Online Percentile Estimation (OPE)

Jorma Kilpi¹ · Timo Kyntäjä¹ · Tomi Rätty¹

Received: 9 March 2020 / Revised: 14 December 2020 / Accepted: 25 May 2021 / Published online: 21 June 2021
© The Author(s) 2021

Abstract

We describe a control loop over a sequential online algorithm. The control loop either computes or uses the sequential algorithm to estimate the temporal percentiles of any univariate input data sequence. It transforms an input sequence of numbers into output sequence of histograms. This is performed without storing or sorting all of the observations. The algorithm continuously tests whether the input data is stationary, and reacts to events that do not appear to be stationary. It also indicates how to interpret the histograms since their information content varies according to whether a histogram was computed or estimated. It works with parameter-defined, fixed-size small memory and limited CPU power. It can be used for a statistical reduction of a numerical data stream and, therefore, applied to various Internet of Things applications, edge analytics or fog computing. The algorithm has a built-in feasibility metric called *applicability*. The applicability metric indicates whether the use of the algorithm is justified for a data source: it works for an arbitrary univariate numerical input, but it is statistically feasible only under some requirements, which are explicitly stated here. We also show the results of a performance study, which was executed on the algorithm with synthetic data.

Keywords Online computation · Sequential estimation · Percentile estimation · Internet of things

1 Introduction

The Internet of Things (IoT) concept reflects the tens of billions connected devices on the Internet in the near future. Many of such devices produce a measurement or monitor data periodically or even continuously. The growth of data will be huge. This offers engineering challenges and business opportunities for the future. Opportunities, however, require first solutions to the challenges, especially the management of increased data due to the amount of networked sensors or measurement agents. Often, at least a portion of the data needs to be stored for later use while, simultaneously, supporting real-time situation awareness is equally important. The transmission of this big data from the network edge to the cloud servers for processing consumes

bandwidth and adds latency [2]. Instead of performing all the computations in a cloud server, some computations can be done already in the proximity of the origins of the data. However, memory, CPU power and communication capacity can be limited where data is measured and/or collected. Occasionally, the observed historical data needs to be recovered with sufficient detail.

When a statistical event appears to be predictable, its information content can be assumed to be small and, therefore, it does not need many bits of information to be stored. A stationary state is such a condition in which the amount of stored data bytes per event can be reduced. Very rare events occur also in a stationary state. However, reacting to all rare events is compulsory, since it is not possible to know beforehand whether an event is significant or not. Afterwards, it may be possible to judge whether the rare event was significant or not, provided that the stored data support the judgement. Responding to a rare event is possible if the latest history of events is available. A rare event is construed as rare in relation to the history of occurred events within a designated period of time. We exploit a probability distribution to identify the rarity of the events, because the percentiles of the distribution provide several fast ways to distinguish a rare event. The process of specifying the probability distribution is informative in itself as it presents indications of both

✉ Jorma Kilpi
jorma.kilpi@vtt.fi

Timo Kyntäjä
timo.kyntaja@vtt.fi

Tomi Rätty
tomi.ratty@vtt.fi

¹ VTT Technical Research Centre of Finland, Espoo, Finland

stationary and non-stationary events. The utilization of probability distributions also facilitates the comparison of current events to previously occurred events simply by comparing the different distributions.

In this paper, we provide the details of an algorithm, which attempts to perform the heuristic described above. It continuously tests whether the input data appear stationary and reacts to events that do not appear to be stationary. The algorithm transforms an input sequence of numbers into an output sequence of histograms. It also indicates how to interpret the histograms since their information content vary according to whether the histogram was computed by sorting a small amount of input data or estimated by assuming stationary input data.

The fields of smart grids and rural industrial systems serve as examples in which the proposed method can be applied. These systems produce massive amounts of data to be analysed, but sending it onward is not always cost effective. An example of this would be *industrial big data* related to systems *predictive maintenance* [22]. These sensing processes (e.g. smart grid cable conditions metering, machine and/or tool vibrations) produce large amounts of data to be analysed to predict possible system faults. Our method can produce both the reduced amount of data and indication of the changes within the measurements. Furthermore, this method can be integrated in 5G communication systems, in which Mobile Edge Computing (MEC) can be used as a service [1, 12].

This paper is structured as follows. In Section 2, we provide background to our solution, relate it to existent, similar research and further motivate our approach. In Section 3, we present our design assumptions, which are simultaneously requirements for our algorithm. Section 4 gives a top-level view of the algorithm and Section 5 its details. In Section 6, we describe the output of the algorithm. Sections 7 and 8 contain the statistical details. In Section 9, we present results of selected performance tests and finally the conclusions are drawn in Section 10.

2 Background and Related Work

Our control loop utilizes a sequential algorithm called *the extended P^2 -algorithm*. We will also discuss other alternatives later in this section. The P^2 -algorithm was first developed by Jain and Chlamtac [10] for the sequential estimation of a single percentile of a distribution without storing the observations, then extended by Raatikainen in [15] for the simultaneous estimation of several quantiles¹, and further analyzed by Raatikainen in [17]. Raatikainen used the extended P^2 -algorithm as a part of his PhD

thesis [16]. The publication [15] of Raatikainen contains the remarkably short pseudocode of the algorithm in such detail that its implementation is straightforward. Recently, the P^2 -algorithm was included in the comparison analysis of Tiwari and Pandey in [21].

The extended P^2 -algorithm is a very fast algorithm since it does not sort the input. However, there exist several computationally simpler algorithms which we will shortly discuss. The P^2 -algorithm approximates the inverse of the empirical *cumulative distribution function* (CDF) by utilizing a piecewise-parabola² to adjust the estimates of the percentiles. If the parabolic adjustment cannot be applied, then linear adjustment is used instead.

The extended P^2 -algorithm has some drawbacks:

1. It is suitable only for *strictly stationary* data sources, that is, for sources whose distribution really have such percentiles which do not depend on the observation time [4, 5, 14]. If the data source is not strictly stationary, then the estimates need not have any meaningful statistical interpretation [21]. This reduces the applicability of the extended P^2 -algorithm.
2. The estimates of the percentile near the median are more accurate, but the tail percentiles are less accurate. However, this is not a problem of the extended P^2 -algorithm alone, as the estimation of the extreme tail probabilities is in general very hard due to a lack of observations [9]. Our solution to this is simply to avoid the use of the extended P^2 -algorithm in the estimation of the extreme tail percentiles.
3. The extended P^2 -algorithm requires initialization data. As in [17], a good statistical representativeness of the initialization values is required. This may be hard to achieve in applications. It is worthwhile to use some computational resources to achieve better representativeness.
4. Initialization values must also be unequal, distinct values. This may cause problems in applications in which this cannot be guaranteed. Placing effort in the previous item (3) also helps this issue.

At first sight, these restrictions may sound hard for useful applications. However, we will show that the area of application can be significantly extended to cover some (not all) of the data sources which are not strictly stationary. The data source is allowed to change from a stationary state to another stationary state and the number of stationary states is not restricted.

Statistical background knowledge includes the research of Heidelberger and Lewis [9], which examines the quantile estimation for dependent data and presents many practical

¹The words *percentile*, *quantile* and *fractile* are synonyms.

²The name $P^2 = PP$ comes from the two P-letters in *P*iecewise-*P*arabola [10].

methods, such as the Maximum Transformation and the Nested Group Quantiles. The work of Feldman and Tucker [7] gives insight what can happen when we try to estimate a non-unique quantile and Zieliński [24] shows that estimation problems can occur even if the quantile is unique.

Alternatives to the extended P^2 -algorithm are the various recursive algorithms based on the *stochastic approximation method* of [18]. These algorithms include the Stochastic Approximation (SA) by Tierney [20], the Exponentially Weighted Stochastic Approximation (EWSA) by Chen et al. [6], Log Odds Ratio Algorithm (LORA) by Bakshi and Hoeflin [3] and the recent Dynamic Quantile Tracking using Range Estimation (DQTRE) of Tiwari and Pandey [21]. Yazidi and Hammer [8, 23], present recursive approaches based on *stochastic learning*. A common property of all the estimation algorithms is the need to obtain good initialization data.

