Aggregation-Aware Compression of Probabilistic Streaming Time Series

Reza Akbarinia, Florent Masseglia

Aggregation-Aware Compression of Probabilistic Streaming Time Series

Reza Akbarinia and Florent Masseglia

Inria & LIRMM
Zenith team - Univ. Montpellier
France
first.last@inria.fr

Abstract. In recent years, there has been a growing interest for probabilistic data management. We focus on probabilistic time series where a main characteristic is the high volumes of data, calling for efficient compression techniques. To date, most work on probabilistic data reduction has provided synopses that minimize the error of representation w.r.t. the original data. However, in most cases, the compressed data will be meaningless for usual queries involving aggregation operators such as SUM or AVG. We propose *PHA* (Probabilistic Histogram Aggregation), a compression technique whose objective is to minimize the error of such queries over compressed probabilistic data. We incorporate the aggregation operator given by the end-user directly in the compression technique, and obtain much lower error in the long term. We also adopt a global error aware strategy in order to manage large sets of probabilistic time series, where the available memory is carefully balanced between the series, according to their individual variability.

1 Introduction

The abundance of probabilistic and imprecise data, in the past decade, has been the center of recent research interest on *probabilistic data management* [8] [14]. This growing production of probabilistic information is due to various reasons, such as the increasing use of monitoring devices and sensors, the emergence of applications dealing with moving objects, or scientific applications, with very large sets of experimental and simulation data (so much so that Jim Gray has identified their management and analysis as the "Fourth Paradigm" [9]). Example 1 gives an illustration of such a scientific application, where one must deal with very large sets of probabilistic data. These data arrive as probabilistic distribution functions (pdf), where each value (*e.g.* the observation of a measure) is associated to a level of probability.

Example 1. Phenotyping is an emerging science that bridges the gap between genomics and plant physiology. In short, it aims to observe the interactions between the functions of a plant (growth, temperature, etc.) and the physical world in which it develops. Then, depending on the genomic background of the plant, conclusions can be drawn about plant productivity, which is a major concern from ecological, economic and societal points of view. In this context, as illustrated by Figure 1, several measures, such as height and weight growth, or evaporation, can be performed by sensors attached to each

plant. Due to device characteristics, each measure is associated to a level of probability and expressed as a pdf. The pdfs of a plant are stored at regular time intervals, leading to a probabilistic time series. In our case, we have as much probabilistic time series as plants in the platform.

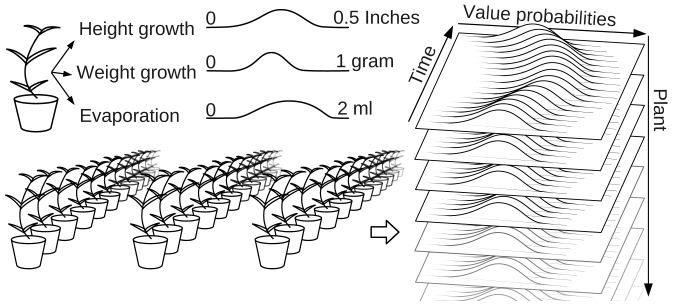


Fig. 1. A phenotyping platform. For each plant, several probabilistic values are measured at regular time intervals and the whole platform produces a large number of streaming probabilistic time series

There are many other possible applications involving streaming probabilistic time series, where the measures may be, for example, the amount of light received by an array of solar panels in a photovoltaic system, the number of gallons passing by a dam, the quantity of pesticides received by the plants of an observation plot, the support of probabilistic frequent itemsets over time in a probabilistic data stream, etc. All the above applications have a common characteristic: they produce time series where, for each interval of time and for each series, we have a pdf corresponding to the possible measures during this interval. In almost all these applications, the user intends to issue specific aggregation queries over the time intervals, e.g. the sum of plant growth in a sequence of time intervals.

Depending on the application, the number of probabilistic values may be very high. Compared to deterministic time series, the data size is multiplied by the range of the pdf (*e.g.* with a thousand of series, ten thousands of measures per day and a hundred of probabilistic values per measure, there are one billion probabilistic values to handle per day). When the number of values is larger than the available memory, data has to be compressed, and when that number gets even much larger, the compression cannot be without information loss. Our goal is to lower the information loss due to probabilistic data compression. Actually, *we take the view that such data are usually intended to be intensively queried, especially by aggregation queries*. In this paper, we propose a compression approach, called *PHA*, that allows high quality results to specific aggregation operators. This is done by incorporating the aggregation in the compression process itself. The idea is that when the granularity of the representation decreases, the aggregation operator tackled by the end-user is used at the core of time series compression. Our approach includes strategies for probabilistic streaming time series management

by considering their tolerance to compression. PHA is distribution independent, so it can work with any kind of distributions (uniform, normal, etc.). We perform extensive evaluations to illustrate the performance of *PHA* over real and synthetic datasets. The results show it's ability to apply compression ratios as high as 95% over probabilistic time series, while keeping extremely low (almost null) error rates w.r.t. the results of aggregation operators.

2 Problem Definition

We are interested in compressing probabilistic time series by aggregating their histograms that are representations of the data distributions in time intervals of time series.

