Knowl Inf Syst (2012) 32:1–23
DOI 10.1007/s10115-011-0448-7

REGULAR PAPER

A clustering approach for sampling data streams in sensor networks

Alzennyr da Silva · Raja Chiky · Georges Hébrail

Received: 28 January 2011 / Revised: 1 July 2011 / Accepted: 22 October 2011 /
Published online: 9 November 2011

Abstract The growing usage of embedded devices and sensors in our daily lives has been profoundly reshaping the way we interact with our environment and our peers. As more and more sensors will pervade our future cities, increasingly efficient infrastructures to collect, process and store massive amounts of data streams from a wide variety of sources will be required. Despite the different application-specific features and hardware platforms, sensor network applications share a common goal: periodically sample and store data collected from different sensors in a common persistent memory. In this article, we present a clustering approach for rapidly and efficiently computing the best sampling rate which minimizes the Sum of Square Error for each particular sensor in a network. In order to evaluate the efficiency of the proposed approach, we carried out experiments on real electric power consumption data streams provided by EDF (Électricité de France).

Keywords Clustering · Data streams · Sampling · Sensor network

1 Introduction

The progressive advances in hardware and software technologies have enabled the generation, transmission and capture of data in very high rates in a wide range of fields. Examples of such fields include finance, network monitoring, telecommunications, web clickstream, manufacturing and sensor networks. The continuous arrival of data in multiple, rapid, time-varying

A. da Silva (✉) · G. Hébrail
BILab (Télécom ParisTech and EDF R&D Lab), LTCI-UMR 5141 CNRS,
INFRES-Télécom ParisTech, 46 rue Barrault, 75013 Paris, France
e-mail: alzennyr.gomes@telecom-paristech.fr

R. Chiky
ISEP-LISITE, Paris, France
e-mail: raja.chiky@isep.fr

G. Hébrail
ICAME-EDF R&D, Clamart, France
e-mail: georges.hebrail@edf.fr

and possibly unbounded streams appears to yield fundamentally new research problems. As research in data streams processing keep progressing, the problem of how to simply and efficiently expose, share and integrate data streams from various sources still remains a significant challenge [23]. Clustering techniques have been widely studied across several disciplines, but only a few of the techniques developed scale to support clustering of very large data. Over the past few years, a number of clustering algorithms for data streams have been put forth [15]. Five different algorithms for clustering data streams are surveyed in [19].

Unlike classical clustering approaches existing in the literature on data streams which are directly applied to raw data represented by sensor measurements, we propose in this article a new clustering approach for sampling multiple source data streams which is applied to the SSE (Sum of Square Errors) matrix. The elements of this matrix represent the error between the original and the interpolated measurement curve of a sensor when its data are sampled at a specific rate. In sensor networks, the rate at which data are collected at each node affects the communication resources and the computational load at the central server. The proposed approach seeks to attribute the best sampling rate for each sensor and to guarantee that the volume of transferred data fits a predefined bandwidth.

The article is organized as follows. Section 2 describes the related work. The formulation of the problem is presented in Sect. 3. Sections 4 and 5, respectively, describe temporal and spatial sampling techniques explored in this work. Section 6 details the sampling algorithm used in our approach. Section 7 presents our clustering approach for sampling data streams. The experimental framework, the implementation details and the result analysis are described in Sect. 8. Finally, concluding remarks and future work are discussed in Sect. 9.

2 Related work

In this section, we describe some related work on sampling and clustering of data streams.

2.1 Sampling

There are many applications where data are continuously produced by a large number of distributed sensors. Adaptive sampling has been developed for these applications to manage limited resources. They aim at conserving network bandwidth and storage by filtering out data that may not be relevant in the current context. The data collection rate becomes dynamic and adaptable to the environment.

Most existing adaptive sampling techniques sample data from each source (*temporal sampling*). An adaptive sampling scheme that adjusts data collection rates in response to the content of the streams was proposed in [12]. A Kalman filter is used at each sensor to make predictions of future values based on those already seen. The sampling interval (SI) is adjusted based on the prediction error. If the needed sampling interval for a sensor exceeds the maximum that is allowed by a specified SI range, a new SI is requested to the server. The central server delivers new SIs according to available bandwidth, network congestion and streaming source priority.

