A Delay-Resilient and Quality-Aware Mechanism over Incomplete Contextual Data Streams

Christos Anagnostopoulos[a,*], Kostas Kolomvatsos[b]

[a] *School of Computing Science, University of Glasgow, G12 8QQ UK*
[b] *Department of Computer Science, University of Thessaly, 35 100 Greece*

Abstract

We study the case of scheduling a Contextual Information Process (CIP) over incomplete multivariate contextual data streams coming from sensing devices in Internet of Things (IoT) environments. CIPs like data fusion, concept drift detection, and predictive analytics adopt window-based methods for processing continuous stream queries. CIPs involve the continuous evaluation of functions over contextual attributes (e.g., air pollutants measurements from environmental sensors) possibly incomplete (i.e., containing missing values) thus degrading the quality of the CIP results. We introduce a mechanism, which monitors the quality of the contextual streaming values and then optimally determines the appropriate time to activate a CIP. CIP is optimally delayed in hopes of observing in the near future higher quality of contextual values in terms of validity, freshness and presence. Our time-optimized mechanism activates a CIP when the expected quality is maximized taking also into account the induced cost of delay and an aging framework of freshness over contextual values. We propose two analytical time-based stochastic optimization models and provide extensive sensitivity analysis. We provide a comparative assessment with sliding window-centric models found in the literature and showcase the efficiency of our mechanism on improving the quality of results of a CIP.

Keywords: Incomplete multivariate context streams, quality of streaming data, Internet of Things, optimal stopping theory.

1. Introduction

Huge volumes of sensory data in Internet of Things (IoT) environments are continuously generated as streams, which need to be analyzed on-line. Multivariate streaming data can be considered as one of the main sources of what is coined *big data*. Many IoT applications deal with multivariate contextual data

*Corresponding author
Email addresses: christos.anagnostopoulos@glasgow.ac.uk (Christos Anagnostopoulos), kolomvatsos@cs.uth.gr;kostasks@di.uoa.gr (Kostas Kolomvatsos)

coming in the form of time series like Wireless Sensors Network (WSN) data. IoT applications for e.g., forest monitoring [34], and statistical analytics applications over large-scale data streams require efficient, accurate, and timely data analysis to facilitate (near) real-time decision-making and situational context awareness [3]. A contextual data stream (or context stream) contains values from contextual parameters corresponding to IoT sources, e.g., humidity sensor. IoT applications exploit all such context, for instance, to (i) infer the top-k recent congested segments of city road networks, or (ii) obtain regularly the highest pollution level within a time horizon in a smart city.

1.1. Motivation & Challenge

All pieces of context captured by IoT **contextual information** sources are considered as continuous context streams, where **Contextual Information Processes** (CIPs) are applied to (i) reason over incomplete data and (ii) infer new knowledge. Recent development in *big data* analytics [9] examines large amounts of contextual data to uncover hidden patterns, correlations, and other insights. With CIPs, it is possible to analyze contextual data from streams almost immediately an effort that is less efficient with more traditional business intelligence solutions. The major challenge in a stream of contextual information is that contextual data are usually imprecise, incomplete, and noisy including missing and out-of-order data. Such incompleteness is due to various errors, e.g., data interference and limitations of sensor equipment, limited WSN resources, and harsh deployment environments. The values of contextual parameters are missing, or not available, or stale. In such cases, we observe values for only a subset of contextual parameters. Hence, a CIP, which ranges from: a **data aggregation function** to an **information fusion engine**, towards to a **context inference process**, cannot be accurately evaluated. This degrades the quality of the CIP result in terms of prediction accuracy and consistent inference.

Accurate CIP results rely on the **information quality** of context stream. Stream quality is expressed by meta-information, e.g., value validity, expiration thresholds, and missingness indicators. Inaccurate observations due to missing values can be either corrected (*data imputation*) [2] or removed. However, this yields bias in the extracted knowledge and the CIP results [12]. The baseline solution is invoking a **Missing Values Substitution (MVS)** process e.g., [6], over context streams at every time *before* the invocation of a CIP. Evidently, this imposes significant computational effort. One has to decide whether such MVS methods should be continuously invoked and at which rate. The trade-off between information quality and computational resource utilization is studied in this paper, which motivated us to introduce an intelligent mechanism for scheduling CIP invocations over incomplete context streams. The motivation here is to compensate the degree of information quality that an IoT application requires with the available computational resources, especially, when dealing with remote sensing devices. An optimally scheduled CIP over incomplete contextual information streams in order to avoid continuous calls of MVS methods establishes a mechanism that achieves the information quality levels of the ap-

plication requirements. A time-optimized CIP scheduling algorithm is deemed appropriate to cope with that trade-off.

Due to incomplete data, it is difficult to determine/predict a *time instance* at which the entire set of contextual values of all context streams are present (not missing) to apply a CIP. The major research challenge here is to decide *when* to apply a CIP over context streams that are of 'good' quality. A CIP process could not be performed continuously but once within a finite time interval, which guarantees at some point quality data, thus saving computational resources. That is a CIP could be executed only when the 'necessary' information for guaranteeing quality results is available. With the term 'necessary' information we denote a degree of completeness of the context streams in order to maximize the quality of CIP results. Ideally, the maximum quality of CIP results is obtained when all values are complete/present/timely/available. The following question arises: *Given incomplete context streams, when one should activate a CIP for maximizing the quality of results by avoiding continuous call of MVS methods to save computational resources?*

2. Literature Review & Contribution

2.1. Literature Review

We report on the CIPs that are applied over incomplete context streams and are activated once necessary information for guaranteeing quality results is available. We distinguish two basic types of a CIP over context streams: (i) CIP for Data Management and (ii) CIP for Knowledge Discovery. CIP for Data Management refers to handling, querying, scheduling, and storage of context sensory data streams. This type of CIP refers basically to data reduction and (statistical) summaries. In both cases, queries (e.g., aggregation queries and top-k queries) over context streams are executed over a summary, which refers to a compact data-structure that captures the underlying distribution of the data streams. Moreover, the well-known 'sliding window' CIP is considered as a fundamental technique for producing approximate answers to a data stream query like aggregation operators `SUM`, `AVG`, and `COUNT`. The idea behind the sliding window is to perform detailed analysis and data processing over the most recent data items. This idea has been adopted in many data stream mining and management systems [1], [31]. It is worth noting that all sliding window methods invoke a CIP operator continuously over a fixed-size window. Once a piece of contextual value is missing/incomplete, then all these methods attempt to predict the missing value and then apply any CIP operator over the window.

CIP for Knowledge Discovery studies methods and algorithms for extracting knowledge from volatile context streams [13], [7]. Among such types of CIPs, the on-line learning and model adaptation, concept drift detection and outliers identification have become important research topics. Pioneer contextual data stream mining processes include stream clustering [17], [8], outliers detection [33], classification and prediction [24], frequent pattern [21], time series [22] and

change detection [18], [26]; the list is not exhaustive. Moreover, contextual information fusion processing has gained significant importance. The objective of this type of CIP is to infer the relevant states and events of the system that is being observed or activity being performed. Finally, contextual inference methods are generally applied in situational context inference [3], where inference is taken based on perceived situational knowledge.

2.2. Rationale

Contextual data streams pose a challenge to large-scale predictive analytics and real-time CIP. There is an increasing need for intelligent methods to check and correct (sensed) context to ensure that is of the highest quality. The above-mentioned CIPs demand computational resources to proceed in a continuous manner. This motivated us to introduce an *optimally scheduled context quality aware mechanism*, which improves the quality of the delivered context to the back-end monitoring IoT system for (near) real time information processing and knowledge extraction through CIPs. The proposed mechanism materializes quality assessment **prior to the delivery of context to the system** by minimizing the induced bias in statistical inference and/or estimation processes due to incomplete context. **Our mechanism decides when to 'deliver' context streams of high quality, by saving computational resources and avoiding the invocation of data imputation methods.** This yields to: (1) *improve* the quality of the CIP results at the expense of a controlled delay, (2) *avoid* the continuous activations of data imputation methods each time incomplete context is received to the back-end monitoring system, as proposed in all approaches of the related work, and (3) *avoid* the continuous activation of CIPs, as adopted in all related works.

To the best of our knowledge, the proposed delay-resilient and quality-aware mechanism over incomplete context streams is novel for improving the quality of the CIP results and saving computational resources. The novel concept of an optimal delay mechanism that controls the scheduling of the application of a CIP operator differentiates our idea with prior approaches for CIPs. Through our mechanism, one does not need to continuously invoke a CIP operator over a given window (of variable or fixed length). Moreover, our mechanism does not rely on the invocation of a MVS method like [27]–[16], for predicting the incomplete data before the invocation of a CIP operator. Many CIPs for Data Management and Knowledge Discovery over incomplete data require the forecasting of any incomplete data (through MVS methods) and then apply a CIP operator continuously, thus, imposing high computation resources. As it will substantiated by our comparative assessment in Section 7, our mechanism (through two stochastic optimization models): (1) avoids the invocation of any MVS method and/or contextual data cleaning/imputation method and (2) minimizes redundant invocations of CIP operators by optimally scheduling the CIP invocations achieving high accuracy results along with efficient resource usage. The trade-off between efficient resource usage and achieved information accuracy is indicated through the comparative assessment of our models with certain sliding window models in [27], [10], and [16] from the literature. The authors would like also

to mention their prior work [19] on dealing with the optimal maintenance of the top-k list of objects over incomplete multivariate data streams. Our previous work and the current one introduce a framework of optimally scheduled mechanisms over contextual data streams.

We propose an optimally scheduled mechanism for finding the most appropriate time instance to activate a CIP (e.g., data aggregation, multivariate fusion, outliers detection, event identification and inference, concept drift, context classification) over incomplete context streams. The introduced mechanism helps CIPs to improve the quality of their output. The proposed mechanism is applied prior to the performance of a CIP in order to provide a time-optimized decision on when to be activated. Moreover, our mechanism can be applied prior to the CIP to maximize the quality of results (e.g., correctness of context inference, minimizing false alarms).

The idea of the proposed mechanism is to continuously assess the quality of the observed context stream (defined later) and on the fly decide whether to activate a CIP or not based on the recent history the quality pattern of the context streams. The objective is to maximize the expected quality of CIP results subject to a quality guarantee level specified by the application/user. We cast this time-optimized decision problem as an optimal stopping time problem derived from the Optimal Stopping Theory (OST) [28]. The OST is proved to be very efficient in cases where we try to find the appropriate time instance to stop the observation of a stochastic process with the objective of maximizing our payoff. Naturally, we build our mechanism on the principles of OST to maximize the quality of CIP results by inducing a delay. Through this delay we attempt to balance between *immediate CIP execution* and *delayed CIP execution* in hopes of observing e.g., more non-missing values before applying aggregation and fusion operators over context streams.

The outcome of the mechanism indicates whether we should stop observing the quality of the context streams and activate a CIP, or to continue. This delay-resilient CIP activation supports applications that can tolerate some delay in hopes of obtaining high quality results. As it will be shown in the performance & comparative assessment, our mechanism provides a wide range of quality results, ranging between medium quality results with almost zero delay and high quality results with a *acceptable* delay. **Through this acceptable delay (in terms of the application tolerance), the system saves computational resources and eliminates redundant CIP activations.**

2.3. Contribution

The contribution of this work is summarized as follows:

- A novel time-optimized, quality-aware mechanism based on the OST, which decides when a CIP should be activated over incomplete context streams by guaranteeing the highest possible quality results.
- Two novel analytical time-optimization stochastic models that derive the optimal time for activating a CIP taking into account: (i) the cost of delay

due to additional observations and (ii) an aging factor over the currently buffered contextual values.

- Comprehensive experimental results showcasing the benefits of our mechanisms over real-data context streams with missing values involving widely applied aggregation and fusion operators vis-à-vis the sliding window-centric model.

- Comprehensive comparative assessment of our models with the sliding window models: the Incremental Mean Model (IMM) and Skewness Sensitive Model (SSM) in [27], the Average Nearest Exponential Smoothing Model (ANESM) in [16], and the Exponential Moving Average Model (EMAM) in [10] over incomplete contextual streams using real datasets with widely applied aggregation CIP operators.

In this paper, we focus on the optimally scheduled mechanism of two widely-used CIPs: aggregation operator process and fusion process over context streams. The former process is an instance of the CIP for Data Management category dealing with aggregate queries (e.g., `COUNT`, `MIN`, `AVG`) over sliding windows with incomplete multivariate streaming contextual data. The latter process refers to an instance of the CIP for Knowledge Discovery category dealing with events detection/inference over incomplete multivariate data streams. Both CIPs are core components of data stream management systems.

3. Preliminaries

3.1. Data Stream Quality

Consider a discrete time domain $\mathbb{T} = \{1, 2, \ldots\}$ and time instance $t \in \mathbb{T}$.

Definition 1 (Quality Context Stream). *A quality stream s of a contextual attribute is an infinite sequence of $\langle t, x, I(x)\rangle$, where $x \in \mathbb{R}$ is the contextual value at time instance $t \in \mathbb{T}$ and $I(x)$ is meta-data representing the quality of x at time t.*

At time t we observe the values x_i of n context streams, thus forming a stream vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$. Let also $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ be the set of context streams s_i, $i = 1, \ldots, n$. We report on certain notions of data stream quality $I(x)$ of contextual value x coming from a stream $s \in \mathcal{S}$. The authors in [29] discuss certain dimensions of the Data Quality (DQ). The generic DQ model allows for a number of DQ dimensions that are adaptable to various application requirements. In our case we adopt such DQ dimensions to improve the accuracy of CIPs results. We report on the DQ dimensions provided in [32] for interpreting the meta-data quality indicator $I(x)$.

