The Effect of Data Granularity on Load Data Compression

Andreas Unterweger[1], Dominik Engel[1], and Martin Ringwelski[2]

[1] Salzburg University of Applied Sciences, Josef Ressel Center for User-Centric Smart Grid Privacy, Security and Control, Urstein Süd 1, 5412 Puch/Salzburg, Austria. firstname.lastname@en-trust.at

[2] Technische Universität Hamburg-Harburg, Institut für Telematik, Am Schwarzenberg-Campus 1, 21073 Hamburg, Germany

Abstract A vast volume of data is generated through smart metering. Suitable compression mechanisms for this kind of data are highly desirable to better utilize low-bandwidth links and to save costs and energy. To date, the important factor of data resolution has been neglected in the compression of smart meter data. In this paper, we review and evaluate compression methods for smart metering in the context of different resolutions. We show that state-of-the-art compression methods are well suited for high resolution, but not for low resolution data. Furthermore, we elaborate on the compression performance differences between appliance-level and household-level load data. We conclude that the latter are practically incompressible at most resolutions.

1 Introduction

In smart grids, the volume of data to be processed, transmitted and stored is considerable. In the distribution grid, smart meters are a source of high data volume. Depending on the use case and regulatory restrictions, different measurements are collected by a smart meter in different granularities, typically in measurement intervals of 60 seconds up to 15 minutes (cf. Table 10 in [7]). Smart meters are also capable of collecting measurements related to power quality. All measurements can technically be done in smaller intervals (i.e., seconds).

It is evident that compressing the data generated in smart metering is highly desirable. Smart meters are typically connected via low-bandwidth links, such as PLC. Through compression, the bandwidth of these links can be utilized more efficiently. The increase in efficiency, of course, depends on the measurement interval and will increase with smaller intervals. Furthermore, transmitting data in compressed form is more energy-efficient than transmitting data in uncompressed form – given that an appropriately light-weight compression scheme is used, the power needed for compression is significantly lower than the power needed for transmission. Finally, at the receiving end, where the data needs to be stored, compression can help to save costs.

It comes at no surprise that a number of proposals have been made for the compression of smart metering data. However, none of these contribution explicitly addresses the issue of resolution and its impact on compression performance.

While there are many benefits to compression, it has to be evaluated how well the raw data is suited for compression at different measurement intervals, i.e., varying data granularity. An overview of standard compression methods applied to smart metering data is given by Ringwelski et al. [11]. Furthermore, the authors propose their own method. Unterweger and Engel [13] propose a compression method that allows resumability. Two contributions that implicitly address resolution, because they both employ the wavelet transform for compression are Ning et al. [10] and Khan et al. [8]. However, neither takes the impact of resolution of the *source* data into account.

In general, there is little research that addresses the resolution of smart metering data, mostly in the area of smart meter privacy. Eibl and Engel [5] give an account on the influence of data granularity on privacy in smart metering. Approaches for privacy-preserving smart metering are presented by Efthymious and Kalogridis [4] and Engel [6]. Sankar et al. [12] introduce an information-theoretic framework for smart meter privacy, which implicitly addresses data resolution as part of the proposed privacy measure.

In this paper, we evaluate the compression algorithms proposed by Ringwelski et al. [11] and Unterweger and Engel [13] in the context of source data resolution. This is an important perspective, as different use cases in the smart grid will require different measuring intervals and therefore different resolutions of load data. An appraisal on how this resolution impacts compression performance gives an important guideline on what amount of data needs to be transmitted for the individual smart metering use cases.

This paper is structured as follows: In Section 2, we describe the compression algorithms that we evaluate in Section 3. Section 4 concludes.

2 Compression algorithms

Several algorithms for compressing load data have been studied in the literature. We focus on those algorithms which have been specifically designed for load data in the context of smart metering, where resources are typically sparse, i.e., execution time and memory consumption have to be minimized.