In [20], the authors present a feedback control mechanism which makes the frequency of measurements in each sensor dynamic and adaptable. Sampled data are compared against a model representing the environment. An error value is calculated on the basis of the comparison. If the error value is greater than a predefined threshold, then a sensor node collects data at a higher sampling rate; otherwise, the sampling rate is decreased. Sensor nodes are completely autonomous in adapting their sampling rate.

In [18], the authors present a method to prevent sensor nodes from sending redundant information; this is predicted by a sink node using an ARIMA prediction model. Energy efficiency is achieved by suppressing the transmission of those samples whose ARIMA-based prediction values are within a predefined tolerance value with respect to their actual values. A similar approach is proposed in [7]. Their results show that reduced communication between sensors and the central server can be sufficient by using an appropriate prediction model. A wide range of queries (including heavy hitters, wavelets and multi-dimensional histograms) can be answered by the central server using approximate sketches.

On the other hand, the authors of [25] use a *spatial sampling* technique called the "backcasting approach". Backcasting operates by first activating only a small subset of sensor nodes which communicate their information to a fusion center. This provides an estimate of the environment being sensed, indicating that some sensors may not need to be activated to achieve a desired level of accuracy. The fusion center then backcasts information based on the estimate to the network and selectively activates additional sensors to obtain a target error level. In this approach, adaptive sampling can save energy by only activating a fraction of the available sensors. A similar approach has been developed in [10] to compute the average of data in a sensor network. A subset of sensors is activated, and their data are used to make statistical inference on data from "inactive" sensors using a Bayesian approach. The algorithm developed is called *infer* and operates in two phases. During the first phase, a distributed algorithm decides which sensors should be active for the next time period based on their levels of energy (this is to increase the lifetime of the sensor network) and their historical data. The second phase occurs at the central server and uses Bayesian inference to predict the average data of all sensors. Even if authors were only interested in the average AVG to validate their approach, Bayesian inference has the advantage of being applicable to a wide range of aggregate queries.

In [21], the authors use Min-Wise sampling [5] to select uniformly and randomly a number r of sensors. The principle of this approach is as follows: given a hash function h applied to sensors ID, the central server selects the r sensors that have the minimum value computed by the hash function. This technique has the advantage of generating a random and uniform sample, and its principle is very close to the Bernoulli sampling. Another technique for sampling sources with spatially distributed data was proposed in [3]. The authors use a uniform random sample of nodes in a sensor network to aggregate data from the sensors. Their approach is based on ancillary data such as geographic data to determine the number of nodes to select.

The main issue that one may ask is: what strategy (temporal or spatial) has to be used to summarize distributed data streams? It is obvious that the answer depends on the application and treatment on data. We will try to give an answer to this issue later in this article as part of the aggregation query processing (especially to estimate SUM) over all or part of the distributed data streams. We also propose a strategy to combine these two approaches, in order to build estimators of aggregate data streams of better quality than if we had made a purely temporal sampling or purely spatial sampling. In addition, we will discuss the possibility of dynamically updating the sample in order to take into account the evolution of data.

2.2 Clustering

Clustering data streams is a well-studied problem in both theory and practice. One of the earliest and best-known clustering algorithms for data streams is BIRCH [26]. BIRCH is a heuristic that computes a preclustering of the data into so-called clustering feature vectors (microclustering) and then clusters this preclustering using an agglomerative bottom-up technique (macroclustering). BIRCH does not perform well because it uses the notion of radius or diameter to control the boundary of a cluster. STREAM [22] is the next main method which has been designed specially for data streams clustering. This method processes data streams in batches of points represented by weighted centroids which are recursively clustered until k clusters are formed. The main limitation of BIRCH and STREAM is that old data points are as important as new data points.

CluStream [2] is based on BIRCH. It uses the clustering feature vectors to represent clusters and expands on this concept by adding temporal features. This method takes snapshots at different timestamps, favoring the most recent data. A snapshot corresponds to q (where q depends on the memory available) micro-clusters stored at particular moments in the stream. New data points will be assigned to one of the micro-clusters in a previous snapshot if it falls within the maximum boundary of that micro-cluster. One weakness of the algorithm is the high number of parameters which depend on the nature of the stream and the arrival speed of elements.