The recursive algorithms EWSA, LORA and DQTRE adapt, to some extent, into some specific types of non-stationary data streams. However, the data source can be non-stationary in a myriad of different ways. If the distribution of data depends on the time of the observation, this dependency can be smooth or non-smooth. For example, EWSA was shown to adapt into a linear trend in [6], but in [21] DQTRE performed better in the simulated case of a sudden non-smooth change of the distribution. The estimation of time-dependent quantiles is conceptually difficult since, if the quantiles depend on time, then also their estimates depend on time. We are not aware of any general theory on the estimation of time-dependent quantiles.

Our contribution is to provide a *control loop* around the extended version of the P^2 -algorithm. The controlled extended P^2 -algorithm will be referred to as *Online Percentile Estimation (OPE)* and it consists of *two separate parts*: the extended P^2 -algorithm of [15] and our control loop. Our intention is to keep the control loop as computationally simple as possible. We emphasize the separation into two parts, because it is possible to replace the P^2 -algorithm with any of the recursive algorithms discussed above. For example, the authors of DQTRE [21] sketch the estimation of several quantiles simultaneously. The extended P^2 -algorithm has a built-in property to keep the estimates of the quantiles strictly monotone while DQTRE temporarily allows non-monotone quantile estimates.

The purpose of the control loop is to detect temporal behavior which is not strictly stationary and to avoid the use of the extended P^2 -algorithm in these situations. In cases which are not strictly stationary, the *sample percentiles*, obtained by sorting a relatively small sample of data, have the most reliable interpretation. For example, the sample median, denoted in this paper as $x_{(\lceil 0.5n \rceil)}$ in which n is the sample size, always has the interpretation that 50%

of the sample values has been smaller than $x_{(\lceil 0.5n \rceil)}$ and 50% has been larger than $x_{(\lceil 0.5n \rceil)}$. This holds true due to sorting of the sample and whether or not the data source was (strictly) stationary when the sample values were observed. In the non-stationary case, the data sample represents also the specific interval of time in which it was observed, and additional information is required for the correct statistical interpretation of it.

The strictly stationary case is just the opposite for the data source and, by its definition, it has a time-invariant distribution. Then the uncertainty, which results from the estimation instead of the computation of percentiles, can be tolerated. In this paper, the term *source percentile* indicates the percentile of the distribution of the (endless) data source. The temporal existence of such source percentiles can be assumed whenever there is *evidence* that the assumption of strictly stationary case holds.

The authors of the P^2 -algorithm [10] considered a hardware implementation (quantile chip) of their algorithm. Tiwari and Pandey [21] suggested a similar, for example FPGA or ASIC, implementation of their DQTRE algorithm. Our viewpoint is that all online quantile estimation algorithms benefit from a simple control loop which attempts to verify that the assumptions of the algorithms are not violated too heavily.

3 Assumptions and Requirements for OPE

For further reference, we present the assumptions that were made during the design process of OPE. These assumptions are also *requirements* for applying the algorithm.

1. It is assumed that there is random variability in the data.
2. It is assumed that the data source can be in some steady state for adequate periods of time that the stationary model can be applied. The number of different transient steady states is unlimited. Momentary periods of behavior, which is not strictly stationary, are allowed but, if the data source cannot ever reach any steady state in which the P^2 -algorithm can be deployed, then one must try to replace the P^2 -algorithm part of OPE by some of the mentioned recursive algorithms which adapt to non-stationary data. A control loop similar of OPE could then be defined to ensure that the changes in the data distribution are such that the adaptation makes sense, for example, the percentile estimates must remain monotone.
3. OPE performs similarly to a low pass filter. It is assumed that the interesting information is not on the small variations of the timescale. Exceptionally small or large values will be visible from the output of OPE, but their exact occurrence in time may be lost unless

a time-stamping feature is added to the algorithm. The handling of time stamps can be added but, for brevity, this issue is not discussed here. Also, trends can be observed by comparing the current histogram to the previous histogram(s), but a trend within a single histogram is not necessarily observed.

The design principle has been that in all circumstances the algorithm must perform rapidly and never produce unreliable data. Even if it is applied in a situation in which its applicability is not the best possible.

4 The Control Loop

The control is obtained by applying specific, statistical counter models and periodical testing of the estimates of the extended P^2 -algorithm against the model percentiles that are computed at the initialization phase. The model consists of

1. the model values of the specified source percentiles,
2. three counters and
3. two alarm mechanisms to test each observed value against the model. The alarm mechanisms include four threshold values and two more counters.

The model will be described in Section 5.1.

The model is *discarded immediately* if the correspondence with the data source is deemed insufficient. When the model is discarded, a new model is built from the new observations. The alarm mechanisms may invalidate the model even during the ongoing observations. The flow chart of Fig. 1 provides an overview of the control loop.

If the model remains valid, it is *updated periodically*. The updating is done only if the observed, new data fits the model. In this case we can assume that the new data contains information which can improve the existing model.

4.1 Phases of the Control Loop

The control loop consists of *configuration*, *initialization*, *model building* and *valid model* phases. We will describe the idea of these phases next.

4.1.1 Configuration

In the configuration phase, certain statistical and computational parameters are set. These values are never changed during the execution of OPE. The main configuration parameter is m , which is the number of percentiles that will be estimated. Section 5.3 will present more details on the configuration parameters. The data structure of the extended P^2 -algorithm requires $2m+3$ initial values. Table 2 contains a quick reference to the numeric values which depend on m .

Figure 1 The high-level flow chart of the control loop. The numbers in the flow chart will be explained in Section 6.

4.1.2 Initialization

In the initialization phase, we need a buffer of size $M_b \geq 2m+3$ observations and an in-place sorting algorithm to sort the buffer. The intention is to obtain $2m+3$ unequal and statistically representative initial values for the extended P^2 -algorithm. The sample percentiles are selected from the buffered M_b values after sorting. The sample percentiles will also be used as initialization values of the model for the source percentiles. Figure 2 is a sketch of the initialization process.

The word “marker” in Fig. 2 refers to the data structure of the extended P^2 -algorithm described in [15]. In the marker selection

$$q_j^* = \frac{j}{2m+2}, j = 1, \dots, 2m+1, \quad (1)$$

and in the model selection

$$q_j = \frac{j}{m+1}, j = 1, \dots, m. \quad (2)$$

Both selections are made from the sorted buffer, hence the initial values of the model are included in the initial values of the marker:

$$q_{2j}^* = q_j, j = 1, \dots, m. \quad (3)$$

Figure 2 Initialization process.

The notation $\lceil \cdot \rceil$ in Fig. 2 refers to the ceiling function and $x_{(i)}$ is the i :th *order statistic*, that is, the i :th smallest value. The q :th sample percentile is just the $\lceil qn \rceil$:th order statistic $x_{(\lceil qn \rceil)}$ in which $n \geq 1$ is the sample size and $0 < q < 1$.

The buffer, the marker and the model are each separate data structures. Each of these structures require memory allocations. The amount of memory that is needed depends on m alone, hence this memory allocation can be done in the configuration phase as it is not changed during the execution.

In this publication, the numbers q_j , $j = 1, \dots, m$ are called *selected probabilities*. They always satisfy (2) and are thus equally spaced. Equal spacing is not necessary, but it is difficult to justify any unequal spacing without making any prior assumptions about the data source. Also, the spacings are organized for the *quartiles* to always be included in them. The inclusion of the quartiles is not necessary either, but it will simplify the presentation and the programming of the algorithm considerably. The consequence of these agreements is that only values $m = 3, 7, 15, 31$ are considered, which may sound restrictive, but should turn out very natural. Splitting the intervals that the selected probabilities determine into halves is the most efficient approach to have qualitatively different granularities. Other choices for m or q_j do not convey any essential, additional prior value.

Sorting the buffer, which is an array of floating point numbers, is computationally the most challenging part of the algorithm, given the assumption that computational resources are limited. Especially in the online computation, in which the sorting will be regularly repeated. When $M_b > 2m + 3$ is sufficiently large, the statistical representativeness of the initialization data usually improves and also the chances that unequal values are obtained increases. Since

sorting is computationally hard and memory usage needs to be minimized, the value of M_b cannot be too large. Also, the values $x_{M_b+1}, x_{M_b+2}, \dots$ can arrive during the sorting which means that they need to be buffered somewhere else. We assume that the input queue is a *first-in-first-out* (FIFO) queue and can store a few values. This storage size and the arrival intensity of the observations also affect the size of M_b . Therefore, M_b is a parameter whose value is chosen according to the given computational resources.