For reference, we use two standardized encodings for load data which do not compress the data. For our measurements in Section 3, we use two tailored compression algorithms. All four approaches are described in the following sections. Although some encodings specify the use of units (e.g., watts), we focus on the value encoding only. Unit signaling can be amended if necessary, but is out of scope of this work.

2.1 Reference algorithms

Two standards for transmitting load data are commonly used: IEC 62056-21 [3] and IEC 61334-6 [2], also referred to as A-XDR. Both specify value encodings which do not perform any compression whatsoever, minimizing computational complexity. In the following, we describe both algorithms briefly since we use them for reference measurements.

IEC 62056-21 Values are encoded in their base 10 representation with a decimal point and encoded as ASCII [1] bytes. The value *123.45*, for example, is encoded as `00110001 00110010 00110011 00101110 00110100 00110101`, requiring six bytes – five digits and the decimal point.

Since the length of each encoded value depends on its magnitude, an additional delimiter between subsequent values is required so that they can be separated during decoding. Without additional signaling information, an underscore (ASCII character 137), for example, can be used as a delimiter. This way, the values *123.45* and *123.56*, for example, are concatenated to *123.45_123.56* before encoding, requiring a total of $6 + 1 + 6 = 13$ bytes.

A-XDR Unsigned integer values are encoded in their base 2 representation with a fixed length, e.g., 16 bits. The value *12345*, for example, is encoded as `00110000 00111001`, requiring 2 bytes. Although floating-point values are not supported, multiplying the floating-point value by 10^n, where n is the number of decimal places after the decimal point, yields an integer value which can be encoded using A-XDR.

Since the number of decimal places does typically not change within a load data time series, no additional signaling for n is required. However, the number of bits required for representation may have to be increased to accommodate for the increased value range due to the multiplication by 10^n. For example, encoding the value *123.45* (as *12345*, see above) requires at least 14 bits, as opposed to the value *123*, which only requires 7 bits.

As stated above, A-XDR coding uses a fixed bit length for representing values. Thus, all values can be decoded without the need for any additional delimiters as opposed to the IEC 62056-21 value coding described above.

2.2 DEGA coding

Unterweger and Engel [13] have proposed a compression algorithm for load data which exploits the data characteristics of load profile data. Their encoding algorithm, which we refer to as DEGA (Differential Exponential Golomb and Arithmetic) coding due to its main elements, is illustrated in Fig. 1 and consists of five steps (labeled A-E).

First, the floating-point input values are normalized (A) to make them integer, as explained for A-XDR in Section 2.1. Second, the differences between consecutive values are calculated (B), since they are typically smaller than the values themselves. Third, the differences are encoded as Signed Exponential Golomb code words of order zero (C) for variable-length coding. Fourth, the code words are concatenated (D) and finally compressed using an adaptive binary arithmetic coder (E).

During processing, the code word concatenation step (D) is usually implicitly contained in the code word generation step (C). A detailed explanation of each step as well as a description of the decoding process can be found in [13].

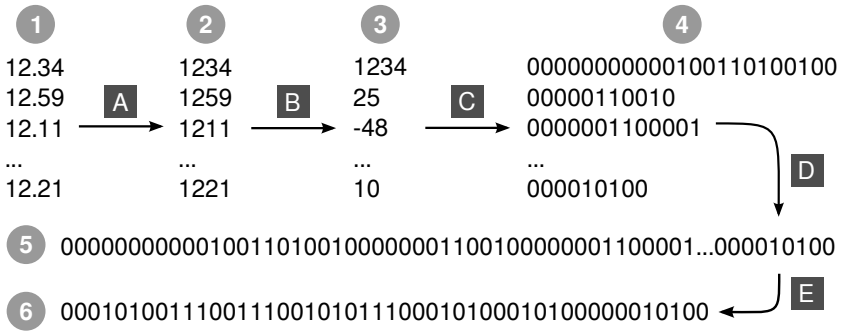

Figure 1. Overview of DEGA coding [13]: Input values (1) are normalized and their differences (3) are represented as Signed Exponential-Golomb code words (4) which are concatenated (5) and arithmetically coded.