StreamSamp [8] is based on CluStream and combines a memory-based reduction method and a time-based reduction module based on tilted windows. In StreamSamp, summaries with a constant size are maintained. The summaries cover time periods of varying sizes: shorter for the present and longer for the distant past. StreamSamp has the advantage of being fast on building the data streams summary. However, its accuracy degrades over time because old elements increase in weight for a given sample size.

Clustering of data streams continues to attract the attention of recent scientific work [19]. In [1], a holistic framework for density-based mining of data streams is presented. The author uses density estimation to compute cluster histograms which summarize data distribution profiles over different stream segments. In [16], the authors present ClusTree, a self-adaptive index structure which is able to provide a snapshot of the clustering model at different points in time by means of feature vectors which maintain values for computing mean and variance of data stream micro-clusters. In the supervised classification domain, [9] propose an architecture for ensemble-based classification to solve the data stream classification problem. In this architecture, the number of models involved in classification is upper-bounded and can dynamically change over time.

3 Problem formulation

The aim of this article is to address the problem of storing infinite data streams from multiple sensors on disk with limited storage space.

The architecture considered in this work consists of a distributed computing environment, describing a collection of N remote sensors that feed a Data Stream Management System (DSMS). We assume that sensor measurements are generated regularly at the same rate for every sensor. We cut out all streams into sampling periods (T)—also called temporal windows—of equal length. A typical sampling period corresponds to a length of one day. A set of measurements issued by a sensor during a sampling period is also referred as a *curve* from now on. A temporal window is composed of p measurements for each sensor (the number of measurements per sensor in the window is the same for all sensors). For each sensor, the number of measurements that have to be stored within a temporal window can vary from m (fixed parameter) to p. Let us define s as the maximum number of measurements which can be stored on the disk from the N sensors during a temporal window ($s < N * p$).

The problem consists of finding the best policy to summarize sensor curves within a time window which respects the constraints of the maximum number of data stored on disk (parameter s) and the minimum number of data taken from a sensor (parameter m). The goal is to answer to aggregate queries using the summary. Each curve r takes the value $C_r(t)$ at time t. In this article, we concentrate on SUM aggregate queries:

$$C(t) = \sum_{r \in P(N)} C_r(t)$$

where $P(N)$ is a subset from $\{1, 2, \ldots, r, \ldots, N\}$.

The data collection model should preserve detailed information as much as possible by reducing summarizing errors. Several criteria can be used to measure errors. In this article, we use the Sum of Square Errors (SSE), which is also the square of the L_2 distance between the original curve $C = \{c_1, c_2, \ldots, c_p\}$ and an estimation from the summarized curve $\hat{C} = \{\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_p\}$. SSE is computed as follows:

$$\mathrm{SSE}(C, \hat{C}) = \sum_{i=1}^{p} (c_i - \hat{c}_i)^2$$

Other error measures can be defined depending on the application. SSE is well adapted to electric power consumption [13].

4 Temporal sampling

Temporal sampling is the process of keeping fewer measurements from a curve, preserving the underlying information as much as possible. Note that if no temporal sampling is applied, then aggregating the whole population curves at each timestamp t consists of summing all values $C_r(t)$ measured on each curve r in the following way:

$$C(t) = \sum_{r=1}^{N} C_r(t).$$

From the observation of what is happening during sampling period $T - 1$, we determine a data collection model to apply to the N sensors on the next sampling period T. During a sampling period T, sensors send data to the DSMS. The engine computes on the fly SSE values for different assumptions of sampling rates between m and p values. It also summarizes data streams at the scheduled sampling rate and stores summarized data on a database. Then, the application solves an optimization problem to find the best sampling rate for each sensor during sampling period $T + 1$.

Several sampling schemes can be considered to summarize data streams during a sampling period [6]. Almost all methods of curve compression can be used depending on the handled data. For the needs of our application, we concentrate on three methods: regular sampling, curve segmentation and wavelet compression.