The initialization phase ends successfully only if

$$x_{(1)} < x_{(\lceil q_1^* M_b \rceil)} < \dots < x_{(\lceil q_{2m+1}^* M_b \rceil)} < x_{(M_b)}. \quad (4)$$

Note that the initial values of the model will then also be distinct. However, the status of the model is not valid after initialization. The status may change while building the model.

4.1.3 Model Building

The phase of model building is divided into two subphases to compromise with the two conflicting criteria that

1st subphase: bad initialization data should be recognized and ignored as soon as possible,

2nd subphase: while checking the correspondence of the model with the data source requires numerous observations.

In the 1st subphase, we test that the model median $x_{0.5}^M$ behaves like the source median. Taking into account the random fluctuations, approximately 50% of the observed, new values should fall in both of the two bins

$$\left] -\infty, x_{0.5}^M \right] \text{ and } \left] x_{0.5}^M, \infty \right[$$

defined by the model median. The superscript “ M ” in $x_{0.5}^M$ stands for the “model”. Before the new value is given as input to the extended P^2 -algorithm, the relationship with the model median is accounted. After N_1 observations, the bin counts are tested. The test is described in Section 7.1.

In the 2nd subphase, we test that the model quartiles behave like the source quartiles. Approximately 25% of the observed, new values should fall in each of the four bins

$$\left] -\infty, x_{0.25}^M \right], \left] x_{0.25}^M, x_{0.5}^M \right], \left] x_{0.5}^M, x_{0.75}^M \right] \text{ and } \left] x_{0.75}^M, \infty \right[.$$

Before the new value is given as input to the extended P^2 -algorithm, this bin information is accounted. After N_2 observed values, the bin counts are tested. The test is described in Section 7.2.

If either of these two tests fail, the control goes back to the initialization and a new attempt is performed. Otherwise the model status is stated as valid.

We use the phrase model *building* instead of model validation, since the model is updated (changed) after the successful termination of the 2nd subphase. We then trust that the initialization phase was successful and that the marker values contain representative information about the source percentiles. The details of the update are described in Section 5.2.

4.1.4 Valid Model

In the model building phase, the model is already tested twice and, in successful cases, it is updated once. After this, the validity of the model is tested periodically after each n values in which n , the *period length*, is a variable. There is a lower bound for a meaningful period length which depends on m : $n \geq n_{\min}(m)$, but there is no upper bound. See Table 2 for values of $n_{\min}(m)$ and Section 8.1 on how these values are computed.

We use the notation $\hat{x}_{q_{2j}}^*$ for the current values of marker variables to distinguish them from the current values of the model variables x_{qj}^M . The test is twofold: 1) the model quartiles are tested that they still behave like the source quartiles and 2) the model percentiles x_{qj}^M and the current estimates of the same percentiles, which are the marker values $\hat{x}_{q_{2j}}^*$, are tested that they are not too far from each other. The first test is called the *bin frequency test* and the latter is called the *bin boundary test*. These tests are explained in more detail in Sections 7 and 8.

Upon the termination of the model building phase, the value of n is first set to $n = n_{\min}(m)$. After each successful test, the size of n is incremented by a step: $n \leftarrow n + s$. As additional tests are accepted by the model, we have more evidence on the strict stationarity and more confidence in the extended P^2 -algorithm. After each successful test the model is updated.

There are also two alarm mechanisms that may invalidate the model during the period. A single value above or below either of the *absolute* alarm thresholds may invalidate the model. Too many values above or below either of the *adaptive* alarm thresholds may invalidate the model.

5 Technical Details of the Control Loop

5.1 The Model

The model of the control loop consists of the m model percentiles

$$x_{q_1}^M, \dots, x_{q_m}^M, \quad (5)$$

which, for simplicity, are assumed to include the model quartiles $x_{0.25}^M, x_{0.5}^M, x_{0.75}^M$. In addition to these, the model contains three periodwise bin counters

$$\begin{aligned} n_1 &= \#\{x_i \mid x_i \leq x_{0.25}^M, i = 1, \dots, n\}, \\ n_2 &= \#\{x_i \mid x_i \leq x_{0.5}^M, i = 1, \dots, n\}, \\ n_3 &= \#\{x_i \mid x_i \leq x_{0.75}^M, i = 1, \dots, n\}, \end{aligned} \quad (6)$$

which depend on the period length n , two absolute alarm thresholds A_{abs}^- and A_{abs}^+ , two adaptive alarm thresholds A_{ada}^- and A_{ada}^+ and two periodwise counters for the adaptive thresholds:

$$\begin{aligned} r^- &= \#\{x_i \mid x_i < A_{ada}^-, i = 1, \dots, n\}, \\ r^+ &= \#\{x_i \mid x_i > A_{ada}^+, i = 1, \dots, n\}. \end{aligned} \quad (7)$$

The negative sign (−) refers to the lower tail of the empirical distribution and the positive sign (+) to the upper tail. All the counters related to the period are naturally initialized to zero in the beginning of the period.

The natural choices for the thresholds of the adaptive alarm are

$$A_{ada}^- = x_{q_1}^M \text{ and } A_{ada}^+ = x_{q_m}^M, \quad (8)$$

but, for example, the marker values

$$A_{ada}^- = \hat{x}_{q_1}^* \text{ and } A_{ada}^+ = \hat{x}_{q_{(2m+1)}}^* \quad (9)$$

can also be selected. The latter choice (9) makes sense if $m = 3$, otherwise (8) is better as their accuracy is typically better. If Eq. 8 are chosen, the expected number of observations in both of the bins $]-\infty, A_{ada}^-[$ and $]A_{ada}^+, \infty[$ are

$$\lceil q_1 n \rceil = \lceil (1 - q_m) n \rceil = \left\lceil \frac{n}{m+1} \right\rceil. \quad (10)$$

The period length n is always set and known before the next period begins. The expected values of the counters are also known beforehand.

The purpose of the absolute alarm thresholds is that, given the model, the events $x < A_{abs}^-$ and $x > A_{abs}^+$ are extremely rare. Since this may require some unavailable prior knowledge, it is always possible to switch off the absolute alarm mechanism by setting $A_{abs}^- = -\infty$ and $A_{abs}^+ = \infty$.

The thresholds of the absolute alarms may also be defined in such a way that they depend only on Eq. 5. We define them as

$$A_{abs}^- = x_{q_1}^M - K^-(x_{q_m}^M - x_{q_1}^M) \quad (11)$$

and

$$A_{abs}^+ = x_{q_m}^M + K^+(x_{q_m}^M - x_{q_1}^M), \quad (12)$$

in which K^- and K^+ are configurable parameters. Here the distance $x_{q_m}^M - x_{q_1}^M$ serves as a yardstick, a similar concept than “sigma” (σ) of the normal distribution $N(\mu, \sigma^2)$. The

choice of the parameters K^- and K^+ requires either prior knowledge or an estimation about the tail behavior of the data source.

The interval $[x_{q_1}^M, x_{q_m}^M]$ is called *the body* bin of the model distribution. If the model is valid, then it can be assumed that this interval contains the majority of the source distribution mass. Its complement consists of two bins which are called *the tail* bins.

5.2 The Updating of the Model

After successful test(s), the model percentiles are updated by the rule

$$x_{q_j}^M \leftarrow (1 - \alpha)x_{q_j}^M + \alpha\hat{x}_{q_{2j}}^*, j = 1, \dots, m, \quad (13)$$

where $0 \leq \alpha \leq 1$ is a configurable parameter.

The boundary value $\alpha = 0$ means that the initialization values are assumed perfect and the model is never changed. In practice, we are not confident that the model could ever be improved by the marker values. The boundary value $\alpha = 1$ means that the model is updated according to the marker values only. This would be plausible if we could know or trust that the marker values converge to the true percentiles. It can be expected that good choices of α will take into account the size of M_b or the ratio $M_b/(2m + 3)$. This is studied in Section 9.2. The parameter α is called a *learning* parameter, which is indicative of the confidence assigned to the marker values when a data source is fed as input.

In the phase of model building, Eq. 13 is the only action. In the phase of valid model, the adaptive and absolute alarm threshold values must also be updated, if they depend on the model percentiles as suggested in Eqs. 8, 11 and 12.

5.3 Configuration Parameters

Table 1 contains the main configurable parameters. Changing their values affect to the output of OPE. In Table 2, we have collected a quick reference on the numerical values that depend on m .