2.3 LZMH coding

Ringwelski et al. [11] have proposed a compression algorithm for load data with low memory requirements. The algorithm is referred to as Lempel Ziv Markov Chain Huffman (LZMH) coding and combines ideas of the Lempel Ziv Markov Chain Algorithm (LZMA) and a variant of Adaptive Trimmed Huffman (AHT) coding as described below and illustrated in Fig. 2. It is designed to process ASCII-coded IEC 62056-21 data as described in Section 2.1 as input.

If at least three of the following characters are found in the history of the last m characters, a reference to it is coded (LZMA-like), consisting of a byte offset and the length, using an optimal prefix code. Conversely, when no sufficient reference is found, it is encoded as a Huffman code word (AHT-like). This code word originates from an adaptive Huffman tree which represents the symbol probabilities that are updated for each encoded character.

To keep memory requirements low, a history buffer of $m = 128$ characters is used and the size of the Huffman tree is limited to the size of the input alphabet which may be reduced to the ten decimal digits and the decimal point. A more detailed description of the algorithm can be found in [11].

3 Evaluation

We analyze the compression performance and execution times of A-XDR, DEGA and LZMH coding for IEC 62056-21 input data. The used data sets are described in detail in Section 3.1. As opposed to prior work, we study the effect of different data granularity on the results.

We evaluate different data granularity levels by summing up c consecutive input data values with inter-value temporal distance t, for example, 5 minute (300 seconds) granularity for $t = 3$ (seconds) with $c = 100$. We use the same

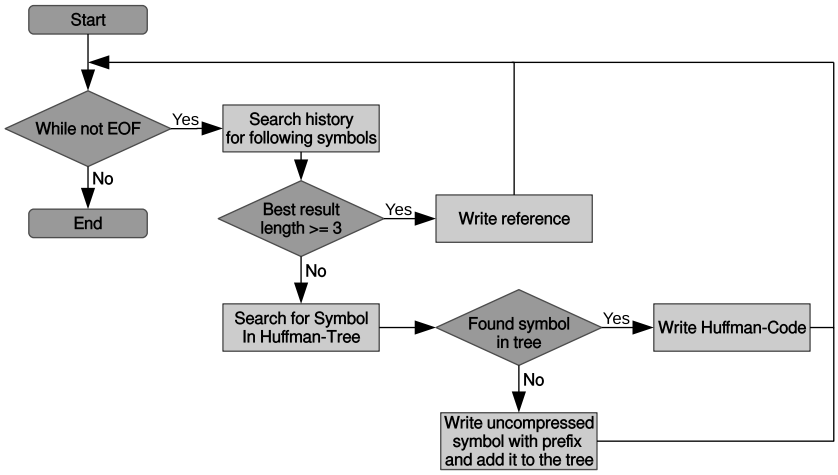

Figure 2. Overview of LZMH coding: Each input symbol is either encoded as a reference to an already processed symbol or as a Huffman code word based on its probability.

granularity levels as Eibl and Engel [5], i.e., 3 s, 9 s, 30 s, 1 min., 5 min., 15 min. and 1 h, if available.

To achieve comparable results, we have reimplemented the A-XDR and DEGA coding algorithms in the C programming language. LZMH is already implemented in C and has only been modified slightly so that it uses the same input/output functions. These changes do not affect its compression performance.

3.1 Load data sets

We use two load data sets for our evaluation: the low-frequency MIT REDD data set [9] and a data set from a local energy provider, referred to as the SAG data set henceforth. Both data sets are described briefly below.

REDD The low-frequency MIT REDD data set is a collection of load data from between 11 and 26 channels of 6 different houses. In total, there are 116 channels. Each channel containing load data is available separately.