The DQ dimension *accuracy* describes the maximal systematic numeric error of a sensor measurement x. It indicates the degree to which x correctly describes the 'real world' phenomenon or event being described. The *confidence* DQ dimension represents the maximal statistical error. The *timeliness* evaluates the

temporal context of the data stream, e.g., x is sufficiently up-to-date for being involved in an aggregation function. There are two perceptions of timeliness: (i) on the one hand, timeliness expresses the age of x as the difference between the recording time instance and the current system time, (ii) on the other hand, timeliness is interpreted as the punctuality of x with respect to the application context. The latter perception presumes the definition of the subjective application and will not be regarded in this work. The *completeness* DQ dimension characterizes missing values in a stream, while the *volume* describes the amount of underlying contextual values. The completeness helps to distinguish between measured values x and imputed or missing ones $\hat{x}$. Completeness up to time t refers to the fraction of the number of the non-missing (or non-imputed) values observed by a stream up to t.

All these DQ dimensions interpret whether the contextual value x of stream s at time t is of a certain degree of quality $I(x)$ for feeding the CIP or not. We abstract DQ evaluation of x by classifying value x at time t as 'usable', notated by $I(x) = 1$, or 'unusable', notated by $I(x) = 0$. For instance, based on the above mentioned DQ dimensions, an unusable x can refer to (i) an expired value with respect to timeliness, or (ii) a missing or imputed value with respect to completeness, or (iii) anything that could assert that x is unusable for further processing, thus, degrading the 'quality' of the data stream and then the quality of the CIP result. A usable x is, for instance, an non-missing, up-to-date, highly reliable value thus usable for further processing. A holistic DQ evaluation function $I(x)$ involving all or some DQ metrics is beyond the scope of this paper. In this work, we abstract the implementation of $I(x)$ and assume that the mechanism is capable of evaluating $I(x)$ over a given stream. We focus on the completeness DQ dimension, that is at time t the mechanism determines immediately if x is missing or not. We represent the value of $I(x) \in \{0, 1\}$ at time t by a random variable (r.v.) indicating whether x is classified as usable or unusable. Let $\beta_i \in (0, 1)$ be the probability that x_i is usable at time t. We then define:

$$I(x_i) = \begin{cases} 1 & \text{if } x_i \text{ is usable w.p. } \beta_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

with $E[I(x_i)] = \beta_i$ and w.p. stands for 'with probability'.

Definition 2 (Vector Quality). *The vector quality Y_t of a stream vector $\mathbf{x}_t = [x_{1(t)}, \ldots, x_{n(t)}]$ at time t is defined as the quantity of the usable values of the n streams, i.e., $Y_t = \sum_{i=1}^{n} I(x_{i(t)})$.*

We assume that at time t, the $I(x_i)$ indicators are independently distributed Bernoulli random variables, each with probability of success β_i. Then $Y = \sum_{i=1}^{n} I(x_i)$ is a Poisson-Binomial random variable with parameters $(\beta_i; i = 1, \ldots, n)$. When all β_i are equal to $\beta > 0$, this reduces to the Binomial distribution with parameters (n, β). Moreover, when n is large and all β_i are not relatively high but not necessarily equal, the distribution of Y is well approx-

imated by a Poisson distribution[1]. For simplicity of analysis, we assume that $\beta_i = \beta, \forall i$, and that n is large, i.e., we assume a high number of context streams, e.g., multi-dimensional stream vectors are coming from a high number of WSN sources. The latter implies that the probability of unusable values $1 - \beta$ is not negligible, which holds true in our case. Moreover, Y may be binomial conceptually, but exact β_i values may be unknown and we may only know the rate on unusable values per time instance t, which can be easily calculated by historical observation of the data streams, as will be discussed in Section 5.3. In this case, the Y is a Poisson random variable with parameter $\lambda = n\beta$, i.e., the average number of usable values, with probability mass function:

$$P(Y = y) = \frac{e^{-\lambda}\lambda^y}{y!}, \lambda = n\beta, \tag{2}$$

with $F_Y(y) = P(Y \leq y) = e^{-\lambda}\sum_{k=0}^{y}\lambda^k/k! = 1 - P(Y > y)$ and the expected vector quality at time t is $E[Y_t] = n\beta$.

Remark 1. The assumption $\beta_i = \beta, \forall i$ does not spoil the theoretical results of our models and is adopted for eliminating the computations of $F_Y(y)$ involved in the optimal stopping criteria discussed later. Obviously, when $\beta_i \neq \beta_j, i, j = 1, \ldots, n$ then $F_Y(y)$ is provided in [11].

We have up to now defined the notion of vector quality over the current values from n streams. To proceed with the stream quality definition over a specific time horizon, we firstly refer to well-known data stream window constructs, i.e., landmark and sliding window, that contain stream vectors used by the CIPs. In both window constructs we assume that within the time interval $[t, t+1)$ we either observe one value x_i or not at all, $\forall i$.

Definition 3 (Landmark Window). *In a landmark window $\mathcal{W}(\tau, t)$, the lower time bound is fixed at a specific time instance $\tau \in \mathbb{T}$ ('temporal landmark') while the upper time bound follows the evolution of time $t > \tau$. Newly arriving stream vectors $\mathbf{x}_t$ are appended to the window without discarding existing ones.*

For instance, at time t with $\tau < t$, a landmark window $\mathcal{W}(\tau, t)$ is a sequence of all stream vectors observed from τ to t (including the vector at t), i.e., $\mathcal{W}(\tau, t) = (\mathbf{x}_\tau, \mathbf{x}_{\tau+1}, \ldots, \mathbf{x}_t)$; e.g., 'get all stream vectors collected after 10 pm'.

Definition 4 (Sliding Window). *A sliding window $\mathcal{W}(h)$ is specified by a fixed-size temporal extent $h > 0$ ('horizon') by appending new stream vectors and discarding older ones on the basis of their appearance.*

For instance, at time t, a sliding window $\mathcal{W}(h)$ is a sequence of all stream vectors observed from $t - h$ to t (including the vector at t), i.e., $\mathcal{W}(h) = (\mathbf{x}_{t-h}, \mathbf{x}_{t-h+1}, \ldots, \mathbf{x}_t)$; e.g., 'continuously return all stream vectors of the past hour, i.e., h=60 minutes'. The sliding window is the most widely used in continuous aggregation and fusion functions over streams [1], [31].

[1]It is also known as Poisson 'law of rare events'.

Definition 5 (Window Quality). *Given a landmark $\mathcal{W}(\tau, t)$ and sliding window $\mathcal{W}(h)$, the window quality Z_t is the cumulative sum of the vectors quality up to t, i.e., $Z_t = \sum_{k=\tau}^{t} Y_k$ and $Z_t = \sum_{k=t-h}^{t} Y_k$, respectively.*

Note, the window quality Z_t is a reference of the completeness DQ dimension provided that we restrict our interest in data streams that contains missing values.

3.2. Data Stream Aggregation and Fusion Operator

The aggregation and fusion are evaluated over the contents of a window. The aggregated/fused value changes over time as the window slides (in a sliding window) or new values append (in a landmark window). We use the classification from [14] that divides aggregation functions into three categories: distributive, algebraic, and holistic. Let $\mathcal{W}$, $\mathcal{W}_1$, and $\mathcal{W}_2$ be windows. An aggregation function f is distributive if $f(\mathcal{W}_1 \cup \mathcal{W}_2)$ can be computed from $f(\mathcal{W}_1)$ and $f(\mathcal{W}_1)$ for all $\mathcal{W}_1$, $\mathcal{W}_2$. An aggregation function f is algebraic if there exists a 'synopsis function' σ such that for all $\mathcal{W}$, $\mathcal{W}_1$, and $\mathcal{W}_2$: (1) $f(\mathcal{W})$ can be computed from $\sigma(\mathcal{W})$; (2) $\sigma(\mathcal{W})$ can be stored in constant memory; and (3) $\sigma(\mathcal{W}_1 \cup \mathcal{W}_2)$ can be computed from $\sigma(\mathcal{W}_1)$ and $\sigma(\mathcal{W}_2)$. An aggregation function is holistic if it is not algebraic. Among the standard aggregates, `MAX` and `MIN` are distributive, `AVG` is algebraic, since it can be computed from a synopsis containing `SUM` and `COUNT`, and `QUANTILE`, `MEDIAN` are holistic.

In our case, the aggregation operator $f_i(\mathcal{W})$ of the i-th stream over window $\mathcal{W}$ (landmark or sliding window) takes into account only the usable values x_i w.r.t. $I(x_i)$. For instance, given a landmark window $\mathcal{W}(\tau, t)$, we can simply define the `SUM`, `COUNT`, `AVG` and `MIN` operators $f_i(\mathcal{W}(\tau, t))$ for each stream s_i at time t, respectively, as follows: $f_i^{sum} = \sum_{k=\tau}^{t} x_{i(k)} I(x_{i(k)})$, $f_i^{count} = \sum_{k=\tau}^{t} I(x_{i(k)})$, $f_i^{avg} = f_i^{sum} / f_i^{count}$, $f_i^{min} = \min\{x_{i(k)} | I(x_{i(k)}) = 1, \tau \leq k \leq t\}$, respectively. Similarly, given a sliding window $\mathcal{W}(h)$, at time t we obtain: $f_i^{sum}(\mathcal{W}(h)) = \sum_{k=h-t}^{t} x_{i(k)} I(x_{i(k)}) / \sum_{k=t-h}^{t} I(x_{i(k)})$ and, $f_i^{min}(\mathcal{W}(h)) = \min\{x_{i(k)} | I(x_{i(k)}) = 1, t - h \leq k \leq t\}$, respectively. Such operators are built-in constructs in application specific continuous queries. For instance, the query 'every minute find the average temperature and the maximum humidity over context streams 'temperature' and 'humidity' collected during the past hour' in Continuous Query Language [5] involving `AVG` and `MAX` operators in a sliding window $\mathcal{W}(h), h = 60min$ can be expressed as follows: `SELECT AVG(temperature), MAX(humidity) FROM Context Streams [RANGE 60 MINUTES SLIDE 1 MINUTE]` Note, typical progressive aggregates like `SUM`, `MIN` and `AVG` requires constant time $O(1)$ per value since there is no need to scan the entire window.

A fusion operator over a subset or all context streams indicates whether a set of conjunctive predicates, e.g., aggregation operators over contextual values fall within pre-specified intervals, or instantly contextual values satisfy a criterion. A fusion operator typically involves more than one context streams with complex event pattern definitions for a specific time horizon. It is defined as the logical conjunction $f(\mathcal{W}) = \bigwedge_{i=1}^{n} (\phi_i(f_i(\mathcal{W}))) \in \{\texttt{TRUE}, \texttt{FALSE}\}$ of n logical operators $\phi_i(f)$ over aggregated (or not) values, e.g., $f \leq y$ or $f \in [y_{low}, y_{high}]$.

We can envisage f as an IF-THEN rule, for instance, when evaluating the situational context which refers to the event stream processing fusion operator for the past ten minutes it refers to the activation of the following rule with conjunctive predicates associated with `AVG` and `MAX` operators over context streams 'temperature' and 'wind-speed': `EVENT := IF AVG(temperature)` $\geq$ `90 AND MAX(wind-speed)` $\in$ `[10,20] WITHIN 10 minutes THEN ACTION is 'warning'`. We define the *aggregation or fusion error* e_i between the estimated result of the applied operator $\hat{f}_i$ over window $\mathcal{W}$ (either landmark or sliding window) of the i-th stream and the actual result f_i applied on $\mathcal{W}$ when all values are usable, i.e., the aggregation result error for stream s_i is: $e_i = |\hat{f}_i(\mathcal{W}) - f_i(\mathcal{W})|$. In the fusion case, e.g., dealing with `EVENT` identification, we define as fusion error from stream s_i as: $e_i = 0$ if $f_i = \hat{f}_i$; $e_i = 1$, otherwise. If $f_i =$ `FALSE` and $\hat{f}_i =$ `TRUE`, we obtain false alarm; the other combinations are similarly defined. The involvement of n context streams at time t, (i.e., when a continuous query engaging n operators over the context streams is executed over $\mathcal{W}$) yields the (total) CIP result error e, which is defined as the 1-norm of the error vector $\mathbf{e} = [e_1, \ldots, e_n]$, i.e., $e = \|\mathbf{e}\|_1 = \sum_{i=1}^{n} e_i$. Finally, the error-per-stream is the fraction e/n, where in the case of the `EVENT` operator, $e/n \in [0, 1]$ indicates the portion of those context streams s_i that correctly identify/evaluate the logical operator $\phi_i(f)$. Hereinafter, the terms: CIP, aggregation operator, and fusion operator are used interchangeably according to the context.