2.1 Preliminaries

We define a *probabilistic histogram* as follows.

Definition 1. [Probabilistic histogram] *A probabilistic histogram H is an approximate representation of a pdf (probabilistic distribution function) based on partitioning the domain of values into m buckets, and assigning the same probability to all values of each bucket. Each bucket b is a pair (I, p) where $I = [s, e)$ is a value interval with s and e as start and end points respectively, and p is the probability of each value in the bucket. We assume that s and e are integer. For each value v, we denote by $H[v]$ the probability of v in the distribution. The buckets of a histogram H are denoted by $bucks(H)$, and the ith bucket by $bucks(H)[i]$. The number of buckets is $|bucks(H)|$.*

Hereafter, unless otherwise specified, when we write histogram we mean a probabilistic histogram. A probabilistic time series is defined as a sequence of histograms as follows.

Definition 2. [Probabilistic time series] *a probabilistic time series $X = \{H_1, \ldots, H_n\}$ is a sequence of histograms defined on a sequence of time intervals $T_1, \ldots, T_n$, such that histogram H_i represents the probability distribution in interval T_i. The intervals $T_1, \ldots, T_n$ are successive, i.e. there is no gap between them.*

Example 2. Let us consider a plant p in the phenotyping application of example 1. The height growth of p has been measured by a sensor. The resulting pdfs are represented by two histograms H_1 and H_2 (see Figure 2): $H_1 = \{([0,1),0), ([1,2),0), ([2,3),0), ([3,4),0.2), ([4,5),0.2), ([5,6),0.4), ([6,7),0.2), ([7,8),0)\}$ from 8 am to 7:59 pm, and $H_2 = \{([0,1),0), ([1,2),0.2), ([2,3),0.4), ([3,4),0.2), ([4,5),0.2), ([5,6),0), ([6,7),0), ([7,8),0)\}$ from 8 pm to 7:59 am. For instance, in H_1, the probability that the plant grew by 5 units is 40%. Meanwhile, during the night, the most probable growth is only 2 units. The probabilistic time series of p is made of these two histograms on their respective time intervals.

Below, we define a data stream based on time series.

Definition 3. [Data stream] *a data stream $S = \{X_1, \ldots, X_m\}$ is a set of time series defined on a sequence of intervals $T_1, \ldots, T_n$ such that each time series X_i has its own histograms for the intervals $T_1, \ldots, T_n$.*

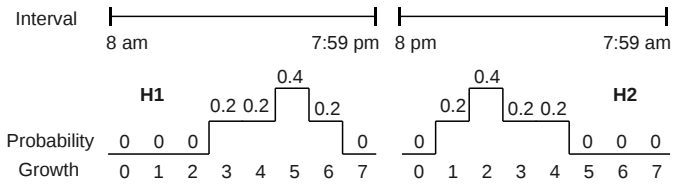


Fig. 2. A probabilistic time series, made of two probabilistic histograms on contiguous time intervals

The aggregation of histograms can be defined using the aggregation of probability distribution functions.

Definition 4. [Aggregation of pdfs] *let G_1 and G_2 be two pdfs, and f be an aggregation function. Then, aggregating pdfs G_1 and G_2 by f means to generate a pdf G such that for any two values $x_1 \in G_1$ and $x_2 \in G_2$ there is a value $x \in G$ where $x = f(x_1, x_2)$, and the probability of x in G is equal to the probability of all combinations of values $x_1 \in D_{G_1}$ and $x_2 \in D_{G_2}$ such that $f(x_1, x_2) = x$.*

Definition 5. [Aggregation of probabilistic histograms] *given an aggregation function f, and two histograms H_1 and H_2 representing two pdfs G_1 and G_2 respectively, then aggregating histograms H_1 and H_2 by f means to make a histogram on the aggregated pdf of G_1 and G_2.*

We compare the probabilistic histograms by measuring their absolute difference defined as follows.

Definition 6. [Absolute difference] *the absolute difference of two histograms H_1 and H_2, denoted as $abs\_dif(H_1, H_2)$, is defined as the cumulative difference of their probability distributions for all possible values. In other words:*

$$abs\_dif(H_1, H_2) = \int_{-\infty}^{+\infty} (|H_1(x) - H_2(x)|)\,\mathrm{d}x.$$

We compress the histograms of time series in order to hold stream management constraints. For each compressed histogram, there is a set of original histograms which we define as follows.

Definition 7. [Original histograms] *Let $X = \{H_1, \ldots, H_n\}$ be a time series defined on time intervals $T_1, \ldots, T_n$. Assume H' is a histogram resulted from compressing the histograms of intervals $T_i, \ldots, T_j$. Then, the original histograms of H', denoted as $org\_hist(H')$, are the histograms of X in time intervals $T_i, \ldots, T_j$. In other words, $org\_hist(H') = \{H_i, \ldots, H_j\}$.*

In our algorithms, sometimes we need to extend the histograms. The extension of a histogram is defined as follows.

