Streaming Piecewise Linear Approximation for Efficient Data Management in Edge Computing

Romaric Duvignau, Vincenzo Gulisano, Marina Papatriantafilou, Vladimir Savic
Chalmers University of Technology (CTH), Sweden
{duvignau,vinmas,ptrianta}@chalmers.se,vladimir.savic@qamcom.se

ABSTRACT

In our digitalization era, where large and continuous data streams are produced by an ever increasing number of sensors, data retrieval and storage from edge devices is hampered when data volumes exceed the communication bandwidth of cyber-physical systems. Piecewise Linear Approximation (PLA), which trades space against precision by representing some portion of data by segments, could reduce the volume of transmitted and stored data and thus be beneficial to a wide range of edge/fog system architectures, saving communication bandwidth and addressing the aforementioned drawback. Porting a well-established tool such as PLA into the streaming paradigm is nonetheless challenging, and attention has to be payed to balance achievable compression, delays and imprecision. We analyze such challenges and propose different solutions to meet them. Our main contribution is a set of streaming PLA techniques that allow compression of the input data stream on the fly, tolerating a bounded maximum error. Through an experimental study based on real data, we demonstrate the superiority of our techniques in all sought aspects over preceding methods.

CCS CONCEPTS

• **Information systems** → *Data streams*; *Data compression*; • **Computer systems organization** → *Sensor networks*; Availability;

KEYWORDS

data compression, piecewise linear approximation, edge computing

ACM Reference format:
Romaric Duvignau, Vincenzo Gulisano, Marina Papatriantafilou, Vladimir Savic. 2019. Streaming Piecewise Linear Approximation for Efficient Data Management in Edge Computing. In *Proceedings of The 34th ACM/SIGAPP Symposium on Applied Computing, Limassol, Cyprus, April 8–12, 2019 (SAC '19)*, 4 pages.
https://doi.org/10.1145/3297280.3297552

1 INTRODUCTION

The recent constant increase in the quantity of continuous data produced by connected devices triggers numerous *big data* challenges, and sets an environment where reducing the volume of sensed data is more important than ever. High speed production of data records induced by large cyber-physical systems such as smart grids, homes and cars, moreover advocates *streaming* algorithms with small overhead. In this context, we identify two interesting scenarios for compression of continuously sensed data: reduction of (1) bandwidth and (2) storage consumption. An efficient compression mechanism could meet one of the prominent challenges in data retrieval and maintenance for large IoT systems: their limited storage and bandwidth could be partially discharged by decreasing the volume of data produced by sensors at the edge.

Computing a Piecewise Linear Approximation (PLA) of time series is a classical problem which asks to represent a series of timestamped records by a sequence of line segments while keeping the error of the approximation within some acceptable bound. In the extensive literature dealing with PLA (among others [2, 4, 5, 7, 9]), it is clearly stated that its main intend is to reduce the size of the input time series for both efficiency of storage and (later) processing. This entails a practical trade-off between a bounded *precision loss* and *space saving* for the time series representation. The advent of edge computing puts a brand new focus on this well-studied problem with new considerations, rather than merely only reducing the number of line segments of the representation as in most previous works about PLA. More concretely, how to produce such segments in a streaming, one-pass fashion and ensure the error bound is respected. Using PLA, we are able to extract in an online fashion an approximation stream with the same number of points of an input stream guaranteeing a fixed absolute maximal error. This is not covered, to the best of our knowledge, by traditional streaming approaches to compression for time series and other stream types [1, 3, 8] that do not guarantee a bounded error, are domain-specific and generally use a different error metric than absolute difference.

In this paper, we revisit and check suitability of PLA as an online compression tool with bounded error for streaming data produced at high speed from small computing devices. We present here fully streaming protocols for producing the PLA representation that, when compared to state-of-the-art PLA techniques, overpass all of them by one to several order of magnitude regarding the rate at which output data is generated for an increase level of compression. We also introduce a safe protocol with the guarantee that data will never get inflated by the compression process and a new original method to compute the PLA of the input stream producing significantly smaller errors for comparable compression rates.

In Section 2, we present an overview of PLA and our new PLA construction method. Using PLA in streaming environments is reviewed in Section 3, as well as our new streaming mechanisms that generate in an online fashion the compressed data. The subsequent section presents a thorough evaluation of our new methods compared to existing PLA techniques on real world time series datasets, while Section 5 concludes the paper.

