Research Article

A stream-sensitive distributed approach for configuring cascaded classifier topologies in real-time large-scale stream mining systems

Abtin Shahkarami[1] · Hossein Bobarshad[1] · Nader Bagherzadeh[2]

Abstract
Stream mining systems have received a great deal of attention in recent years. These systems process incoming data streams from different sources and extract high-level semantic features from them. They do this by passing data streams through an ensemble of classifiers. Owing to dynamic changes in characteristics of the data streams, these classifiers need to be configured dynamically to maximize the performance of the system. As a challenge, different data streams from different sources have different specifications from each other. This causes treating all the incoming data streams identically by a common topology configuration to be not appropriate for an optimal stream mining. Hence, an approach is required which allows each data stream to be processed by consideration of its own specifications. In this paper, by implementing a buffer for each source and using a time-sharing solution, we propose a distributed approach to solve the aforementioned problem for cascaded classifier topologies. We first formally define a utility metric which captures both the performance and the delay of a binary filtering classifier system. We then propose our solution for a base case and evolve it step by step until reaching the most general case for cascaded topologies. We finally test and compare our approach with the state-of-the-art solution on a text detection scenario from the incoming video streams to the system.

Keywords Stream mining systems · Distributed systems · Real-time stream processing · Big data classification

1 Introduction

Every year globally an ever-increasing amount of data is being produced by different sources in various formats [1], including multimedia files [2], medical measurements [3], and information from satellites [4]. This high volume of data streams requires operations such as classification, filtering, aggregation, and correlation [5–7], in plethora of nowadays applications such as search engines [8], fraud detection systems [9], video surveillance systems [10], medical devices [11], sensor networks [12], and autonomous robots control systems [13]. These systems require high computational processing power to process incoming continuous data streams from distributed sources [14]. Distributed stream mining systems have been recently developed in order to perform these tasks [5, 15–18]. They are constructed by using a topology of low-complexity binary classifiers, each performing feature extraction and classification specific to different tasks [14]. Indeed, they decompose applications as topologies of distributed processing operators, which will highly increase reliability, scalability, and performance of the system [19–22].

One of the main issues in the real-time distributed stream mining system is how to handle system overload effectively while maintaining high performance under resource constraints [5]. A common approach is to use a chain of classifiers for intelligent load shedding. In this approach, each classifier in the chain determines when, where, what, and how much of each stream data object (SDO) to discard and what to pass to the next classifier in the chain to reach the desired quality of service (QoS) and meet the delay constraints. This approach can cause a significant volume of

✉ Abtin Shahkarami, abtinshahkarami@alumni.ut.ac.ir | [1]Department of Network Science, Faculty of New Sciences and Technologies, University of Tehran, Tehran, Iran. [2]Henry Samueli School of Engineering, University of California, Irvine, Irvine, USA.

SN Applied Sciences (2019) 1:594 | https://doi.org/10.1007/s42452-019-0565-6

Received: 21 January 2019 / Accepted: 6 May 2019 / Published online: 18 May 2019

data streams to be shed prior to reaching the next classifiers in the chain, resulting in a much less computational burden on the successive classifiers [23–30]. However, optimizing each classifier individually without a joint consideration of misclassification error effects and resource constraints at possibly multiple downstream classifiers in the chain may lead to a high sub-optimality and cause the end-to-end processing delay for a chain of classifiers to become intolerable for real-time applications [1, 31, 32]. As a result, the optimal topology must be based on a joint optimization over the configuration of all classifiers operating points (i.e., ordered pair of the probability of false alarm and detection), while observing delay and resource constraints [1, 33–37]. This concept is similar to [5] where a utility metric for real-time stream processing applications is introduced, which captures the trade-offs between classification accuracy and end-to-end delay of the filtered stream. The algorithm configures each classifier by choosing an operating point to control its performance and throughput to maximize the utility metric in each period. However, the approach in [5] considers a fixed configuration for the ensemble in each interval based upon the defined utility metric in the paper. Thus, the ensemble treats all the incoming data streams from different sources with a common ensemble configuration in each interval. This issue, in applications where the importance and specifications of data streams are different from each other, can lead to non-optimality because the set thresholds of classifiers may be aggressive for some sources and permissive for some others. Hence, many of the data streams may not receive the desired quality of service, which makes them useless in consequence, whereas an approach can be adopted such that the ensemble processes each data stream optimally by the topology configuration that is specifically set for that data stream. In this paper, we propose such an approach for cascaded topologies, an approach in which the ensemble treats each SDO by the required optimal topology configuration of the SDO.

In this paper, we first introduce a utility metric for the single-path–single-source case. We discuss a method for decomposing the utility function into a set of locally observable metrics that can be calculated directly by each classifier (similar to what is proposed in [5]). Each classifier can then exchange these metrics with other classifiers to compute the utility of the entire system for any fixed configuration of classifiers, thereby configuring itself by choosing the proper operating point, which controls its performance and throughput in order to maximize the utility of the entire system. Next, we extend our solution to the single-path–multiple-source case by implementing a buffer for each source at the ensemble side and using a time-sharing approach for serving accumulated SDOs in their buffers periodically. We then extend our approach to the most general case, which is multiple-path–multiple-source.

Note that in this paper we do not modify the underlying classification scheme, but rather focus on configuring operating points of individual classifiers in a fixed processing sequence. This allows the designed algorithms to be applicable to any available type of underlying classification algorithms (e.g., support vector machines, k-nearest neighbors, maximum likelihood, etc.). The paper is organized as follows: In Sect. 2 we discuss our model for cascaded topologies of distributed stream processing systems. In this section we also derive a utility function for capturing the performance of the ensemble in mining a single stream. Then we propose a framework for the single-path–single-source case in Sect. 3. In Sects. 4 and 5, we extend the framework to the single-path–multiple-source and the multiple-path–multiple-source cases, respectively. In Sect. 6 we provide and compare the implementation results of our approach and the state-of-the-art approach for a text detection scenario in incoming video streams from different sources. Finally, we conclude the paper in Sect. 7.

2 Distributed stream mining system model

2.1 Binary classification

In a binary cascaded classifier topology, as shown in Fig. 1, each classifier C_i classifies each incoming SDO as belonging to the class of interest, H_i (positive class), or not belonging to the class of interest, i.e., belonging to the negative class, $\overline{H_i}$. If an SDO is labeled as belonging to H_i by classifier C_i, it is forwarded to the next classifier in the chain; otherwise, it is dropped from the stream. As a result, SDOs which reach the last classifier in the chain are labeled as positive in all of the previous classifiers in the chain. If we denote the classification decision of classifier C_i by $\hat{X}_i$ and the ground truth by X_i, then *the probability of detection* which determines the proportion of correctly forwarded samples is $P_i^D = Pr\left\{\hat{X}_i \in H_i \middle| X_i \in H_i\right\}$, and *the probability of false alarm* which determines the proportion of incorrectly forwarded samples is $P_i^F = Pr\left\{\hat{X}_i \in H_i \middle| X_i \notin H_i\right\}$. Therefore, the probability of forwarding an SDO to the next classifier in the chain by classifier C_i, if the SDO has a priori *probability* (APP) π_i of being positive, can be given by:

$$\Omega_i = \pi_i P_i^D + \left(1 - \pi_i\right) P_i^F. \tag{1}$$

Also, the probability of correctly forwarding an SDO to the next classifier in the chain would be equal to:

$$\delta_i = \pi_i P_i^D. \tag{2}$$

If we assume that each classifier operates at a fixed complexity level, Ω_i and δ_i can become deterministic function of P_i^F by using the APP π_i, and the detection error trade-off (DET) curve of the classifier, which is concave and increasing and relates P_i^D to P_i^F ($P_i^D = f\left(P_i^F\right)$) [30, 38, 39]. Therefore,

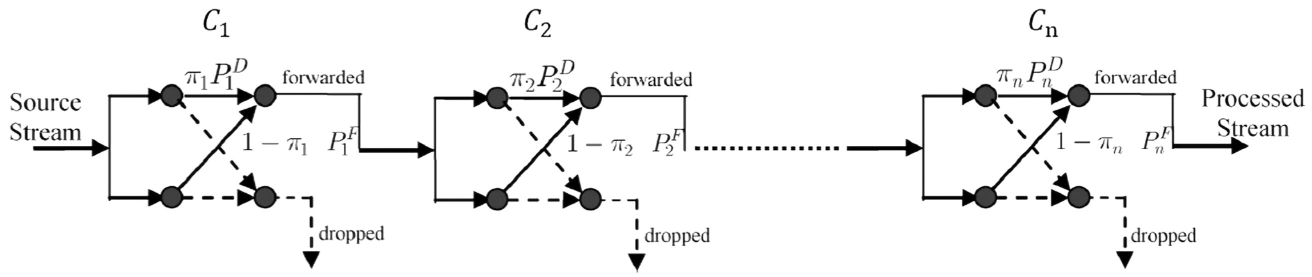

Fig. 1 A cascaded single-path topology [5]

each classifier can be configured by varying its false alarm constraint in order to maximize utility of the system.

2.2 Cascaded classifier model

In this paper, we model the ensemble of the classifiers as a $D/G/1$ queuing facility. Also each classifier C_i can itself be modeled as $G/G/1$ queuing facility followed by stream filtering and splitting operations. Each SDO is forwarded by classifier C_i to the next classifier with the probability Ω_i and is discarded with the probability $1-\Omega_i$. For classifier C_i, the total SDO input rate and the average output rate are denoted by λ_i and μ_i, respectively. Also the input and output rate of the whole ensemble are denoted by λ and μ, respectively. The set of previous-hop and next-hop neighbors of classifier C_i are, respectively, denoted by prev(i) and next(i). The ancestors of classifier C_i are defined as the classifiers which have path to C_i and are denote by anc(i). Similarly, descendants of classifier C_i are defined as the set of classifiers which C_i have path to, and are denoted by des(i). We denote arrival rate from $C_y \in$ prev(i) to C_i by λ_y^i; therefore, $\sum_{y\in \text{prev}(i)} \lambda_y^i = \lambda_i$. If classifier C_i forwards an SDO, the probability of choosing $C_x \in$ next(i) is denoted by Φ_i^x, so $\sum_{x\in \text{next}(i)} \Phi_i^x = 1$and therefore $\lambda_y^i = \Phi_i^x \Omega_i \lambda_i$.