Definition 8. [Extending a probabilistic histogram] *the extension of a probabilistic histogram H, denoted as $Ext(H)$, is a probabilistic histogram H' such that the interval of each bucket in H' is of size one (i.e. $e - s = 1$), and for each value v, $H'(v) = H(v)$.*

Example 3. Let $H = \{([0,2), 0.4), ([2,4), 0.6)\}$ be a histogram with two buckets. The extension of H is $Ext(H) = \{([0,1),\ 0.4),\ ([1,2), 0.4),\ ([2,3), 0.6),\ ([3,4), 0.6)\}$. Thus, in $Ext(H)$ the size of each bucket interval is 1.

Hereafter, when we write extended representation of a histogram we mean its extension. For simplicity of presentation, when each bucket of a histogram is of size one, we write the histogram with the probability values only and we omit the start and end points.

2.2 Problem Statement

Our objective is to compress the histograms of a data stream's time series by aggregating their histograms, while respecting memory constraints on the size of histograms. Concretely, the results of each aggregation should be represented by a histogram of at most m buckets, where m is a predefined value. At the same time, we want to preserve the quality of aggregation, i.e. minimum error in the results.

Our goal can be defined as follows: given a set of given time series $S = \{X_1, \ldots, X_n\}$, an aggregation function f, and an integer number m, our objective is to compress S to a set of time series $S' = \{X'_1, \ldots, X'_n\}$ such that each time series X'_i is composed of a set of histograms $\{H'_1, \ldots, H'_k\}$, while :

1. Minimizing the error of aggregation, given by:
$$\sum_{\forall X' \in S'} (\sum_{\forall H' \in X'} abs\_dif(H', f(org\_hist(H'))))$$
2. Limiting the number of buckets of each histogram to m. In other words:
$$\forall X' \in S', \forall H' \in X', |bucks(H'_i)| \leq m.$$

2.3 Background

In our underlying applications, when the number of buckets becomes larger than the available memory, we must reduce the data size. Since we are working with probabilistic time series (*i.e.* series of histograms) we have two possible dimensions for data compression. The first one is the bucket dimension. In this case, we merge several contiguous buckets into one bucket having larger granularity. Wavelets and regression [7] are two popular methods that can be adapted for this purpose. The second one is the time dimension. In this case, when the number of histograms is too large, we merge several histograms, corresponding to a number of time slots, into one aggregated histogram that correspond to a larger time slot, thus increasing the granularity of our representation.

3 Compression Approach

In this section, we propose our compression approach on histograms called PHA (Probabilistic Histogram Aggregation) and a strategy for histogram selection (for compression) in the global streaming process.

3.1 Bucket Dimension

As stated previously, there are two possible dimensions for compressing probabilistic time series: bucket and time dimensions. In this subsection, we present our approach for compression on the bucket dimension. We first describe our method for aggregating probabilistic histograms of time intervals based on the SUM aggregation operator, which is the running operator of this paper. Then, we explain the compression of aggregated histograms. It is not difficult to extend our approach to other aggregation operators such as AVG (see Appendix 7 for AVG).

Aggregating Histograms with SUM. Given two histograms H_1 and H_2 on two time intervals T_1 and T_2 respectively, aggregating them by a function f means to obtain a histogram H such that the probability of each value k in H is equal to the cumulative probability of all cases where $k = f(x_1, x_2)$ such that $H_1(x_1) > 0$ and $H_2(x_2) > 0$. In deterministic context, given two histograms H_1 and H_2, for each value k the SUM operator returns $H(k) = H_1(k) + H_2(k)$ as output. However, this method does not work for the case of probabilistic histograms, because we need to take into account the probability of all cases where sum is equal to k. Below, we present a lemma that gives a formula for performing the SUM operator over probabilistic histograms.

Lemma 1. *Let H_1 and H_2 be two probabilistic histograms. Then, the probability of each value k in the probabilistic histogram obtained from the sum of H_1 and H_2, denoted as $SUM(H_1, H_2)[k]$, is given by $\sum_{i=0}^{k}((H_1[i] \times H_2[k-i] + H_1[k-i] \times H_2[i])$*

Proof. The probability of having a value k in $SUM(H_1, H_2)$ is equal to the probability of having values i in H_1 and j in H_2 such that $i + j = k$. Thus, we must compute the cumulative probability of all cases where the sum of two values from H_1 and H_2 is equal to k. This is done by the sigma in Lemma 1.

Example 4. Consider the histograms H_1 and H_2 in Example 2. Then, the probability of value $k = 4$ in the histogram $H = SUM(H_1, H_2)$ is computed as follows:

$H[4] = H_1[0] \times H_2[4] + H_1[1] \times H_2[3] + H_1[2] \times H_2[2] + H_1[3] \times H_2[1] + H_1[4] \times H_2[0] = 0 + 0 + 0 + 0.2 \times 0.2 + 0 = 0.04$.