SAC '19, April 8–12, 2019, Limassol, Cyprus

ACM ISBN 978-1-4503-5933-7/19/04…$15.00
https://doi.org/10.1145/3297280.3297552

2 PIECEWISE LINEAR APPROXIMATION

We consider here the problem of computing a piecewise linear approximation of a data stream while tolerating a prescribed maximum error of ε, and more precisely the *online* version of the problem, *i.e.* (i) the time series is processed one record at a time, the output line segments are produced along the way, and (ii) the projected points along the compression line segments always fall within ε from the original tuples. Figure 1 gives an example of a PLA, where original data points are crossed, reconstructed points bulleted, approximation error as vertical bars and information about the PLA streamed along the processing of the input stream.

2.1 Related work

We briefly introduce here the most popular methods to construct the PLA of a time series, summarized in Table 1 by: worst case/amortized complexity, latency introduced in the reconstruction, processing time, optimality criteria for the PLA construction, size of the PLA information generated on the output stream, number of line segments and overall compression on a relative scale −− (worst), −, +/−, +, ++ (best), based on previous experimental surveys complemented by our results. Among the evaluated methods[1], *SwingFilter* is a simple greedy construction, *Continuous* produces the optimal PLA in terms of the number J of continuous segments (*i.e.* consecutive ones must share their endpoint), *Convex-Hull* the optimal PLA in the number D of discontinuous segments, *MixedPLA* the PLA minimizing $2J + 3D$, and finally *Linear* is a new heuristic method.

Most PLA methods follow a *greedy* approach: a longest possible approximation is constructed from an initial point, till encountering a *break-up point* that cannot be approximated with the previous sequence, then it restarts from that point on. In order to efficiently (in $O(1)$ amortized time) find such a break-up point, all methods revolve around calculating extreme slope lines (bounding possible approximation slopes) in order to quickly evaluate if adding a new input point to the approximation is feasible. The **Convex-Hull** method finds the optimal PLA in number of discontinuous segments by surprisingly only following the greedy approach. From a fixed starting point, the longest approximation segment can be efficiently calculated by maintaining two convex hulls[2] while processing new inputs (Figure 2 displays both hulls for some input sequence) and using the hulls to update the extreme slope lines. Once a break-up point is found, a segment line is generated (average of possible slopes) and the algorithm restarts.

[1]For a complete description of each method, refer to the references given in Table 1.
[2]The *convex hull* of a set of points is the minimum convex set containing all of them.

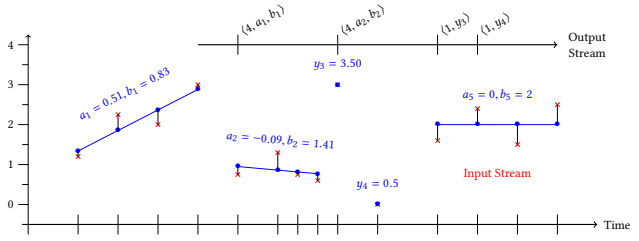

Figure 1: Example of a PLA using maximum error $\varepsilon = 0.5$.

2.2 A new PLA method: Linear

PLA methods developed so far optimize the representation based only on the number of disc/continuous segments. Hence, the intrinsic PLA trade-off compression versus loss of accuracy is only encoded in those methods as "compression" (for one fixed criterion) versus maximum error. Thus individual errors that better characterize the precision loss are lost in the process. We devise here a simple and efficient novel PLA method aiming at reducing those errors while preserving a fixed maximum error. The method follows the greedy approach but uses the best-fit line obtained by a simple regression model for the approximation. Such a line of slope $a = \text{cov}(t, y)/\text{var}(t)$ and y-intercept $\mu_y - a\mu_t$, is calculated in an online fashion from the covariance $\text{cov}(t, y) = (\sum_i t_i y_i)/n - \mu_t \mu_y$, variance $\text{var}(t) = (\sum_i t_i^2)/n - \mu_t^2$, averages μ_t and μ_y and n the number of compressed points; those quantities are maintained keeping track of the different sums and n. Once the line has been calculated, a second step is to decide whether it is associated with a correct approximation for the points involved, or it violates the error requirement for some of them. The simplest way to check that, is to verify sequentially that condition for each currently approximated point, entailing an inappropriate linear cost for the verification. To mitigate this issue, we maintain the two convex hulls associated with the current approximation after each new arrival (as the *Convex-Hull* method). The current best-fit line is then verified by traversing both hulls and checking that the line stays above the lower hull and below the upper one (illustrated by Figure 2 where the plain line is the best-fit one for 10 points, the dashed one for 11 points). The hulls themselves may have a linear size in n, but in practice they are rarely larger than a few units (cf [5, 9]).