The message of $n_{min}(m)$ in Table 2 is that a simultaneous estimation of several percentiles from the same data requires a lot of data. The default values for N_1 and N_2 in Table 1 are obtained by first setting $N_1 + N_2 = n_{min}(m = 7) = 107$, see the last column of Table 2. For K^+ and K^- , values like 1, 2, 3, 4 or 5 can be considered, analogously to “one sigma”, “two sigmas”, and so on, and the defaults are chosen as $K^+ = K^- = 2$. The test level β affects the rate of false positives of the bin boundary test that is explained in Section 8. In addition, we suggest that the increment step could be $s = s(m) = 2m + 3$.

The data structure of the marker of the extended P^2 -algorithm requires memory of the size of $4 \times (2m + 3) =$

Table 1 The main configuration parameters.

Parameter	Explanation	Default
m	The number of the percentiles	
M_b	The internal buffer size	
α	The learning parameter	
s	The period length increment	$2m + 3$
N_1	The period length in the 1st subphase	36
N_2	The period length in the 2nd subphase	71
K^-	Lower tail alarm yardstick multiplier	2
K^+	Upper tail alarm yardstick multiplier	2
β	The bin boundary test level	0.01

$8m + 12$ floating point numbers. The data structure of the model requires $m + 4$ floating point numbers and 5 counters which can be integer variables.

6 The Output of OPE

The percentiles are output periodically. However, there are various choices in defining what the output of OPE should be. First of all, the output should depend on the phase of the control loop.

After n observations, where $n \geq n_{min}(m)$ is a variable, at least the m model percentiles are output. In the initialization phase, M_b input values are reduced to $2m + 3$ marker values and to m initial model values and it is natural to output at least the m model values. In model building, the period lengths are configurable parameters. It is natural to output the model values after updating. In all cases, the amount of input values are reduced to a very small number of output values. There is plenty of room to add *metadata* without affecting much of the data reduction.

After the initialization, the structure of the marker contains information about the minimum and the maximum observed values, denoted in Eq. 2 as $x_{(1)}$ and $x_{(M_b)}$. Until the next initialization, the minimum can only become smaller and the maximum can only become larger. Moreover, they are true observed values, not estimates of any kind. It is natural to include them in the output, so at least $m + 2$ data

Table 2 Quick reference to the values that depend on m .

m	$2m + 3$	$1/(m + 1)$	$n_{min}(m)$
3	9	1/4	20
7	17	1/8	107
15	33	1/16	525
31	65	1/32	2468

values are output. We will use the notation $x_{(1)}$ and $x_{(n)}$ for the minimum and maximum of the period, respectively.

Let us consider, for a moment, what values may be computed from the $m + 2$ output values. In many of the applications, the mean value is of interest. Assuming that the model *cumulative distribution function* (CDF) F^M is the piecewise linear CDF determined by the $m + 2$ parameters

$$(x_{(1)}, x_{q_1}^M, \dots, x_{q_m}^M, x_{(n)}), \quad (14)$$

see Section 8.2, then the expected value of F^M is

$$\bar{x}^M = \underbrace{\frac{1}{m+1} \sum_{j=1}^m x_{q_j}^M}_{\text{robust part}} + \underbrace{\frac{x_{(1)} + x_{(n)}}{2(m+1)}}_{\text{noisy part}}. \quad (15)$$

In addition to the mean $\bar{x}^M$, it is easy to compute the variance of the model distribution F^M . The formula of the variance is presented in the Section 8.3. Essentially, the uncertainty of $\bar{x}^M$ can also be quantified. Also, the robust and the noisy parts are separated in Eq. 15, which allow to check whether a change in the mean occurs merely due to noise or not.

If the model is valid, then Eq. 15 has a well-justified interpretation as an estimate of the source mean, otherwise it is still the mean of the observed sample. Therefore, we need to have the information about the validity of the model together with the *percentile data* (14). If the model is invalidated, then the failure reason is informative metadata.

In Table 3, an output called *phase code* is introduced. These are the numbers in the flow chart of Fig. 1. The purpose of the phase code is to indicate the exact place in the algorithm which caused the *output record* (16). This indicates the validity of the model.

After introducing the phase code, the output format can be defined as a record with a generic format

$$(\text{phase code, percentile data, metadata}). \quad (16)$$

This means that OPE will transform the input sequence of values into an output sequence of records. The phase

code generally helps in interpreting the record correctly and parsing the metadata. The phase code provides also an *applicability criteria*, see the assumption 2) of Section 3. If the output never contains records with the phase code 32, briefly *32-records*, then this algorithm is not applicable. Moreover, the proportion of 32-records is a measure of applicability of OPE. This metric of applicability is used in Section 9.

The phase code is an informative form of metadata. Other metadata, which is beneficial to obtain, is the period length. In the initialization and model building, the period lengths are M_b , $N_1 + N_2$, all of which are configuration parameters. In the valid model phase, the period length n is a variable whose final value depends on whether there was an alarm during the period of execution. The information of the period length helps to keep account of the total number of observations handled. In the cases in which the model is invalidated, it is plausible to add the test parameters and the alarm-causing observations to the metadata. Serial numbers for the output data integrity control and time stamps, if available, are also natural metadata.

We emphasize that OPE is *not* meant to be a *change detection* algorithm. If change detection is important, then a change detection algorithm can be applied to the output of OPE. The control loop is defined in such a way that the statistical tests in it will regularly give false positives. This happens on purpose, as it is possible to check if there has been a significant change in the source distribution from the history of the output data. Reducing the amount of testing will increase the risk of false negatives which is much worse, because it may not be possible to detect them afterwards from the recorded data alone.

Since n is typically considerably larger than m , the data reduction target is fulfilled. What remains to be shown is that the computational requirements are small and that the information loss depends on the parameters. The latter issue is studied in Section 9. The computational complexity results from the regular sorting of the buffer. Other computations that the control loop requires are described in Sections 7 and 8.

Table 3 The output phase codes.

Phase	Phase Code	Explanation
0	0	Configuration
10	11	Failure in the initialization
	12	Successful initialization
20	21	Failure in the 1st subphase
	22	Successful end of the 1st subphase
	23	Failure in the 2nd subphase
	24	Successful end of the 2nd subphase
30	31	Failure during or at the end of a period
	32	Successful end of a period

7 Bin frequency tests

The definition of the counters n_1 , n_2 and n_3 was given in Eq. 6. In Section 2.7 of [19], a standard approach to bin frequencies for i.i.d. data is given. The exact multinomial model of the bin frequency vector

$$(n_1, n_2 - n_1, n_3 - n_2, n - n_3)$$

is approximated by a multivariate normal model. This approach is not computationally fast, so we seek for a more simple approach.

The theoretical model behind the bin frequency tests presented below is based on an assumption of independent trials, reducing the use of the multinomial distribution to the binomial distribution by considering conditional counter values and the normal distribution approximation of the binomial distribution.

7.1 The Testing in the 1st Subphase

Consider the following simple, classical model for the first subphase of model building. Let X be a random variable modeling the observations and let $x_{0.5}^M = x_{(\lceil 0.5M_b \rceil)}$ be the median value of the model obtained in the initialization process of Fig. 2. Let

$$p = \mathbb{P}\{X \leq x_{0.5}^M\}$$

and as in Eq. 6

$$n_2 = \#\{X_i \mid X_i \leq x_{0.5}^M, i = 1, \dots, n\}.$$

If the strictly stationary assumption holds, then the distribution of X_i does not change and, therefore, p is a constant. If the X_i are independent and n is fixed beforehand, then n_2 is binomially distributed random variable with parameters n and p , denoted as $n_2 \sim \text{Bin}(n, p)$. The expected value of n_2 is np and variance is $np(1-p)$. Moreover, the $\hat{p} = \frac{n_2}{n}$ is the maximum likelihood estimator of p . Based on the central limit theorem (see [19]), an approximation can be done as follows

$$\begin{aligned} \mathbb{P}\{|\hat{p} - p| < a\} &= \mathbb{P}\left\{\frac{|n_2 - np|}{\sqrt{np(1-p)}} < \frac{na}{\sqrt{np(1-p)}}\right\} \\ &\approx 2\Phi\left(a\sqrt{\frac{n}{p(1-p)}}\right) - 1 = 0.95, \end{aligned} \quad (17)$$

if $a = z_{0.975}\sqrt{\frac{p(1-p)}{n}}$ where Φ is the CDF of the standard normal distribution and $z_{0.975}$ is the 97.5% percentile of Φ . Furthermore, since $z_{0.975} \approx 1.96 \leq 2$ and $p(1-p) \leq 1/4$, the 95% confidence interval of $\hat{p}$ can be written in a simplified form

$$]\hat{p} - a, \hat{p} + a[\subset \left] \hat{p} - \frac{1}{\sqrt{n}}, \hat{p} + \frac{1}{\sqrt{n}} \right[.$$

We set a requirement that $0.5 = 1/2$ belongs to this simplified approximative 95% confidence interval of $\hat{p}$:

$$\hat{p} - \frac{1}{\sqrt{n}} < \frac{1}{2} < \hat{p} + \frac{1}{\sqrt{n}}$$

which is equivalent with

$$n - 2\sqrt{n} < 2n_2 < n + 2\sqrt{n}. \quad (18)$$

Recall that the motivation of the 1st subphase is to quickly recognize and ignore bad initialization data. The length in the first subphase period, $n = N_1$, can be short provided

that the normal approximation (17) used above is plausible. Since we can assume that $p \approx 0.5$, this is not a hard requirement. Another criteria comes from the extended P^2 -algorithm. The markers, which are next to the median, affect to the accuracy of the estimated median. Therefore, the minimum length of the period (the number of trials) should satisfy $N_1 \geq n_{\min}(3)$.