Aggregation	Value probabilities															Error
SUM(H1,H2)	**0**	**0**	**0**	**0**	**0.04**	**0.12**	**0.2**	**0.28**	**0.2**	**0.12**	**0.04**	**0**	**0**	**0**	**0**	
SUM(Ext(Wav(H1)),Ext(Wav(H2)))	0	0	0.01	0.02	0.07	0.12	0.17	0.22	0.17	0.12	0.07	0.02	0.01	0	0	0.24
SUM(Ext(Reg(H1)),Ext(Reg(H2)))	0	0	0	0.02	0.04	0.14	0.17	0.26	0.17	0.14	0.04	0.02	0	0	0	0.16
PHA(H1,H2)	0.008	0.008	0.008	0.008	0.008	0.12	0.2	0.28	0.2	0.12	0.04	0	0	0	0	0.064

Fig. 3. Comparing the results of SUM operator on histograms compressed using different compression techniques

Compressing Aggregated Histograms. An aggregated histogram is not a compressed histogram by itself (it is just a representation having a lower granularity on the time dimension). Let us consider the SUM operator on H_1 and H_2 from Example 2. The

result of this operator is given by Figure 3 (line "SUM($H1, H2$)" in bold characters). Obviously, the number of buckets is not significantly reduced and the requirement of data compression when the memory budget has been reached is not met. This is due to the fact that the number of values has grown, and now corresponds to the sum of the maximum values in H_1 and H_2. Therefore, if we want to obtain a compressed version of this histogram, we need to aggregate data on the buckets dimensions. We chose the regression approach and apply it to the aggregated histogram H'. Then, we obtain an aggregation on the time dimension, combined to a compression on the buckets dimension.We adopt a bottom-up approach to calculate the regression form of our histograms. First, we build a sorted list of distances between the contiguous buckets. Then, the two most similar buckets are merged and the list of distances is updated. This process is repeated until the desired number of buckets is reached. Here, when two contiguous buckets b_1 and b_2 are merged into the resulting bucket b', then b' inherits from the start point of b_1 and the end-point of b_2. The value of b' is the weighted average of values in b_1 and b_2. The weight of a bucket b' is the number of original buckets that have been merged into b'.Our claim is that paying attention to the order of these operations (*i.e.* compression+aggregation vs. aggregation+compression) makes the difference between appropriate and irrelevant manipulation of probabilistic data. Since an aggregation operator considers the probabilistic meaning of each value in a histogram, it gives better results on the original data, compared to the result obtained on the compressed (*i.e.* damaged) data. This is illustrated in the next example.

Comparative Example. The operations reported in Figure 3 illustrate the main motivation for our approach: *in order to choose a representation for probabilistic time series compression, we should not focus on the error w.r.t. the original histograms, but rather on the error w.r.t.the results of aggregation operators that will run on the compressed data*. The table of Figure 3 gives the results of the SUM operator on H_1 and H_2 from Example 2 (second line, "$SUM(H_1, H_2)$"). We compare the results of this operator on the compressed versions of H_1 and H_2, using the techniques presented Sections 2.3 and 3.1. First, we report the results of SUM on the compressed histograms on the bucket dimension, *i.e.* $SUM(Ext(Wav(H1)), Ext(Wav(H2)))$. In other words, we apply SUM on the extended histograms of the wavelet transform of H_1 and H_2 and obtain the approximate value probabilities on the time interval $T_1, \ldots, T_2$. In the last column, we report the error (24%) of this representation, compared to the true probability values of $SUM(H_1, H_2)$. Second, we apply SUM on the extended histograms of the regression compression on H_1 and H_2, *i.e.* $SUM(Ext(Reg(H1)), Ext(Reg(H2)))$ and also report the error (*i.e.* 16%).

This comparison is provided with SUM operator applied to the histogram compressed on the time dimension. We apply a regression compression to the result of SUM on the original histograms. The compressed histogram is given by the "$Reg(SUM(H1, H2))$" line and its extension is given by the last line. In this case, the error is only 6.4%.

In this illustration, we can observe the difference between the errors of each representation. Intuitively, we observe that PHA gives the best results (bottom line of Figure

3).When the histograms are independent, the regression technique should give better results than *wavelets*.

3.2 Time Dimension

The time dimension is very important in the technique used for compressing time series. In data streams, this dimension has a major impact since it gives the organization of events records (*i.e.* their order of arrival). Moreover, most streams are cohesive on this dimension since the series have usually low variation from one timestamp to an other. By working on this dimension, we are thus able to lower the information loss due to the representation granularity. In this section, we study two strategies under the decaying factor point of view and propose a global optimization framework where the available space is balanced between the series in the model, in order to obtain the best possible global error.

Logarithmic Tilted Time Windows. The logarithmic Tilted Time Windows (TTW) is a management principle based on a decaying factor [5]. When new transactions arrive in the stream, older transactions are merged. The older the transactions, the larger the granularity of representation (and the size of windows). The main advantage of this representation is to give more space to recent transactions. The main drawback is to merge old transactions with a blind approach that does not optimize the available space.

Global Error Optimization. Let us consider S, the data stream of Definition 3. At each step s (a new histogram is added to each probabilistic time series), we want to update the representations and their error rates in our data structure. In order to maintain a globally satisfying error rate, we need to choose the representations that will minimize this error. The main idea is that merging the most similar histogram will have low impact on the resulting error. In other words, similar histograms have higher tolerance to merging error. Therefore, for each new histogram in S, let X_i be the time series having the most similar consecutive histograms, *i.e.* H_1 and H_2, then i) H_1 and H_2 are merged into H_3 ii) the distances between H_3 and its direct neighbors are calculated and iii) the sorted list of distances between consecutive histograms in S is updated. Based on this general principle, we devise the strategies described hereafter.