– **Regular sampling**: Curves are sampled using a step j chosen between 1 and maxStep $= \left\lfloor \frac{p}{m} \right\rfloor$ ($\lfloor x \rfloor$ is the floor function). When $j = 1$, all measurements are selected, when $j = 2$ every two measurement is selected and so on. This technique is described as *regular* because selected points are temporally equidistant, and a jump j is made between two sampled measurements. For the estimation of the original curve from the sampled measurements, linear interpolation is used. Let c_a and c_b be two measurements of a same curve at different timestamps a and b, and an intermediate measurement c_i (where $a < i < b$)

is obtained in the following way:

$$\hat{c}_i = \left(\frac{b-i}{b-a}\right) c_a - \left(\frac{a-i}{b-a}\right) c_b.$$

- **Curve segmentation**: Segmentation is used to extract information from time series using a *piecewise representation*. The goal in segmentation is to decompose the sequence into a small number of straight lines (segments), such that data in each segment can accurately be associated with a numerical value (average here), the number of segments is fixed in advance. The segments could have different size. Segmentation has been considered in many areas, starting from the classic article of Bellman [4]. Many other formulations exist, and the problem has been studied extensively in various settings. There exists a variety of segmentation algorithms which are enumerated in [11,14]. The algorithm used in this framework is the optimal algorithm based on dynamic programming [4]. In fact, the total error is the sum of independent square errors. The optimal segmentation in k segments consists of a first segment followed by the optimal segmentation in $k-1$ segments starting from the end of the first segment. More details of the segmentation algorithm can be found in [11].
- **Wavelet compression**: Wavelet coefficients are projections of the given signal (set of data values) onto an orthogonal set of basis vectors. The choice of basis vectors determines the type of wavelet. Haar wavelets are often used for their ease of computation. The synopsis is a linear combination of j basis vectors ($j < m$). The j numbers are chosen to minimize a suitable error between the original data and the estimate derived from the synopsis. The minimum number of data points in an input signal should be 2 in the case of the Haar wavelet, and the number of data points needed for n times decomposition is 2^n. If the number of input data points is less than this required number, 2^n, zeros may be padded (appended) at the right end of the input data.

At the end of each sampling period T, different values of SSE are computed which correspond to different levels of summarization, indexed by j varying from 1 to maxStep. These SSE values are stored in a matrix $W_{N*\mathrm{maxStep}}$ of N rows and maxStep columns (N is the number of sensors connected to the server). Element w_{ij} of the matrix corresponds to the SSE obtained when collecting data from sensor i with a summarizing level j.

Instead of equally distributing the summarizing levels between all sensors for sampling period $T+1$, an optimization is performed to assign different sampling rates to sensors from the SSE values computed during sampling period T. This problem can be formalized as an optimization problem minimizing the sum of SSEs on sensors:

$$\text{Minimize} \sum_{i=1}^{N} \sum_{j=1}^{\mathrm{maxStep}} (W_{ij} \times X_{ij})$$

subject to:

$$\begin{cases} X_{ij} = 0 \text{ or } 1 \\ \sum_{j=1}^{\mathrm{maxStep}} X_{ij} = 1 \\ \sum_{i=1}^{N} \sum_{j=1}^{\mathrm{maxStep}} \left(\left\lfloor \frac{p}{j} \right\rfloor \times X_{ij} \right) \leq s \end{cases}$$

This is a problem of assignment of sampling rates to the sensor curves respecting the constraints above. A variable $X_{ij} = 1$ means that the sampling step j is assigned to the curve of index i. The second constraint $\sum_{j=1}^{\text{maxStep}} X_{ij} = 1$ imposes only one value of j per curve. Lastly, the third constraint means that the number of data to be stored should not exceed the threshold s. To solve this problem, we used the simplex method applied to linear problems with real variables. The simplex method is coupled with the Branch-and-bound method to reach integer values [17]. Once the optimization problem is solved, the optimal sampling rates are sent to each sensor for sampling.