7.2 Testing of the Quartiles

First notice that the counters n_1, n_2 and n_3 of Eq. 6 are not independent: $n_1 \leq n_2 \leq n_3$. We can start with n_2 and get the same condition as Eq. 18, but then we must consider *conditioned* events $n_1|n_2$ and $n - n_3|n - n_2$. Luckily the same method applies to conditional confidence intervals, that is, we can set a requirement that

$$\frac{n_1}{n_2} - \frac{1}{\sqrt{n_2}} < \frac{1}{2} < \frac{n_1}{n_2} + \frac{1}{\sqrt{n_2}} \quad (19)$$

which is equivalent with

$$n_2 - 2\sqrt{n_2} < 2n_1 < n_2 + 2\sqrt{n_2}, \quad (20)$$

and set a requirement that

$$\frac{n - n_3}{n - n_2} - \frac{1}{\sqrt{n - n_2}} < \frac{1}{2} < \frac{n - n_3}{n - n_2} + \frac{1}{\sqrt{n - n_2}} \quad (21)$$

which is equivalent with

$$n + n_2 - 2\sqrt{n - n_2} < 2n_3 < n + n_2 + 2\sqrt{n - n_2}. \quad (22)$$

Note the use of $1/2$ instead of $1/4$ in Eqs. 19 and 21, respectively, which is due to conditioning.

Together Eqs. 18, 20 and 22 form the test of a conditional approach started from n_2 which we describe symbolically as

$$n_1|n_2 \leftarrow n_2 \rightarrow (n - n_3)|(n - n_2).$$

The notation “|” refer to conditioning and the arrow reads as, for example, “if n_2 then n_1 given n_2 ” ($n_2 \rightarrow n_1|n_2$). However, there are four more cases that have to be considered and we list them symbolically below:

$$\begin{aligned} n_3 &\rightarrow n_2|n_3 \rightarrow n_1|n_2 \\ n_3 &\rightarrow n_1|n_3 \rightarrow n_2|(n_3 - n_1) \\ n_1 &\rightarrow (n - n_2)|(n - n_1) \rightarrow (n - n_3)|(n - n_2) \\ n_1 &\rightarrow (n - n_3)|(n - n_1) \rightarrow n_2|(n_3 - n_1) \end{aligned}$$

Each of these lead to a different group of three inequality chains listed below:

$$\begin{aligned} 3n - 4\sqrt{n} &< 4n_3 < 3n + 4\sqrt{n} \\ 2n_3 - 3\sqrt{n_3} &< 3n_2 < 2n_3 + 3\sqrt{n_3} \\ n_2 - 2\sqrt{n_2} &< 2n_1 < n_2 + 2\sqrt{n_2} \end{aligned}$$

$$\begin{aligned} 3n - 4\sqrt{n} &< 4n_3 < 3n + 4\sqrt{n} \\ n_3 - 3\sqrt{n_3} &< 3n_1 < n_3 + 3\sqrt{n_3} \\ n_3 - n_1 - \sqrt{n_3 - n_1} &< n_2 < n_3 - n_1 + \sqrt{n_3 - n_1} \end{aligned}$$

$$\begin{aligned} n - 4\sqrt{n} &< 4n_1 < n + 4\sqrt{n} \\ 2n + n_1 - 3\sqrt{n - n_1} &< 3n_2 < 2n + n_1 + 3\sqrt{n - n_1} \\ n_2 + n - 2\sqrt{n - n_2} &< 2n_3 < n_2 + n + 2\sqrt{n - n_2} \end{aligned}$$

$$\begin{aligned} n - 4\sqrt{n} &< 4n_1 < n + 4\sqrt{n} \\ 2n + n_1 - 3\sqrt{n - n_1} &< 3n_3 < 2n + n_1 + 3\sqrt{n - n_1} \\ n_3 - n_1 - \sqrt{n_3 - n_1} &< n_2 < n_3 - n_1 + \sqrt{n_3 - n_1} \end{aligned}$$

The bin frequency test accepts a triple (n_1, n_2, n_3) if at least one of the five inequality chain triples is satisfied. If all of them fail, then the test rejects the triple. The combined length of the period $n = N_1 + N_2$ of both subphases requires that the normal approximation is simultaneously plausible near all of the quartiles, which is one criteria. The extended P^2 -algorithm requires that the four markers next to the quartiles should also have some accuracy. Therefore: $N_1 + N_2 \geq n_{min}(7)$ is adequate.

7.3 Adaptive Alarm Bin Counts

Assume that Eq. 8 are selected and let the r^- and r^+ denote the adaptive alarm bin counters. Since n is fixed, r^- and r^+ are not independent. The binomial model is again applicable in which the success probability can be assumed to be $p = 1/(m+1)$ and it can be assumed that $n > (m+1)^2$. In the beginning of a period, we can set approximate 95% bounds to, for example, r^+ as

$$\frac{n}{m+1} - \sqrt{n} < r^+ < \frac{n}{m+1} + \sqrt{n} \quad (23)$$

but the conditioning $r^-|r^+$ leads to

$$\frac{n - r^+}{m+1} - \sqrt{n - r^+} < r^- < \frac{n - r^+}{m+1} + \sqrt{n - r^+} \quad (24)$$

which does not work similarly since the final value of r^+ is not known when the period begins. However, if the r^+ in Eq. 24 is understood as the current value, not the final value, then it can be implemented. Naturally, the roles of r^+ and

r^- in Eqs. 23 and 24 must also be considered in the opposite way:

$$\frac{n}{m+1} - \sqrt{n} < r^- < \frac{n}{m+1} + \sqrt{n} \quad (25)$$

and

$$\frac{n - r^-}{m+1} - \sqrt{n - r^-} < r^+ < \frac{n - r^-}{m+1} + \sqrt{n - r^-}. \quad (26)$$

Finally, the lower bounds in Eqs. 24 and 26 can be dropped since the bin frequency test at the end of the period will handle them. During the period, we are only concerned about drifts away from the current body bin of the observations and take care that the sum $r^- + r^+$ will not become too large.

8 Bin Boundary Test

The bin boundary test is based on the following theorem:

Theorem 1 Let $0 < q_1 < \dots < q_m < 1$. If $X_1, \dots, X_n$ are i.i.d., $X_i \sim F$ in which the q_j percentiles of F are x_{q_j} and F is strictly increasing in the neighborhoods of each x_{q_j} , then for all $\varepsilon > 0$

$$\mathbb{P} \left\{ \left| \frac{1}{m} \sum_{j=1}^m (X_{(\lceil q_j n \rceil)} - x_{q_j}) \right| > \varepsilon \right\} \leq 2 \sum_{j=1}^m e^{-2n\delta_{\varepsilon,j}^2}$$

where

$$\delta_{\varepsilon,j} = \min \{ F(x_{q_j} + \varepsilon) - q_j, q_j - F(x_{q_j} - \varepsilon) \}. \quad (27)$$

The proof of Theorem 1 in case $m = 1$ is given, for example, in Section 2.3.2. of the book [19]. Extending the proof from [19] to the more general case, stated above, requires basic probabilistic reasoning based on the triangle inequality. We omit the proof since it is more important to describe how it is used.