PHA 'a la Ward'. The goal of *PHA* is to minimize the error of a merged histogram w.r.t. the original merged histogram. It is not possible to know, in advance, which couple of histograms will give the lowest error (after merging) among all the histograms in S. Therefore, an optimal strategy, by means of *PHA*, should try all the possible combinations and pick up the best one. This principle is inspired from the Ward criterion in hierarchical clustering. However, as illustrated by our experiments, this does not always give the best results in terms of global error.

Fast PHA. We also propose a fast strategy based on *PHA*. In order to reduce the time spent on each merging operation, we propose to merge two consecutive histograms if the sum of their number of values is minimum. Surprisingly, our experiments illustrate

the good quality of results obtained with this strategy, whereas no information about the histogram's content has been taken into account (only their number of probability values).

4 On Retrieving Original Values

When the histograms have been merged, the remaining question is "how to use this compressed information to retrieve the original histograms?". An ideal solution would be able to get back to the original histograms (even if some information has been lost in the compression). In this section, we first show that in general if the original histograms are different, it is not possible to find them from the compressed histogram, because of information loss in the compression process. Then, by assuming that the original histograms are identical, we propose a method to return them. Let us first study of the case where the original histograms are different. Let us assume that the size of the original histograms is n. The following lemma shows the impossibility of computing them from the merged histogram.

Lemma 2. *Suppose a histogram $H = SUM(H_1, H_2)$, and assume it has been generated from two different probabilistic histograms. Then, we cannot compute H_1 and H_2 by using H.*

Proof. Let $2n$ be the size of histogram $H = SUM(H_1, H_2)$. Let $X_i = H_1[i]$ and $Y_i = H_2[i]$ for $0 \leq i \leq n$. To find the variables X_i and Y_i for $0 \leq i \leq n$, we must be able to solve the following equation system:

$$\begin{cases} X_0 \times Y_0 = H[0] \\ X_0 \times Y_1 + X_1 \times Y_0 = H[1] \\ X_0 \times Y_i + X_1 \times Y_{i-1} + \cdots + X_i \times Y_0 = H[i] \\ \cdots \\ X_n \times Y_n = H[2n] \end{cases} \quad (1)$$

In this system, there are $2 \times n + 1$ equations, and $2 \times n + 2$ variables, i.e. X_i and Y_i for $0 < i < n$. Since the number of equations is less than the number of variables, it is not possible to find the value of variables. □

This is usual for data streams, where decreasing the granularity of representation leads to approximate representations and information loss. However, our experiments show that our approach allows very high compression rates with no, or very few, loss for the queries performed on the stream (while traditional approaches show very important loss). Furthermore, when the original histograms are equal, the following lemma gives us a formula for retrieving their exact original values.

Lemma 3. *Suppose an aggregated histogram H, and assume it has been generated from H_1 and H_2 with the SUM operator, i.e. $H = SUM(H_1, H_2)$. Suppose that H_1 and H_2 are two equal probabilistic histograms, i.e. $H_1 = H_2$, then H_1 can be reconstructed from H using the following formulas:*

$$H_1(k) = \begin{cases} \frac{H[k] - \sum_{i=1}^{k-1} H_1[i] \times H_1[k-i]}{2 \times H_1[0]} & \text{for } k > 0; \\ \sqrt{H[0]} & \text{for } k = 0 \end{cases} \quad (2)$$

Proof. The proof is done by using Lemma 1 and the assumption that $H_1[k] = H_2[k]$ for all k. We first prove the equation for the case of $k = 0$, and then for $k > 0$. From Lemma 1, we have : $H_1[0] \times H_2[0] = H[0]$. By replacing $H_2[0]$ with $H_1[0]$, we obtain: $H_1[0] = \sqrt{H[0]}$. To prove the second case, i.e. $k > 0$, let us write the equation of Lemma 1 as follows:

$H[k] = H_1[0] \times H_2[k] + H_1[k] \times H_2[0] + \sum_{i=1}^{k-1}(H_1[i] \times H_2[k-i] + H_1[k-i] \times H_2[i])$

Then, by replacing $H_2[i]$ with $H_1[i]$ for $0 \leq i \leq k$, we have

$H[k] = 2 \times H_1[0] \times H_1[k] + 2 \times \sum_{i=1}^{k-1}((H_1[i] \times H_1[k-i]$

Thus, we have :

$H_1[k] = \frac{H[k] - \sum_{i=1}^{k-1} H_1[i] \times H_1[k-i]}{2 \times H_1[0]}$ □

By assuming the equality of the original histograms of a merged histogram, Algorithm 1 computes the original histograms.

