Reading Notes

TVStore Yanzhe An and Wang (2022)

Abstract

A pressing demand emerges for storing extreme-scale time series data, which are widely generated by industry and research at an increasing speed. Automatically constraining data storage can lower expenses and improve performance, as well as saving storage maintenance efforts at the resource constrained conditions. However, two challenges exist: 1) how to preserve data as much and as long as possible within the storage bound; and, 2) how to respect the importance of data that generally changes with data age.

To address the above challenges, we propose time-varying compression that respects data values by compressing data to functions with time as input. Based on time-varying compression, we prove the fundamental design choices regarding when compression must be initiated to guarantee bounded storage. We implement a storage-bounded time series store TVStore based on an open-source time series database. Extensive evaluation results validate the storage boundedness of TVStore and its time-varying pattern of compression on both synthetic and real-world data, as well as demonstrating its efficiency in writes and queries.

Introduction

- "Time series databases are becoming the most popular type of databases in recent years." Newest data seems to suggest otherwise? gmbh (2023)
- "... the fast increasing volume of time series data has placed an unprecedented requirement on computing resources, especially storage space [6, 79]." pg 83
- "as the significance of time series data is highly correlated with the age of the data [22,37,89], it is desirable to have a storage management strategy that takes data ages into account [3, 7]." pg 83
- " Significant prior work has addressed the storage-control problem by compression, which can be lossless or lossy. Lossless compression [10, 33, 57, 71, 73] preserves the complete data, but its achievable upper bound on compression ratio [93] might not be satisfactory for applications." pg 83
- "Hence, time series databases commonly control storage consumption by directly discarding data older than a given time [43] or exceeding a storage threshold [67]. But discarding historical data causes a loss [94]." pg 83
- "Another common approach is to exploit lossy compression [15,41,65], which preserves partial data and trades off precision for space. But existent approaches to lossless and lossy compression are only best-effort about the final size of compressed data size [13, 24, 99]." pg 83
- *Problem statement* "We consider the problem of automatically bounding the storage of a time series store by compression. To enable this, our key insight is that time series data can be compressed losslessly or lossily according to its importance, which is in turn related to its age, as users commonly accept information loss on less important old data [12, 14, 23, 38, 40]." pg 83
- "The goal is to reduce computing resource consumption and improve performance." pg 84

Background and Motivation

- "…mounting demands have emerged for keeping time series data for future analysis [94]. But time series data are generated at a growing speed that is outpacing the increase of computing capabilities [17, 79]. Many application scenarios cannot afford enough computing resources such as storage and network bandwidth to accommodate the processing needs for time series data." pg 84
- "Sensors of a connected car can generate about 30 terabytes (TB) of data per day [62, 77]…. Since a 30TB disk can cost around $1200, a month's worth of data can fill up a 960TB disk, causing a cost of $30,000." pg 84
- "In the oil and gas industry, a typical offshore oil platform generates more than 1TB of data [19] daily. But common data transmission via satellite connection allows only a speed from 64 Kbps to 2Mbps for these offshore oil platforms. If all data are transmitted back for processing, it would take more than 12 days to move 1 day's worth of data to the processing backend [8]." pg 84
- "… cosmological simulations generate petabytes of data per simulation run [34] and climate simulations generate tens of terabytes per second [29]." pg 84
- "Data reduction is necessary to enable data processing and analytics within a reasonable amount of resource and time [92]." pg 84
- "The importance of time series data changes along with time, as reflected by applications' favoring recent data over old data [5, 18, 31], or favoring some events at certain moments over others [49,83]." pg 85
- "As a result, we have seen a plethora of research on data series analysis and prediction considering the timechanging pattern [9,22,36,37,89]." pg 85
- "As time series data can be identified by timestamps, we use a time-dependent function to denote the changing importance of data. Hence, the compression ratios can also be deduced from the function." pg 85
- "We must deduce the proper moments for compression initiation when 1) it is not too late that the storage space is exceeded during compression; and, 2) it is not too early that unnecessary compression is applied to some recent data or that an improperly high compression ratio is used." pg 85
- "Challenges: As a result, two main challenges exist in automatically bounding the time series storage by timevarying compression: 1) how time-varying compression can be executed on an ever-increasing volume of data and with error bounds computed, when the compression ratios keep changing… and, 2) how to automatically decide the conditions for running time-varying compression such that storage space is always bounded but not too much… " pg 85
- time-varying compression (TVC)
- "To guarantee that data are compressed to the compression ratio sequence r_1,r_2,…,r_k, TVC groups r_i data chunks into the ith segment. Each segment is then compressed to an output chunk. Hence, the ith chunk of the compression output has a compression ratio r_i, complying to the definition of *r(t)*." pg 86
- "…given the data chunks to be compressed in round n, TVC *virtually* decompresses them, by mathematical mapping, to the original raw data chunks for computing the sequence of compression ratios. Then, the conditions for ratio sequencing are considered." pg 86
- "Virtual decompression enables the generation of a compression ratio sequence based on the original raw data even after rounds of compression. Only by virtual decompression could the compressed data always follow the time-dependent function *r(t)*'s definition. Otherwise, data can only be compressed following the exponentially decaying pattern, as compression on compressed data leads to the multiplication of compression ratios. This would limit the applications of TVC, as *r(t)* can only be an exponential function." pg 86