We use the marker $\hat{x}_{q_{2j}^*}$ of the extended P^2 -algorithm to approximate $X_{(\lceil q_j n \rceil)}$. Furthermore, the CDF F^M is strictly increasing with percentiles $x_{q_j}^M$, so we use F^M in place of F and $x_{q_j}^M$ in place of x_{q_j} .

The probabilistic interpretation of $\delta_{\varepsilon,j}$ is that the smaller it is, the more uncertain the estimation of x_{q_j} is. Theorem 1 states that the most uncertain percentile dominates the convergence of all the estimates. Figure 3 visualizes the relationship between ε and $\delta_{\varepsilon,j}$ when $F = F^M$.

What remains to be specified is the $\varepsilon > 0$. When the piecewise linear F^M is used as the model, the selection of ε depends on the scale of the bins. In order to simplify the next formulas, the following convenient notation is now taken

Figure 3 Interpretation of $\delta_{\epsilon,j}$ is easy with F^M .

into use:

$$q_0 = 0 \text{ and } x_{q_0}^M = x_{(1)} \quad (28)$$

$$q_{m+1} = 1 \text{ and } x_{q_{m+1}}^M = x_{(n)}. \quad (29)$$

Note that $x_{(1)}$ and $x_{(n)}$, the period's minimum and maximum observations, are not part of the model of the control loop, but they are part of the piecewise linear CDF model F^M . The superscript M is, therefore, justified.

With the notations Eqs. 28 and 29, the Eq. 27 of $\delta_{\epsilon,j}$ simplifies to

$$\delta_{\epsilon,j} = \min \left\{ \frac{1}{(x_{q_j}^M - x_{q_{j-1}}^M)}, \frac{1}{(x_{q_{j+1}}^M - x_{q_j}^M)} \right\} \frac{\epsilon}{m+1},$$

$j = 1, \dots, m$. Also, with the Eqs. 28 and 29 we define the maximum and the minimum bin widths as follows:

$$\epsilon_1 = \max \{x_{q_j}^M - x_{q_{j-1}}^M \mid j = 1, \dots, m+1\} \quad (30)$$

$$\epsilon_2 = \min \{x_{q_j}^M - x_{q_{j-1}}^M \mid j = 1, \dots, m+1\}. \quad (31)$$

Now, if $\epsilon \geq \epsilon_1$, then for all $j = 1, \dots, m$

$$\delta_{\epsilon,j} \geq \frac{1}{m+1}, \quad (32)$$

and if $\epsilon \leq \epsilon_2$, then for all $j = 1, \dots, m$

$$\delta_{\epsilon,j} \leq \frac{1}{m+1}, \quad (33)$$

We want to minimize the use of F^M since it is only one out of infinitely many CDFs which have the same percentiles $x_{q_j}^M$, $j = 1, \dots, m$. Due to computational feasibility with F^M choosing any $\epsilon < \epsilon_2$ would sound attractive, but it also means that we assume that F^M is an accurate model of the source distribution. This is hardly true, at least if $m = 3$. The accuracy improves according to the larger values of m .

The choices of $\epsilon = \epsilon_1/m$ and $\epsilon = \epsilon_2/m$ both seem quite plausible. When the model is updated, the values of ϵ_1 or ϵ_2 are recomputed and the test will remain the same.

Assume that either $\epsilon = \epsilon_1/m$ or $\epsilon = \epsilon_2/m$ has been selected. The bin boundary test accepts the current marker values $\hat{x}_{q_{2j}}^*$, $j = 1, \dots, m$, if

$$\left| \frac{1}{m} \sum_{j=1}^m (\hat{x}_{q_{2j}}^* - x_{q_j}^M) \right| \leq \epsilon \quad (34)$$

or else it accepts if

$$2 \sum_{j=1}^m e^{-2n\delta_{\epsilon,j}^2} > \beta \quad (35)$$

or else in all the other cases it results in rejection. The parameter β is assumed to have values such as $\beta = 0.05$ or $\beta = 0.01$ (default).

8.1 The Minimum Period Length $n_{\min}(m)$

Based on Theorem 1 we define $n_{\min}(m, \epsilon)$ such that

$$2 \sum_{j=1}^m e^{-2n\delta_{\epsilon,j}^2} < \frac{1}{2} \text{ when } n \geq n_{\min}(m, \epsilon). \quad (36)$$

The idea is that the probability estimate of Theorem 1 is not trivial when $n \geq n_{\min}(m, \epsilon)$.

The choice $\epsilon = \epsilon_1$ of Eq. 30 gives a lower bound Eq. 32 to $\delta_{\epsilon,j}$ which does not depend on ϵ or j and, therefore, it gives an upper bound to the probability bound of Theorem 1. We can find a candidate for $n_{\min}$ if we set a requirement that

$$2 \sum_{j=1}^m e^{-2n\delta_{\epsilon_1,j}^2} \leq 2me^{-2n/(m+1)^2} < \frac{1}{2}$$

from which $n_{\min}(m) = n_{\min}(m, \epsilon_1)$ is

$$n_{\min}(m) = \left\lceil \frac{1}{2} (m+1)^2 \log 4m \right\rceil.$$

8.2 The Piecewise Linear CDF Model

The piecewise linear model F^M is used mainly for visualization purposes. However, in the bin boundary test, it is also used to model the source distribution near the bin boundary points $x_{q_j}^M$, $j = 1, \dots, m$. The formula, with the Eqs. 28 and 29, is the following:

$$F^M(x) = \begin{cases} 0, & x < x_{q_0}^M \\ \frac{x - x_{q_{j-1}}^M}{(m+1)(x_{q_j}^M - x_{q_{j-1}}^M)} + \frac{j-1}{m+1}, & x_{q_{j-1}}^M \leq x < x_{q_j}^M \\ 1, & x \geq x_{q_{m+1}}^M \end{cases}$$

8.3 The Variance of F^M

Let $X \sim F^M$. Since $\text{Var}(X) = \mathbb{E}(X^2) - (\mathbb{E}X)^2$ and, since the formula $\mathbb{E}(X) = \bar{x}^M$ is already given in Eq. 15, only

the second moment remains to be computed. According to Eqs. 28 and 29, the formula is

$$\mathbb{E}(X^2) = \frac{1}{3(m+1)} \sum_{j=1}^{m+1} (x_{q_j}^M)^2 + x_{q_j}^M x_{q_{j-1}}^M + (x_{q_{j-1}}^M)^2.$$

9 Testing of OPE with the PoC

We have implemented a version of OPE using the programming language C [11]. It contains all the features described here except for the time stamps and the FIFO input queue which is not part of OPE itself. In this publication, this implementation is called *the proof of concept (PoC)*. In this Section, we will present a collection of tests which are based on this PoC.

Even though OPE is designed to be an online algorithm, only offline testing is discussed here. Offline testing allows to run PoC with exactly the same input, but with different values for the parameters. Also, the focus is on the correspondence with the source percentiles and the model percentiles of OPE. Our testing framework is to consider it as a *deterministic (time) one pass (or single pass)* algorithm, because this framework already has some theoretical results about *the space complexity*, see [13]. The space complexity of a deterministic one pass algorithm which computes the (population) median of a file with size N numbers is $\Omega(N)$, that is, *at least* of order N asymptotically.

If the global estimates of the source percentiles are computed from all 32-records' temporally local estimates, (e.g., the mean of the percentiles), this whole offline process can be considered as a deterministic, one-pass algorithm with a large input file as a source. This is achieved when the offline data source file is an output of a simulated, stationary time series. The abstract source percentiles are then practically the same as the population percentiles of the simulated file. In this special case, the size of all the output records (16) corresponding to a given input file is considered as a measure of the space complexity of OPE and it is $\Omega(N)$.

We mention just briefly that the PoC has been tested in internal projects at VTT with light sensor data, wireless mesh network delay measurement data, and vibration sensor data from a revolving machine.

9.1 Performance Metrics

First, we present the performance metrics which are considered within the above described framework of a deterministic, single-pass algorithm. These metrics are accuracy, bandwidth savings and the buffer size.