2.3 Objective function for mining a stream

The objective of mining a stream data object is to minimize misclassification cost, subject to an end-to-end delay constraint, which leads to an optimization problem. Misclassification cost consists of two types of errors in classification task, which are false alarms and misses. (Misses are those SDOs which are actually in the class of interest but not detected as belonging to the class of interest by the classifier.) Hence, misclassification cost of classifying an SDO by classifier C_i can be given by $\wp_i = (\pi_i - \delta_i) + \theta(\Omega_i - \delta_i)$, where δ_i and Ω_i are specific to the data stream of the SDO. $\pi_i - \varphi_i$ and $\Omega_i - \delta_i$ specify the fractions of misses and false alarms, respectively, and θ denotes the weight of false alarms to misses. Therefore, the end-to-end misclassification cost of classifying an SDO by a single-path chain of classifiers, labeled from 1 to n, can be given by:

$$\begin{aligned}\wp &= (\pi - \delta) + \theta(\Omega - \delta)\\ &= \pi - \prod_{i=1}^{n} \delta_i + \theta\left(\prod_{i=1}^{n} \Omega_i - \prod_{i=1}^{n} \delta_i\right)\end{aligned} \tag{3}$$

where $\prod_{i=1}^{n} \Omega_i$ is the total fraction of forwarded SDOs and $\prod_{i=1}^{n} \delta_i$ is the total fraction of correctly forwarded SDOs. Because the parameter π in (3) depends just upon stream characteristics, it can be regarded as a constant and be omitted. Therefore, it is possible to produce a utility function for the system classification performance by removing π and inverting (3), which results in:

$$U = \prod_{i=1}^{n} \delta_i - \theta\left(\prod_{i=1}^{n} \Omega_i - \prod_{i=1}^{n} \delta_i\right) \tag{4}$$

In real-time stream mining applications, there is also a delay penalty in SDOs classification process. This type of penalty captures the loss of utility due to the volume of SDOs which have not been processed in the specified hard deadline Λ. Hard deadline Λ is the time limit from where if the processing delay of an SDO (from capturing to coming its final result out of the ensemble) cross, its result will not be useful anymore and it can be considered as a miss (considering a hard delay deadline is common in most of the embedded real-time systems [40–43]). Hence, a function $\psi(\tau)$ must be defined to estimate the proportion of SDOs whose processing delays (from capturing) have not crossed the hard deadline [31, 32, 44]. Trivially, this function can be given by $\psi(\tau) = Pr\{\tau \le \Lambda\}$, which is the probability of the processing delay of a processed SDO (from capturing) that has not exceeded the hard deadline. Hence, it captures the fraction of data that is useful for the system. In practice, this probability can be obtained by time stamping the SDO packets and calculating the fraction of processed SDOs which have crossed the deadline. Also, alternatively, it can be estimated analytically if an exact model (e.g., $D/M/1$) for the arrival and service times is used. By combination of system classification performance utility and delay penalty, a single objective function $U.\psi(\tau)$ can be achieved, which satisfies the concept of fairness between accuracy and delay implemented by Nash product [5, 45]. Therefore, if we consider the general form for the ensemble of classifiers, i.e., multi-path topology, the objective function which the system must attempt to maximize by configuring each classifier false alarm probability will be as follows [5]:

$$\Theta\left(P^F, \Phi\right) = \sum_r \Phi_r \psi\left(\tau_r\right) \times \left(\prod_{i\in r} \delta_i - \theta\left(\prod_{i\in r} \Omega_i - \prod_{i\in r} \delta_i\right)\right)$$
$$\text{s.t} \quad 0 \le P_i^F \le 1$$
$$\sum_r \Phi_r = 1, \quad \Phi_r \ge 0 \tag{5}$$

where $P^F = \left\{P_1^F, \ldots, P_N^F\right\}$ is the set of false alarm probabilities of all classifiers, Φ is the set of all end-to-end path selection probabilities, Φ_r is the probability of choosing the end-to-end path r, and τ_r is the expected end-to-end delay across the path r. Also $i \in r$ means classifier C_i is a classifier along the path r.

In the following sections, we discuss what needs to be done to maximize $\Theta\left(P^F, \Phi\right)$ for cascaded topologies in different scenarios. We first propose a distributed framework for the single-path–single-source case (a case where the ensemble has just one end-to-end path and there is only one source which is sending data). Then we extend our approach step by step to reach the solution for the most general case, which is multiple-path–multiple-source (the case where there are several end-to-end paths in the ensemble and also there are multiple sources which are sending data to the ensemble).

3 Distributed approach for single-path–single-source case

3.1 Fundamentals of the solution

In single-path–single-source case, there is only one path; hence, the objective function for this case is:

$$\Theta\left(P^F\right) = \psi(\tau) \times \left(\prod_{i=1}^{n} \delta_i - \theta\left(\prod_{i=1}^{n} \Omega_i - \prod_{i=1}^{n} \delta_i\right)\right) \tag{6}$$
$$\text{s.t} \quad 0 \le P_i^F \le 1$$

which is obtained from (5).

The specifications and characteristics of data streams change over time (e.g., getting day and night or getting more and less important). This causes APPs and θ of the data stream to be changed dynamically. These dynamic changes also cause the current configuration of the ensemble to lose its optimality during the time and require to be updated dynamically. To cope effectively with this event, the time needs to be split into smaller intervals. Before each interval, APPs and θ of the data stream should be updated. Then, a new configuration should be adopted in each new interval based upon the updated APPs and θ, such that for each classifier C_i the following condition is met [5]:

$$P_i^F(t) = \arg\max_{P_i^F} E\left[\Theta^{t+1}\left(P^F\right)\right] \tag{7}$$

where $\Theta^{t+1}\left(P^F\right)$ means the achieved utility in $t+1$th interval and $P_i^F(t)$ is the configuration that is adopted by C_i at time t.

Framework 1 Distributed framework for single-path-single-source case

- **Initialize** the configuration $P_i^F(0)$ for each classifier C_i and exchange initial APPs across all classifiers
- **Repeat** at the beginning of each interval $[t, t+1)$
  1. Make each classifier update its APP based upon extrapolating from actual values of input data in previous intervals or by using a model (e.g., multivariate Gaussian model)
  2. Update $\psi(\tau)$ by using observed end-to-end delays of the processed SDOs in the previous intervals
  3. Update θ
  4. Make each classifier exchange its Ω_i and δ_i of the interval $[t-1, t)$, with other classifiers to obtain an approximation of Θ^t $(\tilde{\Theta}^t)$.
  5. Make each classifier configure its operating point based upon theoretical analysis, empirical analysis, modeling, etc., such that $P_i^F(t) = \arg max_{P_i^F} E[\Theta^{t+1}(P^F)]$.

3.2 Framework for single-path–single-source case

Because classifiers are distributed, none of them has enough information on its own to configure its operating point to maximize (6), due to the need of other classifier APPs and operating points. But each classifier can estimate its own APP and knows its own current operating point. As a result, for every parameter in (6), there exists a classifier which has access to it and knows

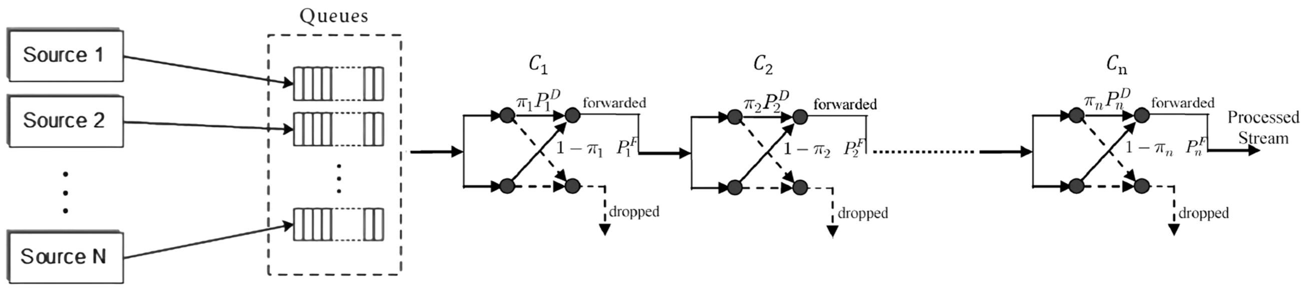

Fig. 2 An illustration of our proposed approach. In this approach, SDOs sent from a source are buffered in their specific queue, first. Then they are discharged periodically and processed by the corresponding optimal topology configuration of their source

its value. Therefore, by exchanging these parameters across all the classifiers and using updated θ and $\psi(\tau)$, each classifier can determine its own configuration for the upcoming interval to maximize the defined objective function. Therefore, based on the foregoing, we can propose Framework 1 as our approach for single-path–single-source case.

In the following section, we will extend Framework 1 to a framework which can support the case where several sources are sending data streams (these streams can have different importance and characteristics from each other) simultaneously to the ensemble. We will then provide the implementation results of the approach in Sect. 6.

4 Distributed approach for single-path–multiple-source case

In most stream mining applications, there is more than one source which sends data stream to the ensemble (e.g., video surveillance system of a city). In some of these systems, different sources have different specifications and importance from each other. For example, in a city video surveillance system, the importance of a data stream from a camera in an important square is not identical to a camera in an ordinary alley. Furthermore, the importance of the data stream from a camera in a location may be increased greatly in some periods of time. Also, characteristics of data streams from different sources can be different from each other (e.g., some are captured at dark and some others in light scenes). These differences cause each data stream to have its own APPs, weight of false alarm to misses (θ), and required thresholds for classification tasks. Even, different classification algorithms may be required to be applied for some sources. Hence, it is not optimal to mine all the incoming data streams from different sources identically by a common topology configuration.

In order to preserve QoS, each data stream should be mined by its own specific required configuration. In this section, we propose a time-sharing approach for solving this issue, such that a specific queue is implemented for each source (in this paper we assume that these queues are implemented at the ensemble side) at where the SDOs are buffered. Then the ensemble dedicates an amount of time periodically to each queue to discharge and mine SDOs inside it. We define a cycle as a repeating time interval during which all the queues get service once from the ensemble. We also call the part of a cycle when a queue is getting service, as the queue's turn in the cycle. In a queue's turn, the SDOs inside the queue are discharged and processed by consideration of their source's specific δ_i, Ω_i, θ, APPs and $\psi(\tau)$. That is to say, they are processed by their specific needed topology configuration. (An illustration of this approach is shown in Fig. 2.)

We model each buffer as a queue with FIFO discipline where the input rate is constant and deterministic. Also during the emptying periods, the queue obeys $D/D/1$ model. (Note that we have modeled the ensemble of classifiers as a $D/G/1$ queue facility which allows the SDOs to enter the ensemble deterministically.) In this approach, choosing the optimal amount of time (T^***)**, dedicated to each queue in order to process accumulated SDOs inside it, is the most important concept that we must investigate. In order to solve this problem, we first consider the case

Table 1 Descriptions of some of the parameters in the paper

Parameter	Description	Unit
α_i	Acquisition rate of source *i*	SDOs/s
λ	Queue discharge rate	SDOs/s
μ	Output rate of the ensemble	SDOs/s
σ	Expected value of end-to-end processing delay of the ensemble	s
Δ	Maximum acceptable end-to-end delay (deadline) for an SDO from entering the queue to coming its result out of the ensemble	s
N	Number of sources	N/A
T_i	Dedicated amount of time to the queue of source *i* to serve its SDOs	s
c	Time needed for switching from serving one queue to the next one and setting new configuration	s

where source acquisition rates and the required classification algorithms are fixed and identical for all the sources. Then, we solve the problem for the case where sources can have adaptive acquisition rates or different classification algorithms can be applied on their streams. Table 1 lists new parameters that we will use in this section.