- “Keeping data at an extremely high error rate is no better than discarding it. Therefore, TVC allows users to specify a compression ratio r_{max} or an error rate e_{max}. TVC automatically discards data compressed at a ratio higher than r_{max} or at an error rate larger than e_{max}.” pg 87
- “Lossless and lossy compression methods exist for time series data. Lossless compression can reconstruct the original data accurately. For lossless compression, specialized compressors for integer [10, 33, 55, 73] and for floating-point values [57, 71] outperform general-purpose compressors [20,28,32,58,66]. It is common practice that general-purpose compression is applied along with specialized compressors [43, 54, 96] in time series stores.” pg 94
- “To manage the ever-increasing volume of time series data, most time series stores either have a native architecture to support a distributed deployment such as InfluxDB [43], or exploit a distributed storage for scalability, e.g., KairosDB [47] on Cassandra [53], TimescaleDB [90] on PostgreSQL database [72], Druid [100] on HDFS [35], BtrDB [4] on Ceph [97], Chronix [54] on Solr [84], and Peregreen on an object store [94].”
- “Besides lossless and lossy compression, time series databases commonly exploit the data retention policy to reclaim storage space by removing data exceeding a time period [42] or exceeding a storage quota, e.g., RRDtool [67], for further storage reduction.”

MSc Thesis Steinarsson (2013)

Abstract

- “The focus of this thesis is to explore methods for downsampling data which can be visualized in the form of a line chart, for example, time series.” pg vii

Introduction

- “If values on the x-axis are evenly spaced over time, the data is most often referred to as a time series.” pg 1
- “A line chart entailing a data point distribution high in density proffers limited information to an observer of that chart.” pg 1
- “. . . the focus of this thesis is mainly to explore downsampling algorithms which return a subset of data points existing in the original data.” pg 2
- “sometimes it might suffice to use only every other data point or maybe every tenth data point, depending on the data and canvas size. Still, such a method is only suitable if the data is “smooth” in nature and has little or no fluctuations. If the data is somewhat irregular, then using only every nth data point to draw the line chart will almost guarantee that many interesting peaks and troughs will not be included in the chart. . . ” pg 2
- “. . . it is important to emphasize that the downsampled data is only intended to visually represent the original data for human observation and not data analysis, statistical or otherwise. . . When processing information for visual representation, it is only important to retain the data which offers the most information and is actually perceived by people, the rest can be discarded.” pg 2
- “The primary objective is to design and implement a few algorithms that can effectively downsample any data which can be directly drawn as line chart, with special emphasis on time series. The downsampling algorithms then need to be compared in regard to several factors. Those factors include but are not limited to efficiency, complexity and correctness.” pg 3
- “The second objective is to be able to say something about the human perception on downsampled data drawn as a line chart. That includes conducting a simple comparative survey using different downsampling algorithms.! pg 3