9.1.1 Accuracy

Measuring accuracy is reasonable only when the true source percentiles $x_{q_j}^{src}$, $j = 1, \dots, m$, exist and are known. Also, only 32-records can be assumed to contain accurate information about the source percentiles. For brevity, we will only report an overall accuracy which we define next. First, let

$$X_{q_j}^{32} = \frac{1}{n_{32}} \sum_{l=1}^{n_{32}} X_{q_j,l}^M, j = 1, \dots, m, \quad (37)$$

in which n_{32} is the number of 32-records and $X_{q_j,l}^M$ is the q_j :th model percentile value reported in the l :th 32-record. The averaging in Eq. 37 is reasonable in the test framework since all the 32-records model the same stationary state. The overall accuracy can be defined as the average over the m differences between the estimates $X_{q_j}^{32}$ and the true values:

$$\frac{1}{m} \sum_{j=1}^m (X_{q_j}^{32} - x_{q_j}^{src}). \quad (38)$$

This is a single number and the average over m allows a comparison between different m .

9.1.2 Bandwidth Savings

The IoT application idea of OPE is that the output records (16) are not stored where they are computed, but transmitted to a cloud server. Therefore, *bandwidth savings*, defined as

$$\frac{\text{Input(Bytes)} - \text{Output(Bytes)}}{\text{Input(Bytes)}},$$

is a plausible metric. The output is according to the provision of the PoC, which includes the phase code, the model percentiles, the period lengths and some other metadata on failure records. The output size depends on the parameters of the PoC. All the output records are included in Output(Bytes) .

9.1.3 The Size of the Buffer M_b

As the content of the buffer is sorted frequently and sorting is computationally expensive, the buffer size is a performance metric.

9.2 Study Case: Buffer Size vs. The Learning Parameter

As we discussed in Section 5.2, the relationship between the buffer size M_b and the learning parameter α is of interest. The effect of α is included in the performance case study which is described here.

9.2.1 Input Files

The input files were simulated realizations from a certain *moving average (MA)* model of *order* r which we will specify here. See Chapter 3.5.5 of [14], Section 3.4.3 of [5] or Chapter 3 in [4] for general technical details of $MA(r)$ time-series models. The different orders $r = 2, 4, 8, 16, 32$ and 64 are considered. For each r , there was a separate input file generated with $N = 500\,000$ values. The specific $MA(r)$ model used here is the following. Assume r is chosen and let Z_t be an infinite sequence of independent $N(0, 1)$ distributed random variables and define

$$X_t = Z_t + \sum_{i=1}^r \beta_i Z_{t-i}, \quad (39)$$

in which the parameters β_i are chosen as

$$\beta_i = \frac{1}{\sqrt{r}}, i = 1, \dots, r. \quad (40)$$

With these parameters, the corresponding $MA(r)$ model is *invertible*. Invertibility ensures that there is a unique $MA(r)$ model for a given *autocorrelation function (ACF)* [5]. With these choices $\mathbb{E}(X_t) = 0$ for all t and, due to independence of Z_t ,

$$\text{Var}(X_t) = 1 + \sum_{i=1}^r \beta_i^2 = 1 + r \left(\frac{1}{\sqrt{r}} \right)^2 = 2. \quad (41)$$

For $k \geq 0$, the ACF at lag $k \geq 0$ is

$$\rho(k) = \begin{cases} 1 & k = 0, \\ \frac{\sqrt{r+r-k}}{2r} & k = 1, \dots, r, \\ 0 & k > r. \end{cases} \quad (42)$$

For lags $k < 0$, the ACF is defined as $\rho(-k)$ and, therefore, $\rho(k) \geq 0$ for all lags k . The time series (X_t) is strictly stationary and, since Z_t are $N(0, 1)$ distributed, by Eq. 41 the X_t are $N(0, 2)$ distributed. The source percentiles $x_{q_j}^{src}$ of the distributions of X_t are the same with all the input files and their numerical values are known with high precision.

The previously defined $MA(r)$ models with the ACF (42) were chosen because, with larger orders r , they have the property that the realizations tend to have long sequences of consecutive values which are either above or below zero. Zero is also the source median for all of the specified $MA(r)$ models. Therefore, the property $\rho(k) > 0$ for all the lags $|k| \leq r$ challenges the initialization and the model building phases of the control loop, which are the novel features. This is visualized in Fig. 4. If it were that $\rho(k) < 0$ with some small lag k , then the initialization and the model building would benefit from this.

Figure 4 Applicability when maximized over all parameters (k, h) .

9.2.2 Parameter Ranges

For each of the six input files, there are different values considered for the parameter $m = 3, 7, 15$. For each of these m , the buffer sizes

$$M_b(k) = k \times (2m + 3), k = 1, \dots, 12, \quad (43)$$

are considered. Thus, the buffer sizes are multiplicities of $2m + 3$, which is the minimal possible buffer size. For each pair $(m, M_b(k))$, the discretized learning parameter values

$$\alpha(h) = \frac{h}{24}, h = 0, \dots, 24, \quad (44)$$

are considered. Then $\alpha(h) - \alpha(h-1) = 1/24 \approx 0.04$. Note that the end points $0 = \alpha(0)$ and $1 = \alpha(24)$ are included. Each time-series file is then given as input to the PoC with different (k, h) values for the parameters $(m, M_b(k), \alpha(h))$. The accuracy and the bandwidth savings are then considered as functions of (m, M_b, α) . Other parameters were fixed to their defaults shown in Table 1.

9.2.3 Results

First, a study of the applicability of OPE for the specified $MA(r)$ input files with different orders r was made. The proportion of 32-records in the output is the criterion of applicability as defined in Section 6. In Fig. 4, an exhaustive search was made over all pairs (k, h) , $k = 1, \dots, 12$ and $h = 0, \dots, 24$, to find the pair with the maximum applicability with the given m and r . If every initialization were successful and always ended up in exactly one 32-record, then the proportion of 32-records would be quite exactly 25%. The proportion of 32-records can exceed the 25% level if the control loop model stays valid, recall the flow chart of Fig. 1. The applicability reduces when the autocorrelation (42) becomes worse since the statistical

Figure 5 The parameter pairs (k, h) which provide the maximum applicability.

representativeness of the available data in the initialization and model building phases becomes poorer.

In Fig. 5, the discretizing parameter pairs (k, h) , which produced the maximum applicability in Fig. 4, with the given m and r , are visualized in the space of (k, h) . The pairs (k, h) are not necessarily unique. In the non-unique case, the pair with the smallest k is shown. The cases $m = 3, 7, 15$ are considered and for each m the points corresponding to input data of orders $r = 2, 4, 8, 16, 32$ and 64 are plotted and joined. The other end points, with $r = 64$, are marked. The horizontal axis is in scale $k = M_b(k)/(2m + 3)$ only. The vertical scale is given in h and in $\alpha(h)$ scale. There is not much structure in Fig. 5, but it appears that two simple policies may be behind the best applicability: either the buffer size M_b is large and α is small, or the smaller buffer size is accompanied with a larger α . The notable things are that the choice of the parameters M_b and α can significantly affect the performance when the applicability is low ($r = 64$) and that the maximum considered buffer size $M_b(12) = 12(3m + 3)$ is not automatically needed.

Figure 6 shows how the accuracy depends on the applicability. The gray area is a rough estimate of the 99% confidence interval in the case $m = 15$, obtained from Eq. 38 by assuming in Eq. 37 that (see Chapter 2.1.1 of [19])

confidence interval in the case $m = 15$, obtained from Eq. 38 by assuming in Eq. 37 that (see Chapter 2.1.1 of [19])

$$X_{qj}^{32} \sim N\left(x_{qj}^{src}, \frac{(1 - q_j)q_j}{n_{32}}\right), j = 1, \dots, m, \quad (45)$$

which is considered as the null hypothesis of accuracy in our test framework.

Finally, the bandwidth savings is presented in Fig. 7. It is computed for the maximizing pairs (k, h) . If the applicability of OPE is high and the data source is stationary, then the bandwidth savings with $m = 15$ can be more than 97%. This indicates that there is not much information in the simulated input data with $r = 2$. This would be interpreted as it is almost white noise with a minor memory effect. Recall that all the output records are included in bandwidth savings.

If the buffer size M_b is limited beforehand in an application, but some time-series samples of the application data are available in the design phase, one can try to estimate a sufficient value for the α of OPE in a similar way as presented in this section. This would involve estimating the sample autocorrelation function of the application data, simulating data with a similar autocorrelation structure, feeding it to OPE and then choosing the α with the best applicability, that is, the largest proportion of 32-records.