4.1 Fix and identical source acquisition rates case

When acquisition rates of all the sources are equal to each other, there would be an identical number of accumulated SDOs in each queue at the beginning of their turn. Therefore, the amounts of time that the system should dedicate to each queue would be identical to each other. Hence, $T_1 = T_2 = T_3 = \cdots = T_N = T$.

The time a queue has to wait for getting its turn only just after the end of its passed turn is:

$$\text{QWT} \cong (\sigma + c) + (N-1)(T + \sigma + c) \tag{8}$$

where σ is the expected amount of time that the result of the last SDO in the queue comes out from the ensemble after entering to it. Also c is the required amount of time for switching to servicing the next queue and adopting a new configuration for it. Note that because we assumed the queues are placed at the ensemble side, the transmission rate from queue to the ensemble can be neglected.

We define $\eta = (\sigma + c)$ and $\omega = \eta + \text{T}$. Therefore, if we dedicate time T to each queue, the number of received SDOs by the queue from end of a turn until the end of its next turn would approximately be equal to:

$$\text{NAS} \cong [\eta + (N-1)\omega + T]\alpha. \tag{9}$$

Trivially, all the accumulated SDOs inside a queue must be discharged from the queue in its turn. Otherwise, the queue will overflow after some cycles. Hence, the following condition must be met:

$$T \geq \frac{[\eta + (N-1)\omega + T]\alpha}{\lambda} = N\omega\frac{\alpha}{\lambda} = N\eta\frac{\alpha}{\lambda} + NT\frac{\alpha}{\lambda} \tag{10}$$

where λ is equal to the queue discharge rate. Hence,

$$T - TN\frac{\alpha}{\lambda} \geq N\eta\frac{\alpha}{\lambda} \Rightarrow T\left(1 - N\frac{\alpha}{\lambda}\right) \geq N\eta\frac{\alpha}{\lambda}. \tag{11}$$

Because all the parameters in (11) are greater than zero, we would have $N\eta\frac{\alpha}{\lambda} > 0$. Hence,

$$1 - N\frac{\alpha}{\lambda} > 0 \Rightarrow N\alpha < \lambda \Rightarrow N < \frac{\lambda}{\alpha}. \tag{12}$$

Therefore, in the design process of the stream mining systems, the condition in (12) has to be considered and met. Also note that, when adjusting λ, the condition $\mu \geq \lambda$ must hold in order for the average delay to be finite; otherwise, the utility is always 0 [5]. Hence, if the condition in (12) is satisfied, by (11) we get:

$$T \geq \frac{(N\eta\alpha/\lambda)}{\left(1 - N\frac{\alpha}{\lambda}\right)} = \frac{N\eta\alpha}{\lambda - N\alpha}. \tag{13}$$

Therefore, (13) determines the lower bound of the domain of T, which is the minimum value T can have. Also, it must be considered that the chosen T should not cause the end-to-end delay of any SDOs to exceed the deadline Δ, in theory. (It is obvious that Δ is equal to Λ minus the expected value of the time that an SDO arrives at its queue after capturing.) For satisfying this condition, we split the incoming SDOs into the two following groups:

- SDOs which arrive at their queues when it is not their queues' turn;
- SDOs which arrive at their queues when it is their queues' turn.

We find upper bound of T for each of these groups.

1. SDOs which arrive at their queues when it is not their queues' turn:

In this case, the total delay (from entering to the queue to coming its result out of the ensemble) of the first SDO which arrives at its queue after the queue's passed turn is:

$$D(1) \cong \eta + (N-1)\omega - \frac{1}{\alpha} + \frac{1}{\lambda} + \sigma \tag{14}$$

where $1/\lambda$ is the time taken for discharging the SDO from the queue, σ is expected value of the ensemble end-to-end processing delay for the SDO, and $1/\alpha$ is the approximated amount of time that the SDO arrives after the end of the passed queue's turn. Also, η, as defined earlier, is the time it takes from end of the just past queue's turn to the beginning of the next queue's turn. Now, if we consider ith arrived SDO, there exist $i-1$ SDOs in front of it in the queue, which require $(i-1)/\lambda$ seconds to be discharged. Also, ith SDO is approximately arrived at i/α seconds after the end of its queue's turn. Therefore, the total delay of ith SDO from entering to the queue to coming its result out of the ensemble is:

$$\begin{aligned} D(i) &\cong \eta + (N-1)\omega - \frac{i}{\alpha} + \frac{i-1}{\lambda} + \frac{1}{\lambda} + \sigma \\ &= \eta + (N-1)\omega - i\left(\frac{1}{\alpha} - \frac{1}{\lambda}\right) + \sigma. \end{aligned} \tag{15}$$

Because for each i, it must be $D(i) \leq \Delta$, the following condition must be satisfied:

$$T \leq U(i) = \left[\Delta + i\left(\frac{1}{\alpha} - \frac{1}{\lambda}\right) - \sigma - N\eta\right] \times \frac{1}{N-1}. \tag{16}$$

Hence, it must be $T \leq \min U(i)$. For finding $\min U(i)$, we differentiate U with respect to i. We have

$$\frac{\partial U}{\partial i} = \frac{1}{N-1}\left(\frac{1}{\alpha} - \frac{1}{\lambda}\right). \tag{17}$$

Because $\alpha < \lambda$ (otherwise, overflow will happen), it would be $\partial U/\partial i > 0$. Hence, U is strictly increasing and meets its minimum at the lower bound of its domain, i.e., $i = 1$. Hence,

$$T \leq \left[\Delta + \left(\frac{1}{\alpha} - \frac{1}{\lambda}\right) - \sigma - N\eta\right] \times \frac{1}{N-1}. \tag{18}$$

By (13) and (18) we have:

$$\frac{N\eta\alpha}{\lambda - N\alpha} \leq \left[\Delta + \left(\frac{1}{\alpha} - \frac{1}{\lambda}\right) - \sigma - N\eta\right] \times \frac{1}{N-1} \tag{19}$$

which results in:

$$\Delta + \frac{1}{\alpha} + c - (N+1)\eta - \frac{N\eta\alpha(N-1)}{\lambda - N\alpha} \geq 0. \tag{20}$$

Therefore, the condition in (20) must be considered to be satisfied all the time.

Note that, if we denote the left-hand side of (20) by M, it is trivial that M will increase by increasing Δ. Also $\partial M/\partial\alpha = (-1)/\alpha^2 - N\eta\lambda(N-1)/(\lambda - N\alpha)^2 < 0$. Hence, M will increase by decreasing α.

2. SDOs which arrive at their queues when it is their queues' turn:

When an SDO arrives at its queue after t seconds from the beginning of its current queue's turn, $t\lambda$ SDOs will have been discharged since the beginning of the turn. Also, $t\alpha$ new SDOs will have arrived at the queue. Hence, the number of accumulated SDOs in the queue after t seconds would be equal to:

$$A(t) = [(N-1)\omega + \eta]\alpha - t\lambda + t\alpha = [(N-1)\omega + \eta]\alpha - t(\lambda - \alpha) \tag{21}$$

where $[(N-1)\omega + \eta]\alpha$ is the number of SDOs inside the queue at $t = 0$.

If we differentiate function A with respect to t, we would have $\partial A/\partial t = -(\lambda - \alpha) < 0$. Therefore, function A is strictly decreasing. Hence, it meets its maximum at the lower bound of its domain, which is $t = 0$. Therefore, the maximum possible number of SDOs in front of an incoming SDO during a turn is $A(0)$.

The system should be designed such that the total delay of the SDO coming at $t = 0$ (from entering the queue to coming its result out of the ensemble) does not exceed the deadline. Hence, the following condition must be satisfied:

$$\frac{[(N-1)\omega + \eta]\alpha}{\lambda} + \frac{1}{\lambda} + \sigma \leq \Delta \tag{22}$$

which results in:

$$T \leq \frac{(\Delta - \sigma)\lambda - N\eta\alpha - 1}{(N-1)\alpha}. \tag{23}$$

By (13) and (23), the following condition must be satisfied:

$$\frac{N\eta\alpha}{\lambda - N\alpha} \leq \frac{(\Delta - \sigma)\lambda - N\eta\alpha - 1}{(N-1)\alpha}. \tag{24}$$

Therefore, stream mining systems should be designed such that the following condition is met:

$$\frac{(\Delta - \sigma)\lambda - N\eta\alpha - 1}{(N-1)\alpha} - \frac{N\eta\alpha}{\lambda - N\alpha} \geq 0. \tag{25}$$

Note that, if we denote the left-hand side of (25) by M and differentiate it with the respect to α, we have:

$$\frac{\partial M}{\partial \alpha} = \frac{-[(\Delta-\sigma)\lambda-1](N-1)}{((N-1)\alpha)^2} - \frac{N\eta(\lambda-N\alpha)-N\eta\alpha(-N)}{(\lambda-N\alpha)^2} < 0. \tag{26}$$

Hence, M will increase by decreasing α. Also, we have $\partial M/\partial\Delta = \lambda/(N-1)\alpha > 0$. Hence, M will increase by increasing Δ.

Therefore, by (13), (18), and (23), the range of T is derived as follows:

$$T \in \left[\frac{N\eta\alpha}{\lambda - N\alpha}, \min\left(\left[\Delta + \left(\frac{1}{\alpha} - \frac{1}{\lambda}\right) - \sigma - N\eta\right] \times \frac{1}{N-1}, \frac{(\Delta-\sigma)\lambda - N\eta\alpha - 1}{(N-1)\alpha}\right)\right]. \tag{27}$$

Various objectives can be considered for choosing the best T in its range, which are ultimately the designer's choice. But to highlight possible approaches, in this paper, we assume that the best T is the one which minimizes the expected value of the total delay of an SDO from entering its queue to coming its result out of the ensemble. For this purpose, once more, we consider the two groups of SDOs we discussed earlier, and find the solution for each of them. Then we will achieve the general optimal T by combining the results.

1. SDOs which arrive at their queue when it is not their queues' turn:

For this group of SDOs, if we denote the total delay of ith arrived SDO, from entering the queue to coming its result out of the ensemble, by $D(i)$, the optimal T is the one that maximizes $D(E[i])$.

