *Relative Compression Ratio (RCR)* and *Relative Root-Mean-Square-Error (RRMSE).* The former represents the amount of compression with respect to the LZ compression, and the latter gives the normalized root-mean-square error of reconstructed time series, i.e.

$$RRMSE = \frac{RMSE}{RMS_{\text{orig}}} = \frac{\sqrt{\sum_{t=1}^{n} (x_t - \hat{x}_t)^2 \,/\, n}}{\sqrt{\sum_{t=1}^{n} (x_t)^2 \,/\, n}} \tag{1}$$

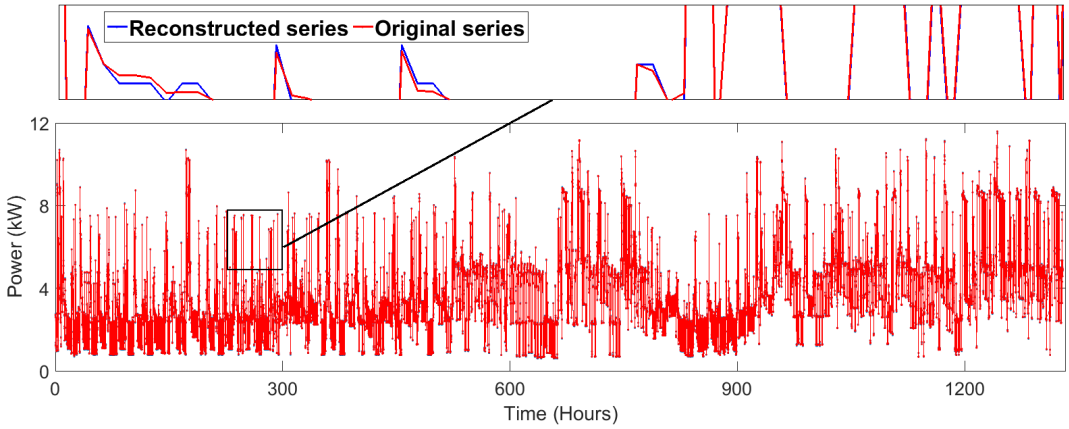


Figure 2: Home-C's power measurements: Comparison of the original and reconstructed time series

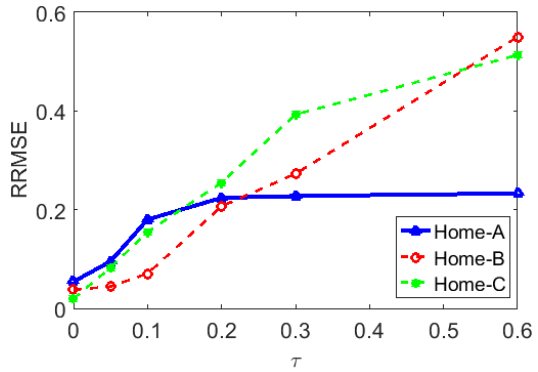
Notice that a lower RRMSE represents higher accuracy (or fidelity) and vice versa. The combined RRMSE of multiple measurements (i.e. temperature. humidity, wind velocity and power usage) is obtained by taking the root of the sum of squares of the individual RRMSE values. Because of this reason, with higher dimensions, the combined RRMSE has a higher value compared to the RRMSE of the individual time series.

Comparison of *OT* and *RT*: To show the accuracy of the proposed compression scheme we compare the original power measurements of home-C along with the reconstructed one for AVSC-C. The result is shown in Figure 2 where τ is 0.1. From this figure, we can observe how well the reconstructed samples approximate the original measurements. The combined RRMSE corresponding to Figure 2 is found to be 0.15.