9.3 Complexity Issues

Our control loop alternates between sorting a fixed size buffer M_b and the extended P^2 -algorithm. Even though the general complexity of sorting is of the order $O(N \log N)$, since the buffer size is constant, the complexity of the sorting phases is at most of constant order, $O(M_b \log M_b) = O(1)$. Also, the control loop adds complexity to the whole and our main effort has been to guarantee that this complexity of the control loop is also constant. Thus, the overall complexity of OPE is of constant order, but this

Figure 6 Accuracy vs. applicability.

Figure 7 Bandwidth savings.

constant is not the minimal possible. Indeed, there is space for further research on developing an algorithm with lower complexity. For example, replacing the P^2 -algorithm by the other algorithms that were discussed in Section 2 may provide a controlled algorithm with lower complexity.

10 Conclusion

We have described a sequential, online algorithm OPE, which either computes or estimates the temporal percentiles of the observed, empirical distribution of the data. This is done without storing all of the observations. It works with a parameter-defined, fixed-size small amount of memory and limited CPU power. It can be used for the statistical reduction of a numerical data stream. It transfers the univariate input stream into the output stream of vectors in which each vector can be interpreted as an empirical distribution of data starting immediately after the previous vector. For IoT applications, there is significant reduction in the amount of data (bytes) needed to be transmitted to a cloud server. The output data is structured in a (cumulative) histogram format, which quickly recognizes such trends or anomalies that are visible in the histogram scale. We provided requirements that should be satisfied when OPE is applied and we illustrated that OPE contains a built-in applicability metric, the proportion of 32-records. We also presented a performance study, which was specially designed to challenge the control loop part of OPE. A lesson learned from this performance study is that 32-records are self-informative in the sense that their interpretation does not require more metadata than the knowledge that they are 32-records and they remain sufficiently accurate even when the applicability of OPE decreases.

Acknowledgements The authors would like to thank Heikki Ailisto and Heli Helaakoski for supporting this study at VTT.

Author Contributions JK is responsible for the algorithm development and the theoretical foundations of the control loop. JK programmed the PoC with valuable help from TK. JK, TK and TR all contributed in the writing and revising of the manuscript. All authors read and approved the final manuscript.

Funding Open access funding provided by Technical Research Centre of Finland (VTT). This study was funded by ECSEL-MegaM@Rt2 (MegaModelling at Runtime - scalable model-based framework for continuous development and runtime validation of complex systems) and BusinessFinland-RAGE (Real-Time AI-Supported Ore Grade Evaluation for Automated Mining) projects.

Code Availability The C-code of the PoC is available from the corresponding author on reasonable request.

Declarations

Conflict of Interests The authors declare that they have no conflict of interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Abbas, N., Zhang, Y., Taherkordi, A., Skeie, T. (2018). Mobile edge computing: a survey. *IEEE Internet of Things Journal*, 5(1), 450–465. <https://doi.org/10.1109/JIOT.2017.2750180>.
2. Adegbija, T., Rogacs, A., Patel, C., Gordon-Ross, A. (2018). Microprocessor optimizations for the internet of things: a survey. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits And Systems*, (Vol. 37 pp. 7–20), <https://doi.org/10.1109/TCAD.2017.2717782>.
3. Bakshi, Y., & Hoeflin, D.A. (2006). Quantile estimation: a minimalist approach. In *Proceedings of the 2006 winter simulation conference*.
4. Brockwell, P.J., & Davis, R.A. (1991). *Time series: theory and methods*, 2nd edn. Berlin: Springer.
5. Chatfield, C. (2004). *The analysis of time series, an introduction*. Florida, US: CRC Press.
6. Chen, F., Lambert, D., Pinheiro, J.C. (2000). Incremental quantile estimation for massive tracking. In *Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining, KDD '00* (pp. 516–522). New York: ACM. <https://doi.org/10.1145/347090.347195>.
7. Feldman, D., & Tucker, H.G. (1965). Estimation of non-unique quantiles. *The Annals of Mathematical Statistics*, 37, 451–457.
8. Hammer, H.L., & Yazidi, A. (2017). Incremental quantile estimators for tracking multiple quantiles. In Benferhat, S., et al. (Eds.) *IEA/AIE 2017 Part I, no. 10350 in LNAI* (pp. 202–210). https://doi.org/10.1007/978-3-319-60042-0_23.
9. Heidelberger, P., & Lewis, P.A.W. (1984). Quantile estimation in dependent sequences. *Operations Research*, 32(1), 185–209. <https://doi.org/10.1287/opre.32.1.185>.
10. Jain, R., & Chlamtac, I. (1985). The P^2 algorithm for dynamic calculation of quantiles and histograms without storing observations. *Communications of the ACM*, 28(10), 1076–1085.
11. Kernighan, B.W., & Ritchie, D.M. (1988). *The C programming language*, 2nd edn. Prentice Hall Professional Technical Reference.
12. Leligou, H.C., Zahariadis, T., Sarakis, L., Tsampasis, E., Voulkidis, A., Velivassaki, T.E. (2018). Smart grid: a demanding use case for 5g technologies. In *2018 IEEE international conference on pervasive computing and communications workshops (percom workshops)* (pp. 215–220). <https://doi.org/10.1109/PERCOMW.2018.8480296>.

13. Munro, J.I., & Paterson, M.S. (1980). Selection and sorting with limited storage. *Theoretical Computer Science*, 12(3), 315–323. [https://doi.org/10.1016/0304-3975\(80\)90061-4](https://doi.org/10.1016/0304-3975(80)90061-4). <https://www.sciencedirect.com/science/article/pii/0304397580900614>.
14. Priestley, M.B. (1981). *Spectral analysis and time series*. Academic Press: London, New York.
15. Raatikainen, K. (1987). Simultaneous estimation of several percentiles. *Simulation*, 49(4), 159–163. <https://doi.org/10.1177/003754978704900405>.
16. Raatikainen, K. (1990). Modelling and analysis techniques for capacity planning. Department of Computer Science University of Helsinki.
17. Raatikainen, K. (1990). Sequential procedure for simultaneous estimation of several percentiles. *Transactions of the Society for Computer Simulation*, 7(1), 21–44.
18. Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3), 400–407.
19. Serfling, R.J. (1980). *Approximation theorems of mathematical statistics*. New York: Wiley.
20. Tierney, L. (1983). A space-efficient recursive procedure for estimating a quantile of an unknown distribution. *SIAM Journal on Scientific and Statistical Computing*, 4(4), 706–711. <https://doi.org/10.1137/0904048>.
21. Tiwari, N., & Pandey, P.C. (2018). A technique with low memory and computational requirements for dynamic tracking of quantiles. *Journal of Signal Processing Systems*. <https://doi.org/10.1007/s11265-017-1327-6>.
22. Yan, J., Meng, Y., Lu, L., Li, L. (2017). Industrial big data in an industry 4.0 environment: challenges, schemes, and applications for predictive maintenance. *IEEE Access*, 5, 23,484–23,491. <https://doi.org/10.1109/ACCESS.2017.2765544>.
23. Yazidi, A., & Hammer, H. (2016). Quantile estimation in dynamic and stationary environments using the theory of stochastic learning. *Applied Computing Review*, 16(1), 15–24. <https://doi.org/10.1145/2924715.2924717>.
24. Zieliński, R. (1998). Uniform strong consistency of sample quantiles. *Statistics & Probability Letters*, 37, 115–119.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

tems, IoT networks, and 5G security.

Timo Kyntäjä graduated as a M.Sc. (Information Technology) from the University of Jyväskylä at 1994. He has been a researcher for two decades in the areas of telecommunication and distributed systems. His competence is targeted on communications systems software design and implementation. The most recent application areas are Smart Energy Sys-

he has served as a Reviewer for multiple journals and at numerous conferences.

Tomi Rätty received the Ph.D. degree in information processing science from the University of Oulu, Finland. He is a Principal Research Scientist with the VTT Technical Research Centre of Finland, Espoo. His current research interests include machine-learning technologies, data science, and software platforms. He is the author or co-author of more than 40 papers published in various conferences and journals, and

Jorma Kilpi received the M.S. and Lic.Phil. degrees in mathematics from the University of Helsinki, Department of Mathematics, Finland. From 1994 to 1998 he was a research assistant at the Department of Mathematics, University of Helsinki. Since 1998, he has been working at VTT Technical Research Centre of Finland as a Senior Scientist. His research interests include applied statistics, stochastics, machine learning and information theory.

His application domains include telecommunications, earth observations and vehicular traffic. Mr. Kilpi is a member of the Finnish Mathematical Society and the European Mathematical Society.