Intutive Downsampling Algorithms

- *Mode-Median-Bucket:* "The algorithm uses primarily two methods, mode and median, to govern which data points are returned, thus the name Mode-Median-Bucket. The bucket part in the algorithm name refers to the data being split up into buckets, each containing approximately equal number of data points. The algorithm then finds one data point within each bucket as follows. If there is a single y-value which has the highest frequency (the mode) then the leftmost corresponding data point is selected. If no such data point exists a point corrosponding to the median of the y-values is selected. An exception to these rules is when a global peak or a global trough is found within the bucket. This is to ensure that those points are included in the downsampled data." pg 5

- "One of the most obvious issues with this algorithm is that it is very likely to exclude local peaks and troughs within each bucket (see figure 2.1). The reason for this is because it does not take into account what the y-value actual is, only how frequent it is within each bucket. The only exception is when the global peak or trough occur in the bucket."pg 6

- "Another minor issue arises if the global peak and trough both occur in the same bucket. Then the peak is always used no matter what the value of the trough is. Incidentally, the absolute height of the trough might be much greater than that of the peak." pg 7

- *Min-Std-Error-Bucket:* "It is based on linear regression and uses the formula for the standard error of the estimate (SEE) to downsample data. The SEE is a measure of the accuracy of predictions made with a regression line when a linear least squares technique is applied. The greater the error the more discrepancy between the line and the data points. Of course one line through all data is not very useful. Thus, the original data needs to be split up into buckets and then multiple lines calculated between points in the buckets." pg 8

- "it is worth mentioning that it is not really a practical solution, merely an exploration of using certain concepts, e.g., the standard error. Even if the algorithm produces a statistically good line chart, it is not a very good visual representation since it smooths out many of the peaks and troughs of the line chart." pg 8

- "In downsampling, it is only normal to miss some minor fluctuation but when major fluctuations are skipped, the line chart can suffer perceptually, giving a poor representation of the original chart." pg 11

- "Another downside to this algorithm is that it has a lot of loose ends. There are other ways to solve the dynamic optimization problem and maybe a more greedy approach is adequate. Finding the absolute minimal sum of the standard errors might not be necessary when a fairly low sum might suffice." pg 12

- *Longest-Line-Bucket:* "It starts off exactly the same [as the MSEB algorithm], splitting the data points into buckets and calculating lines going through all the points in one bucket and all the points in the next bucket... The main difference is that instead of calculating the standard error for each line segment it simply calculates its length (Euclidean distance between the two points defining the line). Then, as with the MSEB algorithm, the points and lines segments are converted to a directed acyclic graph (DAG) and the weight of an edge is the length of the corresponding line segment... All that remains is to find the longest path through the graph. The path will contain one point per bucket which forms the longest total line length through all the bucket." pg 13

- "Finding the longest path in a general graph is a NP-Hard problem and cannot be computed in polynomial time. However in the case of the graph being a DAG, the edge weight can simply be changed to its negation, thus changing it to a shortest path problem. This problem can then be solved with dynamic programming in exacly the same way as done in the MSEB algorithm, except that maximization rather than minimization is applied." pg 13

- "…like with the MSEB algorithm the problem with this algorithm is how complicated and inefficient the current implementation is. If this algorithm is to become a practical option in a real application, it would need to be simplified and optimized." pg 14