3 STREAMING PLA AND PROTOCOLS

Context and Metrics. We identify two situations in the context of edge computing where a simple compression procedure as PLA can be easily implemented and greatly beneficial:

- (1) **Reduce bandwidth consumption:** in order to decrease the volume of data sent over the network from a large set of embedded devices to a distant datacenter monitoring them;
- (2) **Reduce storage consumption:** in order to decrease the volume of data stored by sensors at the edge before transmission/processing (or the amount of data saved by the datacenter).

These scenarios form a common framework where an input stream is processed and a compressed one is generated and transmitted and/or stored, to be later (or on reception) reconstructed in an online fashion. They lead us to investigate the performance associated with the full compression process for existing PLA algorithms

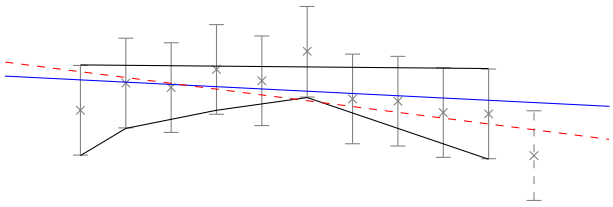

Figure 2: Best-fit lines of a set of points.

Table 1: Summary of the evaluated PLA methods and their respective features.

PLA method [ref] – Key	Complexity	Latency	Proc. time	Opt. criterion	Size of PLA output	No. of segm.	Compr.
SwingFilter [2, 7] – Sw	$O(1)/O(1)$	++	++	none	2 doubles	−−	−−
Convex-Hull [2, 9] – Sl, C1/C2	$O(n)/O(1)$	+/−	+/−	# disc. segm.	3 doubles	+	+
Continuous [4] – C	$O(n)/O(1)$	−	+/−	# cont. segm.	2 doubles	−	+
MixedPLA [5] – M	$O(n)/O(1)$	−−	−−	global size	2 or 3 doubles	++	++
Linear (new) – L1/L2	$O(n)/O(n)$	+	+	none	2 doubles + 1 byte	+/−	+/−

and our new solutions, independently of concrete implementation aspects, with respect to the following three aspects:

- **Compression:** capturing the space gain between the compressed stream and the original input stream; it corresponds to the ratio between the amortized size in bytes taken by each tuple in the compressed stream versus the original size.
- **Latency:** capturing the delay in number of records introduced by the compression process; it measures the logical latency corresponding to the number of points that occurred between the emission of an input tuple and its reconstruction.
- **Error:** capturing the precision loss by considering continuous and average error rather than only maximum error as in previous works; it corresponds to the absolute differences between approximated tuples and their original counterpart.

It is important for us to highlight that decision making should be considered irreversible, as once information about a new line segment has been transmitted over the network, it is hardly modifiable. Hence, particular attention will be given to peak values of these metrics in addition to the aggregated average.

Streaming PLA Protocols

Known studies of PLA methods in the literature do not focus on the cost of when and how the information is output and reconstructed. In our study that takes place in streaming environments (where the performance of any tool depends both on its semantics and its algorithmic implementation), it is, however, particularly important. The compression's performance will indeed depend both on the algorithm chosen to construct the PLA and on how results are output, *i.e.* the corresponding *streaming PLA protocol*. Below we outline several alternatives:

Implicit: A streaming protocol implicitly assumed in previous PLA works, consists in using as output information directly the segments' endpoints of the PLA representation. This allows to reconstruct quickly tuples, however, this strategy may generate an *inflated stream* when compression is not possible (up to 2 - 3 times more data is produced in the worst case).

We introduce two new protocols to improve compression and latency as well as to avoid inflation. They differ from the *implicit* one through: (1) introducing "singleton" tuples, that are not part of any approximation segment, allowing (partial) deactivation of PLA compression; (2) using a counter n for indicating the segment length, removing the dependency on including timestamps in the output stream. The latter leads to a potential gain in compression, as n may need much less space to be represented compared to timestamps (n is bounded, also implying a maximum latency).

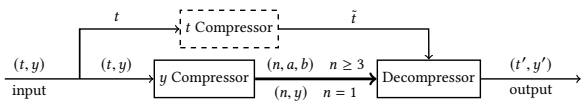

Figure 3: SingleStream protocol flowchart.