Algorithm 1 Retrieving original histograms in the case where they are identical

Require: H: a probabilistic histogram generated by applying SUM operator on two identical histograms; $2 \times n$: highest value in the domain of H;
Ensure: H_1: the original probabilistic histogram from which H has been generated;

```
1: H1[0] = sqrtH[0];
2: for k = 1 to n do
3:     s = 0;
4:     for i = 1 to k − 1 do
5:         s = s + H1[i] × H1[k − i];
6:     end for
7:     H1[k] = (H[k] − s)/(2 × H1[0]);
8: end for
9: return H1
```

5 Experiments

We have implemented the strategies described in Section 3.2 with our *PHA* compression technique.

We evaluated our approach on a real-world dataset and two synthetic datasets. The real world dataset has been built over probabilistic frequent itemsets (PFI) [1] extracted from the *accident* dataset of the FMI repository[1]. The original accident dataset contains 11 millions of events, 340K transactions and 468 items. We have added an existential probability $P \in [0..1]$ to each event in these datasets, with a uniform distribution. Then, we have extracted PFIs [3] from this file. Each PFI has a probability distribution of its frequentness at regular time intervals in the file. Our interest in this dataset is due to the significant number of PFIs that can be extracted from it. Actually, we have recorded the evolution of these distributions for 15 itemsets, over 574K timestamps, with a range of 49 possible values. For synthetic datasets, we have written a generator that builds a set of

[1] http://fimi.ua.ac.be/data/

S probabilistic time series having length N of histograms having H buckets (normal or uniform distributions) as follows. For a normal distribution, we first generate a random time series of length N. Then, for each value in the time series, we consider it as the "mean" value of the distribution and we chose a random "variance". We have generated two datasets: U.S20N100K.H50 and N.S10N100K.H40. For both of these datasets, S is the number of series, N the series length and H the size of each histogram.

In our tests, we don't adopt Wavelets as a compression strategy on probability distributions because this technique does not fit the constraints of probabilistic streaming time series. Actually, Wavelets need to be applied to static datasets (they don't apply to streaming or incremental environments) and they require datasets size to be a power of two (for instance, in our context, both the number of timestamps and the number of buckets should be a power of two for a Wavelet compression).

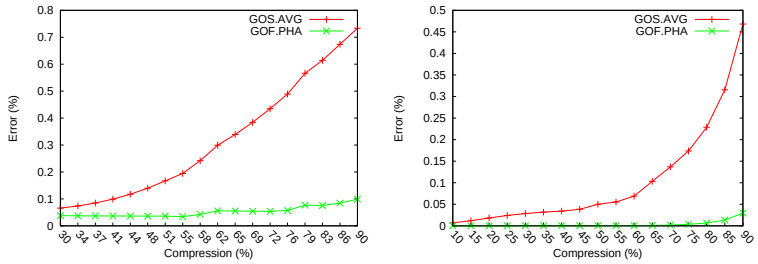
We compare our compression approach with extensions of a competitive approach, which we call *averaging*, that compresses two given histograms by computing their average for any given value in the domain. Formally, averaging works as follows. Let H_1 and H_2 be two given histograms, then their compression means to generate a histogram H' such that for any value v in their domain, $H'(v) = AVG(H_1(v), H_2(v))$. Actually, in our context, the time dimension is very important since successive histograms are more likely to be similar than random histograms in the series. Therefore, we consider that the most adequate approach, for a comparison to *PHA*, is the *averaging* compression technique.

In the following, *TTW.AVG* is the *averaging* compression in the Tilted Time Windows strategy, *TTW.PHA* is the *PHA* compression in the TTW strategy, *GOF.PHA* is *PHA* in the Global Optimization strategy with the *Fast* approach (Section 3.2), *GOW.PHA* is *PHA* in the global optimization 'a la Ward' (Section 3.2) and *GOS.AVG* is the *averaging technique* in the global optimization strategy with SSE as a distance between histograms. When quality is measured, it is with regards to the goal defined in Section 2.2. For this purpose, when a stream has been summarized by means of *PHA* and a technique T, we compare the quality of the chosen operator (in our case, it is SUM) over the intervals built by *PHA* on the corresponding histograms in the representation of T and in the original data. Our goal is to measure the information loss of various compression techniques, compared to that of *PHA*, having the original data as a reference. Since the intervals built by *PHA* are not the same from one experiment to another, we have one comparison diagram per dataset and per couple of compression techniques. Measures are given according to a compression ratio. For instance, a compression of 90% for a file with $1,000,000$ histograms means that our representation is done with $100,000$ compressed histograms.

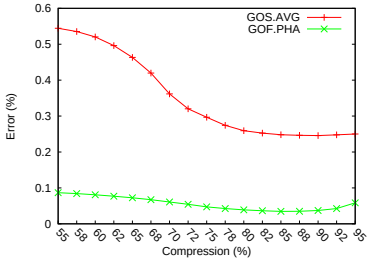
5.1 Global Optimization: Evaluation of the Synopses Techniques

During our experiments, we observed that *GOW.PHA* does not justify the difference in response time. Actually, the results in terms of quality were similar to that of *GOF.PHA* with much higher response times. This is the case for all the datasets we have tested, but due to lack of space, we don't report this result in our figures. Our first set of experiments aims to compare the synopses techniques based on *averaging*, and *PHA*. For this purpose, we use the global optimization principle described in Section 3.2.