Cartographic Generalization

- "There are several well known techniques in cartographic generalization regarding polyline simplification. The evaluation of these algorithms is still an issue of active research and there is large number of different spatial measurements which can be used as criteria [9]." pg 15
- "One of the most common line simplification method is the Douglas-Peucker algorithm [5]. It selects the point that is over a specified threshold and furthest away from an imaginary line which is initially between the first and the last point of the polyline. The algorithm then calls itself recursively with the polylines on both sides of the selected point (including the selected point) until all the points have been either selected or discarded (fall under the threshold). When the recursion is done the points which were selected define the simplified polyline." pg 15
- "Another, more recent, algorithm to simplify lines is called the Visvalingam–Whyatt algorithm [11]. The basic idea behind this algorithm is to give all the points a certain rank based on their significance. The significance of a point is the area size of a triangle which it forms with its two adjacent points on each side. This is usually called the effective area of a point… The least significance points are then excluded in the simplified line and the effective areas of the remaining adjacent points recalculated as it has changed." pg 16
- "One obvious restriction is that the algorithm should not skip too many data points in a row. That is because that would result in long line segments over parts in the line chart with minor fluctuations. Such minor fluctuations are indeed information and their presence has a noticeable perceptual impact for a line chart." pg 16
- "The perception of line charts and other data visualization techniques has been the topic of discussion and research for a long time [3, 6]. The scope of the research is very broad and often borders on the field of psychology [2]. For now it suffices to say that when downsampling data to be displayed as line chart, it is important to retain as much visual characteristics of the line chart as possible and not suggest any false patterns because minor fluctuations are indeed a type of visual characteristic." pg 17

Largest Triangle Algorithms

- *Largest-Triangle-One-Bucket:* "This algorithm is very simple. First all the points are ranked by calculating their effective areas. Points with effective areas as null are excluded. The data points are then split up into approximately equal number of buckets as the specified downsample threshold. Finally, one point with the highest rank (largest effective area) is selected to represent each bucket in the downsampled data." pg 19
- "One issue with this algorithm is that the ranking (effective area) of a point only depends on its two adjacent points. It is perhaps not an apparent issue but it can lead to bad representation of the original line chart in certain cases." pg 20
- *Largest-Triangle-Three-Buckets:* "With the algorithm discussed in this section, the effective area of a point does not depend on the position of its two adjacent points but on the points in the previous and next buckets, making the possible effective area much larger. The first step is to divide all the data points into buckets of approximately equal size. The first and last buckets however contain only the first and last data points of the original data… This is to ensure that those points will be included in the downsampled data. The next step is to go through all the buckets from the first to the last and select one point from each bucket. The first bucket only contains a single point so it is selected by default. The next bucket would then normally contain more than one point from which to choose." pg 21

- “The algorithm works with three buckets at a time and proceeds from left to right. The first point which forms the left corner of the triangle (the effective area) is always fixed as the point that was previously selected and one of the points in the middle bucket shall be selected now… add a temporary point to the last bucket and keep it fixed. That way the algorithm has two fixed points; and one only needs to calculate the number of triangles equal to the number of points in the current bucket. The point in the current bucket which forms the largest triangle with these two fixed point in the adjacent buckets is then selected.” pg 22
- “There is still the matter of how this temporary point in the next bucket should be decided. A simple idea is to use the average of all the points in the bucket.” pg 23
- “The biggest problem is not really how the points are selected within the buckets but rather how the points are divided into buckets. This algorithm uses roughly equal sized buckets (except for the first and last buckets) to make sure a point is selected for every fixed interval on the x-axis. The problem is that some line charts have somewhat irregular fluctuations.” pg 24
- “The problem is that not all buckets can be visually represented fairly with just one point and some buckets might not even need to be represented at all (if the local fluctuation is very small). This is at least the case if all the buckets have approximately the same number of points and the algorithm selects only one point. from each bucket. This problem also exists in all the previous algorithms which rely on this roughly equal bucket dividing concept.” pg 24
- *Largest-Triange-Dynamic:* “…this algorithm does not rely on equal size buckets but allows a dynamic bucket size. If a part of the line chart is fluctuating greatly, the buckets become smaller; and if another part is relatively calm, the buckets become larger. The algorithm is really an attempt to address the biggest problem with the LargestTriangle-Three-Buckets (LTTB) algorithm… ” pg 25
- “The first step is to assign a number to all buckets which indicates whether a bucket needs to be smaller or larger, if only one point is supposed to represent the bucket. An obvious way to calculate this number is to apply a simple linear regression for all buckets.” pg 26
- “If the SSE for a bucket is relatively high, it means that the data points within the bucket would most likely be better represented as two buckets. If however the SSE is relatively low for a bucket, it can be merged with either one of the adjacent buckets, if one of them also has a relatively low SSE. After all the initial buckets have been ranked, the next step is to either split or merge them accordingly.” pg 26
- “After a given number of iterations the algorithm needs to halt (on some predetermined condition).” pg 27
- “For example, if 1,000 points are to be downsampled down to 900 points, the algorithm runs 11 iterations. If however the 1,000 points need to be downsampled down to 50, the algorithm runs 200 iterations.” pg 28
- “Although this algorithm can produce a good visual representation of a line chart, it is not without issues. As mentioned before, this algorithm gives the best results if the data is irregular. When the data is highly regular over the whole line chart, the algorithm appears no better than the LTTB algorithm (it might even sometimes be a little worse).” pg 29
- “Perhaps the main problem has to do with how this algorithm could be optimized for better performance because it is currently rather slow in comparison with the LTTB algorithm.” pg 30
- “Another issue is determining the halting condition. As it is implemented, the halting condition is calculated with a simple formula which takes in the data point count and downsample threshold. It does not take into account the nature of the line chart.” pg 30