4.1 General approach with summarizing method selection

At time $t = 0$, the DSMS has no knowledge of sensor data. It requests sensors to send $\lfloor \frac{s}{N} \rfloor$ values all throughout the first sampling period. At the end of this period, each sensor computes a total of maxStep SSEs for each summarizing method and sends the minimum of them to the central server. The DSMS computes the whole error matrix $W_{N*\text{maxStep}}$ and applies the optimization module to find j corresponding to: (i) sampling steps in the case of regular sampling, (ii) the double of the number of segments in the case of segmentation or (iii) the number of wavelet coefficients in the case of compression. It then sends the result of optimization to the sensors. If the SSE by applying a step of sampling j to sensor i is lower than the SSE by segmenting the curve into $\left\lfloor \frac{p}{j\times 2} \right\rfloor$ or compressing using $\left\lfloor \frac{p}{2^j} \right\rfloor$ wavelet coefficients, then the sensor decides to apply a regular sampling and sends the samples to the DSMS. If SSE by segmenting the curve is the smallest, the sensor sends the averages of the segments with the number of points constituting the segment; otherwise, it sends the wavelet coefficients. At the end of the next sampling period, the sensors send the SSEs to update the matrix of errors in the same way. This process continues as long as data streams arrive on line.

4.2 Experimental validation

Having described our approach, we now seek to evaluate its effectiveness by comparing it against a standard non-adaptive approach that we call *fixed* approach. It means that each sensor divides the total number of elements it can perform in a sampling period T equally. Specifically, we are interested in comparing *empirically* the cumulative SSE gathered in the central server at each sampling period.

Experiments were carried out on two real data sets from electric power industry. Each sensor corresponds to a real electricity meter installed at a household by the French energy group (EDF). Each meter produces a curve representing the electric power consumption of one customer, called “load curve”. The first data set is produced by a network composed of 1,000 m. Each meter produces one measurement every 10 min during one day (144 measurements per meter per day). It is used to assess the efficiency of the approach in the case of a stationary consumption for each customer: the optimized sampling rates are assessed on the same curve which is used to choose them. This experiment will be referred as “stationary window”. The second data set is produced by a network composed of 140 m each producing one measurement every 30 min during one year (365 days). It is used to assess the efficiency on a non-stationary consumption for each customer: the optimized sampling rates are assessed on the curve of a different day than the one used for choosing them. This second experiment will be referred as “jumping window”.

4.2.1 Stationary window

In our experiments, we used $m = 7$ (maxStep $= \lfloor \frac{144}{7} \rfloor = 20$), it means that sampled curves will consist of at least 7 values if it is built regularly, of 3 values if it is built with segmentation

Table 1 Experimental results

Threshold s	144,000	72,000	48,000	28,800	20,571	14,400	9,600
SSE opt samp LI	0	434.35	996.34	1,969.67	2,781.54	3,883.9	5,676.6
SSE opt seg	0	161.62	385.13	874.1	1,346.58	2,027.17	3,432.16
SSE opt wav	0	381.15	916.22	1,881.66	2,724.08	3,888.6	6,118.8
SSE fix LI	0	1,956	2,821	4,245	5,501	7,024	9,087
SSE fix seg	0	505.8	860.8	1,535	2,042.6	2,761.5	4,677.1
SSE fix wav	0	1,385.32	2,725.35	4,766.09	4,766.09	7,179.5	7,179.5
Avg data/meter	144	72	48	28.8	20.57	14.4	9.6
Std data LI	0	51	45.45	33.6	24.7	16.5	4.9
Std data seg	0	43	35.73	25.32	17.64	10.6	3.9
Std data wav	0	48.9	42	30	21.8	13.5	3

SSE sum square error. *Opt* sampling steps (#segments or #wavelet coefficients) obtained by optimization, and *fix* same sampling step (same #segments or #wavelet coefficients) for all curves. *Avg* average and *std* standard deviation. *Samp LI* regular sampling with linear interpolation, *seg* curve segmentation and *wav* wavelet compression

and of 9 coefficients with wavelet compression. In these experiments, we used the first data set. We point out that a load curve in our example consists of 144 points. We tested several values of threshold, s and we drew up a summary table of the results (Table 1). Each column of the table corresponds to a value of threshold s. $s = 144{,}000 = 144 * 1{,}000$ means that we collect the whole curves. We experiment thresholds equal to the total number 144,000 divided by each of $\{1, 2, 3, 5, 7, 10, 15\}$. For instance, $s = 48{,}000$ is the third of total number of data $\left(\frac{144{,}000}{3}\right)$.