3.3. Optimal stopping rule problem

The theory of optimal stopping [28], [30] is concerned with the problem of choosing a time instance to take a certain action, in order to minimize an expected loss (or maximize an expected payoff). A stopping rule problem is associated with: (1) a sequence of random variables (r.v.) $Y_1, Y_2, \ldots$, whose joint distribution is assumed to be known and (2) a sequence of loss functions $(L_t(y_1, \ldots, y_t))_{1 \leq t}$ or payoff functions $(G_t(y_1, \ldots, y_t))_{1 \leq t}$ which depend only on the observed values $y_1, \ldots, y_t$ of corresponding r.v.s. An optimal stopping rule problem is described as follows: We are observing the sequence of the r.v.s $(Y_t)_{1 \leq t}$ and at each time instance t we choose either to stop observing or continue. If we stop observing at time instance t, we induce loss L_t or gain payoff G_t. We desire to choose a stopping rule or stopping time to minimize our expected loss, or equivalently, maximize our expected payoff.

Definition 6. *An optimal stopping rule problem is to find the optimal stopping time* t^* *which minimizes the expected loss* $E[L_{t^*}] = \inf_{0 \leq t \leq \mathcal{T}} E[L_t]$. *In the case of payoff,* t^* *maximizes the expected payoff, i.e.,* $E[G_{t^*}] = \sup_{0 \leq t \leq \mathcal{T}} E[G_t]$. *Note,* $\mathcal{T}$ *might be* ∞.

The available information up to t is a sequence $\mathcal{F}_t$ of values of the r.v.s $Y_1, \ldots, Y_t$ (a.k.a. filtration).

Definition 7. *The 1-stage look-ahead (1-sla) stopping rule/time is*

$$t^* = \inf\{t \geq 0 : L_t \leq E[L_{t+1}|\mathcal{F}_t]\} \text{ and } t^* = \inf\{t \geq 0 : G_t \geq E[G_{t+1}|\mathcal{F}_t]\} \quad (3)$$

In other words, t^* calls for stopping at the first time instance t for which the loss L_t (or payoff G_t) for stopping at t is (at most) as small as (or as high as) the expected loss (payoff) of continuing to the next time instance $t+1$ and then stopping.

Definition 8. *Let A_t denote the event $\{L_t \leq E[L_{t+1}|\mathcal{F}_t]\}$ (or $\{G_t \geq E[G_{t+1}|\mathcal{F}_t]\}$). The stopping rule problem is monotone if $A_0 \subset A_1 \subset \ldots$ almost surely (a.s.)*

A monotone stopping rule problem can be described as follows: The set A_t is the set on which the 1-sla rule calls for stopping at time instance t. The condition $A_t \subset A_{t+1}$ means that if the 1-sla rule calls for stopping at time t, then it will also call for stopping at time $t+1$ no matter what Y_{t+1} happens to be. Similarly, $A_t \subset A_{t+1} \subset A_{t+2} \subset \ldots$ means that if the 1-sla rule calls for stopping at time t, then it will call for stopping at all future times no matter what the future observations turn out to be.

Theorem 1. *The 1-sla rule is optimal for monotone stopping rule problems.*

Proof. See [28] □

4. Problem Statement

Consider a sliding window $\mathcal{W}(h)$ and assume that we require exactly h usable values to apply an operator f_i, e.g., `SUM`, `AVG`, `MIN`, or `MAX`, over a specific stream s_i within $\mathcal{W}(h)$, $i = 1, \ldots, n$. In the multidimensional case, i.e., dealing with stream vectors, we require nh usable values (h usable values for each stream) to apply operators f_i, within $\mathcal{W}(h)$. The operator f_i involves only the usable values x_i that are, on average, $E[\sum_{k=t-h+1}^{h} I(x_{i(k)})] = h\beta < h$. (It is the expected number of successes out of a number of h trials each one with success probability β.) The aggregation result $\hat{f}_i$ over $h\beta$ usable values deviates from the ideal one f_i, at which we have full information, i.e., exactly h usable values. The more usable values we have in $\mathcal{W}(h)$, the lower the error we get, with result error $e_i = |f_i(\mathcal{W}(h)) - \hat{f}_i(\mathcal{W}(h))|$. In the multidimensional case, the total CIP error is $e = \sum_{i=1}^{n} e_i$.

Evidently, the CIPs over $\mathcal{W}(h)$ would not guarantee zero error e that would be obtained if all values in $\mathcal{W}(h)$ were usable. That is because we on average obtain $nh\beta < nh$ usable values, as discussed above. We could prolong the horizon of the window to $h' = h/\beta$ in hopes of receiving more usable values. However, we would obtain nh usable values on average with variance $n\beta$, thus not exactly nh (or at least very close to nh). Moreover, in the sliding window new vectors are appending and older ones are discarded thus we cannot accumulate more than nh' usable values.

The fixed horizon of the sliding window h does not provide us with the flexibility to gather **more** usable values in hopes of minimizing the global CIP error e. Nonetheless, we can exploit the flexibility of the landmark window in which the upper bound can be extended until assembling a desired number of

usable values. This implies that, we **delay** the activation of the CIP operators f_i by continuously observing more stream vectors expecting to gather more usable values. Ideally, we want to apply CIP operators f_i over a window with as much usable values as possible, or exactly nh usable, if possible. (We assume here that all streams are equally important thus we desire h usable values for each stream.)

Consider a landmark window $\mathcal{W}(\tau, t)$. Instead of applying the operators f_i at time t, with $t - \tau = h$, we apply them at some **unknown** time $t^* > t$ over $\mathcal{W}(\tau, t^*)$ such that the number of usable values for each stream s_i be h or, counting for all streams, be nh. In this case, more vectors have to be appended to window $\mathcal{W}(\tau, t^*)$. This does not necessarily means that $t^* - \tau = h$ since at each time an usable value appears w.p. $\beta < 1$. Generally, we obtain that $t^* - \tau \geq h$ where the equality holds w.p. $\beta^{(t^* - \tau)}$. Nonetheless, it is unpredictable when we can gather nh usable values in a landmark window, i.e., when it is the time t^* of gathering nh usable values. The problem here is to find the time instance t^* at which we stop appending vectors $\mathbf{x}$ to $\mathcal{W}$ and the sum of the usable values be as close to nh as possible, i.e., the quality Z_{t^*} of the window be as close to nh as possible. In that case, the CIP error e is minimized and becomes zero when $Z_{t^*} = nh$.

Starting at a time landmark $\tau \in \mathbb{T}$, we apply the f_i operators over a landmark window $\mathcal{W}(\tau, t^*)$ at time $t^* > \tau$ such that t^* minimizes the difference $|Z_{t^*} - nh|$, i.e., we desire that $Z_{t^*} = \sum_{k=\tau}^{t^*} Y_k \simeq nh$. The time t^* refers to the optimal stopping time in our case, which minimizes the above difference and the value of $T = nh$ specifies the lowest bound of the total CIP error over the window. The difference $t^* - \tau$ refers to the (dynamic) size of the landmark window, which evidently cannot be determined a priori due to the stochastic nature of the window quality measure. The T value refers to **quality guarantee** and indicates the minimum acceptable quality level of the context streams. It is worth mentioning that by maximizing the window quality Z_t with respect to guarantee T implicitly indicates the minimization of the CIP error e. One could reform this (stochastic) optimization problem by finding the time t^* at which we minimize the cumulative total CIP error $\sum_{k=h-t+1}^{t} e_k$ given a sliding window $\mathcal{W}(h)$. This, however, is not applicable since the missing values are never revealed (by definition) thus we cannot actually estimate the CIP error. Finally, by using landmark window, the cumulative total CIP error will be continuously non-decreasing thus could never be minimized, since new vectors are appended and old ones are nor discarded.

We propose a time-optimized mechanism which determines the optimal time upper bound $t^* > \tau$ of a landmark window given a fixed landmark $\tau \geq 0$ such that the derived window quality Z_{t^*} is as close to T as possible. When t^* is determined then we apply the f_i operators involving the maximum possible number of usable values, thus, minimizing the CIP error. Then, the mechanism starts-off a new era with setting the time landmark $\tau = t^* + 1$. Evidently, the size of the landmark window is dynamic and its value is governed by the stochasticity of the appearance of the usable values in time. We proceed with the definition

of two problems that implicitly minimize the CIP error e by estimating the optimal stopping time t^*.

4.1. Quality Distance Minimization

The first problem refers to finding the optimal stopping time which minimizes the expected distance between window quality and quality guarantee. Let the landmark time $\tau = 0$. We apply the CIP operators at time $t > \tau$ when $Z_t = \sum_{k=0}^{t} Y_k$ is close to T. We do not know and cannot forecast when Z_t will reach T since $Z_1 = Y_1, Z_2 = Y_1 + Y_2, \ldots, Z_t = Y_1 + Y_2 + \cdots + Y_t$ is a Poisson process independent of T with mean $\lambda = n\beta$. We delay the observation process by appending stream vectors in hopes of accumulating usable values. If we gathered more usable values than T, i.e., $Z_{t^*} > T$, then we should have stopped before t^* since we performed redundant observations. We treat this problem as a stopping time rule problem by defining as loss L_t the absolute difference of Z_t from T:

$$L_t = |T - Z_t|, \text{ for } t = 0, 1, 2, \ldots \text{ and } L_\infty = \infty, \tag{4}$$

where Z_0 is defined to be zero and $L_0 = T$ represents the loss if we apply the CIP operators immediately, i.e., without any delay (no extra observation taken).

Problem 1. *Given a quality guarantee $T > 0$, find the optimal stopping time t^* such that $\inf_{1 \le t \le \infty} E[L_t]$ is attained. The minimum expected loss is $E[L_{t^*}]$.*

4.2. Cost delay-aware and Time-decaying aware Quality Maximization

In Problem 1, our target is to reach T as close as possible. In the case that $Z_t > T$, then immediately at that time t we stop the observation and activate the CIP operators, since we have accumulated much more usable values than required. Obviously, the latter case is not undesirable but it induces a 'penalty' that we should have stopped earlier before getting $Z_t > T$. This penalty is quantified by $Z_t - T$. However, we can incorporate a specific cost due to additional delay for observing one more stream vector until we stop. Each time t we do not stop, thus, not activating CIP we induce a certain cost $c > 0$. This cumulative cost up to t enforces the mechanism to take a decision on whether to stop or continue with another observation by balancing between the trade-off: prolong the observation for possible more usable values and additional delay to activate the CIP operators. Such cost could represent the urgency of a monitoring application that requires near real-time CIPs. On the other hand, if the application is in need of highly accurate CIP results, a certain (controlled) delay must be tolerated.

Moreover, we consider the case where the stream vectors in $\mathcal{W}$ are subject to the timeliness DQ dimension. Each time we continue the observation by appending a new stream vector to $\mathcal{W}$, the least recent stream vectors turn gradually obsolete to a certain degree. Obviously, we cannot delay for ever to obtain the highest window quality, since the buffered stream vectors have to represent the current (or better near past) state of nature. An aging factor

$a \in (0, 1]$ over all buffered stream vectors enforces the mechanism to take into account both: delay for gathering possibly more usable values and timeliness of the buffered stream vectors to avoid involving almost obsolete values. Based on the above interpretation, we provide the following payoff function $G_t(Z_t)$, which involves (i) the quality guarantee T, (ii) the delay cost per observation $c \geq 0$, and (iii) the aging factor $0 < a \leq 1$:

$$G_t(Z_t) = \begin{cases} a^t Z_t - ct & \text{if } Z_t \leq T, \\ 0 & \text{if } Z_t > T. \end{cases} \tag{5}$$

Our target is to reach T by maximizing the expected window quality. If Z_t is over T then the return is zero, since the mechanism should have activated CIP, thus, any extra delay occurred and aging discount applied was of no avail.

Problem 2. *Given a quality guarantee T, delay cost $c \geq 0$, and aging factor $a \in (0, 1]$, find the optimal stopping time t^* such that $\sup_{1 \leq t \leq \infty} E[G_t]$ is attained. The maximum expected return is $E[G_{t^*}]$.*

5. Quality-aware Mechanisms

5.1. Distance Quality Model

We provide a solution to the Problem 1 through the Distance Quality Model (DQM). Based on Theorem 1 we first provide an 1-sla rule for the Problem 1 and then prove that this rule is optimal with respect to the loss function L_t defined in (4). In Appendix A, we also provide certain insights (indigenous characteristics) of a family of quality loss functions applicable to Problem 1.

Let the landmark time $\tau = 0$ and window $\mathcal{W}(\tau, t)$, $\tau < t$. On the event $\{Z_t < T\}$, the loss L_t is simply $L_t = T - Z_t$. On the event $\{Z_t \geq T\}$, it is definitely optimal for the DQM to stop the observation of the context streams and activate the CIP. Now consider that $Z_t < T$ and let us decide whether to continue observing the context streams at the next time instance $t + 1$, i.e., receiving one more context vector $\mathbf{x}$, and not stopping at t. Given that $\{Z_t < T\}$ we calculate the expected loss at the next time $t + 1$: $E[L_{t+1}|\mathcal{F}_t] = E[|T - Z_t - Y||\{Z_t < T\}]$, where $Z_{t+1} = Z_t + Y$ and Y is the Poisson variable with parameter (mean) λ defined in (2). To apply the 1-sla rule (5), we have to assert that it is optimal to stop at the first time instance at which it holds true that $L_t = T - Z_t \leq E[L_{t+1}|\{Z_t < T\}]$. This implies that the difference $E[L_{t+1}|\{Z_t < T\}] - T + Z_t$ is a.s. monotonically non-decreasing with Z_t with $Z_t < T$; recall that when $Z_t > T$ then we stop immediately.