Survey

- “In order to collect more data the survey was designed in such a way that people were asked to order the downsampled lines charts from the best representation to the worst. Thus, the survey would yield some information about all choices of the downsampled line charts and also how good or bad a representation a downsampled line chart is relative to the other choices.” pg 31

Questions

- “In order to keep the survey short it only consisted of nine questions (not including two questions about participant’s age and education). In each question the participant was shown the original line chart and a number of downsampled line charts (displayed in a random order for each participant) using different methods and settings. The participant’s task was then to order the downsampled line charts from the best to the worst representation relative to the original line chart.” pg 32
- “One of the assumptions of this thesis is that both aesthetics and resemblance matter and this survey was designed to give some insight into what a good visual representation really means for people in the context of downsampled line charts.” pg 33

Participants

- “The survey was open to everybody and 58 participants took part over the course of about two weeks. DataMarket advertised the survey on their Twitter page and the University of Iceland also sent out emails to all students in computer science and software engineering.” pg 35

Survey Results

- The Largest-Triangle-Dynamic algorithm yielded the best results according to the survey. This comes as no surprise since extreme downsampling of irregular data is one of its strong points. Not far behind was the Longest-Line-Bucket algorithm. That was a bit surprising seeing as it is one of the intuitively designed algorithms, with little theoretical foundations (at least in the papers I reviewed).” pg 43

Overview Comparison of the Downsampling Algorithms

- Speed and scalability pg 45
- Complexity and portability pg 45
- Correctness pg 46

Conclusion

- “As it turned out, the Largest-Triangle-Three-Buckets algorithm (see section 4.2) was both efficient and produced good results for most cases. Already it is being put to use by DataMarket and is reported to be a great improvement over their previous method of downsampling data (see section 2.1).” pg 49
- “For future work, it might be prudent to explore other polyline simplification algorithms applied in cartographic generalization to adapt them for the purpose of simplifying line charts. Furthermore, similar to cartographic generalization, what is needed is a deeper understanding of how line charts are perceived [11], what characteristics have the most perceptual impact and to what degree.” pg 50

Change Point Detection Algorithms Burg and Williams (2022)

Abstract

- "Change point detection is an important part of time series analysis, as the presence of a change point indicates an abrupt and significant change in the data generating process." pg 1
- "While many algorithms for change point detection have been proposed, comparatively little attention has been paid to evaluating their performance on real-world time series." pg 1
- ". . . we present a data set specifically designed for the evaluation of change point detection algorithms that consists of 37 time series from various application domains. Each series was annotated by five human annotators to provide ground truth on the presence and location of change points. We analyze the consistency of the human annotators, and describe evaluation metrics that can be used to measure algorithm performance in the presence of multiple ground truth annotations. Next, we present a benchmark study where 14 algorithms are evaluated on each of the time series in the data set." pg 1