SingleStream: In this protocol (cf. Figure 3), the stream of input tuples (t, y) is half-duplicated such that the t values form a separate stream (that will be stored or compressed), then fed to a PLA compressor that generates line segments which are output if representing more than 3 values; else the compressor is restarted after outputting the first tuple as a singleton. Each segment is represented by a triplet (n, a, b) where (a, b) are the linear coefficients and n the segment length in tuples. Singleton values are generated on the same stream, preceded by a counter set to 1. The compression is upper bounded by some N which allows n to be represented in $\log_2(N)$ bits. To reconstruct the input stream from timestamp t_i, one checks the value of the next counter n on the compression stream: if $n = 1$, then the reconstructed record (t_i, y) is generated (where y is the value following the counter); otherwise ($n \geq 3$), the two coefficients a and b given right after the counter are used to generate the sequence $(t_j, a.t_j + b)_{i \leq j \leq n}$ of n reconstructed tuples consuming along the way the $n - 1$ timestamps succeeding t_i.

TwoStreams: The protocol uses two streams to encode a single input stream of (t, y) tuples, by forwarding the latter to a segment line compressor, which produces: one stream of uncompressed y-values and one of line segments represented by quadruplets (t_0, n, a, b), where t_0 is the starting time of the segment, whereas n, a, b have the same meaning as in the previous protocol. A minimum of 4 values are compressed by line segments, redirecting otherwise the y-values to the singleton stream. This ensures data inflation never occurs and can be of interest in streaming situations where the quantity of transmissions cannot be increased by any means. By comparing the current timestamp t to t_0, one can easily deduce whether to consume a value on the singleton stream or reconstruct the n values associated with the current segment.

4 EVALUATION

We thoroughly evaluate against the metrics introduced in § 3 using datasets of realistic streamed data the five PLA methods of Table 1. The four classical PLA methods use the implicit protocol described in the previous section, whereas C1/L1 is associated with SingleStream protocol (SS) and C2/L2 with TwoStreams (TS).

Datasets: We use three different datasets: (a) *GPS* consists of trajectories collected within the scope of the (Microsoft Research Asia) Geolife project [10]; (b) *LiDAR* consists of LiDAR scans from the *Ford Campus* dataset [6]; (c) *URBAN* consists of average vehicle speed measured at different locations in the city of Rio de Janeiro.

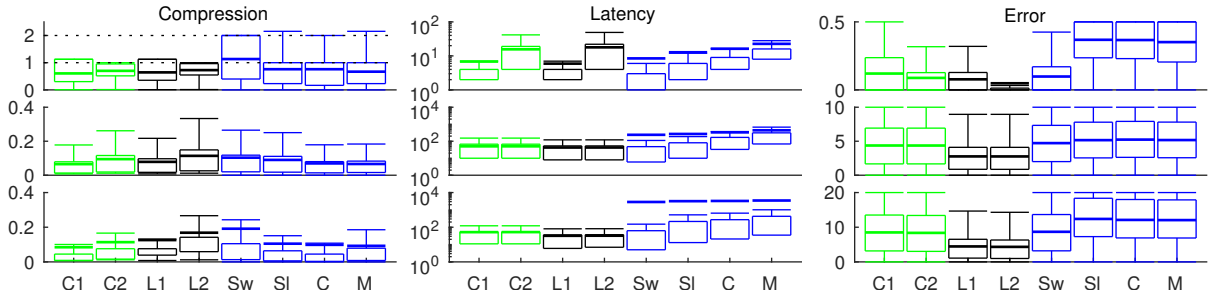

Figure 4: Statistics for 3 levels of compression for URBAN (low), GPS (medium) and LiDAR (high) datasets.

Setup: When using SS and TS, the compression is bounded to 256 values per segment and the segment length is coded on a single byte. All other input data numbers require at least double precision in order not to break the error threshold and thus use 8 bytes.

Results: Detailed results for three levels of compression are depicted in Figure 4, whereas the remaining ones are omitted due to space constraints. Each row presents a different error threshold (low, medium and high compression) for one of the dataset. Statistics are shown by a box plot (first and third quartiles with whiskers extending to 1.5 the interquartile range) and a bold line for the average. Figure 5 presents the ranking of the different PLA methods and protocols obtained by summing the mean value of each statistics in all the experiments and normalizing them by the highest value. Our thorough evaluation shows that:

- **Compression:** TS is the only safe protocol that never inflates data (especially noticeable at low compression phase); SS achieves the best compression (only comparable with *MixedPLA*).
- **Latency:** TS/SS achieve smaller latencies with SS the lowest (delays are significantly smaller than with previous methods).
- **Error:** *Linear* produces the smallest errors (with comparable delays and compression ratio to the other evaluated methods); New protocols present better trade-offs compression/error (most errors are smaller with our implementations).