Because the number of arrived SDOs at a queue during the end of its just past turn and the beginning of its next turn is $[\eta + (N-1)\omega]\alpha$, the expected value of i is:

$$E[i] = \sum iPr(i) = \sum_{i=1}^{\alpha(N\eta+(N-1)T)} i\frac{1}{\alpha(N\eta + (N-1)T)} = \frac{\alpha(N\eta + (N-1)T) + 1}{2}. \tag{28}$$

By (15), we have:

$$D(E[i]) \cong \eta + (N-1)\eta - \frac{\alpha(N\eta + (N-1)T) + 1}{2}\left(\frac{1}{\alpha} - \frac{1}{\lambda}\right) + \frac{1}{\mu} = \frac{(N\eta + (N-1)T)}{2} - \frac{1}{2\alpha} + \frac{1}{\lambda}\left[\frac{\alpha(N\eta + (N-1)T) + 1}{2}\right] + \frac{1}{\mu}. \tag{29}$$

Now, we have:

$$\frac{\partial D(E[i])}{\partial T} = \frac{N-1}{2} + \frac{\alpha(N-1)}{2\lambda} > 0. \tag{30}$$

Hence, D is strictly increasing with respect to T and meets its minimum at the lower bound of the domain of T. Therefore, the optimal T for this group of SDOs is:

$$T^* = \frac{N\eta\alpha}{\lambda - N\alpha}. \tag{31}$$

2. SDOs which arrive at their queues when it is their queues' turn

If an SDO arrives at second t after starting its queue's turn ($t \le T$), the total delay of the SDO from entering the queue to coming its result out of the ensemble, by (21), would be:

$$K(t) = \max\left(\frac{[(N-1)T + N\eta]\alpha - t(\lambda-\alpha)}{\lambda}, 0\right) + \sigma. \tag{32}$$

The optimal T is the one that minimizes $K(E[t])$. Because SDOs are captured at a specific rate (i.e., every $1/\alpha$ seconds), the incoming arrival time of SDOs obeys a uniform distribution, so $E[t] = T/2$. Hence, we have

$$K(E[t]) = \max\left(\frac{[(N-1)T + N\eta]\alpha - \left(\frac{T}{2}\right)(\lambda-\alpha)}{\lambda}, 0\right) + \sigma. \tag{33}$$

If $[(N-1)T + N\eta]\alpha - (T/2)(\lambda - \alpha) \ge 0$, then the differentiation of (33) with respect to the parameter T would be as follows:

$$\frac{\partial K(E[t])}{\partial T} = \frac{(N-1)\alpha}{\lambda} - \frac{1}{2}\left(1 - \frac{\alpha}{\lambda}\right) = \frac{\alpha}{\lambda}\left(N - \frac{1}{2}\right) - \frac{1}{2}. \tag{34}$$

As a result:

$$\frac{\lambda}{\alpha} \le 2N - 1 \Rightarrow \frac{\partial K(E[t])}{\partial T} \ge 0\,; \tag{35}$$

hence, $K(E[t])$ would be increasing with respect to T, if $\lambda/\alpha \le 2N - 1$. Therefore, $K(E[t])$ would meet its minimum at lower bound of domain of T which is $\frac{N\eta\alpha}{\lambda - N\alpha}$, by (13). Also,

$$\frac{\lambda}{\alpha} > 2N - 1 \Rightarrow \frac{\partial K(E[t])}{\partial T} < 0\,; \tag{36}$$

hence, $K(E[t])$ will be strictly decreasing with respect to T, if $\lambda/\alpha > 2N - 1$. Therefore, $K(E[t])$ would meet its minimum at the upper bound of the domain of T, and inversely, upper bound of T is the value which minimizes $K(E[t])$. The minimum of $K(E[t])$ occurs when

$$[(N-1)T + N\eta]\alpha - \left(\frac{T}{2}\right)(\lambda - \alpha) = 0 \tag{37}$$

which results in $T = 2N\eta\alpha/(\lambda - (2N-1)\alpha)$. Therefore, when $\lambda/\alpha > 2N-1$, for the group of SDOs which arrive during the beginning and end of their queues' turn, the optimal T is $2N\eta\alpha/(\lambda - (2N-1)\alpha)$.

Also if $[(N-1)T + N\eta]\alpha - (T/2)(\lambda - \alpha) < 0$, then the function becomes constant. As a result, the output of the all values in the domain of T is optimal. Hence, we can again consider if $\lambda/\alpha \leq 2N-1$ the optimal T is $\frac{N\eta\alpha}{\lambda - N\alpha}$ and if $\lambda/\alpha > 2N-1$ the optimal T is $2N\eta\alpha/(\lambda-(2N-1)\alpha)$.

Now the question is, What is the optimal T, when $\lambda/\alpha > 2N-1$, since, for the group of SDOs which arrive at their queues' turn, the optimal T is $N\eta\alpha/(\lambda - N\alpha)$, and for the group of SDOs which arrive during the beginning and end of their queues' turn, the optimal T is $2N\eta\alpha/(\lambda-(2N-1)\alpha)$? To solve this question a bargaining problem is formed where the wealth of the first group is κ_{Pr} times more than the wealth of the second group (κ_{Pr} is the ratio of the probability that an arrived SDO belongs to the first group, to the probability of belonging to the second group). The two parties should finally agree on a common T which we call T_{aggred} and is equal to:

$$T_{\text{aggred}} = \arg\min_{T \in \left[\frac{N\eta\alpha}{\lambda - N\alpha}, \frac{2N\eta\alpha}{\lambda-(2N-1)\alpha}\right]} \mathcal{C}(T) \tag{38}$$

where

$$\mathcal{C}(T) = \kappa_{\text{Pr}} \times P\left(T - \frac{N\eta\alpha}{\lambda - N\alpha}\right) + P\left(T - \frac{2N\eta\alpha}{\lambda - (2N-1)\alpha}\right). \tag{39}$$

Function P is a cost function which determines the cost of deviation from choosing the optimal T for an SDO. By the Nash bargaining problem [45], $T^* = T_{\text{aggred}}$.

In a cycle, the proportion of the times at when if an SDO arrives it is placed in the first group, to the times at when if an SDO arrives it is placed in the second group, is $N-1$ to one. Therefore, in a modest approach, it can be assumed that $\kappa_{Pr} = N-1$. (Note that this is not an accurate approach at all.) Also, in almost all of non-extraordinary conditions, because there is no distinction between SDOs, the cost of any deviation from the interest would be equal for all the SDOs. Hence, if we take one step closer to one SDO's interest in the second group, we will get one step farther from $N-1$ SDOs' interest in the first group. Therefore, although we achieve 1 point we will lose $N-1$ points. Thus, in almost all of non-extraordinary conditions, $T_{\text{aggred}} = \frac{N\eta\alpha}{\lambda - N\alpha}$.

In conclusion, in this subsection, we discussed how to extend the approach for single-path–single-source case to a single-path–multiple-source case where source acquisition rates are fixed and identical. We also discussed the conditions that must be considered to be satisfied in designing process of stream mining systems. We can summarize our discussion in this subsection as Framework 2. Indeed, this framework proposes our approach for a single-path–multiple-source case where source acquisition rates are fixed and identical.

In the following subsection, we will extend Framework 2 to an approach for the case where sources can have adaptive acquisition rates, in order to cope with the stream importance and specification dynamics. We will then represent the implementation results of the approach in Sect. 6.

4.2 Adaptive source acquisition rates case

In the previous subsection, we proposed an approach for fixed and identical source acquisition rates, but as we mentioned earlier, the importance and characteristics of different data streams change dynamically during the time.

Framework 2 Distributed framework for the single-path-multiple-source case

- **Satisfy** all the following conditions regularly:

  - $\frac{\lambda}{\alpha} > N \;\&\&\; \mu \geq \lambda$
  - $\Delta + \frac{1}{\alpha} + c - (N+1)\eta - \frac{N\eta\alpha(N-1)}{\lambda - N\alpha} \geq 0$
  - $\frac{(\Delta-\sigma)\lambda - N\eta\alpha - 1}{(N-1)\alpha} - \frac{N\eta\alpha}{\lambda - N\alpha} \geq 0$

- **Initialize** T^*, and the configuration $P_i^F(0)$ of each source for each classifier C_i and exchange initial APPs of all sources across all the classifiers
- **Repeat** at the beginning of each iteration $[t, t+1)$ (time is split into smaller intervals called iteration)
  1. Make each classifier updates its APPs, based upon extrapolating from actual values of the input data in the previous intervals or by using a model (e.g., multivariate Gaussian model)
  2. Update $\psi(\tau)$ and σ of each source by using the observed end-to-end delays of the processed SDOs in the previous intervals
  3. Update θ of each source
  4. Update T^* by the following function:

$$T^* = \begin{cases} \frac{N\eta\alpha}{\lambda - N\alpha}, & \frac{\lambda}{\alpha} \leq 2N - 1 \\ argmin_{T \in \left[\frac{N\eta\alpha}{\lambda - N\alpha}, \frac{2N\eta\alpha}{\lambda - (2N-1)\alpha}\right]} \mathcal{C}(T), o.w \end{cases}$$

  5. Make each classifier C_i exchange corresponding Ω_i and δ_i of each source in the interval $[t-1, t)$, with other classifiers to obtain an approximation of Θ^t ($\tilde{\Theta}^t$) for each source.
  6. Make each classifier configure its operating point (based upon theoretical analysis, empirical analysis, modeling, etc.), at the beginning of each source's turns, such that $P_i^F(t) = arg\,max_{P_i^F} E\left[\tilde{\Theta}^{t+1}(P^F)\right]$, where all the parameters are specific to the source.

These changes may cause the required acquisition rates for some sources to be increased. Even, it is probable that the classification algorithms for data streams coming from them need to be replaced. In this case, the aforementioned dedicated amount of time for serving accumulated SDOs inside these source queues would not be sufficient, and it needs to be increased. We denote the amount of time that is needed to be increased for source i by ΔT_i. For example, if the acquisition rate of source i was to increase m times, we would have $\Delta T_i = \frac{N\eta m\alpha}{\lambda - Nm\alpha} - \frac{N\eta\alpha}{\lambda - N\alpha}$. This leads the total required amount of time which must be dedicated to other sources for serving their accumulated SDOs to be decreased by ΔT, to preserve the overall sustainability of the system. Hence, the main challenge is how to reduce this amount from dedicated times to other sources in an optimal way. This time reduction can be done by either changing the classification algorithms, or by reducing source acquisition rates. However, changing the classification algorithms (if it is possible) is not reasonable and has a significant cost, due to spoiling the previous learnings. Therefore, in the following, we will focus on reducing the acquisition rates of the sources in an optimal way.

If we increase dedicated amount of time to a source, two types of penalties will be incurred by other sources: (1) loss of details penalty and (2) additional delay penalty.

1. Loss of details penalty

If we decrease the acquisition rate of a source to compensate an increase in the dedicated amount of time to another source, some details at the location of the source would not be captured and sent by the source, which causes a penalty. For capturing this penalty we first define the details loss function and denote it by $\mathfrak{A} : \mathbb{R} \to \mathbb{R}$. This function determines what will be the amount of details loss by $m\%$ decrease in source acquisition rate. Because the natures of data streams are the same (e.g., video of a location), this function is the same for all of the sources and gives output regardless of the source. But sources have different importance, so a certain decrease in source acquisition rate for different sources would not bring about equal costs. We denote the function that determines the importance of a source by $\mathcal{I} : \mathbb{N} \to \mathbb{R}$. Therefore, the cost function of $m\%$ decrease in acquisition rate of source i can be given by

$$\mathfrak{T}(i, m) = \mathcal{I}(i)\mathfrak{A}(m). \tag{40}$$

2. Extra delay penalty

If dedicated amount of time to a source is increased by ΔT, the SDOs inside the queues of the other sources would wait more, as well. This has a cost and causes the function $\psi(\tau)$ to output a smaller value for all or some of the sources. We define $\psi^c(\tau_i) = 1 - \psi(\tau_i) = Pr\{\tau_i > \Lambda\}$, where τ_i indicates the specific τ for source i. Therefore, the larger the $\Delta\psi^c(\tau)$, the more the information in the data streams of source i will be lost ($\Delta\psi^c(\tau_i)$ means the difference between $\psi^c(\tau_i)$ of the source i, before and after dedicating an additional time to the other source). Hence, $\Delta\psi^c(\tau)$ can be considered as a metric for cost of incurred additional delays.