Theorem 2. *Given a sequence of window quality r.v.s $Z_1, \ldots, Z_t$, the optimal stopping rule t^* for Problem 1 is*

$$t^* = \inf\{t \geq 0 : \sum_{y=0}^{T-Z_t} yP(Y = y) + (T - Z_t)(1 - F_Y(T - Z_t)) + \frac{\lambda}{2} \geq 0\}. \tag{6}$$

PROOF. For any $\ell > 0$ we have

$$\begin{aligned} E[|\ell - Y|] &= E[\ell - Y] + 2E[\max(0, Y - \ell)] = \ell - \lambda + 2P(Y > \ell)E[Y - \ell | Y > \ell] \\ &= \ell - \lambda + 2E[Y - \ell, Y > \ell] = \ell - \lambda + 2 \sum_{y=\ell+1}^{\infty} (y - \ell)P(Y = y) \end{aligned}$$

Hence, given the event $\{Z_t < T\}$, we obtain with $\ell = T - Z_t$ that $E[|T - Z_t - Y|] = T - Z_t - \lambda + 2\sum_{y=T-Z_t+1}^{\infty}(y - T + Z_t)P(Y = y)$. Therefore, by comparing with $T - Z_t$ we obtain that the DQM stops at the first time instance t at which $T - Z_t \leq E[|T - Z_t - Y|]$ or $\sum_{y=T-Z_t+1}^{\infty}(y - T + Z_t)P(Y = y) \geq \lambda/2$. Evidently, the problem is monotone since $\sum_{y=\ell+1}^{\infty}(y - \ell)P(Y = y)$ with $\ell = T - Z_t$ is monotonically non-decreasing with Z_t for $Z_t < T$, given that Z_t is a.s. increasing, by definition. □

When the window quality Z_t at time t is such that the stopping criterion in (6) becomes non-negative then the DQM immediately stops appending stream vectors and activate the CIP with the stream vectors in the landmark window $\mathcal{W}(\tau, t)$. Due to the 1-sla rule and the monotone nature of the Problem 1, no other stopping rule can guarantee us as much, i.e., to minimize the expected 'distance' of the window quality from quality guarantee T. Note that with a low λ value, the mechanism delays the CIP invocation since, with small incremental steps (i.e., relatively small values of Y) it attempts to reach T. On the other hand, a high λ value yields the DQM to quit at an early stage of the observation process (thus low delay), since large incremental steps would possibly results to Z_t that significantly overpasses T.

Remark 2. The stopping criterion evaluation should be as quick as possible with trivial complexity thus avoiding time-consuming decision making whether to activate the CIP or not. From (6), the stopping criterion depends on the calculation of a summation from 0 to $T - Z_t$ at time instance t. Evidently, we can recursively evaluate this sum at time t by simply use the sum up to time $t - 1$ plus a loop of length $E[|Z_t - Z_{t+1} + 1|]$. Hence, the time complexity of evaluating the sum at t is $O(\lambda)$.

5.2. Delay-aware and Time-decaying Quality Model

We provide a solution to Problem 2 by defining the delay-aware and time-decaying quality model (DAQM). Consider again that landmark time $\tau = 0$ and window $\mathcal{W}(\tau, t)$, with $\tau < t$. The DAQM attempts to stop the observation (and appending) of context vectors $\mathbf{x}$ at which the window quality Z_t is close to T but also takes into consideration the cumulative delay cost up to that time and the aging factor over the buffered stream vectors. We report on an 1-sla stopping rule based on the principle of optimal stopping in Theorem 1, at which we stop at the first time t such that $G_t(Z_t) \geq E[G_{t+1}(Z_{t+1})|\mathcal{F}_t]$, with the event $\{Z_t \leq T\} \in \mathcal{F}_t$. That is any additional observation at time $t + 1$ would not additionally contribute to the payoff maximization. The 1-sla rule is optimal

when the difference $E[G_{t+1}(Z_{t+1})|\mathcal{F}_t] - G_t(Z_t)$ is monotonically non-increasing with Z_t.

Theorem 3. *Given a sequence of window quality r.v.s $Z_1, \ldots, Z_t$, the optimal stopping rule t^* for Problem 2 is*

$$t^* = \inf\{t \geq 0 : \sum_{y=0}^{T-Z_t} (a^{t+1}(Z_t + y) - c(t+1))P(Y = y) \leq a^t Z_t - ct\}. \quad (7)$$

PROOF. Given that $Z_t \leq T$, the conditional expectation $E[G_{t+1}(Z_{t+1})|Z_t \leq T]$ is given by

$$\begin{aligned} E[G_{t+1}(Z_{t+1})|Z_t \leq T] &= E_Y[G_{t+1}(Z_t + Y)|Z_t \leq T, Z_t + Y \leq T]P(Z_t + Y \leq T) \\ &\quad + E_Y[G_{t+1}(Z_t + Y)|Z_t \leq T, Z_t + Y > T]P(Z_t + Y > T) \\ &= E_Y[a^{t+1}(Z_t + Y) - c(t+1)|Y \leq T - Z_t]P(Y \leq T - Z_t) \\ &= \sum_{y=0}^{T-Z_t} (a^{t+1}(Z_t + y) - c(t+1))P(Y = y) \end{aligned}$$

Hence, the mechanism stops at the first time instance t with Z_t such that $E[G_{t+1}(Z_{t+1})|Z_t \leq T] \leq a^t Z_t - ct$. The corresponding difference is monotonically non-increasing with Z_t with $Z_t < T$, i.e., the 1-sla rule is optimal. □

Since the 1-sla rule in Theorem 1 is optimal, then the DAQM with fixed T and taking into consideration the delay cost c and the aging factor a can guarantee that the expected window quality based on the stopping criterion is as much close to T as possible and no other stopping rule can guarantee us much. Based on the c and a parameters, the DAQM is flexible to treat and control the expected delay and the freshness of the contextual values involved in the CIP. We define the variants of DAQM w.r.t. the cost and aging parameters.

- Case $c = 0, a \in (0, 1)$. Here, mechanism does not consider any delay cost, thus, it is only enforced to stop the observation once the timeliness of the buffered stream vectors is discounted. In this case, the stopping criterion is given by $t^* = \inf\{t \geq 0 : \sum_{y=0}^{T-Z_t} yP(Y = y) \leq Z_t(\frac{1}{a} - F_Y(T - Z_t))\}$.
- Case $c > 0, a = 1$. Here, DAQM cares about the delay cost of observation while assuming that all buffered stream vectors are fresh for being applied to the CIP. The DAQM here is enforced to stop the observation to avoid waiting for a long time, especially when λ is relatively small, thus it is required a high number of usable values to sum up to T.
- Case $c = 0, a = 1$. Here, we require that the window quality be as much close to T as possible but not greater than T[2]. In this variant, the stopping

[2]This variant reduces to the discrete case of the continuous time *black jack* OST problem in [15] where the sum of values must be close to T, otherwise the payoff is zero

criterion is: $t^* = \inf\{\sum_{y=0}^{T=Z_t} yP(Y=y) \leq Z_t(1-F_Y(T-Z_t))\}$. Note that this variant is different with the DQM, where in the DQM if Z_t overpasses T then the penalty is the difference $Z_t - T$ and we stop immediately.

Remark 3. DAQM requires $O(\lambda)$ time to evaluate the stopping criterion (7).

5.3. The Algorithm of the DQM and DAQM

The algorithm for both models the DQM and DAQM for an era is shown in Algorithm 1. The input of the algorithm is the probability of an usable value β, the quality guarantee $T = nh$ and, in the case of the DAQM, the input is the delay cost per observation c, $0 \leq c \leq \lambda$ as a portion of λ to quantify the penalty of the delay as the mean value of the vector quality, and aging factor $a \in (0, 1]$.

The probability β can be incrementally estimated by the maximum likelihood estimation of β of the Poisson distribution with mean $\lambda = n\beta$ after observing a series of t instances of $(Y_\tau)_{\tau=1}^t$, $t > 1$. The log-likelihood $\mathcal{H}_t(\beta)$ of a series of t samples of $Y_1, \ldots, Y_t$ with the probability distribution in (2) is $\mathcal{H}_t(\beta) = \sum_{\tau=1}^t [Y_\tau \log n\beta - n\beta - \log Y_\tau!] = \log n\beta \cdot \sum_{\tau=1}^t Y_\tau - tn\beta - \sum_{\tau=1}^t \log Y_\tau!$. Since $\mathcal{H}_t(\beta)$ is a continuous function of β given t observations, i.e., $\beta = \beta_t$, its maximum value derives from the derivative of $\mathcal{H}_t(\beta)$ with respect to β_t by setting it equal to zero. After this calculation, we obtain that up to the t-th observation, the probability β_t is: $\beta_t = \frac{1}{nt}\sum_{\tau=1}^t Y_\tau$. Hence, we can incrementally estimate the β_t value by the previous β_{t-1} and the current value of Y_t by using the recursion $\beta_t = \frac{t-1}{t}\beta_{t-1} + \frac{1}{nt}Y_t$, with $\beta_1 = \frac{1}{n}Y_1$. After a series of t observations, we can estimate the $\beta = \beta_t$ and then initiate our mechanisms.

ALGORITHM 1: The algorithm of the quality-aware mechanism.

Input: Probability of usable value β, quality guarantee T; observation cost $c \in [0, \lambda]$, aging factor $a \in (0, 1]$.
Output: Optimal stopping time t^*, i.e., size of the landmark window.
/* the algorithm runs for an era starting at temporal landmark $\tau \geq 0$ */ ;
STOP $\leftarrow$ FALSE; $t \leftarrow 0$; $Z_0 \leftarrow 0$; $\lambda \leftarrow n\beta$;
repeat
  $Y_t \leftarrow 0$ /* initialize the vector quality */ ;
  observe vector $\mathbf{x}$;
  for *each context stream* $s_i \in \mathcal{S}$ **do**
    $Y_t \leftarrow Y_t + I(X_i)$;
  end
  $Z_t \leftarrow Z_{t-1} + Y_t$ /* update the window quality */ ;
  if *criterion in (6) (or in (7)) is satisfied* **then**
    STOP $\leftarrow$ TRUE;
    $t^* \leftarrow t$ /* activate the CIP and start-off a new era*/ ;
  else
    $t \leftarrow t + 1$ /* continue with the next context vector*/ ;
  end
until STOP=TRUE;

6. Performance Evaluation

6.1. Methodology

We report on the performance of the models DQM and DAQM over real-data streams with respect to a sensitivity analysis of the basic models' parameters and examine the corresponding (i) CIP results error, (ii) expected delay $E[t^*]$, i.e., expected size of the landmark window, (iii) scalability in terms of number of context streams n and coefficient h of quality guarantee T. We also compare the performance of both models with the baseline solution of the Sliding Window Model (SWM) with window size h referring to the quality guarantee T used for the DQM and DAQM. We chose SWM for performance comparison with our proposed models since it is widely adopted by numerous data stream management systems e.g., [1] and [31]. The idea of the comparative assessment is to demonstrate how a time-optimized delay of the CIP activation, as achieved by our models, could come along with benefits in minimizing the CIP error and saving computational resources of a back-end system by invoking the CIP when appropriate usable information is accumulated.

We use the real data-streams $\mathcal{D}$ from the GreenOrbs [25] application for forest surveillance and forestry observation. There are $n = 450$ TelosB sensor nodes (context streams) scattered on the Tianmu Mountain, China and capture context parameters: temperature, light, humidity, illumination, and carbon dioxide titer, once every half minute with 20000 sensing intervals for over one year. In the experiments, the usable value probability β ranges from 0.1 to 0.9. If $\beta = 0$ then all data are lost, thus, no method can work. If $\beta = 1.0$ the context streams are complete, thus, it is unnecessary to pay any attention. In order to measure the CIP result error, based on the dataset $\mathcal{D}$, we create a corresponding incomplete dataset $\mathcal{D}'$ where at each time instance t, a value x_i of the context vector $\mathbf{x} \in \mathcal{D}$ is missing with probability $1 - \beta$. Hence, on average we obtain $n(1-\beta)$ missing entries in each incoming incomplete context vector $\hat{\mathbf{x}} \in \mathcal{D}'$. For experimentation of CIP operators over the n context streams, we apply representative CIPs from each class of CIPs, i.e., `MAX` for standard aggregates, `AVG` for algebraic aggregates, and `EVENT` for fusion operator involving the aggregates: `MIN`, `MAX`, and `AVG` in its definition. The DQM and DAQM initiate at $\tau = 0$ with a landmark window, where they stop the observation process at stopping time t^*. Then, immediately, they apply the CIP operators over the corresponding window $\mathcal{W}(\tau, t^*)$. We then measure the CIP result error e_i for each context stream given the complete and incomplete context vectors, the 1-norm CIP error vector $e = \|\mathbf{e}\|_1 = \sum_{i=1}^{n} e_i$ and the error-per-stream e/n. Note, in the `EVENT` case, the $e/n \in [0, 1]$ denotes the number of context streams that provide opposite logical predicates ϕ_i with the actual ones in the IF-THEN definition rule of an event. After the stopping at t^*, both models initiate a new 'era' through a new landmark window acting with the same way. The expected delay after a large number of eras is $E[t^*]$. The SWM operates on a sliding window of size h corresponding to the same guarantee threshold T. We measure at each time instance t, the 1-norm error vector e and error-per-stream e/n applying the same aggregate/fusion operators as in the DQM and DAQM. In the SWM there is not

Parameter	Description	Default value/range
n	number of context streams	$\{10, 100, 200, 300, 450\}$
β	probability of usable value	$\{0.1, \ldots, 0.9\}$
λ	mean value of vector quality	$n\beta$
h	sliding window size; coeff. of quality guarantee	$[10, 1000]$
T	quality guarantee	nh
a	aging factor	$(0, 1]$
c	delay cost per observation	$[0, \lambda]$

Table 1: Notations and default range/values of parameters.

delay. We also define the percentage decrease of error $\nu = \frac{e_{SWM} - e_x}{e_{SWM}}\%$ between the error of SWM, e_{SWM}, and of a model $x \in \{DQM, DAQM\}$. A positive $\nu\%$ value indicates that a model x achieves lower CIP error than SWM.