The load data values are average apparent power readings in Watts with two decimal places, i.e., they are effectively stored with an accuracy of one hundredth of a Watt. They have an inter-value temporal distance of $t = 3$ (seconds) for all channels but the mains, which have $t = 1$. The values cover measurement intervals of between 2.7 and 25.8 days.

SAG The SAG data set is a collection of load data from 508 households and industrial plants. As opposed to the REDD data set, only the mains of each

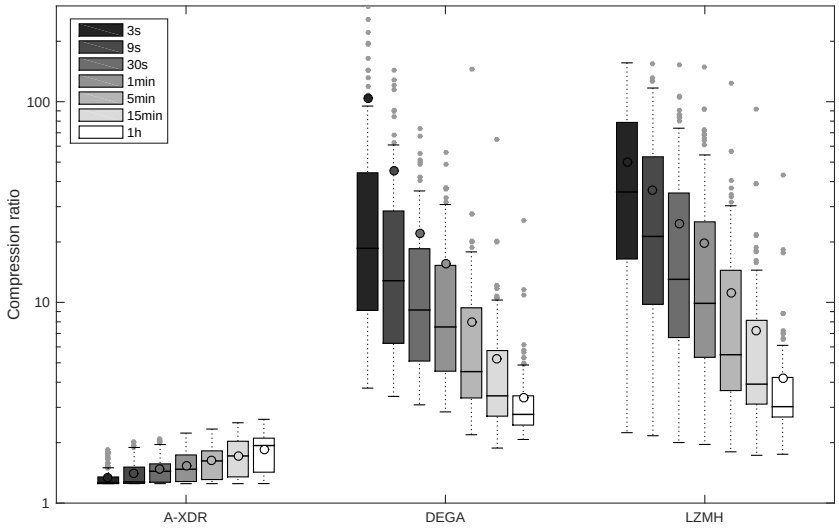

Figure 3. Compression performance of different algorithms compared to IEC 62056-21 value encoding for the REDD load data set at different data granularity levels.

household are available. They are summed up in one single value, i.e., they are not available as separate channels.

The load data values are accumulated energy readings in kWh with three decimal places, i.e., they are effectively stored with Wh accuracy. They have an inter-value temporal distance of $t = 300$, i.e., 15 minutes. The values cover measurement intervals of exactly one year.

3.2 Compression performance

Due to the different characteristics of the two load data sets described in Section 3.1, we analyze the compression performance for each data set separately. All input data is encoded in the form of IEC 62056-21 values as described in Section 2.1, which we use as reference. The results are described in the following sections.

REDD data set Figure 3 shows an overview of the compression performance of the A-XDR, DEGA and LZMH algorithms for the REDD data set. Each channel is compressed separately and its compressed size is expressed relative to the input data size as a ratio. A compression ratio of 5, for example, means that the compressed data requires only 20% of the size of IEC 62056-21 value encoding.

The compression ratio distribution for all channels is depicted as a box plot with added mean compression ratios (filled circles with black borders) and outliers (gray circles without borders). The y axis is logarithmic and capped at 300. Thus, four outliers representing all-zero valued channels are not depicted.

Obviously, DEGA and LZMH exhibit significantly better compression performance than A-XDR, which does not compress by design. Still, it achieves compression ratios greater than 1 compared to IEC 62056-21 value encoding. This is due to the fact that all input values are at least five bytes long (one decimal digit before the decimal point, two thereafter and one delimiter), but typically longer, whereas A-XDR values are always four bytes in size.

In general, LZMH outperforms DEGA at all granularity levels, where the performance difference increases with data granularity. At the finest granularity level (3 s, dark gray boxes), DEGA and LZMH achieve compression ratios of 18.59 and 35.48, respectively. They drop to 2.77 and 3.02, respectively, at the coarsest granularity level (1 h, white boxes).