Therefore, the cost of decreasing the acquisition rate of source i by $m\%$, to compensate the increase in dedicated amount of time to another source, can be given by $\mathfrak{T}(i, m). \sum_{i=1}^{N} (\mathcal{I}(i)\Delta\psi^c(\tau_i))^2$. Also, if the acquisition rates of a number of sources are changed, the cost function can be given by the following:

$$\mathcal{P}(M) = \left(\prod_{i=1}^{N} \mathfrak{T}\left(i, m_i\right) \right) \times \sum_{i=1}^{N} \left(\mathcal{I}(i) \Delta \psi^{c}\left(\tau_i\right)\right)^2 \tag{41}$$

where $M = \langle m_1, m_2, \dots, m_N \rangle$ is the vector of the amounts of decrease (by percentage) in acquisition rates of the sources. For example, when we just decrease the acquisition rate of source i by $m_i\%$, it is $M = \langle 0, 0, \dots, m_i, \dots, 0 \rangle$. Hence, if the dedicated amount of time to a source is required to be increased by ΔT, we must find and apply the optimal M, which can be expressed as:

$$M^* = \arg\min_{M} \mathcal{P}(M) \quad \text{s.t}\, R(M^*) = 0 \tag{42}$$

where

$$\mathcal{R}(M) = \frac{(N\eta)\left(M^T \langle \alpha \rangle\right)}{\lambda - N\left(M^T \langle \alpha \rangle\right)} - \Delta T. \tag{43}$$

Intuitively, $\mathcal{R}(M)$ determines how much of ΔT is compensated by the chosen M.

Note that the case of non-equal acquisition rates for the sources from the beginning can easily be modeled by assuming that changes occur at $t = 0$.

Therefore, by what we proposed in this subsection, we can extend our approach in the previous subsection to the adaptive source acquisition rate case and complete our approach for single-path–single-source cases.

5 Distributed approach for multiple-path–multiple-source case

In the previous section, we proposed our approaches for single-path cases. But, in most of the stream mining systems, there are several replicated classifiers in the ensemble for each classification task, to increase the system robustness, consistency, reliability, speed, etc. [46–49]. This causes several end-to-end paths to be created in the ensemble. In this section, we will discuss how to extend the approach in the previous section to multiple-path topologies.

5.1 Discussion on extending the previous approach to multiple-path case

As we discussed in Sect. 4, multiple-source case requires a time-sharing approach for entering captured and buffered SDOs into the ensemble. Our proposed time-sharing approach in the previous section would not be changed for multiple-path case, because multiple-path concept is an in-ensemble issue, and we consider the ensemble as a whole and only consider end-to-end parameters of the ensemble (e.g., σ, which is the expected end-to-end processing delay of the ensemble). Hence, the only thing that needs to be changed is the way of determining the configuration of classifiers for each turn at the beginning of iteration. Therefore, if the approach in Sect. 3 is extended to multiple-path case, the approach in IV can easily be extended to multiple-path case, as well. This is done by changing the method of steps 5 and 6 in the repeat section of Framework 2, to the method in the following subsection.

5.2 Ensemble configuration mechanism in multiple-path case

By extending (5), we have

$$\begin{aligned} \Theta\left(P^F, \Phi\right) &= \sum_{r} \Phi_r \psi(\tau) \times \left(\prod_{i \in r} \delta_i - \theta \left(\prod_{i \in r} \Omega_i - \prod_{i \in r} \delta_i \right) \right) \\ &= \sum_{r} \Phi_r \psi(\tau) \times \left((1+\theta) \prod_{i \in r} \delta_i - \theta \prod_{i \in r} \Omega_i \right). \end{aligned} \tag{44}$$

Now, if we consider an arbitrary classifier C_i in the ensemble, we can rewrite (44) based on C_i as follows [5]:

$$\Theta\left(P^F, \Phi\right) = \psi(\tau). \begin{bmatrix} (1+\theta) \displaystyle\sum_{j \in \mathrm{sib}(i) \cup \{i\}} \mathfrak{G}_0^j\left(P_{0,j}^F, \Phi_0^j\right) \\ \times \displaystyle\sum_{y \in \mathrm{next}(j)} \Phi_j^y \mathfrak{G}_y^N\left(P_{y,N}^F, \Phi_y^N\right) \\ \\ -\theta \displaystyle\sum_{j \in \mathrm{sib}(i) \cup \{i\}} \mathfrak{F}_0^j\left(P_{0,j}^F, \Phi_0^j\right) \\ \times \displaystyle\sum_{y \in \mathrm{next}(j)} \Phi_j^y \mathfrak{F}_y^N\left(P_{y,N}^F, \Phi_y^N\right) \end{bmatrix} \tag{45}$$

where sib(i) is the set of classifiers that perform the same classification task as C_i, in the ensemble,

$$\begin{aligned} \mathfrak{G}_0^j\left(P_{0,j}^F, \Phi_0^j\right) &= \left(\delta_j\right) \sum_{v_{\mathrm{anc}(j)}} \Phi_{v_{\mathrm{anc}(j)}} \left(\prod_{x \in \mathrm{anc}(j)}^{n} \delta_x \right) \\ \mathfrak{F}_0^j\left(P_{0,j}^F, \Phi_0^j\right) &= \left(\Omega_j\right) \sum_{v_{\mathrm{anc}(j)}} \Phi_{v_{\mathrm{anc}(j)}} \left(\prod_{x \in \mathrm{anc}(j)}^{n} \Omega_x \right) \end{aligned} \tag{46}$$

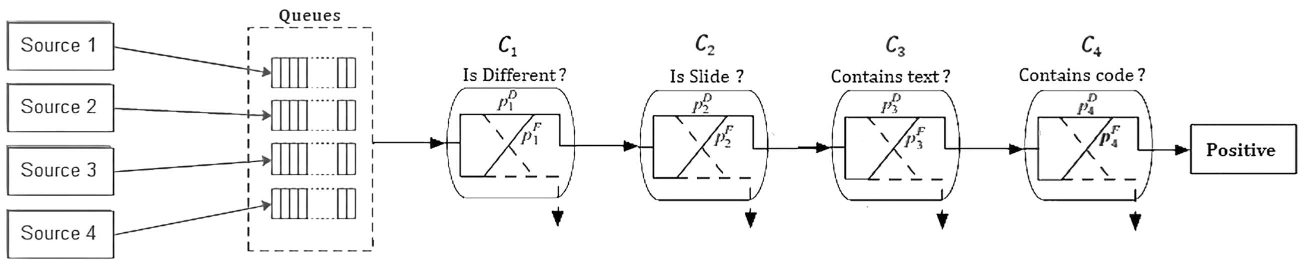

Fig. 3 Diagram of the implementation scenario

and

$$\mathfrak{G}_y^N\left(P_{j,N}^F,\Phi_y^N\right)=\left(\delta_y\right)\sum_{v_{\mathrm{des}(y)}}\Phi_{v_{\mathrm{des}(y)}}\left(\prod_{z\in\mathrm{des}(y)}^{n}\delta_z\right)$$
$$\mathfrak{F}_y^N\left(P_{y,N}^F,\Phi_y^N\right)=\left(\Omega_y\right)\sum_{v_{\mathrm{des}(y)}}\Phi_{v_{\mathrm{des}(y)}}\left(\prod_{z\in\mathrm{des}(y)}^{n}\Omega_z\right). \tag{47}$$

Hence, if local parameters and path selection probabilities of each classifier are exchanged with all the other classifiers, each classifier can adjust its operating point to maximize the objective function. However, there is no need of exchanging parameters with all the other classifiers, which has an information exchange overhead of $O\left(N^2\right)$, because each classifier only needs the product of the parameters of its ancestors and descendants. Therefore, by using forward propagation of products with the following recursive formula:

$$\mathfrak{G}_0^j\left(P_{0,j}^F,\Phi_0^j\right)=\left(\delta_j\right)\sum_{x\in\mathrm{prev}(j)}\Phi_x^j\mathfrak{G}_0^x\left(P_{0,x}^F,\Phi_0^x\right)$$
$$\mathfrak{F}_0^j\left(P_{0,j}^F,\Phi_0^j\right)=\left(\Omega_j\right)\sum_{x\in\mathrm{prev}(j)}\Phi_x^j\mathfrak{F}_0^x\left(P_{0,x}^F,\Phi_0^x\right) \tag{48}$$

and backward propagation of products with the following recursive formula:

$$\mathfrak{G}_i^N\left(P_{i,N}^F,\Phi_i^N\right)=\left(\delta_j\right)\sum_{y\in\mathrm{next}(i)}\Phi_i^y\mathfrak{G}_y^N\left(P_{y,N}^F,\Phi_y^N\right)$$
$$\mathfrak{F}_i^N\left(P_{i,N}^F,\Phi_i^N\right)=\left(\Omega_j\right)\sum_{y\in\mathrm{next}(i)}\Phi_i^y\mathfrak{F}_y^N\left(P_{y,N}^F,\Phi_y^N\right). \tag{49}$$

information exchange overhead can be reduced to $O(dN)$, where d is the average of out-degree/in-degree of the classifier nodes.

6 Experiments and results

We tested our approach on a text detection scenario in a large volume of incoming video streams from a number of sources which had adaptive acquisition rates. We implemented a multiple-path topology to extract different semantic features from the incoming frames. On this scenario, we also tested the state-of-the-art approach proposed in [5], which can be also called a fundamental approach in the field of SMAs as it is mathematically proven, and almost all the succeeding proposed approaches have been built on top of that and assumed it as a principal true approach[1] [1]. Also, this approach has been awarded the prize of the National Science Foundation of USA [50] and many patents have been registered on that [51].

In this section, we will describe the aforementioned scenario in detail, and by discussing the results of the implementations, we will show how our approach outperformed the state-of-the-art in the considered scenario.

6.1 Description of the application

To test our approach, we considered a scenario where four video lectures were playing from different sources. Some of the scenes in the videos were showing the slides that were being taught and presented in the lectures. The slides in each video had different templates and styles from the slides in other videos. We decided to construct a chain of classifiers intended to detect scenes of the videos in where a slide was showing, and the slide contained a snippet of programming code. We considered the end-to-end delay

[1] As a matter of fact, in this paper also, we do not intend to cast aspersions on [5], but we have brought up a scenario from the real world in which we showed [5] alone is not responsive and it needs an extension to become effective, a scenario which we could not find a paper debating about; otherwise, we would definitely compare our approach with that as well.