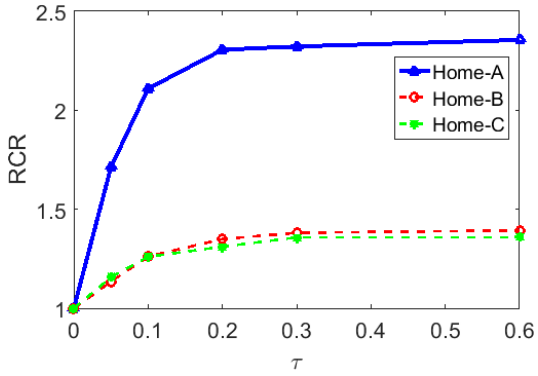
Evaluation of uncorrelated compression AVSC-NC: Figure 3 shows the variation of RRMSE and RCR with τ corresponding to the data obtained from three homes using AVSC-NC compression technique. From Figure 3 we can observe that AVSC-NC provides better compression ratio for the dataset from home-A than that of homes B and C. Notice that in case of home-A the readings are obtained at every hour, whereas for homes B and C the samples are taken at every 5 minutes. Because of this reason, the identified patterns obtained in case of homes B and C cannot replace a large section of the remaining subsequences with reasonable accuracy. We can also observe that the improvement in compression ratio becomes marginal beyond $\tau = 0.2$, however, the RRMSE keeps increasing especially in case of homes B and C. This brings the notion of finding the optimal τ for a given measurement data, which needs further investigation in future.

Figure 3 also shows the discretization error (i.e. RRMSE at $\tau = 0$) due to the binning operation, which is found to be significantly smaller compared to the accuracy loss due to compression.

Evaluation of correlated compression (AVSC-C): While evaluating the effect of correlated compression with the above datasets, we couldn't identify a correlated pattern with more than 32 bins. In case of home-A, we had to decrease the number of bins even further to 16 however, even after doing so we obtain no compression gain until τ reaches 0.6. We, therefore, ignore the case of home-A, and only show the comparison of homes B and C in Figure 4, while keeping the number of bins equal to 32.



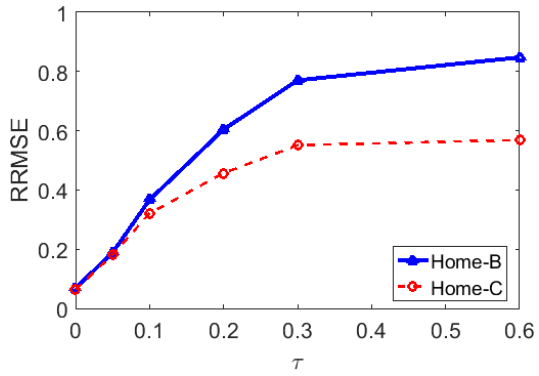
(a) RRMSE



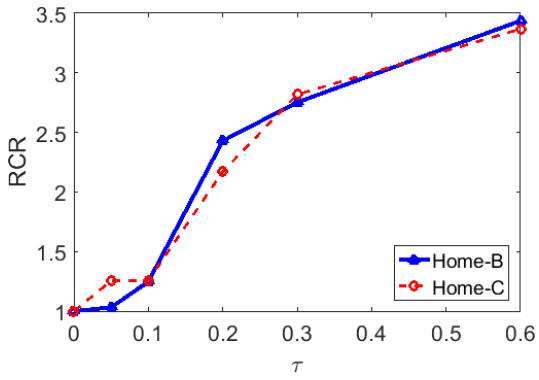
(b) RCR

Figure 3: AVSC-NC Evaluation

From Figure 4 we can observe that RCR increases with the increase in τ as more subsequences are approximated with the identified patterns in the dictionary. However, doing so increases the RRMSE significantly, i.e. using the above datasets, even though we can obtain a considerable gain in terms of compressibility, it comes at the cost of low accuracy.



(a) RRMSE



(b) RCR

Figure 4: AVSC-C Evaluation

Comparison of AVSC-NC and AVSC-C: In order to directly compare the benefits and downsides of two compression techniques, we repeated experiments for homes B and C for 32 bins discretization. In the repeated experiments we observed that AVSC-C method in comparison with AVSC-NC provides similar compression ratio for smaller thresholds τ, but the loss in accuracy is almost 60% worse for AVSC-C (From 0.10 RRMSE for AVSC-NC to 0.18 RRMSE for AVSC-NC). Since AVSC-C does a somewhat better job in preserving correlation, a more detailed analysis is required for a fair comparison; however, based on the results so far, it appears that AVSC-C is not a very attractive method. It is worth noting that the observed behavior is domain specific. There is a strong correlation between weather and energy usage and discretization smooths the differences between the vectors such that there are enough patterns that AVSC could exploit.