Compared to A-XDR coding with a median compression ratio of 1.94 at this granularity level, it becomes clear that both, DEGA and LZMH, are practically ineffective at compressing load data with high (1 h) inter-value temporal distances. A-XDR is expected to outperform both compression algorithms at even coarser granularity levels, e.g., at inter-value temporal distances of 24 h.

In general, increased inter-value temporal distances yield larger input values, i.e., they have more decimal digits and therefore yield longer IEC 62056-21 values. Since A-XDR values are of constant size, their compression ratio increases relatively at coarser granularity levels, whereas DEGA and LZMH coding become less efficient in terms of compression performance. This is mainly due to the increased input entropy.

Coarser data granularity impacts compression performance due to the summing of values. Thus, the mains (channels 1 and 2) of all houses from the REDD data set deserve special attention. They, too, are effectively sums of multiple other channels and therefore likely to behave differently than the other channels. Figure 4 shows the compression performance of only the mains.

As expected, the compression performance of DEGA and LZMH coding for the mains is significantly poorer than the respective performance for all channels depicted in Figure 3. Although the best median compression ratio for fine-grain data (3 s, dark gray in Figure 4) is 4.90, double-digit compression performance is not achievable for the mains.

Interestingly, when compressing only the mains, DEGA outperforms LZMH at all granularity levels. The reverse is true when looking at the compression performance of all channels in Figure 3. Still, at medium granularity levels (1 min., medium gray in Figure 4), compression becomes ineffective when compared to uncompressed A-XDR coding.

Even more surprisingly, at coarser granularity levels (15 min., light gray), A-XDR actually outperforms DEGA coding with a median compression ratio of 2.39 vs. 2.25 despite the fact that A-XDR does not compress by design. This means that the mains are effectively incompressible at this resolution.

SAG data set Figure 5 shows the compression results of the A-XDR, DEGA and LZMH algorithms for the SAG load data set. Since the latter only has 15-

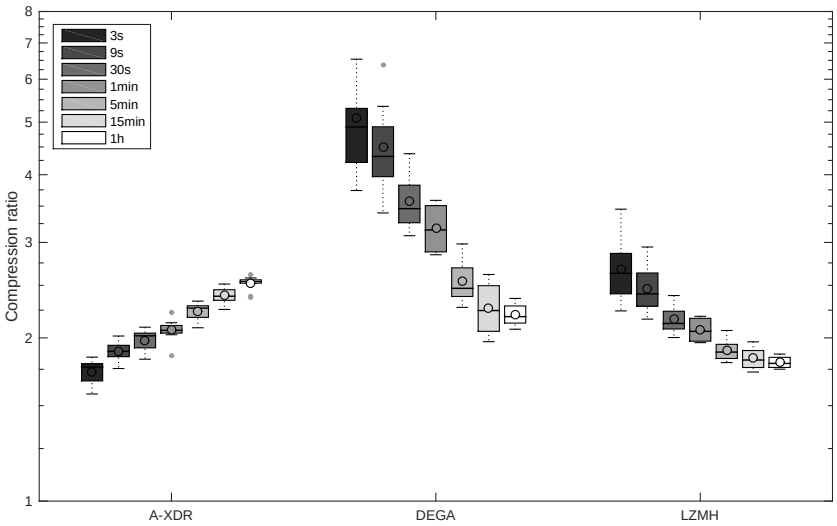

Figure 4. Compression performance of different algorithms compared to IEC 62056-21 value encoding for only the mains from the REDD load data set.

minute resolution, finer granularity levels cannot be evaluated. The visualization is identical to the one in Figure 3 for the REDD data set.

Since the SAG data set only contains measurements from the mains and not from individual channels, the results are similar to the results of the mains from the REDD data set illustrated in Figure 4. Again, DEGA outperforms LZMH coding, but the compression ratios of both are higher when compared to A-XDR, i.e., the data can be compressed to some extent, even at a temporal inter-value distance of 1h.