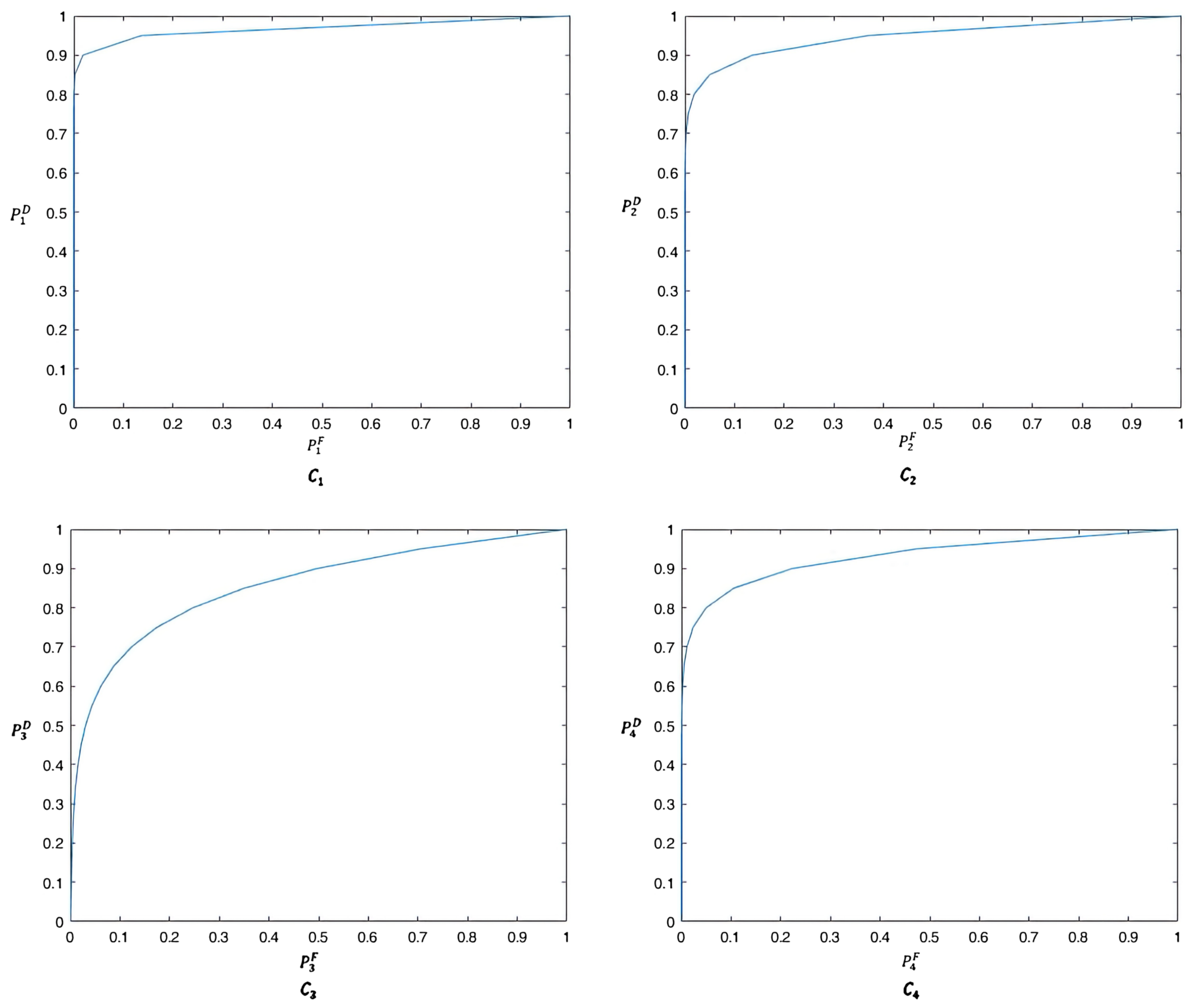

Fig. 4 DET curves of the classifiers in the chain

Table 2 Node specifications

Node	Implemented on	Programming languages	Avg. processing delay
C_1	Sony Xperia Z with Android 5.1, Quad-core 1.5 GHz Krait, and 2 GB RAM	JavaScript	0.13 s
C_2	Sony Xperia Z1 with Android 5.1, Quad-core 2.2 GHz Krait 400, and 2 GB RAM	JavaScript & Java	0.37 s
C_3 (1)	Sony Xperia Z2 with Android 5.1, Quad-core 2.3 GHz Krait 400, and 3 GB RAM	JavaScript & Java	4.67 s
C_3 (2)	Samsung Galaxy S7 edge with Android 7.0, Quad-core 2 × 2.15 GHz Kryo & 2x1.6 GHz Kryo, and 4 GB RAM	JavaScript & Java	4.44 s
C_4	Samsung Galaxy Note 10.1 N8000, with Android 4.4.2, Quad-core 1.4 GHz Cortex-A9, and 2 GB RAM	JavaScript	0.52 s
Queues	Lenovo Z570, Windows 7 PC, Intel-core i5-2410 M 2.3 GHz, 6 GB RAM	JavaScript	N/A

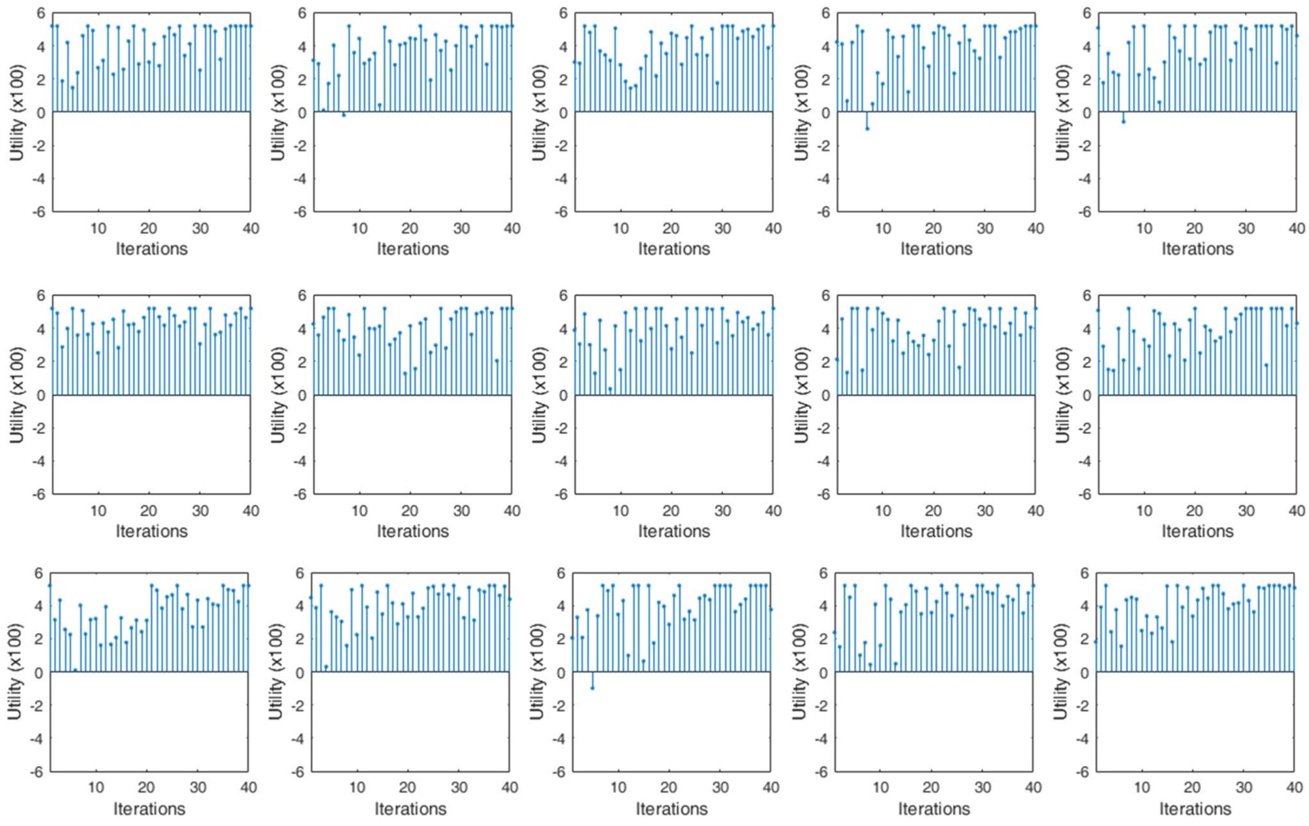

Fig. 5 Illustrative view of the achieved utilities in our proposed approach experimentations

Table 3 Details of the overall implementation results of the approaches

Approach	Average of the gained utilities (100×)	Average of the variances	Number of negative achieved utilities	Average of negative achieved utilities (100×)	Number of maximum achieved utilities
Ours	4.007	1.582	4	−0.688	150
State-of-the-art	3.532	3.932	56	−0.380	252

deadline of $\Lambda = 13$ s, for processing an SDO (from capturing to coming its result out of the ensemble).

Regarding dataset, we collected a video dataset consisting of more than 3 million frames (see [52] for access), from the video lectures of Google I/O conferences [53], MIT OpenCourseWare [54], Stanford Online [55], and NE Scala Symposium conferences [56], to test our approach and the state-of-the-art, which is without any time-sharing solution and buffer, on them.

6.2 Implementation details

We created a cascaded topology of 4 classifiers to reach the goal of our scenario. The task of first classifier (C_1) was to determine whether or not there was enough difference between the frame which C_1 was processing, and its previous frame. (If there is no noticeable difference between the frames, it means no significant changes happened, so no new information is received. Therefore, there is no need to process the current frame.) The task of second classifier (C_2) was to determine if a scene was showing a slide or not. The aim of third classifier (C_3) was to detect if there was any text on the slide or not. (We implemented two replicated nodes for this time-consuming task to increase scalability and reliability.) Also the fourth classifier (C_4) was for detecting whether or not the text on the slide contained any snippets of code. All classifiers were devised and trained based on SVM algorithm. The classifiers C_1 and C_2 were doing their analysis directly on RGB images, while the classifiers C_3 and C_4 were preceded by an OCR neural network which converted an image to a Unicode text file containing some characters which could be meaningful or meaningless. Therefore, the classifiers C_3 and C_4 were performing their task on text files. Indeed, the task of C_3 was to detect a set of meaningful sentences or words in a bunch of Unicode characters and the task of C_4 was to detect whether or not there is a set of computer code among those meaningful characters. The diagram of the scenario is illustrated in Fig. 3, The DET curve of each classifier is shown in Fig. 4, and the specifications of the nodes are listed in Table 2. The bandwidths of the links between the sources and queues were set to 1024 Kb/s, and the bandwidths of links between queues and the ensemble were set to 150 Mb/s as well as the bandwidths of intra-ensemble links. Also, the frames were sent in HD quality mode.