5 CONCLUSIONS AND FUTURE WORK

In this paper we discussed mechanisms for lossy data compression that provide a trade-off between compression ratio and data usefulness for offline statistical analysis in IoT. We considered sequence pattern mining based compression strategies for multiattribute IoT data streams. For each method, we evaluated the compressibility of the method vs. the level of similarity between original and compressed time series in the context of the home energy management system. In particular, we showed gains over the plain lossless compression for a specified amount of accuracy for purposes of identifying the state of the system. This comparison shows a clear tradeoff in between the compressibility and fidelity with respect to distance threshold τ, i.e., in general, the amount of fidelity increases with small thresholds, but at the cost of poor compressibility.

A similar trade-off between compressibility and accuracy can also be drawn with respect to the number of bins used in the discretization process. We will explore these characteristics in more detail in the future to change the distance threshold τ and the number of bins dynamically with age as well as for different regions of interest in a time series. Along the same lines, we want to extend AVSC to explore the notion of progressive compression, where we re-compress the older at a higher compression level in order to keep maintenance of the measurement information scalable. We also need to investigate how to devise the optimal threshold τ and optimal breakup of long non-matching subsequences. More investigation is needed to find a better fidelity metric and then more validation is needed using more data from different domains. Another line of investigation is the accuracy of the compressed representation for specific event detection, such as home occupancy detection, intrusion detection etc. Finally, we do a more comprehensive evaluation of the proposed compression scheme and its enhancements.

REFERENCES

[1] Eleanor Ainy et al. 2015. Approximated Summarization of Data Provenance. In *ACM CIKM*.
[2] Muhammad Naveed Aman et al. 2017. Secure Data Provenance for the Internet of Things. In *ACM IoTPTS*. 11–14.
[3] Khaled Bachour et al. 2015. Provenance for the People: An HCI Perspective on the W3C PROV Standard Through an Online Game. In *ACM CHI*. 2437–2446.
[4] Sean Barker et al. 2012. Smart*: An open data set and tools for enabling research in sustainable homes. (2012).
[5] Sabine Bauer et al. 2013. Data provenance in the Internet of things. In *Conference Seminar SS*.
[6] Kaushik Chakrabarti et al. 2002. Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. *ACM Trans. Database Syst.* 27, 2 (2002), 188–228.
[7] Christos Faloutsos et al. 1994. Fast Subsequence Matching in Time-series Databases. In *ACM SIGMOD*. 419–429.
[8] Sorabh Gandhi et al. 2010. Space-efficient online approximation of time series data: Streams, amnesia, and out-of-order. In *IEEE ICDE*. 924–935.
[9] Jayavardhana Gubbi et al. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems* 29, 7 (2013), 1645–1660.
[10] Luc Moreau et al. 2008. *The Open Provenance Model: An Overview*. 323–326.
[11] Themistoklis Palpanas et al. 2004. Online Amnesic Approximation of Streaming Time Series. In *IEEE ICDE*. 339–349.
[12] Ivan Popivanov et al. 2002. Similarity Search Over Time-Series Data Using Wavelets. In *IEEE ICDE*. 212–221.
[13] Karen Rose et al. 2015. The internet of things: An overview. *The Internet Society* (2015), 1–50.
[14] Joan Serra et al. 2014. An empirical evaluation of similarity measures for time series classification. *Knowledge-Based Systems* 67 (2014), 305–314.
[15] Salmin Sultana et al. 2015. A distributed system for the management of fine-grained provenance. *Journal of Database Management* 26, 2 (2015), 32–47.
[16] Nikolaj Tatti et al. 2012. The long and the short of it: summarising event sequences with serial episodes. In *ACM KDD*. 462–470.
[17] Rob van der Meulen. 2017. Gartner Says 8.4 Billion Connected âĂŸThingsâĂŹ Will Be in Use in 2017, Up 31 Percent From 2016. (2017).
[18] Yulai Xie et al. 2011. Compressing Provenance Graphs.. In *TaPP*.
[19] Yulai Xie et al. 2013. Evaluation of a hybrid approach for efficient provenance storage. *ACM Transactions on Storage* 9, 4 (2013), 14.
[20] Byoung-Kee Yi et al. 2000. Fast Time Sequence Indexing for Arbitrary Lp Norms. In *VLDB*. 385–394.