An overall key comment is that the *most influential factors* on the different metrics are the protocol (first) and the PLA method (second). The benchmark confirms that applying PLA compression using our protocols with a bounded segment length achieves comparable compression level while showing significant improvements (up to two orders of magnitude) on the measured latency of the compression. To be noted as well, due to the possibility to output singleton values, errors produced by our protocols are on average smaller than with any other PLA methods. Finally, the *Linear* method presents errors in general about half smaller than what is currently achieved by state-of-the-art algorithms, while the compression loss is mitigated by using our streaming protocols.

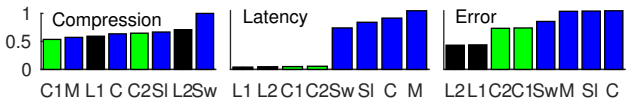

Figure 5: Ranking (from best to worst) of the PLA methods and protocols combinations.

5 CONCLUSIONS

We suggest using PLA to reduce the volume of streamed data in large IoT systems. Moreover, we introduce several streaming techniques and a novel PLA method to balance the trade-offs when using PLA as a compression tool. Our experimental results show that for comparable compression ratios, our techniques reduce significantly both the reconstructed stream's delays and individual errors when compared to baseline PLA methods. Our work demonstrates the benefits of using streaming PLA compression, which can help decrease significantly the communication needs in large cyber-physical systems such as vehicular swarms or smart grids.

ACKNOWLEDGMENTS

Supported by the projects: OODIDA, program Strategic Vehicle Research and Innovation (DNR 2016-04260), Swedish Agency for Innovation Systems (VINNOVA/FFI); FiC, grant nr. GMT14-0032, Swedish Foundation for Strategic Research; INDEED and STAMINA by the Chalmers Energy AoA framework; HARE, grant nr. 2016-03800, Swedish Research Council (Vetenskapsrådet).

REFERENCES

[1] Graham Cormode, Minos Garofalakis, Peter J Haas, Chris Jermaine, et al. 2011. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends® in Databases* 4, 1–3 (2011), 1–294.
[2] Hazem Elmeleegy, Ahmed K Elmagarmid, Emmanuel Cecchet, Walid G Aref, and Willy Zwaenepoel. 2009. Online piece-wise linear approximation of numerical streams with precision guarantees. *VLDB Endowment* 2, 1 (2009), 145–156.
[3] Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. 2016. *Data Stream Management: Processing High-Speed Data Streams.* Springer.
[4] S Louis Hakimi and Edward F Schmeichel. 1991. Fitting polygonal functions to a set of points in the plane. *CVGIP: Graphical Models and Image Processing* 53, 2 (1991), 132–136.
[5] Ge Luo, Ke Yi, Siu-Wing Cheng, Zhenguo Li, Wei Fan, Cheng He, and Yadong Mu. 2015. Piecewise linear approximation of streaming time series data with max-error guarantees. In *31st Int'l Conf. on Data Engineering (ICDE).* IEEE, 173–184.
[6] Gaurav Pandey, James R McBride, and Ryan M Eustice. 2011. Ford campus vision and lidar data set. *The International Journal of Robotics Research* 30, 13 (2011), 1543–1552.
[7] Tom Schoellhammer, Eric Osterweil, Ben Greenstein, Mike Wimbrow, and Deborah Estrin. 2004. Lightweight Temporal Compression of Microclimate Datasets. In *29th Annual IEEE Int'l Conf. on Local Computer Networks.* IEEE Computer Society, 516–524.
[8] Mihaela Van der Schaar and Philip A Chou. 2011. *Multimedia over IP and wireless networks: compression, networking, and systems.* Elsevier.
[9] Qing Xie, Chaoyi Pang, Xiaofang Zhou, Xiangliang Zhang, and Ke Deng. 2014. Maximum error-bounded Piecewise Linear Representation for online stream approximation. *The VLDB Journal* 23, 6 (2014), 915–937.
[10] Yu Zheng, Xing Xie, and Wei-Ying Ma. 2010. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* 33, 2 (2010), 32–39.