We set the initial weight of false alarm to misses on $\theta = 0.5$. We also considered when a slide in the concept of interest was detected from a sent SDO by a source, then it would be more likely that another slide in the concept of interest appeared again in that source's stream, so the sensitivity of detection would be better to be increased a bit. Hence, we set a plan to decrease the weight of false alarms to misses of a source by 0.004, when an SDO in the concept of interest was detected in the streams of the source. We also assumed that the importance of a source was increased when an SDO in the concept of interest was detected; because when in a video lecture there exist several slides, then more probably the lecture has more important information than a lecture with fewer slides. (Note that this assumption is only applicable to this scenario, and in most of the other scenarios, the situation is totally inverse, i.e., more quantity leads to less quality.) Hence, in light of these assumptions, we decided to set a mechanism to increase the acquisition rate of a source by 8%, for every 10 positive detected SDOs from the streams of it, in order to capture more details by the source. (Note that we knew the acquisition rates would not cross a limit because as the acquisition rate of a source would be increased in some moments, it also would be decreased in some other times due to an increase in the acquisition rate of another source.) Also, based on specifications of the classifiers and our prediction about APPs, we estimated the initial η and μ of the ensemble as 0.72 s and 2.12 SDO/s, respectively. To satisfy the proposed conditions in Framework 2, by solving the optimization problem of the trade-off between increasing Δ and decreasing α, we chose $\lambda = 2.09$ SDO/s, $\alpha = 0.42$ SDO/s, and $\Delta = 11.92$ s, which made initial T^* be 0.41 s. We also set the initial P^F as $P^F(0) = \{0.04, 0.06, 0.15, 0.11\}$.

We used open-source projects: ML-Optimization [57] for the optimization tasks, Gauss project [58] for data analysis and predicting values, Tesseract [59] for OCR, and WebRTC [60] for data transmission between nodes.

6.3 Experimentation results and comparison

We performed 15 experiments based on the aforementioned scenario using different video lectures of the dataset to evaluate our approach. In all of these experiments, each iteration was set as 30 complete cycles. The utilities achieved in each iteration during the experimentations are shown in Fig. 5. Also, the details of the results are provided in the first row of Table 3. As Fig. 5 shows, almost all of the iterations achieved positive utility and an upward trend in gained utilities at iterations is clear in each experiment. Also, no crashes and overflows occurred during all the experiments. These were likely because significant amounts of data were shedding prior to reaching C_3 by C_2 and C_1 (because they were quickly detected as "not new scene" or "not slide"), which was also causing expected processing delay of the ensemble (σ) to be reduced significantly (for each of the sources). In consequence, the acquisition rates of sources were able to be increased, which resulted in capturing more details. More importantly, this was also accompanied by classifying SDOs by their own desired ensemble configuration, which made their classification utility be very close to the maximum.

We also carried out exactly the same experimentations on the state-of-the-art approach [5], which is without any time-sharing solution and buffers. Given our scenario and the specifications of the nodes, we encountered several problems. The first problem was about the classifier C_1. It did not have any memory, so it could not store pixel values of any source SDOs, in order to use them for comparing the next SDO of the source with it, to see if there is an enough difference or not. (Note that sources were sending SDOs simultaneously; therefore, in the state-of-the-art approach, after processing an SDO from a source, the next SDO that must be processed might not come from the same source.) Naturally, this led C_1 to be not practical for our scenario and made us get it out of the ensemble,[2] which caused a higher processing burden to be incurred by the subsequent classifiers in the ensemble. Furthermore, since in the sate-of-the-art approach, a general algorithm for detecting slides had to be used,

[2] This issue also demonstrated; if we implement some mechanism to compress data at the source and decompress it at the ensemble by using a data differencing method, the mechanism will not be applicable for the sate-of-the-art approach, because it cannot decompress SDOs due to no access to the value of their previous SDO. Even if a software solution for temporary storing values is implemented, retrieving the associated previous SDO's value from the stored values has an overhead, which can cause problems in large-scale systems. Therefore, in the sate-of-the-art approach compressing data can poorly be implemented, which cause SDOs to arrive at the ensemble with a higher delay.

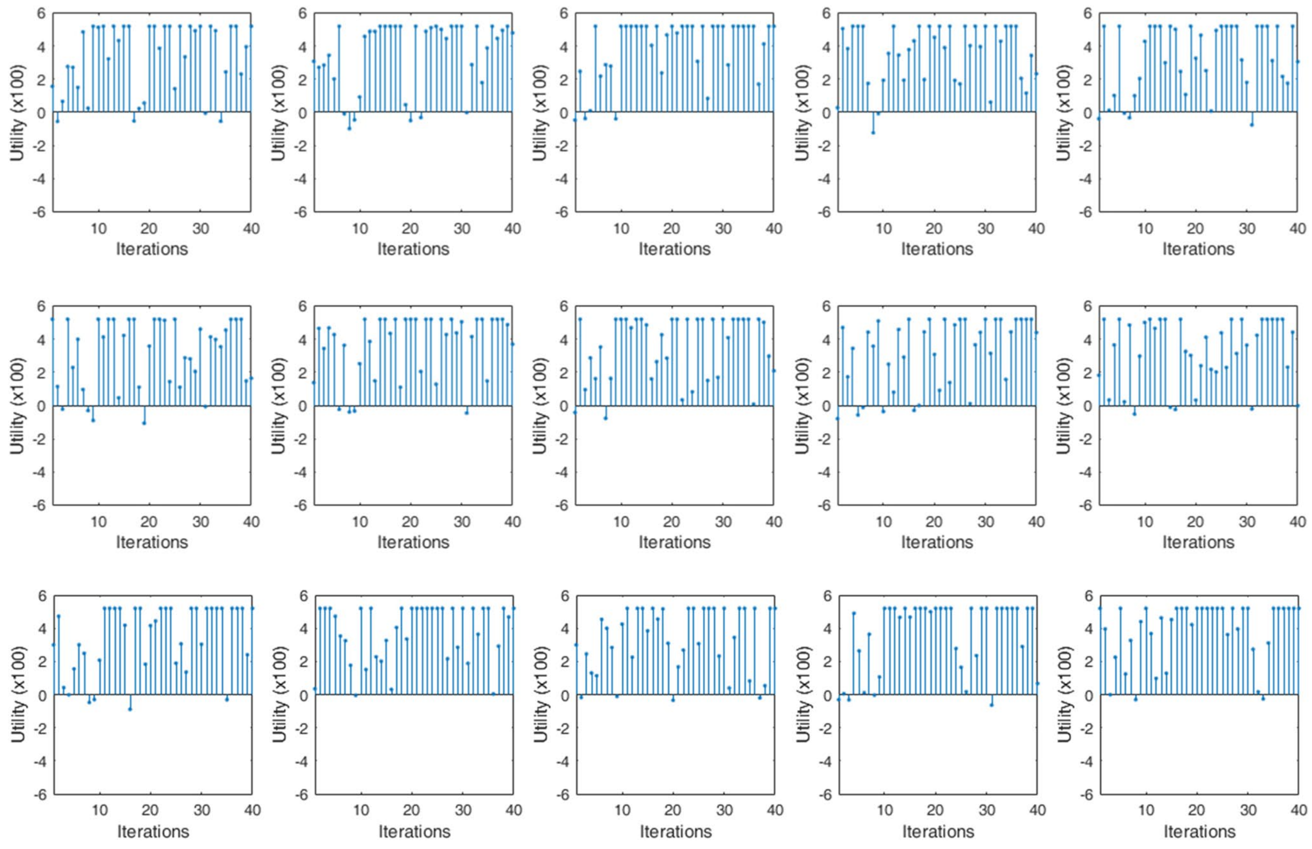

Fig. 6 Illustrative view of the achieved utilities in the state-of-the-art approach experimentations

Table 4 Comparison of the implementation results

Factors	Values
Number of the iterations where our approach gained higher average utility	267
Number of the iterations where the state-of-the-art approach gained higher average utility	270
Number of the experiments where the overall gained utility in our approach is higher	15
Number of the experiments where the overall gained utility in the state-of-the-art approach is higher	0

the average processing delay of C_2 was increased about 0.98 s, which caused the expected value of the end-to-end processing delay of the ensemble to be increased about 1 s, and the ensemble output rate to be decreased to 1.48 SDO/s. These issues caused several intolerable system failures during the experimentations. All of these failures occurred when some videos with a high number of continuous scenes of slides were playing. We noticed that the reason for the failures was because, when the number of detected SDOs of a source in the concept of interest was highly increased, the source acquisition rates of all the other sources were increased as well. This led the number of incoming SDOs to the ensemble to be increased significantly, which was together with more delayed classification methods. As a result, the capacity of the system would become full rapidly and lead to system failure.

This problem made us remove the method of increasing source acquisition rates when positive SDOs are detected and perform the experimentations again (although we knew that it would cause a loss of details). In the new experimentations, we also implemented a queue for each classifier with enough capacity to avoid any failure and set the fixed and identical acquisition rates of $\alpha = 0.09$ SDO/s for all the sources. The achieved utilities in each experiment are shown in Fig. 6, and the details of the results are provided in the second row of Table 3. As it is illustrated, in these experimentations, a considerable number of iterations achieved near-zero utility in each experiment. After investigation, we detected that the chief reason for this problem in most of the iterations was a decrease in $\psi(\tau)$ due to an increase in the SDOs wait time in the embedded queues before the classifiers. Also, besides this, erroneous learning and predicting values owing to fusion of SDOs were not something negligible.

The overall comparison between the implementation results of the two approaches is provided in Table 4. As this table explicates, the average of the gained utilities in each of the state-of-the-art approach experiments is less than the corresponding average in the results of our approach experiments. Moreover, according to Table 3, based on the average of the gained utilities during the whole experimentations, our approach has achieved a 13.45% superiority over the state-of-the-art approach, which along with the other statistics proves that our approach has outperformed the state-of-the-art in the implemented scenario.

7 Conclusion

In this paper, we proposed a distributed and adaptive approach for configuration of cascaded classifier topologies, in such a way that when several numbers of sources are sending data to the ensemble simultaneously, the system serves each stream data object (SDO) by the corresponding SDO optimal topology configuration. We defined a utility metric for the system performance, subject to delay constraints, on mining an SDO, given the SDO source. We showed that by adjusting acquisition rates of sources and implementing a queue for each source, in which the SDOs sent by the source are buffered, we can take advantage of a time-sharing solution to extend the approach for single-source cases to the cases where multiple sources are sending data. We also provided the conditions that must be satisfied prior to applying the time-sharing solution. By defining a term called stable cycle and investigating the conditions for fulfilling its requirements, we also succeeded in extending our approach to the adaptive source acquisition rate case. More importantly, we showed that our proposed time-sharing approach is roughly topology-free, which means it is applicable to all types of topologies (e.g., tree, directed acyclic graphs, etc.). Hence, it can be used in most of the multimedia processing applications and also other informationally distributed or restricted stream mining systems.

There are several directions for future research. One of the important directions involves detecting and solving synchronization issues among the classifiers. Another notable research area is to integrate the proposed time-sharing approach in this paper with other types of topologies such as trees and acyclic directed graphs model. Also, besides these, modeling and involving data privacy challenges in the optimization problem for stream mining systems would be a very interesting and important extension to explore in future.