We set the aging factor $a \in (0, 1]$ and the delay cost per observation $c = [0, \lambda]$ to quantify the maximum penalty of an extra delay, with respect to the vector quality Y, as a portion of its mean value λ. The application quality guarantee threshold is set to $T = nh$, for diverse values of h. Note that, a high h value implies (i) high storage capacity, (ii) the applied CIP operators could refer to a high portion of obsolete values (especially when h is large), and (iii) the dynamics (e.g., concept drifts, peaks, sudden changes) of the context streams are more or less 'flattened'. Hence, shorter window length would be more applicable to catch the current dynamics of the context streams and to deal with freshness of the CIP results. For that reasons we experimented with $h = 10$, which refers to a time horizon of five minutes. However, in the comparative assessment of the SWM, DQM and DAQM we study the impact of higher h values on the derived CIP result error. The experimental parameters and their default values are shown in Table 1.

6.2. Experimental Evaluation

Figures 1(left), and 2(left) shows the error-per-stream of the DQM and the SWM with respect to probability β for `AVG` and `EVENT` operators, respectively, for $n \in \{10, 100\}$ and sliding window size $h = 10$ for the SWM. We observe the applicability of the DQM especially in the case of high degree of loss (i.e., $\beta < 0.5$) in context streams, which is not rare in real-life environmental monitoring applications [20]. Specifically, with a low β value denoting high portion of observed unusable values, we obtain a very high error with SWM compared with the DQM. This error percentage difference is quantified by $\nu\%$ in Figures 1(right), and 2(right), respectively. The DQM is robust in all β values since it attempts to optimally delay the process in hope of receiving more usable values thus decreasing the CIP error. Evidently, as $\beta \to 1$ then less unusable values are observed, thus both models DQM and SWM assume the same error. But even in this case, the DQM on average achieves to 10%-15% lower error than SWM. The benefit gained from our time-optimized mechanism is clear compared to SWM, which is also robust in terms of the number of context

streams n. The more context streams we have the higher the error we obtain with SWM. On the other hand, the DQM attempts to optimally reach the quality guarantee given all context streams simultaneously each one with equal importance. However, as we will show later, our models (DQM and DAQM) are highly scalable (linearly increase) in terms of accuracy with an increase number of streams compared with the DQM, which the latter does not scale well (the corresponding error exponentially increases with the number of context streams); see Figure 6 (right).

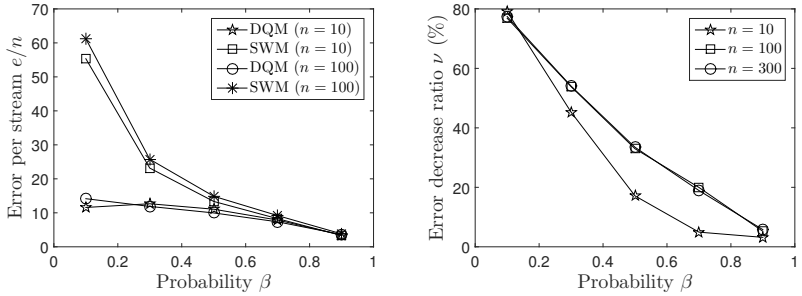

Figure 1: (Left) Error per stream e/n for `AVG` operator and (right) percentage ν% of the DQM and the SWM against probability β.

Note that in Figure 2 (left) the error-per-stream in the `EVENT` operator indicates the portion of the context streams that produce false alarms out of the n streams when evaluating an event. In this case the accuracy of the fusion result plays a significant role in certain event-based monitoring applications, especially when evaluating events with high confidence [3]. That is because the error refers not only to estimation errors of the aggregate operators in the involved predicates but also on the evaluation of the situational context itself, which is of prime importance for context-aware applications. In the DQM, 27% (on average) of the context streams falsely evaluate an event for all β values. In SWM around 80% of the context streams falsely evaluate events for low β values and at least 40% of streams proceed with erroneous event inference for $\beta \sim 0.5$. The DQM is more precise in event detection than SWM as shown in Figure 2 (right). In that case, if a certain delay on proceeding with an event inference/detection is tolerated then one could obtain more precise fusion results or at least obtain usable contextual information for further processing. Nonetheless, we have to study the trade-off of achieving low result error and robust behavior of the DQM in highly incomplete context streams with the incurred delay.

Figure 3 shows the the expected delay $E[t^*]$ of the DQM, i.e., the average window size of the landmark window used for achieving low error given the quality guarantee T; over `MAX,AVG,EVENT` operator. Obviously, in the case of $\beta \leq 0.1$ the delay is high corresponding to a landmark window with a size of ten times that of the sliding window size h for the SWM. This is attributed to the fact that, the DQM delays the observation process in observing usable pieces of context for minimizing the quality distance to the threshold T. In this case

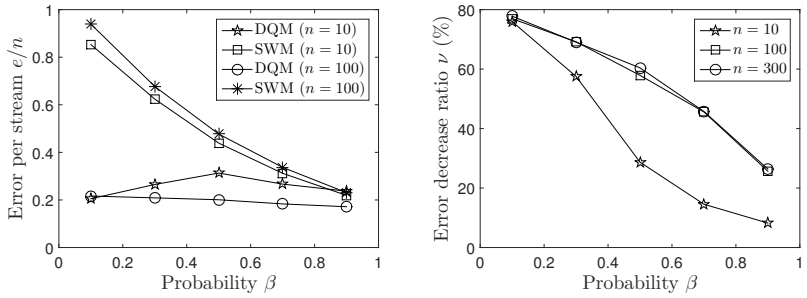

Figure 2: (Left) Error per stream e/n for `EVENT` operator and (right) percentage ν% of the DQM and SWM against probability β.

we obtain on average 80% less error than SWM for all operators. Nevertheless, with a relatively high incomplete data streams, i.e., $\beta = 0.3$, the DQM requires much less delay, specifically, only a factor of 2.03 of window size h for achieving 55% less error w.r.t. SWM. This implies the efficiency of the proposed model and its adaptability to the underlying data quality indicators in order to autonomously decide when to stop the observation process for achieving its goal under uncertainty. Obviously, when $\beta \to 1$ then the delay obtained by the DQM is close to the SWM case, i.e., the average (expected) landmark window length is of the same length h as the sliding window size (but, we need to examine also the variance of t^* as will be discussed later). This is due to the fact that the underlying context streams do not contain many missing values thus any extra delay is of no avail. This also depicts the adaptability of the proposed model even when the underlying data streams are not so incomplete. The capability of the DQM to efficiently and optimally adapt to the degree of incompleteness of the underlying streams balancing between low error and relatively low expected delay render it as an appropriate model for stream-based context aware applications. Moreover, it is worth mentioning that the expected delay of the DQM is independent of the aggregation and fusion operators rendering thus our model applicable to certain CIP applications over context streams.

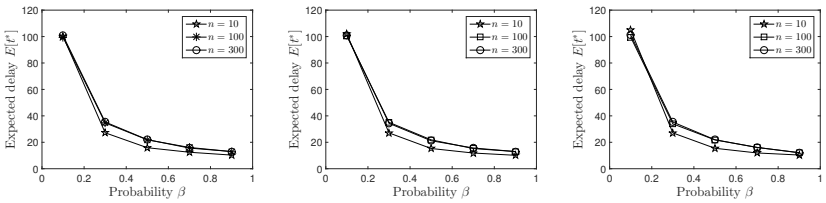

Figure 3: Expected delay $E[t^*]$ for (left) `MAX`, (middle) `AVG`, and (right) `EVENT` operator of the DQM against probability β.

Figure 4 shows the error-per-stream and the percentage decrease error ν% of the DAQM with respect to SWM having aging factor $a \in \{0.8, 0.9, 0.99\}$ and

delay cost $c = 0$ for `AVG` (similar results are observed for `MAX,EVENT`) and $n \in \{10, 200\}$. In this case, the DAQM attempts to reach the quality guarantee T taking into consideration the aging factor of the pieces of values in the landmark window. We observe that the DAQM behaves very efficient in cases where β is low and achieves a high relative difference with the quality of the aggregation results compared to SWM.

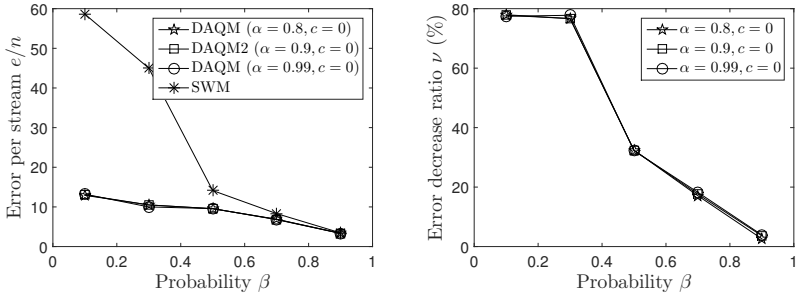

Figure 4: (Left) Error per stream e/n for `AVG` operator and (right) percentage ν% of the DAQM and the SWM against probability β; $n = 200, c = 0$.

Notably, for $c = 0$ the aging factor a does not play so significant role in expected delay and error. In case where we desire to control the expected delay and the achieved accuracy, the combination of c and a provide us this flexibility. The higher the cost c along with low a value, the more conservative the DAQM becomes in terms of stopping the observation process at an earlier stage of an era. This denotes that the expected size of the landmark window size, expressed by $E[t^*]$ is shorter however at the expense of higher error compared to the DQM, but also very lower than that of the SWM. Evidently, when $c = 1 \cdot \lambda$ then the delay cost is of utmost importance thus the DAQM obtains a high error, here in the case of low n, it achieves almost the same accuracy as SWM with $E[t^*] \sim h$. However, in case where n is relatively large, even with high cost c the DAQM achieves lower error than the SWM denoting its applicability with a high number of context streams.

To further demonstrate the flexibility of DAQM to trade-off accuracy with expected delay, we examine the impact of a factor of DAQM on the error and the expected delay. Figure 5 shows the error and expected delay for DAQM with $c = 0.5\lambda$, the SWM and DQM against a range of a values (the SWM and DQM are shown for comparison reasons, since they do not depend on a). DAQM provides us with the flexibility to achieve low delay compare with the DQM but at the expense of high error and on the other hand to achieve very low error with a relatively high delay. Note that, the DAQM through a can reach the accuracy results of the DQM, i.e., with high a values and also achieves quite the same delay of the DQM. Moreover, DAQM achieves lower error than SWM for $a > 0.6$. A low a value results to a high aging factor, denoting that the values in the landmark window turn obsolete with a high rate. Hence, in this case, the DAQM stops the observation process in a very short time with expected delay

$E[t^*]$ lower than h, i.e., lower than that of SWM. An a value higher than 0.6 results to lower error than SWM and higher than the DQM. Nonetheless, in this case DAQM achieves lower delay than the DQM. Hence, DAQM through $a \in [0.6, 1)$ can balance the trade-off accuracy and acceptable delay. In the extreme values, i.e., $a \to 1$ DAQM behaves the same as the DQM in terms of accuracy and delay. Indicatively, to achieve 50% and 80% lower error than SWM, we require and expected delay $E[t^*]$ of two and four times the sliding window size h for DAQM and the DQM, respectively.

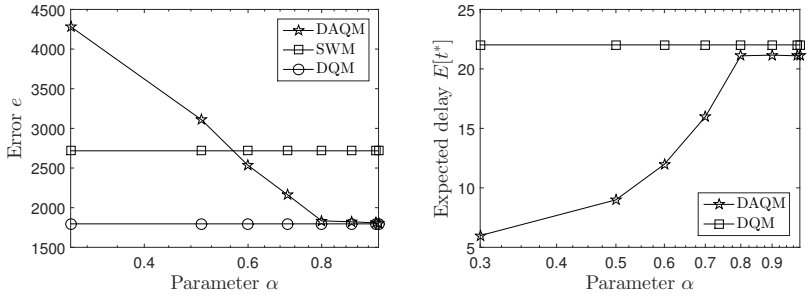

Figure 5: (Left) Error e and (right) expected delay $E[t^*]$ against a of the DAQM and the DQM for `AVG` operator with $\beta = 0.5$; $c = 0.5\lambda$.