The main reason for this, considering that the mains from the REDD data set are incompressible as explained above, is the different accuracy of the data. While the REDD mains data has an accuracy of one hundredth of a Watt, the SAG data has an accuracy of only one Watt-hour. This significantly reduces the entropy since the highly volatile least significant digits are missing.

Apart from the lower accuracy, the value range is also reduced, i.e., the kWh readings (SAG) are significantly smaller than Watt readings (REDD). This also explains the high number of outliers (gray circles without borders) in Figure 5 for DEGA and LZMH coding: Households with a lower power consumption yield smaller values which can be more compressed more easily.

3.3 Execution time

The DEGA and LZMH compression algorithms reduce the data rate in a number of cases as described above. However, they are computationally more complex

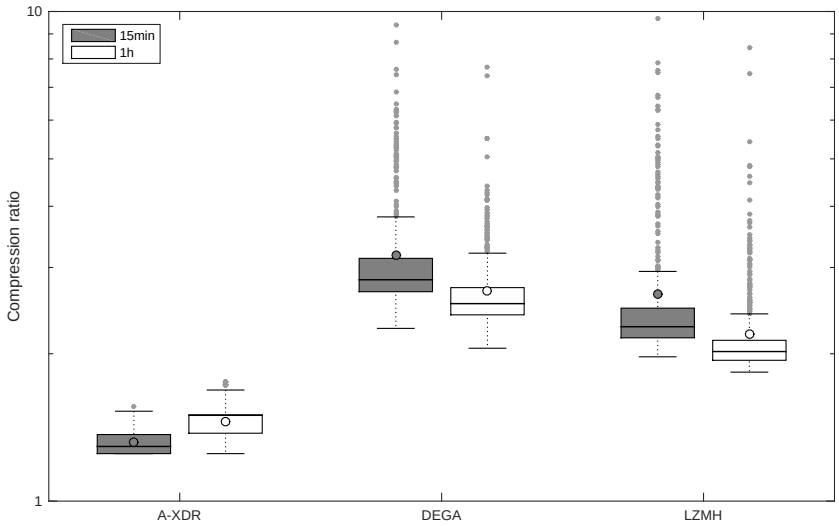

Figure 5. Compression performance of different algorithms compared to IEC 62056-21 value encoding for the SAG load data set at different data granularity levels.

than uncompressed data transmission. Thus, the additional code execution time has to be analyzed.

We measure the execution time similar to Unterweger and Engel [13]. Each channel (REDD data set) or household (SAG data set) is processed, as a whole, three times for cache warming and then five times for the actual time measurements. The five time results are averaged and divided by the number of data values in the processed channel/household to yield the average processing time per data value.

Again, the REDD and SAG data sets are evaluated separately due to their different data characteristics. A-XDR encoding is used as reference for uncompressed processing. All results have been obtained on a virtualized 64-bit *Ubuntu* 14.04 machine with *gcc* 4.8.2 running on an Intel Xeon W3503 CPU.

REDD data set Figure 6 shows an overview of the execution time per data value required by the A-XDR, DEGA and LZMH algorithms for the REDD data set. Despite the powerful CPU used for benchmarking, the processing time is in the microsecond range, i.e., most likely in the 10- or 100-microsecond range on less powerful hardware, e.g., smart meters. This can be considered feasible.

LZMH coding is clearly faster than DEGA coding. Surprisingly, it is, in the majority of cases, even faster than uncompressed A-XDR coding. This can be explained by the fact that both algorithms process data on a per-character basis, but A-XDR requires a conversion to floating point values which involves