Acknowledgements We would like to express our immense gratitude to Prof. Mihaela van der Schaar for her constructive and precious comments toward reviewing this paper.

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Canzian L, Schaar MVD (2015) Real-time stream mining: online knowledge extraction using classifier networks. IEEE Netw 29(5):10–16
2. Zhou P, Zhou Y, Wu D, Jin H (2016) Differentially private online learning for cloud-based video recommendation with multimedia big data in social networks. IEEE Trans Multim 18(6):1217–1229
3. Muller H, Unay D (2017) Retrieval from and understanding of large-scale multi-modal medical datasets: a review. IEEE Trans Multimed 19(9):2093–2104
4. Berriel RF, Lopes AT, Souza AFD, Oliveira-Santos T (2017) Deep learning-based large-scale automatic satellite crosswalk classification. IEEE Geosci Remote Sens Lett 14(9):1513–1517
5. Foo B, Schaar MVD (2010) A distributed approach for optimizing cascaded classifier topologies in real-time stream mining systems. IEEE Trans Image Process 19(11):3035–3048
6. Shah M, Hellerstein J, Franklin M (2003) Flux: an adaptive partitioning operator for continuous query systems. In: Proceedings of 19th international conference on data engineering, pp 25–36
7. Schapire Y (1999) A brief introduction to boosting. In: Proceedings of 16th international joint conference on artificial intelligence, pp 1401–1406
8. Catena M, Tonellotto N (2017) Energy-efficient query processing in web search engines. IEEE Trans Knowl Data Eng 29(7):1412–1425
9. Zanetti M, Jamhour E, Pellenz M, Penna M, Zambenedetti V, Chueiri I (2019) A tunable fraud detection system for advanced metering infrastructure using short-lived patterns. IEEE Trans Smart Grid 10(1):830–840
10. Lienhart R, Liang L, Kuranov A (2003) A detector tree for boosted classifiers for real-time object detection and tracking. In: Proceedings of ICME, pp 277–280
11. Kiourti A, Nikita KS (2017) A review of in-body biotelemetry devices: implantables, ingestibles, and injectables. IEEE Trans Biomed Eng 64(7):1422–1430
12. Ding X, Tian Y, Yu Y (2016) A real-time big data gathering algorithm based on indoor wireless sensor networks for risk analysis of industrial operations. IEEE Trans Ind Inf 12(3):1232–1242
13. Lee S-H, Chung CC (2017) Robust multirate on-road vehicle localization for autonomous highway driving vehicles. IEEE Trans Control Syst Technol 25(2):577–589
14. Foo B, Turaga DS, Verscheure O, Schaar MVD, Amini L (2011) Configuring trees of classifiers in distributed multimedia stream mining systems. IEEE Trans Circuits Syst Video Technol 21(3):245–258

15. Olston C, Jiang J, Widom J (2003) Adaptive filters for continuous queries over distributed data streams. In: Proceedings of international conference on management data, pp 563–574
16. Amini L, Andrade H, Eskesen F, King R, Park Y, Selo P, Venkatramani C (2005) The stream processing core. Technical Report, RSC 23798
17. Xing Y, Zdonik S, Hwang J-H (2005) Dynamic load distribution in the borealis stream processor. In: Proceedings of 21st international conference on data engineering, Tokyo, Japan, pp 791–802
18. Cherniack M, Balakrishnan H, Balazinska M, Carney D, Cetintemel U, Xing Y, Zdonik S (2003) Scalable distributed stream processing. In: Proceedings of conference innovative data system research, Asilomar, CA, pp 257–268
19. Cherniack M, Balakrishnan H, Balazinska M, Carney D, Cetintemel U, Xing Y, Zdonik SB (2003) Scalable distributed stream processing. In: Proceedings of 2nd Biennial CIDR
20. Schapire RE (1999) A brief introduction to boosting. In: Proceedings of 16th international joint conference on artificial intelligence, vol 2, pp 1401–1406
21. Garg A, Pavlovic V, Huang TS (2002) Bayesian networks as ensemble of classifiers. In: Proceedings of 16th ICPR, pp 779–784
22. Balazinska M, Balakrishnan H, Madden S, Stonebraker M (2005) Fault tolerance in the borealis distributed stream processing system. In: Proceedings of international conference management data (SIGMOD), pp 13–24
23. Babcock B, Babu S, Datar M, Motwani R (2003) Chain: operator scheduling for memory minimization in data stream systems. In: Proceedings of international conference management data, pp 253–264
24. Tatbul N, Cetintemel U, Zdonik S, Cherniack M, Stonebraker M (2003) Load shedding in a data stream manager. In: Proceedings of 29th international conference on very large databases, pp 309–320
25. Babcock B, Datar M, Motwani R (2003) Cost-efficient mining techniques for data streams. In: Proceedings of 2nd workshop Australasian information security, data mining, web intelligence, and software international, vol 32, pp 109–114
26. Tatbul N, Zdonik S (2006) Dealing with overload in distributed stream processing systems. In: Proceedings of IEEE international workshop network meets databases, Atlanta, GA, p 24
27. Tatbul N (2002) QoS-driven load shedding on data streams. In: Proceedings of EDBT Ph.D. Workshop, Prague, Czech Republic, pp 566–576
28. Eide V, Eliassen F, Granmo O, Lysne O (2003) Supporting timeliness and accuracy in distributed real-time content-based video analysis. In: Proceedings 11th ACM international conference on multimedia, pp 21–32
29. Turaga D, Verscheure O, Chaudhari U, Amini L (2006) Resource management for networked classifiers in distributed stream mining systems. In: Proceedings of 6th IEEE international conference on data mining, pp 1102–1107
30. Fu F, Turaga D, Verscheure O, van der Schaar M, Amini L (2007) Configuring competing classifier chains in distributed stream mining systems. IEEE J Sel Topics Signal Process 1(4):548–563
31. Kumar V, Cooper B, Schwan K (2005) Distributed stream management using utility-driven self-adaptive middleware. In: Proceedings of 2nd international conference on autonomic computing, pp 3–14
32. Horvitz E, Rutledge G (1991) Time-dependent utility and action under uncertainty. In: Proceedings of 7th conference on uncertainty artificial intelligence, pp 151–158
33. Fong S, Wong R, Vasilakos A (2015) Accelerated PSO swarm search feature selection for data stream mining big data. IEEE Trans Serv Comput 9:33–45
34. Rutkowski L, Jaworski M, Pietruczuk L, Duda P (2015) A new method for data stream mining based on the misclassification error. IEEE Trans Neural Netw Learn Syst 26(5):1048–1059
35. Mencagli G, Torquati M, Danelutto M, Matteis TD (2017) Parallel continuous preference queries over out-of-order and bursty data streams. IEEE Trans Parallel Distrib Syst 28(9):2608–2624
36. Kolomvatsos K, Anagnostopoulos C, Hadjiefthymiades S (2017) Data fusion and type-2 fuzzy inference in contextual data stream monitoring. IEEE Trans Syst Man Cybern Syst 47(8):1839–1853
37. Canzian L, Zhang Y, Schaar MVD (2015) Ensemble of distributed learners for online classification of dynamic data streams. IEEE Trans Signal Inf Process Netw 1(3):180–194
38. Poh N, Chan CH (2015) Generalizing DET curves across application scenarios. IEEE Trans Inf Forensics Secur 10(10):2171–2181
39. Narasimhan H, Agarwal S (2017) Support vector algorithms for optimizing the partial area under the ROC curve. Neural Comput 29(7):1919–1963
40. Makki B, Fang C, Svensson T, Nasiri-Kenari M, Zorzi M (2017) Delay-sensitive area spectral efficiency: a performance metric for delay-constrained green networks. IEEE Trans Commun 65(6):2467–2480
41. Argibay-Losada PJ, Yoshida Y, Maruta A, Kitayama K-I (2016) Optical versus electronic packet switching in delay-sensitive 5G networks: myths versus advantages. J Opt Commun Netw 8(11):B43–B54
42. Alinia B, Hajiesmaili MH, Khonsari A, Crespi N (2017) Maximum-quality tree construction for deadline-constrained aggregation in WSNs. IEEE Sens J 17(12):3930–3943
43. Li X, Wang C-C, Lin X (2017) Inter-session network coding schemes for 1-to-2 downlink access-point networks with sequential hard deadline constraints. IEEE/ACM Trans Netw 25(1):624–638
44. Ren S, Deligiannis N, Andreopoulos Y, Islam MA, Schaar MVD (2014) Dynamic scheduling for energy minimization in delay-sensitive stream mining. IEEE Trans Signal Process 62(20):5439–5448
45. Nash J (1950) The bargaining problem. Econometrica 18(2):155–162
46. Savage D, Zhang X, Chou P, Yu X, Wang Q (2017) Distributed mining of contrast patterns. IEEE Trans Parallel Distrib Syst 28(7):1881–1890
47. Boem F, Ferrari RMG, Keliris C, Parisini T, Polycarpou MM (2017) A distributed networked approach for fault detection of large-scale systems. IEEE Trans Autom Control 62(1):18–33
48. Hespanha JP, Naghshtabrizi P, Xu Y (2007) A survey of recent results in networked control systems. Proc IEEE 95(1):138–162
49. Mosaddegh A, Canizares CA, Bhattacharya K, Fan H (2017) Distributed computing architecture for optimal control of distribution feeders with smart loads. IEEE Trans Smart Grid 8(3):1469–1478
50. National Science Foundation of United States, Awards Database. https://www.nsf.gov/awardsearch/showAward?AWD_ID=1016081. Accessed 5 Jan 2017
51. United States Patent and Trademark Office. Patent Full-Text Databases, Patents Nos. 8990134, 8856051, 8819024, 8533134. http://patft.uspto.gov/netahtml/PTO/index.html. Accessed 5 Jan 2017
52. Shahkarami A, Bobarshad H, Bagherzadeh N. A stream-sensitive distributed approach for configuring cascaded classifier topologies in real-time large-scale stream mining systems. https://www.ravannevis.org/paper/2018/sma.html/. Accessed 15 Feb 2018
53. Google, Google I/O. https://events.google.com/io/. Accessed 28 Apr 2017
54. Massachusetts Institute of Technology, MIT OpenCourseWare. https://ocw.mit.edu/. Accessed 28 Apr 2017

55. Standford University, Stanford Online. http://online.stanford.edu/. Accessed 28 Apr 2017
56. Northeast Scala Symposium, NE Scala. http://www.nescala.org/. Accessed 28 Apr 2017
57. Zasso MCAB, Hernández JJ, Castillo A, Asencio MA. "mljs/optimization," *GitHub*. https://github.com/mljs/optimization. Accessed 05 May 2017
58. Galoso F, Tellis P, Fitzgerald B, Sam. "Gauss," GitHub, 01-Nov-2015. https://github.com/fredrick/gauss. Accessed 06 May 2017
59. Webster G, Kwok K, Hadley M, Zarco H, Suarez R. "tesseract.js," GitHub. https://github.com/naptha/tesseract.js. Accessed 05 May 2017
60. "WebRTC." https://webrtc.org/. Accessed 06 May 2017