Figure 4 illustrates the difference in quality of approximation according to the compression ratio on the real and synthetic datasets. We can observe that *GOF.PHA* gives very good results in terms of quality when *GOS.AVG* reaches high error levels (up to 75% on *real* data). For instance, on the *normal* dataset (Figure 4(b)) *GOF.PHA* is able to keep an error close to zero with compression rates up to 80%. This is a very good result, illustrating the capability of *PHA* to keep most of the information during the compression, compared to the high error rates of *GOS.AVG*.



(a) accident (b) N.S10N100K.H40



(c) U.S20N100K.H50

Fig. 4. Quality of approximation: Global Optimization strategy

5.2 TTW vs. Global Optimization

Here, our goal is first to investigate the difference between *TTW.PHA* and *TTW.AVG*. Figure 5(d) illustrates the difference between both approaches in terms of compression and error rates on the *normal* dataset. We observe similar differences, compared to our previous experiments between *GOF.PHA* and *GOS.AVG*. Then, we have measured the error rate, timestamp by timestamp, for a TTW strategy on one hand, compared to a Global Optimization, on the other hand. We expect TTW to give better results on the most recent histograms, since they are designed for this purpose. Figure 5(e) gives a detailed error comparison between *GOF.PHA* and *TTW.AVG* on the accident data set with a compression ratio of 85%. Figure 5(f) gives the same comparison on the *normal* dataset with a compression ratio of 95%. We observe that the difference is negligible

for the most recent histograms, while it increases greatly for old histograms (one can observe the quality plateau due to changes of TTW in the representation).

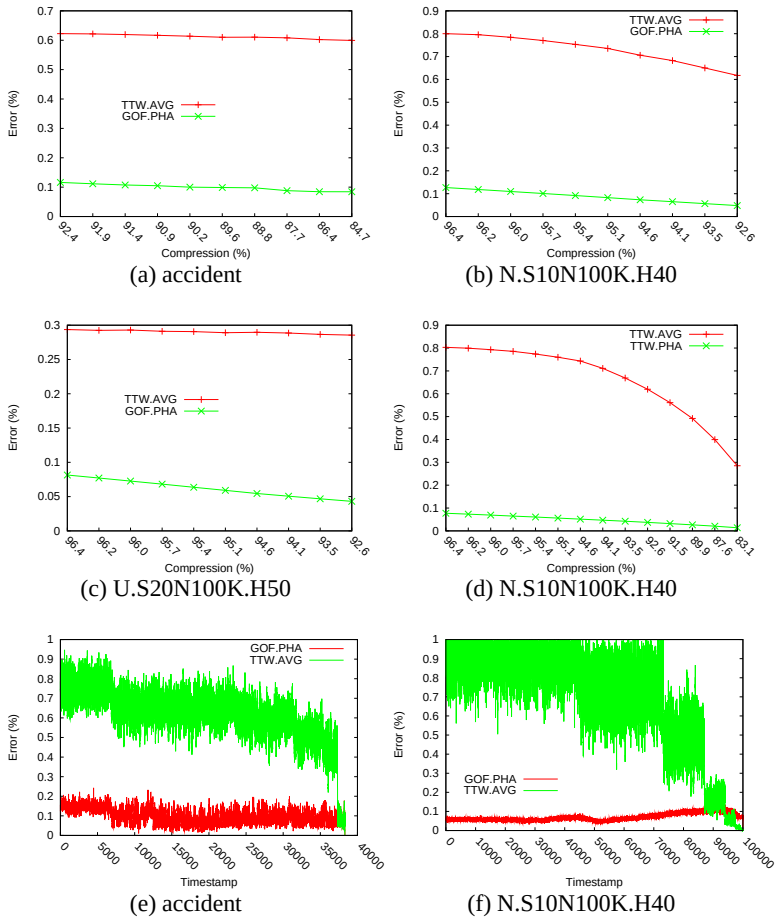


Fig. 5. Quality of approximations: TTW vs. Global Optimization (a,b,c), TTW only (d) and detailed quality of approximation, transaction by transaction (e,f)

6 Related Work

Today, probabilistic data management is recognized as a major concern [8], where an important challenge is to provide reliable models that will allow data querying with relevance and cohesion [14]. In this section, we review the works done on the problems that are related to the problem that we address. Aggregate query processing is an important issue of probabilistic data management. In this context, some works were