To better illustrate this trade-off, Figure 6 (left) shows the expected delay against the error-per-stream for the `EVENT` for all models with $\beta \in \{0.1, 0.3, 0.5\}$ and different values of a. The DQM lies on the left side of the 'box' determined by the expected delay and error-per-stream with the least error and the highest delay. DAQM ranges from the left side of the box to the bottom right according to the a factor. The SWM results to the highest error without delay. DAQM is more applicable than the DQM in terms of a high range of applications. That is because, in DAQM one can tune (i) the cost c parameter, which expedites the application of the aggregation/fusion operators over the window thus adapting the 'speed' of obtaining results, e.g., in time critical applications with a restricted tolerance, (ii) one can tune the aging factor a, which refers to a 'degree of freshness' / 'timeliness' of the contextual values being involved in the aggregation/fusion operation, or (iii) a hybrid scheme of supporting both speediness and freshness of results being as much accurate as possible.

We also experiment with the impact of the number of context streams n on the derived CIP error. Note: in order to obtain a high number of context streams $n = 20,000$ (higher than 450 from the dataset $\mathcal{D}$) for experimenting with the scalability limit of the models, we replicate each context stream s_i many times (approx. 40 times) and generate its (time series) values x_i drawn from a Gaussian and uniform distribution (with equal probability each) using its corresponding mean and variance. Figure 6 (right) shows the impact of n on error e for the SWM, DQM, and DAQM for the `EVENT` operator with $\beta = 0.3$. We observe that SWM does not scale well with a high number streams, thus being inappropriate when dealing with parallel processing of data streams. Our

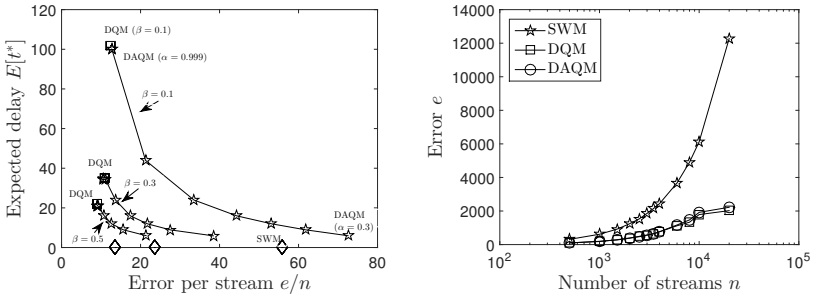

Figure 6: (Left) Expected delay $E[t^*]$ vs. error-per-stream e/n for $\beta \in \{0.1, 0.3, 0.5\}$ of DAQM (star $\star$) with $a = 0.999$ (top left) towards $a = 0.3$ (bottom right); DQM (square $\square$) with $\beta = 0.1$ (top left) towards $\beta = 0.5$ (bottom left); SWM (diamond $\diamond$) with $\beta = 0.5$ (bottom left) towards $\beta = 0.1$ (bottom right); $n = 200$. (Right) Scalability of error e for `EVENT` operator with $\beta = 0.3$ of the DQM, DAQM ($a = 0.95, c = 0.5\lambda$), and the SWM.

both models scale efficiently with the number of streams in terms of accuracy. This is because, the 'dimension' of n is orthogonal to the time optimization process already included in the quality guarantee threshold T. This renders both models to deal with stochastic optimization of the multivariate space (i.e., multivariate context vector $\mathbf{x}$ of n components) ahead of time minimizing the quality distance. The decision is just scaled with the number n and does not significantly influence the stopping criterion. Note that, for both models DQM and DAQM the number of streams n does not influence the expected delay as discussed above thus render them scalable and robust when dealing with

7. Comparative Assessment

In this section we report on the comparison models: the Incremental Mean Model (IMM) [27], the Skewness Sensitive Model (SSM) [27], the the Average Nearest Exponential Smoothing Model (ANESM) [16], and Exponential Moving Average Model (EMAM) [10] adopted for applying a CIP over a window of contextual streams. We provide a comprehensive comparative assessment with our model in terms of a series of performance metrics.

7.1. Comparison Models

Incremental Mean Model (IMM). The IMM [27] is applied independently to each context stream s_i given a sliding window $\mathcal{W}(h)$. The idea is to deliver a series of contextual values that are replaced by plausible estimates (a.k.a. *imputation*) in case they are missing. That is, **each time** the observed value $x_{i,t}$ from a context stream s_i is missing, then the IMM imputes this value. Then, the IMM applies the CIP operator over a sliding window $\mathcal{W}(h)$. This model does not take into consideration any quality guarantee. Instead, it invokes the CIP operator over the window $\mathcal{W}(h)$ and proceeds with a missing value imputation method in case $I(x_{i,t}) = 0$ at time instance t.

Assume that at time t, the $x_{i,t}$ value is missing. The IMM relies on the *global* incremental mean value $\mu_{i,t-1}$ over the stream s_i up to time $t-1$ in order to impute the missing value $x_{i,t}$. The $\mu_{i,t}$ value up to time t for the stream s_i is incrementally calculated by the $\mu_{i,t-1}$ value up to time $t-1$ and the current value $x_{i,t}$. The adopted recursion for the estimation of the mean value is: $\mu_{i,t} = \mu_{i,t-1} + \frac{1}{t}(x_{i,t} - \mu_{i,t-1})$. The interested reader could refer to Appendix B for the generation of this recursion. IMM extrapolates the missing $x_{i,t}$ with the $\mu_{i,t-1}$, which is calculated up to $t-1$. After this substitution, then the IMM applies the CIP operator over the $\mathcal{W}(h)$ with the recent h actual and imputed values for each stream s_i. Note here that, the IMM involves in the CIP operator the values within a sliding window where a portion of them (statistically, they are βh in number) refer to imputed values with the corresponding incremental mean value. On average, for the n streams, the CIP operator involves $n\beta h$ imputed values given a sliding window $\mathcal{W}(h)$. Obviously, for a given $\beta > 0$ the number of actual values $n(1-\beta)h$ is less than the quality guarantee threshold $T = nh$. Our models, instead, ensure to deliver at least T actual values.

Skewness Sensitive Model (SSM). The SSM [27] exploits the incremental median and incremental mean value in order to impute the missing values before proceeding with a CIP operator over a sliding window $\mathcal{W}(h)$. The SSM takes into consideration the skewness of each stream s_i to obtain an insight on the degree of asymmetry of the underlying probability distribution of s_i. The SSM at **each time** instance t incrementally updates the mean value $\mu_{i,t}$ and the median value $m_{i,t}$. When a missing value $x_{i,t}$ is observed at time instance t, i.e., $I(x_{i,t}) = 0$, then the SSM compares the incremental $\mu_{i,t-1}$ with $m_{i,t-1}$ up to the time $t-1$. Specifically, let us define the indicator function $\mathcal{J}(x_i; h)$ which is 1 if x_i is less than the average (mean) value of all values within a sliding window $\mathcal{W}(h)$. Otherwise, $\mathcal{J}(x_i; h) = 0$. Based on this indicator function, two cases are distinguished in the SSM: [Case 1]: If $\mu_{i,t-1} > m_{i,t-1}$ then the underlying distribution of stream s_i is *right* skewed, also referred to as positive skewness. In that case, the missing value $x_{i,t}$ is extrapolated with the average value of those values within the sliding window $\mathcal{W}(h)$, which are less than $\mu_{i,t-1}$, i.e., for those values with indicator $\mathcal{J} = 1$. That is: $x_{i,t} = \frac{\sum_{k=t-h+1}^{h-1} \mathcal{J}(x_{i,k};h) x_{i,k}}{\sum_{k=t-h+1}^{h-1} \mathcal{J}(x_{i,k};h)}$. [Case 2]: If $\mu_{i,t-1} \leq m_{i,t-1}$ then the underlying distribution of stream s_i is *left* skewed, also referred to as negative skewness. In that case, the missing value $x_{i,t}$ is extrapolated with the average value of those values within the sliding window $\mathcal{W}(h)$, which are equal or greater than $\mu_{i,t-1}$, i.e., for those values with indicator $\mathcal{J} = 0$. In this case, $x_{i,t} = \frac{\sum_{k=t-h+1}^{h-1} (1-\mathcal{J}(x_{i,k};h)) x_{i,k}}{\sum_{k=t-h+1}^{h-1} (1-\mathcal{J}(x_{i,k};h))}$. The SSM at each time instance t has to calculate the incremental mean value $\mu_{i,t}$ and the incremental median $m_{i,t}$. For each time instance, the SSM invokes the CIP operator over the sliding window $\mathcal{W}(h)$. In the case of a missing value, the SSM based on the median and mean value imputes the $x_{i,t}$ missing value and then invokes a CIP operator over the sliding window. As in IMM, the SSM involves in the CIP operator the values within a sliding window where a portion of them (statistically, they are βh) refer to imputed values corresponded with the conditional mean

value as discussed in Cases 1 and 2. On average, for the n streams, the CIP operator involves $n\beta h$ imputed values given a sliding window $\mathcal{W}(h)$. Given that $\beta > 0$, the number of actual values is $n(1-\beta)h$ with respect to n streams.

Remark 4. The IMM depends only on the current mean value for imputation and invokes a CIP operator over the sliding window $\mathcal{W}(h)$ at every time instance t. The SSM depends on the current mean and median and depends on the conditional mean value for imputation over the sliding window $\mathcal{W}(h)$. In addition, the SSM invokes a CIP operator over $\mathcal{W}(h)$ at every time instance t.

The computation complexity of the SSM has as follows: Given a sliding window $\mathcal{W}(h)$, the SSM at each time instance t for each stream s_i calculates the incremental mean with $O(1)$ and calculates the incremental median with $O(h \log h)$ (dynamic insertion of the new value and sorting of h values), given that it exploits only the most recent values in the window $\mathcal{W}(h)$. Then it checks the indicator $I(x_{i,t})$ for each stream s_i. If for a stream s_i it holds true that $I(x_{i,t}) = 0$, then the SSM imputes the missing value $x_{i,t}$ by calculating the conditional mean over $\mathcal{W}(h)$ with $O(h)$; it is the same complexity for both Cases 1 and 2. Then, after all required imputations, the SSM invokes the CIP operator over the $\mathcal{W}(h)$ for all n streams with $O(f(\mathcal{W}(h)))$. Hence, at each time t, on average, the SSM requires $O(n\beta h \log h) + O(f(\mathcal{W}(h)))$. In the case of the IMM, only the incremental mean is calculated with $O(1)$. Hence, for all the streams, at time instance t, the processing over $\mathcal{W}(h)$ requires $O(n\beta) + O(f(\mathcal{W}(h)))$ time.

Average Nearest Exponential Smoothing Model (ANESM). The ANESM [16] depends on (i) the Average Nearest Observation (ANO) method and (ii) the Holt-Winters Double Exponential Smoothing (DSE) [23] method for imputing a missing value at time instance t for every stream s_i. The ANO method is applied over a sliding window $\mathcal{W}(h)$ and substitutes each value x_i with the average x'_i of the nearest *backward* and *forward* values. That is given the series of $x_{i,k}$, $k = t-h+1, \ldots, t-1$ from the sliding window $\mathcal{W}(h)$, the corresponding smoothed values $x'_{i,k}$ are: $x'_{i,k} = \frac{1}{h} \sum_{j=k-\frac{h-1}{2}}^{k+\frac{h-1}{2}} x_{i,j}$. Once the x'_i values are generated in the $\mathcal{W}(h)$, then the missing value at time instance t, $x_{i,t}$, is imputed using the DES method. Specifically, once the series of values within $\mathcal{W}(h)$ are smoothed using the ANO method, the $x_{i,t}$ missing value is predicted by applying DES over those smoothed series. The DES method is represented as: $a_{i,t} = \delta x'_{i,t} + (1-\delta)(a_{i,t-1} + b_{i,t-1})$ and $b_t = \gamma(a_{i,t} - a_{i,t-1}) + (1-\gamma)b_{i,t-1}$, where $x'_{i,t}$ is the actual smoothed value from the ANO method at time t, a_t and a_{t-1} are the intercepts at time instance t and $t-1$, respectively. The b_t and b_{t-1} are the corresponding slopes (time series trends) at time t and $t-1$, respectively. The δ and γ are smoothing constants in (0,1). The δ value is used to smooth the new actual and trend-adjusted previously smoothed intercept, while the γ value is used to smooth the trend. The smoothing constants determine the weight given to most recent past values and control the weight of smoothing. Values close to 1 give weight to more recent values and near to 0 distribute the weights to consider values from the more distant past within the window $\mathcal{W}(h)$. We set $\delta = 0.7$ and $\gamma = 0.9$ as in [16]. Hence, upon the event of

a missing value at time instance t at stream s_i, the $x_{i,t}$ value is imputed as: $x_{i,t} = a_{i,t-1} + b_{i,t-1}$. Note that $a_{i,t-1}$ takes into consideration the smoothed series $x'_{i,k}$, $k = t-h+1, \ldots, t-1$, from the ANO method. **For each time** instance, the ANESM invokes the CIP operator over the sliding window $\mathcal{W}(h)$. In the case of a missing value, the ANESM based on the ANO and DES method imputes the $x_{i,t}$ value and, then, invokes a CIP operator over the sliding window. On average, for the n streams, the CIP operator involves $n\beta h$ imputed values given a sliding window $\mathcal{W}(h)$. This means that the number of actual values is $n(1-\beta)h$ with respect to n streams.