We have computed empirically the Sum Square Errors for each method and the standard deviation to measure the dispersion of number of data computed by optimization around the average which is equal to $\frac{s}{1{,}000}$.

As can be seen in Table 1, the optimization performs well for all summarizing methods, it considerably reduces the Sum Square Errors compared to "fixed" approach. For instance, in the case of regular sampling with Linear Interpolation, the optimization reduces the SSE by about 77% for the threshold $s = 72{,}000$ compared to a"fixed regular sampling".

We observe high values of the standard deviations mainly for high values of threshold s. The sampling rates are significantly deviated from the average. Figure 1 is an example of sampling steps distribution ("Linear Interpolation") for a threshold of $s = 28{,}800$. We notice that sampling steps assigned by optimization are most concentrated on limits ($j = 1$ and $j = 20$), whereas fixed step sampling is $j = 5$. This fact confirms the importance of selecting sampling steps by optimization. The same observation is verified for all summarizing methods.

4.2.2 Jumping window

Previous experiments evaluate the error made over a sampling period from an optimization carried out over the same period. In the current experiments, we used the second data set. Our approach consists of summarizing curves over a sampling period based on optimization results carried out over a previous period. We present here the method adopted for a streaming environment as well as first experiments completed on 140 load curves available over one-year period.

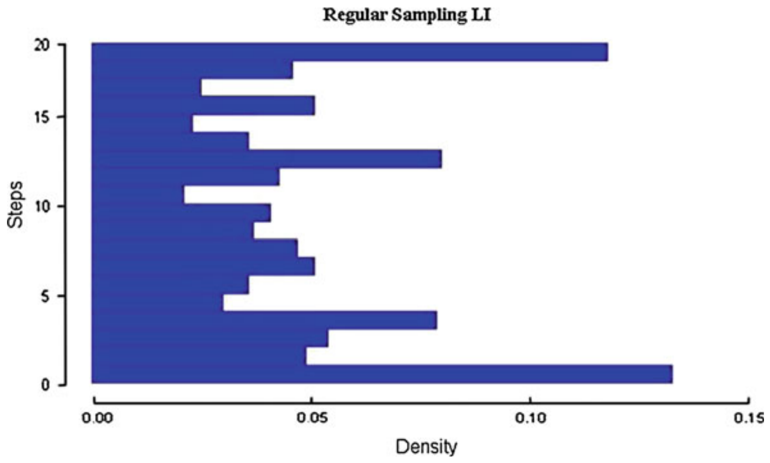

Fig. 1 Sampling steps distribution

Table 2 Percentage of days whose SSE with optimization is better than "fixed" methods

Threshold s	6,720	3,360	2,240	1,344	960	672	448
$\%d$ < 10% SSE LI opt-1	0	84%	84%	84%	85%	85%	85%
$\%d$ < 10% SSE esc opt-1	0	84%	84%	85%	86%	97%	100%
$\%d$ < 10% SSE seg opt-1	0	83%	83%	84%	84%	84%	84%
$\%d$ < 10% SSE wav opt-1	0	84%	85%	94%	89%	100%	100%

Opt-1 optimization at day $d - 1$ and sampling at day d. $\%d$ < 10% SSE corresponds to the percentage of days whose SSE opt-1 is lower by 10% than the SSE obtained by "fixed" methods

Optimization using d-1 We used a data set of 140 load curves measured every 30 min over one year (365 days) to validate the efficiency of our proposed data collection scheme. Table 2 gives the percentage of periods (temporal windows) for which the SSE using optimized summarizing is lower than the SSE using "fixed" summarizing. The length of the temporal window used for the experiments is one day. The optimization stage uses data of the previous day, and resulting steps j are applied to sample data of the current day. Results in Table 2 show that our approach is very efficient. In fact, the optimized summarizing methods decrease SSE by at least 10% compared to the fixed methods of more than 83% of all days and for all tested threshold values.