devoted to developing efficient algorithms for returning the expected value of aggregate values, e.g. [4] [10]. In [6] and [13], approximate algorithms have been proposed for probabilistic aggregate queries. The central limit theorem [12] is one of the main methods to approximately estimate the distribution of aggregation functions, e.g. SUM, for sufficiently large numbers of probabilistic values. In [11], Kanagal et al. deal with continuous aggregate queries over correlated probabilistic data streams. They assume correlated data which are Markovian and structured in nature. In [2], a dynamic programming algorithm for evaluating the SUM aggregate queries was proposed. The goal of almost all the work on probabilistic aggregate query processing is to efficiently process these queries over uncompressed probabilistic data. But, the goal of our work is to develop compression techniques such that the aggregate queries return high quality results on the compressed data. Despite the data overload and the quality issues that are at stake, we only find a few contributions on probabilistic data reduction [15] [7]. Currently, the main grip for research in this domain is to consider the difference between deterministic and probabilistic data in the compression techniques. In [7] the authors build synopses of probabilistic data represented by histograms that associate a value to its probability. In [15], a wavelet approach is designed for probabilistic time series compression. The authors work on a series of probabilistic histograms and consider two dimensions in their Wavelet algorithm. Aside from [15, 7], most of existing works on probability distributions has focused on normal distributions. We didn't find solutions for probabilistic streaming time series where the numbers of series, buckets and timestamps have no constraints. and they don't consider the possible use of aggregation operations that can be made on these data.

7 Conclusions

We addressed the problem of probabilistic time series compression with the objective of allowing high quality results to aggregation operators. We proposed PHA, a new probabilistic approach that takes into account the aggregation operator in the compression process. We evaluated our approach through experiments over several datasets. Experimental results illustrate that our approach is capable to obtain compression ratios as high as 95% over probabilistic time series, with very low (close to zero) error rates for the results of aggregation operators. This shows that PHA fits well to analytical applications where specific aggregation operators are frequently issued over compressed probabilistic time series.

References

1. Reza Akbarinia and Florent Masseglia. Fast and exact mining of probabilistic data streams. In *ECML/PKDD (1)*, pages 493–508, 2013.
2. Reza Akbarinia, Patrick Valduriez, and Guillaume Verger. Efficient evaluation of sum queries over probabilistic data. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints), 2012.
3. Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz, Florian Verhein, and Andreas Zuefle. Probabilistic frequent itemset mining in uncertain databases. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 119–128, New York, NY, USA, 2009. ACM.

4. Doug Burdick, Prasad M. Deshpande, T. S. Jayram, Raghu Ramakrishnan, and Shivakumar Vaithyanathan. Olap over uncertain and imprecise data. *The VLDB Journal*, 16(1):123–144, January 2007.
5. Yixin Chen, Guozhu Dong, Jiawei Han, Benjamin W. Wah, and Jianyong Wang. Multi-dimensional regression analysis of time-series data streams. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 323–334. VLDB Endowment, 2002.
6. Graham Cormode and Minos Garofalakis. Sketching probabilistic data streams. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 281–292, 2007.
7. Graham Cormode and Minos Garofalakis. Histograms and wavelets on probabilistic data. *IEEE Trans. on Knowl. and Data Eng.*, 22(8):1142–1157, August 2010.
8. Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, October 2007.
9. Anthony J G Hey, Stewart Tansley, and Kristin Michele Tolle, editors. *The fourth paradigm : data-intensive scientific discovery*. Redmond, Wash. : Microsoft Research, 2009.
10. T. S. Jayram, Andrew McGregor, S. Muthukrishnan, and Erik Vee. Estimating statistical aggregates on probabilistic data streams. *ACM Trans. Database Syst.*, 33(4):26:1–26:30, December 2008.
11. Bhargav Kanagal and Amol Deshpande. Efficient query evaluation over temporally correlated probabilistic streams. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, ICDE '09, pages 1315–1318, 2009.
12. Grzegorz Rempala and Jacek Wesolowski. Asymptotics for products of sums and u-statistics. *Electron. Commun. Probab.*, 7:no. 5, 47–54, 2002.
13. Robert Ross, V. S. Subrahmanian, and John Grant. Aggregate operators in probabilistic databases. *J. ACM*, 52(1):54–101, January 2005.
14. Saket Sathe, Hoyoung Jeung, and Karl Aberer. Creating probabilistic databases from imprecise time-series data. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE '11, pages 327–338, Washington, DC, USA, 2011.
15. Yuchen Zhao, Charu Aggarwal, and Philip Yu. On wavelet decomposition of uncertain time series data sets. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 129–138, 2010.

AVG Operator. To extend PHA to the average (AVG) operator, we need a method for computing the average of two probabilistic histograms. Given two probabilistic histograms H_1 and H_2, aggregating them by an AVG operator means to make a histogram H such that the probability of each value k is equal to the cumulative probability of cases where the average of two values x_1 from H_1 and x_2 from H_2 is equal to k, i.e. $k = (x_1 + x_2)/2$. In the following lemma, we present a formula for aggregating two histograms using the AVG operator.

Lemma 4. *Let H_1 and H_2 be two probabilistic histograms. Then, the probability of each value k in the probabilistic histogram obtained from the average of H_1 and H_2, denoted as $AVG(H_1, H_2)[k]$, is computed as:*

$$AVG(H_1, H_2)[k] = \sum_{i=0}^{k}((H_1[i] \times H_2[2 \times k - i] + H_1[2 \times k - i] \times H_2[i])$$

Proof. The probability of having a value k in $AVG(H_1, H_2)$ is equal to the cumulative probability of all cases where the average of two values i in H_1 and j in H_2 is equal to k. In other words, j should be equal to $2 \times k - i$. This is done by the sigma in the above equation.