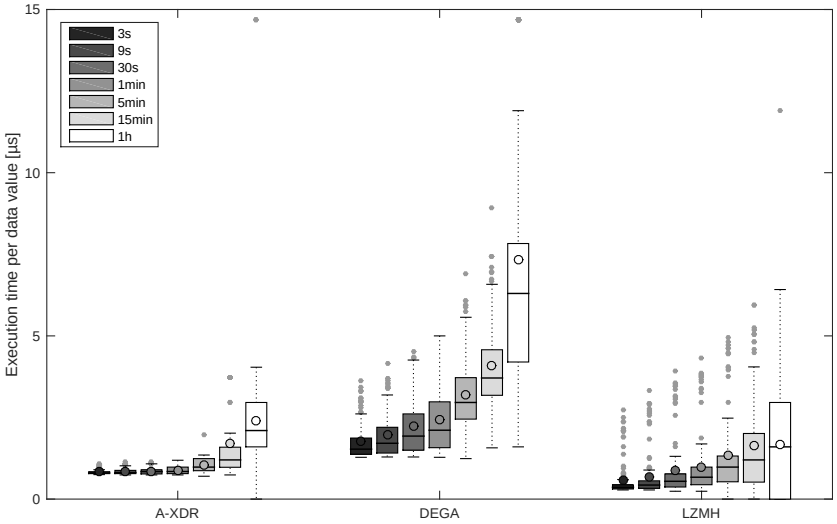

Figure 6. Execution times per value of different algorithms when compressing channels of the REDD load data set at different data granularity levels.

expensive floating point operations. These require about as much time as the whole compression step of LZMH coding, which is very compact.

Execution times increase at finer data granularity levels for all algorithms due to the relative increase in size of the input data. Since the (summed) values are larger in terms of magnitude, their IEC 62056-21 representations are longer. This explains the increased slopes of the median execution times for A-XDR and DEGA coding at coarser granularity levels. As LZMH coding does not convert the representation of the values, its slope is not affected by their magnitude, but by their redundancy, resulting in a smaller slope.

SAG data set Figure 7 shows an overview of the execution time per data value required by the A-XDR, DEGA and LZMH algorithms for the SAG data set. The visualization is identical to the one in Figure 6 for the REDD data set, with the exception of the data granularity range due to the 15-minute inter-value temporal distance of the original data.

The order of magnitude of the execution times is the same as for the REDD data set. However, the absolute values are lower for all algorithms due to the smaller (and therefore shorter) input values. Interestingly, also the differences between 15 min. and 1 h granularity are significantly smaller for the SAG data set than for the REDD data set. Again, this is due to the range (and therefore the length) of the input values.

The slope between the 15 min. and 1 h granularity levels for the SAG data set in Fig. 7 is comparable to the slope for 3 s to 5 min. granularity levels for the

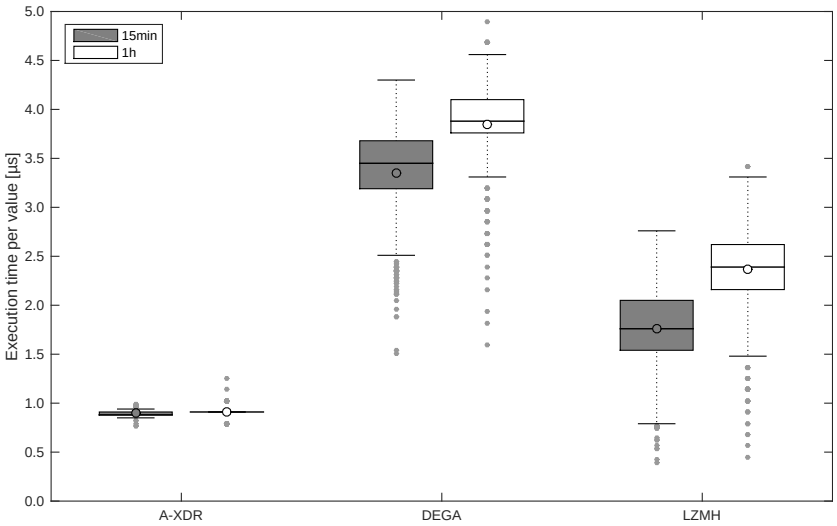

Figure 7. Execution times per value of different algorithms when compressing households of the SAG load data set at different data granularity levels.