Introduction

- "Moments of abrupt change in the behavior of a time series are often cause for alarm as they may signal a significant alteration to the data generating process. Detecting such change points is therefore of high importance for data analysts and engineers." pg 1
- "Existing work often follows a predictable strategy when presenting a novel detection algorithm: the method is first evaluated on a set of simulated time series with known change points where both the model fit and detection accuracy are evaluated. An obvious downside of such experiments is that the dynamics of the simulated data are often particular to the paper, and any model that corresponds to these dynamics has an unfair advantage. Subsequently the proposed method is typically evaluated on only a small number of real-world time series. These series are often reused (e.g., the univariate well-log data of O Ruanaidh and Fitzgerald ´ , 1996), may have been preprocessed (e.g., by removing outliers or seasonality), and may not have unambiguous ground truth available. When evaluating a proposed method on real-world data, post-hoc analysis is frequently applied to argue why the locations detected by the algorithm correspond to known events. Clearly, this is not a fair or accurate approach to evaluating novel change point detection algorithms." pg 2

Related Work

- "Methods for CPD can be roughly categorized as follows: (1) online vs. offline, (2) univariate vs. multivariate, and (3) model-based vs. nonparametric. We can further differentiate between Bayesian and frequentist approaches among the model-based methods, and between divergence estimation and heuristics in the nonparametric methods." pg 3
- "For a more expansive review of change point detection algorithms, see e.g., Aminikhanghahi and Cook (2017) or Truong et al. (2020)." pg 3

Change Point Data Set

- Use f1 score of human annotators to ensure there is a baseline familiarity with annotating change points, no contextual information is provided to ensure external knowledge is not used in the process, and no dates, time, or values are shown on the axis. pg 6-7
- "Time series were collected from various online sources including the WorldBank, EuroStat, U.S. Census Bureau, GapMinder, and Wikipedia. . . The main criterion for including a time series was whether it displayed interesting behavior that might include an abrupt change. Several time series were included that may not in fact contain a change point but are nevertheless considered interesting for evaluating CPD algorithms due to features such as seasonality or outliers." pg 7

Benchmark Study

- code and data used in study can be found here: : https://github.com/alan-turing-institute/TCPDBench pg. 9

Discussion

- "We have introduced the first dedicated data set of human-annotated real-world time series from diverse application domains that is specifically designed for the evaluation and development of change point detection algorithms." pg 14

OMˆ3 Yunhai Wang and Yu (2023)

- "We present a novel multi-level representation of time series called OM3 that facilitates efficient interactive progressive visualization of large data stored in a database and supports various interactions such as resizing, panning, zooming, and visual query." pg 145:1
- "To improve the user exploration efficiency, a variety of interaction techniques [1] have been developed, e.g., SignalLens [16], multi-focus zooming [12, 29], Zenvisage [27], and a few visual query tools [10, 21]. Recently, Siddiqui et al. [28] proposed an expressive shape search algebra, enabling users to interactively search for various desired patterns. However, almost all existing techniques require the whole data to be quickly loaded onto the client side for efficient rendering. Yet, the latency is often high for large time-series data stored on remote databases. This hinders smooth interactive exploration of temporal patterns at various resolutions." pg 145:2
- "A common approach to reducing the latency is data reduction [4], by aggregating the data to reduce its size before visualization. Various strategies have been proposed to preserve salient features in the reduction. However, most of them do not account for the perceptual effects of, e.g., resizing the display, and thus often produce erroneous visualizations that can significantly distort the shape of the rendered data." pg 145:2
- "Time-series visualization [1] has been extensively explored for facilitating users to discover trends and patterns at varying time scales. Line chart is still the most popular vehicle for presenting time series. However, displaying many time series as line charts on a limited screen inevitably produces heavy overplot. To address this issue, various lens-based interaction techniques [16] and overview detail interaction techniques [12, 29] have been developed for simultaneously showing the regions of interest in detail with an overview of the whole data." pg 145:4
- "Point Aggregation. To ensure a manageable size for the query data, point aggregation methods often first group the entire time series into a few time spans and then compute an aggregated value and an aggregated timestamp for each interval using aggregation functions such as minimum, maximum, mean, and median." pg 145:4
- " A common method is the piece-wise aggregate approximation (PAA) [14], which selects the minimum (first) timestamp and computes an average value. Based on these aggregation functions, a few methods have been proposed to offer fast approximate query [2, 6]. Yet, regardless of the aggregation function being used, the line chart of the aggregated data may distort the original shape of the time series. To address the issue, M4 [13] aims to preserve the shape by selecting two data points with the minimum and maximum values, and another two with the minimum and maximum timestamps in each pixel column, connecting the four data points in time order with the rest of the time series, and rasterizing the line segments in the pixel column." pg 145:5
- "Line Simplification. Given a time-series curve, line simplification is another data-reduction approach that aims to preserve its rough shape with fewer curve segments." pg 145:5