Remark 5. The computation complexity of the ANESM has as follows: Given a sliding window $\mathcal{W}(h)$, the ANESM at each time instance t checks the indicator $I(x_{i,t})$ for each stream s_i. If for a stream s_i it holds true that $I(x_{i,t}) = 0$, then the ANESM first proceed with the smoothing of the values within the $\mathcal{W}(h)$ and then imputes the missing value of $x_{i,t}$. This requires $O(h^2)$ time for smoothing all values in $\mathcal{W}(h)$ and $O(h)$ time for imputation. Then, after all required imputations, the ANESM invokes the CIP operator over the $\mathcal{W}(h)$ for all n streams with $O(f(\mathcal{W}(h)))$. Hence, at each time t, on average, the ANESM requires $O(n\beta h^2) + O(f(\mathcal{W}(h)))$.

Exponential Moving Average Model (EMAM). The EMAM [10] weighs more the nearest neighbors of the current missing value $x_{i,t}$ through an exponential decaying factor. The idea here is to update **at each time** instance the weighted mean value over a sliding window $\mathcal{W}(h)$. Upon an event of a missing value at time instance t, the missing value $x_{i,t}$ corresponding to stream s_i is estimated by: $x_{i,t} = \frac{\sum_{k=t-h+1}^{t-1} x_{i,k} e^{-\phi|k-t|}}{\sum_{k=t-h+1}^{t-1} e^{-\phi|k-t|}}$. The $\phi \in (0,1)$ parameter controls the decaying factor. For a small decaying factor, all values in window $\mathcal{W}(h)$ are assigned similar weights. This effect diminishes as window size h increases and ϕ introduces a higher variation among the weights of the near and distant neighbors of the current missing value at time instant t. When we adopt a high value for the decaying factor, $\phi = 1$, the effect of the distant neighbors diminishes faster, thus, rendering the size of the sliding window irrelevant for the performance of the missing value imputation accuracy. We set $\phi = 0.5$ as in [10]. For each time instance, the EMAM invokes the CIP operator over the sliding window $\mathcal{W}(h)$. In the case of a missing value, the EMAM imputes the missing value $x_{i,t}$ and then invokes a CIP operator over the sliding window. It should be noted here that the EMAM requires to compute the weighted mean value only upon the occurrence of a missing value. On average, for the n streams, the CIP operator involves $n\beta h$ imputed values given a sliding window $\mathcal{W}(h)$. This means that the number of actual values is $n(1-\beta)h$ with respect to n streams.

Remark 6. The computation complexity of the EMAM has as follows: Given a sliding window $\mathcal{W}(h)$, the EMAM at each time instance t checks the indicator $I(x_{i,t})$ for each stream s_i. if for a stream it holds true that $I(x_{i,t}) = 0$, then the EMAM imputes the missing value of $x_{i,t}$. This requires $O(h)$ computational time. Then, after all required imputations, the EMAM invokes the CIP operator

Model	Computational Complexity
DQM	$O(n\beta) + p^* O(f(\mathcal{W}(\tau, t^*)))$
DAQM	$O(n\beta) + p^* O(f(\mathcal{W}(\tau, t^*)))$
SWM	$O(f(\mathcal{W}(h)))$
IMM	$O(n\beta) + O(f(\mathcal{W}(h)))$
SSM	$O(n\beta h \log h) + O(f(\mathcal{W}(h)))$
ANESM	$O(n\beta h^2) + O(f(\mathcal{W}(h)))$
EMAM	$O(n\beta h) + O(f(\mathcal{W}(h)))$

Table 2: Time Complexity for the Comparison Models; $p^* \in (0, 1)$ is the probability that at a given time instance the optimal stopping time criterion for the DAQM and DQM holds true.

over the $\mathcal{W}(h)$ for all n streams with $O(f(\mathcal{W}(h)))$. Hence, at each time t, on average, the EMAM requires $O(n\beta h) + O(f(\mathcal{W}(h)))$. Table 2 shows the computational complexity of all models.

7.2. Comparative Metrics & Setup

For the comparative assessment of the models DQM, DAQM, IMM, SSM, ANESM and EMAM, we adopt the real dataset provided by the Intel Berkeley Research Lad, called Intel Lab Data[3]. The dataset contains environmental contextual data reading, such as temperature, humidity, and light, reported by $n = 54$ Mica2Dot sensors. The sensors were installed in an indoor area, and had the same reporting frequency of once per 31 seconds. In our comparative evaluation, we choose the contextual parameter of temperature and the dataset consists of $N = 43047$ 54-dimensional contextual vectors, where each dimension corresponds to a context stream. The sliding window size h takes values in $\{10, 50\}$, which correspond to a sliding window of 310 seconds (5.17 minutes) and 1550 seconds (25.8 minutes), respectively. Hence, the quality guarantee for the DQM and DAQM is, respectively, $T = nh \in \{540, 2700\}$. In the experiments, we set the usable probability $\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. That is, at each reporting time instance t, for each context stream s_i, $i = 1, \ldots, 54$, we consider a value to be usable with probability β. Hence, on average each context stream s_i has βN usable values, for different values of β. We deal with the `MAX` standard aggregate and `AVG` algebraic aggregate operators for all models. The comparison performance metrics for all models are: (1) the error-pre-stream e/n as defined in Section 6.1, (2) the total number of *CIP invocations* per stream s_i, notated by C_i, for the DQM and DAQM (landmark window methods) and the sliding window methods, i.e., SWM, IMM, SSM, ANESM and EMAM, up to N, and (3) the per stream *invocation factor*, notated by M_i, of window accesses (sliding or landmark) due to CIP invocation and the invocation of a MVS method over window $\mathcal{W}$ of length $|\mathcal{W}|$ out of N.

The C_i metric denotes how many times each model invokes a CIP operator over a sliding/landmark window $\mathcal{W}$ up to N contextual data readings. The

[3]http://db.csail.mit.edu/labdata/labdata.html

M_i metric denotes how many times each model invokes a CIP operator over a window $\mathcal{W}$ weighted by the length $|\mathcal{W}|$ of the window and the *extra* required window access due to the (possible) invocation of a MVS algorithm/method adopted by a model, upon the event of a missing value. Since all methods SWM, IMM, SSM, ANESM and EMAM each time instance t invokes the CIP operator over a sliding window $\mathcal{W}(h)$, then the corresponding M_i/N value is at least h, i.e., CIP invocation per every contextual value reading. Moreover, for such methods, if at time instance t the event of a missing value holds true, then they invoke their corresponding MVS method by accessing the sliding window. Our models: DQM and DAQM based explicitly on the quantity of the usable pieces of data for each context stream attempt to control/tune the rate of CIP invocations in light of saving computational resources. The CIP invocation in our models involves the length of the current landmark window, which is variable. Recall also that in our models, there is no invocation of a MVS method. However, this comes at the expense of the provided accuracy of CIP results, as will be discussed below.

Formally, let us define the CIP invocation indicator function $I_1(t)$ at time instance t, which takes the value of 1 if a CIP operator is invoked over a window, otherwise zero. Evidently, in the case of SWM, IMM, SSM, ANESM and EMAM we obtain: $I_1(t) = 1, \forall t$, while in the case of DQM and DAQM, we obtain that: $I_1(t) = \begin{cases} 1 & \text{if } t = \tau^* \text{ w.p. } p^* = P(t = \tau^*) \\ 0 & \text{otherwise.} \end{cases}$ Recall τ^* is the optimal stopping time that DQM and DAQM decide to invoke a CIP operator over a landmark window. In the case of SWM, IMM, SSM, ANESM and EMAM the (sliding) window length is fixed h. In the case of DQM and DAQM, the (landmark) window length is variable and depends on the length or an era between two consecutive stopping decisions. This is captured by the introduced metric M_i per stream s_i, which is defined as: $M_i = \frac{1}{N}\sum_{t=h}^{N}(I_1(t) + 1 - I(x_{i,t}))|\mathcal{W}$ for the SSM, ANESM and EMAM. Since both SWM and IMM do not depend on the event of a missing value, then we obtain that: $M_i = \frac{1}{N}\sum_{t=h}^{N} I_1(t)|\mathcal{W}|$.

The length of window $|\mathcal{W}| = h$ in the case of SWM, IMM, SSM, ANESM and EMAM (sliding window) and all these methods start from time instance $t = h$ since they require at least h values to operate on an initialized sliding window of length h. Given that the probability of a usual value in a context stream is β, then the expected value of M_i for a context stream s_i is: $E[M_i] = \frac{(N-h)}{N}h(2-\beta)$ for SSM, ANESM and EMAM, which operate over $N - h$ streaming values. In the case of SWM and IMM, the expected value of $E[M_i] = \frac{(N-h)}{N}h$, being independent of β. The average value for all context streams with respect to the M_i is $M = \frac{1}{n}\sum_{i=1}^{n} M_i$.

In the case of DQM and DAQM, when $t = \tau_k^*$, i.e., it is the k-th decision to stop and invoke a CIP operator at time instance t, then the length of the landmark window $|\mathcal{W}_t| = t - t'$, which corresponds to the number of pieces of values between the k-th and $(k-1)$-th stopping eras at time $t = \tau_k^*$ and $t' = \tau_{k-1}^*$ with $t' < t$. The M_i per stream metric is defined for the DQM and DAQM as: $M_i = \frac{1}{N}\sum_{t=1}^{N} I_1(t)|\mathcal{W}_t|$. It is worth noting that in the case of DQM and DAQM,

the M_i value is at most 1. Specifically, let us assume that up to N readings of a context stream s_i, the DQM (or DAQM) method optimally decides to stop K times, i.e., K eras. At each era, the length of the landmark window is: $|\mathcal{W}_k| = \tau_k^* - \tau_{k-1}^*$, assuming that the optimal stopping time $\tau_0^* = 0$; fictitious state. Evidently, the sum of the products of the $I_1(t)$ with $|\mathcal{W}_t|$ at the time instances $t = \tau_k^*$, for $k = 1, \ldots, K$, equals to $\tau_K^* = \sum_{k=1}^{K} |\mathcal{W}_k| = \sum_{k=1}^{K} (\tau_k^* - \tau_{k-1}^*) \leq N$. That is, the corresponding M_i value up to N is at most 1, or more specifically, $\frac{\tau_K^*}{N}$. The $M_i = M_j$ for every pair of context streams s_i and s_j since the stopping time depends on all the values of the streams, thus, $M = M_i$.

Finally, the C_i per stream metric is defined for all models as: $C_i = \sum_{t=1}^{N} I_1(t)$, i.e., the total number of CIP invocations for the DQM and DAQM (landmark window methods) and the sliding window methods, i.e., SWM, IMM, SSM, ANESM and EMAM, up to N. The average value for all context streams with respect to C_i is $C = \frac{1}{n} \sum_{i=1}^{n} C_i$. Overall, we require from a model a low error-per-stream e/n value along with a low factor of window access M and, consequently, low number of CIP invocations, to reflect efficient resource usage for data stream processing.

7.3. Comparative Evaluation

We first compare the quality of the delivered context for all models in terms of the error-per-stream e/n metric. Figure 7 shows the error per stream against the probability of a usable value β for $h = 10$ and $h = 50$ over $n = 54$ streams from the Intel Lab Data using the `MAX` operator. The IMM and SSM methods achieve the highest error due to the fact that they only rely on the statistical mean and median information to impute missing values, which are not reliable when β is low. Moreover, the substitution of the missing values with the current mean (in the case of IMM) and the conditional mean (in the case of SSM) does not offer better accuracy results, especially when the sliding window length is high (see Figure 7(right) with $h = 50$). The ANESM and EMAM methods achieve better accuracy in terms of error-per-stream compared with the DQM for high β values. It is however interesting to note that our DAQM achieves significantly similar accuracy with the ANESM and EMAM methods for all β values. Furthermore, the improvement of ANESM and EMAM methods in absolute values compared with our models is marginal and is only observed with high β. Both ANESM and EMAM require high computational time (see Table 2) to achieve this accuracy level compared with our models. It is also worth noting that the ANESM does not behave better than our models for low β values with high window length. This is due to the fact that the ANESM operates over an exponential smoother which is highly unstable (due to the smoothing function) in the case of a high portion of missing values.

In Figure 8 we show the error per stream against the probability of a usable value β for $h = 10$ and $h = 50$ over $n = 54$ streams from the Intel Lab Data using the `AVG` operator. Since now the CIP operator is the `AVG` both the the IMM and SSM methods achieve low error due to the fact that they exploit the cumulative knowledge of the statistical mean and median. However, for a

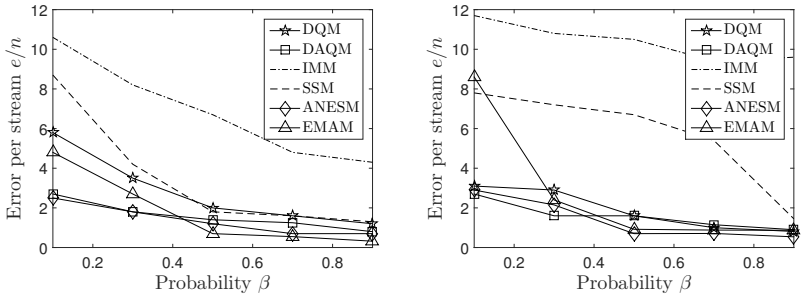

Figure 7: Impact of probability β on error per stream e/n for `MAX` operator over DQM, DAQM ($a = 0.95, c = 0.5$), IMM, SSM, ANESM, and EMAM, using $n = 54$ context streams and (left) $h = 10$, (right) $h = 50$; Intel Lab Data.

relatively high window length (see Figure 8 (right)), the impact of a high h on the cumulative mean and median results in higher error for the IMM and SSM methods. On the other hand, the EMAM method achieves the lowest error for this type of CIP operator. This is due to the fact that the weighted exponential mean value follows the current average of the values in a window, thus, the CIP error is relatively small. However, this does not hold true for the ANESM method, which achieves quite the same error with our models, but it requires high computational resources. Especially, when h is high, then the ANESM obtains higher error than DAQM and similar error with DQM at the expense of a high number of window accesses and CIP invocations.