REDD data set in Fig. 6. This shows that both, execution times and execution time differences are highly dependent on the input data length.

In addition, the differences in terms of execution time between DEGA and LZMH coding are smaller. This is due the lower compression efficiency of LZMH. This also explains why, in contrast to the execution times for the REDD data set (see Fig. 6), LZMH is slower than A-XDR coding for the SAG data set.

4 Conclusion

Load data from the evaluated data sets is compressible, but only at fine data granularity levels, e.g., 3 second intervals. At coarser granularity levels, compression becomes less effective or even futile, i.e., the reduction in data rate is practically insignificant compared to uncompressed encoding. This effect is stronger for the mains of a household than for per-room or per-device channels which have lower entropy and are therefore easier to compress. When compressing the tested load data sets, LZMH coding by Ringwelski et al. is recommended for the latter type of channels at fine data granularity levels. For coarser granularity levels as well as the mains, DEGA coding by Unterweger and Engel offers higher compression ratios at the cost of longer execution time.

5 Acknowledgements

The authors would like to thank Günther Eibl for his help in visualizing the compression ratio results. They would also like to thank their partner Salzburg AG for providing additional real-world load data.

The financial support by the Austrian Federal Ministry of Economy, Family and Youth and the Austrian National Foundation for Research, Technology and Development is gratefully acknowledged.

References

1. Coded Character Sets – 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII) (1986)
2. Distribution Automation Using Distribution Line Carrier Systems – Part 6: A-XDR Encoding Rule (2000)
3. Electricity Metering – Data Exchange for Meter Reading, Tariff and Load Control – Part 21: Direct Local Data Exchange (2002)
4. Efthymiou, C., Kalogridis, G.: Smart Grid Privacy via Anonymization of Smart Metering Data. In: Proceedings of First IEEE International Conference on Smart Grid Communications. pp. 238–243. Gaithersburg, Maryland, USA (2010)
5. Eibl, G., Engel, D.: Influence of Data Granularity on Smart Meter Privacy. IEEE Transactions on Smart Grid 6(2), 930–939 (2015)
6. Engel, D.: Wavelet-based Load Profile Representation for Smart Meter Privacy. In: Proc. IEEE PES Innovative Smart Grid Technologies (ISGT'13). pp. 1–6. Washington, D.C., USA (2013), http://dx.doi.org/10.1109/ISGT.2013.6497835
7. European Commission: Cost-benefit analyses & state of play of smart metering deployment in the EU-27. Tech. rep., European Commission Report (2014), http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52014SC0189&from=EN
8. Khan, J., Bhuiyan, S., Murphy, G., Arline, M.: Embedded zerotree wavelet based data compression for smart grid. In: Industry Applications Society Annual Meeting, 2013 IEEE. pp. 1–8 (2013)
9. Kolter, J., Johnson, M.J.: Redd: A Public Data Set for Energy Disaggregation Research. In: Workshop on Data Mining Applications in Sustainability (SIGKDD). pp. 1–6 (Aug 2011)
10. Ning, J., Wang, J., Gao, W., Liu, C.: A Wavelet-Based Data Compression Technique for Smart Grid. Smart Grid, IEEE Transactions on 2(1), 212–218 (2011)
11. Ringwelski, M., Renner, C., Reinhardt, A., Weigel, A., Turau, V.: The Hitchhiker's guide to choosing the compression algorithm for your smart meter data. In: 2012 IEEE International Energy Conference and Exhibition (ENERGYCON). pp. 935–940 (Sep 2012)
12. Sankar, L., Raj Rajagopalan, S., Mohajer, S., Vincent Poor, H.: Smart meter privacy: A theoretical framework. IEEE Transactions on Smart Grid 4(2), 837–846 (2013)
13. Unterweger, A., Engel, D.: Resumable Load Data Compression in Smart Grids. IEEE Transactions on Smart Grid 6(2), 919–929 (2015), http://dx.doi.org/10.1109/TSG.2014.2364686