- "One widely-used method is the top-down Ramer-Douglas-Peucker [7, 9] algorithm, which joins the first and last points in the curve and divides the curve into two segments, if the distance from the farthest point to the line is larger than a threshold. Then, each curve segment is recursively divided until all points of the original curve are within the simplification's tolerance. Fu et al. [5] further attempts to preserve salient points in the simplification. However, these methods have a high computational complexity, 0 (n2) or 0(n logn), (n is the number of data points in the time series), so interactive queries cannot be supported for large data. Instead, INCVISAGE [23] uses online sampling-based algorithms to incrementally generate curve segments, attaining 46x speedup relative to baselines. Yet, it provides only approximate visualizations and does not support interactive operations like zooming." pg 145:5

- "We have four observations from Fig. 7. First, M4-P, M4-I, M3, OM3, and OM3-NP all produce error-free visualizations, whereas Haar wavelet cannot preserve the original shape of the time series. Second, all methods have high data reduction ratios larger than 0.98 on average, while M4-P performs relatively poorly with a data reduction ratio of only around 0.77 for some data caused by the join operation. Third, Haar wavelet, M3, and OM3 are all able to support interactive exploration with the average response time around 100 ms, 280 ms, and 210 ms, respectively. In contrast, the average response time of M4-P is larger than 500 ms for all the test data, while the one of M4-I is more than 1 minute for 16M data and it quickly grows as the dataset size increases. For the query time, Haar wavelet, M3, OM3 , and OM3 -NP all take less than 100 ms, whereas the other methods take more than 1 second for most data. Fourth, M3 requires more coefficients to load and transfer, resulting in a slightly smaller data reduction ratio than OM3 (0.996 vs. 0.998) and higher query time (45 ms vs. 30 ms) and response time (280 ms vs. 210 ms) on average. Last, our pruning strategy helps avoid around half of the data points in the OM3 search, contributing to a reduction of query time by around 200 ms." pg 145:17 (Fig 7 image saved)

MinMaxLTTB Donckt et al. (2023)

Bibliography

Burg, Gerrit J. J. van den, and Christopher K. I. Williams. 2022. "An Evaluation of Change Point Detection Algorithms." https://arxiv.org/abs/2003.06222.

Donckt, Jeroen Van Der, Jonas Van Der Donckt, Michael Rademaker, and Sofie Van Hoecke. 2023. "MinMaxLTTB: Leveraging MinMax-Preselection to Scale LTTB." https://arxiv.org/abs/2305.00332.

gmbh, solidIT consulting & software development. 2023. "DBMS Popularity Broken down by Database Model." https://db-engines.com/en/ranking_categories.

Steinarsson, Sveinn. 2013. "Downsampling Time Series for Visual Representation." Faculty of Industrial Engineering, Mechanical Engineering; Computer Science, School of Engineering; Natural Sciences, University of Iceland, Reykjavik, Iceland: University of Iceland.

Yanzhe An, Yuqing Zhu, Yue Su, and Jianmin Wang. 2022. "TVStore: Automatically Bounding Time Series Storage via Time-Varying Compression." In *Proceedings of the 20th USENIX Conference on File and Storage Technologies*, 83–99. USENIX Conference on File and STorage Technologies. Santa Clara, CA, USA: USENIX Association.

Yunhai Wang, Xin Chen, Yuchun Wang, and Xiaohui Yu. 2023. "Om3: An Ordered Multi-Level Min-Max Representation for Interactive Progressive Visualization of Time Series." In *Proc. ACM Manag. Data*, 1:145:1–24. 2. ACM. https://doi.org/10.1145/3589290.