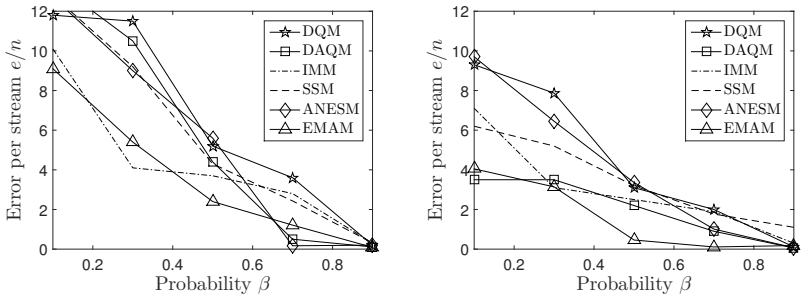

Figure 8: Impact of probability β on error per stream e/n for `AVG` operator over DQM, DAQM ($a = 0.95, c = 0.5$), IMM, SSM, ANESM, and EMAM, using $n = 54$ context streams and (left) $h = 10$, (right) $h = 50$; Intel Lab Data.

We also experiment with the required invocation factor and CIP invocations for each model to achieve a specific error per stream. This will provide us with certain insights on the computational resources needed for each model to efficiently proceed with a CIP operator and to deliver high quality context. Figure 9 (left) shows the average invocation factor M for all models against the probability of a usable value β in logarithmic scale. One could observe that our

models do not depend on the probability β as concerns the invocation factor M, since they adopt the landmark window and the rate of CIP invocations is optimally scheduled. Moreover, our models do not require a MVS method to impute any missing value and, more interestingly, they do not invoke a CIP operator each time a new piece of value is received. This is optimally controlled by the proposed delay mechanism. Hence, the invocation factor refers only to those cases that both the DQM and DAQM method intelligently decide to invoke a CIP operator. On the other hand. the IMM, SMM, ANESM and EMAM methods should invoke a CIP operator at every time, since there is not mechanism to schedule the rate of invocation. Moreover, such models have access to a fixed window l and the quality of the delivered context depends only on the invocation of a MVS method. This extra data access makes these models to require high computational resources compared to our models. Hence, the non-sophisticated schedule of the CIP operator invocation and the mandatory invocation of a MVS method upon an event of a missing value result into a high invocation factor. However, the ANESM and EMAM methods in certain cases, i.e., with a relatively small window length and high β value, obtain better performance in terms of the error-per-stream compared to our model. Nonetheless, in order for these models to achieve a slightly better accuracy they require, on average, 15 times more computational resources (in terms of the M factor) than our models. This is also observed in Figure 10, which plots the trade-off: invocation factor M against the achieved error-per-stream for all models for different β values. For instance, the DAQM for $\beta = 0.5$ achieves the same error with the EMAM model with 93% less invocation factor. In terms of efficiency (error and invocation factor), the IMM and SSM are the less efficient methods for all β values. In case that we require very high accuracy, at the expense of a high invocation factor, the ANESM and EMAM methods are deemed appropriate. Nonetheless, for low β values and for high efficiency, the DAQM model is the most appropriate since it achieves quite the same accuracy level with a highly computational resource demand method (EMAM, ANESM) over streams with a low number of usable values. The same hold true for the DQM method, which operates efficiently with low requirements of computational resources and satisfactory accuracy levels.

To further present the trade-off of accuracy and computational resources, we show in Figure 9 (right) the average CIP invocations C for both the `MAX` and `AVG` operators against β values for all models: the DQM and DAQM landmark window models and the IMM, SSM, ANESM, and EMAM, notated as sliding window models. Through this metric C, we show the number of times a CIP operator is invoked independently of the window access. Evidently, all the sliding window models simply invoke a CIP operator at each time instance, thus, without imposing any delay. However, this comes at the expense of high computational resources demand, compared with the C metric for our models. It is also worth noting that the DAQM and DQM methods increase the value of C as the probability of a usable value β increases. This is due to the fact that both methods minimize the imposed delay in light of a good quality of context. Even in the case where $\beta = 0.9$, the C value obtained by our models

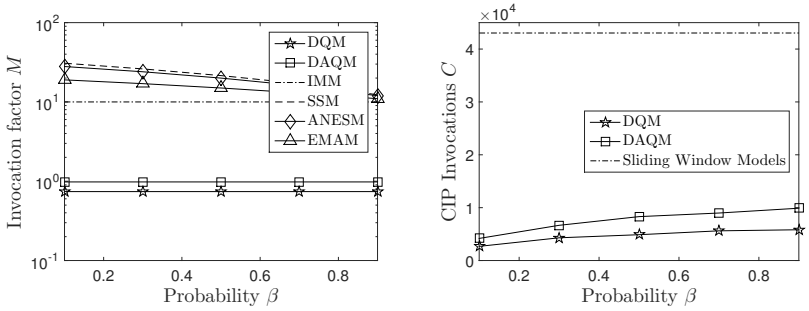

Figure 9: Impact of probability β on (left) the invocation factor M and (right) CIP invocations, over DQM, DAQM ($a = 0.95, c = 0.5$), IMM, SSM, ANESM, and EMAM (sliding window models), using $n = 54$ context streams and $h = 10$; Intel Lab Data.

is more than four times lower than the C value obtained by the sliding window models. This indicates the efficiency achieved by our models balancing between accuracy and demands in computational resources.

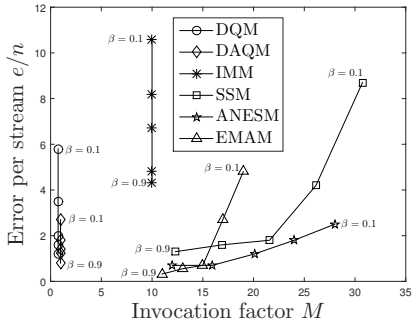

Figure 10: The trade-off invocation factor M and error per stream e/n for different probability β values for all models; $n = 54$ context streams and $h = 10$; CIP is `AVG`; Intel LabData.

8. Conclusions

We studied the case of a time-optimized mechanism for improving the quality of results from contextual information processes. We proposed two time-optimization models (the DQM and DAQM) based on the principles of the theory of optimal stopping, where they evaluate the quality of the incomplete contextual data streams and, then, optimally decide on when to activate a contextual information process (aggregation and/or fusion function) with the aim to increase the quality of results. Both models can swiftly take optimal decisions on-line thus being highly adaptable to on-line analytics tasks over contextual data streams. We show that our models improve the quality of the CIP results and avoid the continuous redundant activation of CIPs each time (incomplete)

context is received to the back-end monitoring system, as proposed in the approaches of the related work. Our mechanism compensates between efficient resource usage and achieved accuracy over incomplete context streams.

References

[1] D. J. Abadi, D.Carney, U.Cetintemel, M.Cherniack, C.Convey, S.Lee, M. Stonebraker, N.Tatbul, S.Zdonik, 'Aurora: a new model and architecture for data stream management', J. VLDB, 12(2):120–139, 2003.

[2] C. Anagnostopoulos, P. Triantafillou, 'Scaling out big data missing value imputations: pythia vs. godzilla', ACM KDD '14, 651–660, 2014.

[3] C. Anagnostopoulos, S. Hadjiefthymiades, 'Enhancing situation-aware systems through imprecise reasoning', IEEE Transactions on Mobile Computing, 7(10):1153–1168, 2008.

[4] C. Anagnostopoulos, Y. Ntarladimas, S. Hadjiefthymiades Situational computing: an innovative architecture with imprecise reasoning J. System and Software, 80 (12):1993–2014, 2007.

[5] A. Arasu, S. Babu, J. Widom, 'The CQL continuous query language: semantic foundations and query execution', J. VLDB, 15(2):121–142, 2006.

[6] I. B. Aydilek, A. Arslan, 'A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm', Information Sciences, 233(1), 25–35, 2013.

[7] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. ACM PODS, 1–16, 2002.

[8] S. Bandyopadhyay, C. Gianella, U. Maulik, H. Kargupta, K. Liu, S. Datta, Clustering distributed data streams in peer-to-peer environments. Information Sciences, 176(14):1952–1985, 2006

[9] T.-M. Choi, H. K. Chan, X. Yue, Recent Development in Big Data Analytics for Business Operations and Risk Management, IEEE Transactions on Cybernetics, 99:1–12, 2016.

[10] M. Dallachiesa, B. Nushi, K. Mirylenka, T. Palpanas. 2012. Uncertain time-series similarity: return to the basics. Proc. VLDB, 5, 11, 1662–1673.

[11] C. Daskalakis et al. 'Learning poisson binomial distributions', ACM STOC '12, 709–728.

[12] A. Farhangfar et al, 'Impact of imputation of missing values on classification error for discrete data', Pattern Recognition, 41(12): 3692–3705, 2008.

[13] M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: A review. SIGMOD, 34(2):18–26, 2005.

[14] J. Gray, S. Chaudhuri, et al. 'Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals'. Data Mining and Knowledge Discovery, 1(1):29–53, 1997.

[15] A. Z. Grzybowski, 'Simulation approach to optimal stopping in some blackjack type problems', Scientific Research of the Institute of Mathematics and Computer Science, 2011, 10(2):75–86.

[16] P. Gupta and R.Srinivasan,'Missing Data Prediction and Forecasting for Water Quantity Data', IPCSIT, 10, 2011, IACSIT Press.

[17] S. Guha, N. Mishra, R. Motwani, L. O' Callaghan, Clustering Data Streams', Annual Symposium on Foundations of Computer Science, 2000, 359–366.

[18] K. Hiramatsu, T. Hattori, T. Yamada, T. Okadome, Finding small changes using sensor networks. Ubicomp'05, 37–44.

[19] K. Kolomvatsos, C. Anagnostopoulos, S. Hadjiefthymiades, 'A time optimized scheme for top-k list maintenance over incomplete data streams', Information Sciences, 311:59–73, 2015.

[20] L. Kong, M. Xia; Xiao-Yang Liu, Min-You Wu, X. Liu, 'Data loss and reconstruction in sensor networks', IEEE INFOCOM, 2013, pp.1654–1662.

[21] C. Leung, Q. Khan, B. Hao, Distributed mining of constrained patterns from wireless sensor data, IEEE/WIC/ACM WI-IATW06.

[22] J. Lian, K. Naik, L. Chen, Y. Liu, G. Agnew, Gradient boundary detection for time series snapshot construction in sensor networks. IEEE Transactions on Parallel and Distributed Systems, 2007.

[23] A. Lynwood, C. Johnson Douglas, Montgomery and John S. Gardiner. 'Forecasting and Time Series Analysis'. McGraw–Hill, 2nd ed., 1990.

[24] S. McConnell, D. Skillicorn, A distributed approach for prediction in sensor networks. SIAM Intl Conf Data Mining, pp. 28–37, 2005.

[25] L. Mo, Y. He, Y. Liu, J. Zhao, S. Tang, X. Li, G. Dai, 'Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in Forest', ACM SenSys '09, 99–112.

[26] T. Palpanas, D. Papadopoulos, V. Kalogeraki, D. Gunopulos, Distributed deviation detection in sensor networks. ACM SIGMOD, 32(4):77–82, 2003.

[27] K. Patel, R. G. Mehta, M. M. Raghuvanshi, N. N. Vandere, 'Incremental Missing Value Replacement Techniques for Stream Data', Intl J. Computer Applications, 122(17):0975–8887, 2015.

[28] G. Peskir, A. Shiryaev, 'Optimal Stopping and Free-Boundary Problems', ETH Zuerich, Birkhauser Basel, 2006.

[29] L. L. Pipino, Y. W. Lee, R. Y. Wang, 'Data quality assessment', Commun. ACM 45, 4:211–218, 2002.

[30] A. Shiryaev, 'Optimal Stopping Rules', Series: Stochastic Modelling and Applied Probability, 8, 2007.

[31] M. Stonebraker, U. Cetintemel, S. Zdonik, 'The 8 requirements of real-time stream processing', ACM SIGMOD, 34(4):42–47, 2005.

[32] R. Y. Wang, D. M. Strong, 'Beyond accuracy: what data quality means to data consumers', J. Management Information Systems, 12(4):5–33, 1996.

[33] Z. Yang, N. Meratnia, P. Havinga, Outlier Detection Techniques for Wireless Sensor Networks: A Survey," Communications Surveys & Tutorials, IEEE, 12(2):159–170, 2010.

[34] E. Zervas, A. Mpimpoudis, C. Anagnostopoulos, O. Sekkas, and S. Hadjiefthymiades. Multisensor data fusion for fire detection. Information Fusion, Elsevier, 12(3):150-159, 2011.