Optimization using d-7 Previous experiments are done on data of day $d - 1$ to summarize data of the day d. Figure 2 shows the evolution of SSEs during one month (30 days) for a threshold $s = 1{,}344$. We observe a periodicity on the curve corresponding to the evolution of Sum Square Errors by applying the sampling rates obtained by optimization during the previous day (curve opt-1). This is due to the weekly cycle of electric consumption. In fact, the consumption remains generally stable during 5 working days, and the consumption level changes on weekends. In our approach, we determine the sampling rates for Monday based on data of Sunday and those of Saturday based on data of Friday. We experiment with the case where the sampling stage of the current day d is based on sampling rates computed from day $d-7$. Result is illustrated in curve opt-7 of Fig. 2 (curve opt-7). We notice that this

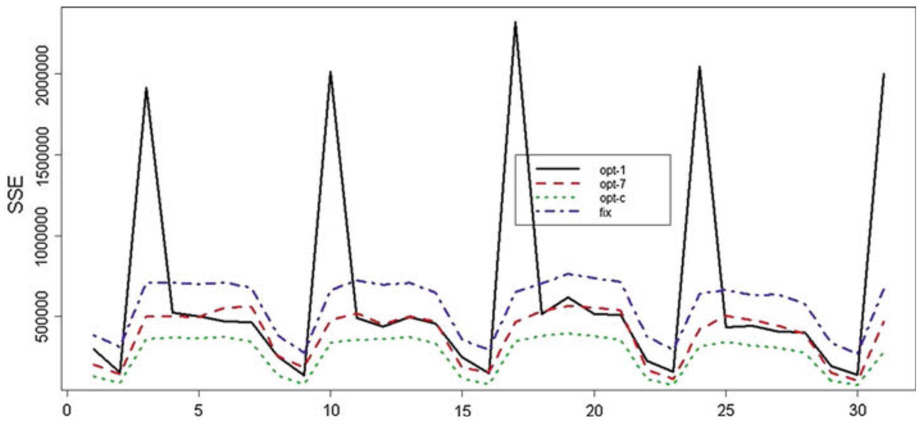

Fig. 2 Evolution of SSEs during 30 days. Opt-7: optimization at day $d - 7$ and sampling at day d. Opt-1: optimization at day $d - 1$ and sampling at day d. Opt-c: optimization and sampling at day d (current day). Fix: fixed regular sampling

method makes it possible to *smooth* the curve of SSE evolution and thus to decrease errors due to the periodicity of electric consumption.

Optimization using w-1 We experiment with a period t of length equal to one week, and we follow the evolution of SSEs during 52 weeks. Table of results is not reported in this article by lack of space, but the approach makes it possible to decrease the summarizing errors compared to fixed summarizing for more than 96% of all weeks, all thresholds and all summarizing methods. Figure 3 shows errors evolution during a year period and using a week as temporal window with threshold $s = 9,408$. Unlike the optimization using $d - 1$, there is no cycle in the evolution of errors, and Sum Square Errors using optimized sampling are clearly reduced for all weeks than fixed sampling method. Figure 4 depicts boxplots that provide a visual summary of many important aspects of Sum Square Error distribution over the year for fixed and optimized regular sampling, using a threshold of $s = 9,408$. We can clearly see in Fig. 4 that optimized regular sampling performs better than fixed regular sampling.

To ensure that the results provided by the optimized sampling method and by the non-adaptive method are statistically different, we performed the Student's T test with a confidence interval of 95%. We first realize a Fisher's F test that allowed us to verify the non-equality of variances of the two data sets analyzed (364 days or 51 weeks). The tests were performed for all sampling methods presented in this paper by varying the parameters (threshold s and the sampling period). The tests give all the same results. Table 3 shows an excerpt of the results for regular sampling with linear interpolation. As we can see in this table, p-values obtained are less than 0.05 for all the performed tests. This confirms that we can reject the null hypothesis H_0 of equality of SSEs means for the optimized sampling method and the non-adaptive method.

5 Spatial sampling

Spatial sampling is the process of selecting an appropriate set of sensors so that the sample allows estimating unknown quantities (queries) of the population (the N sensors). This is similar to a standard sampling survey problem.

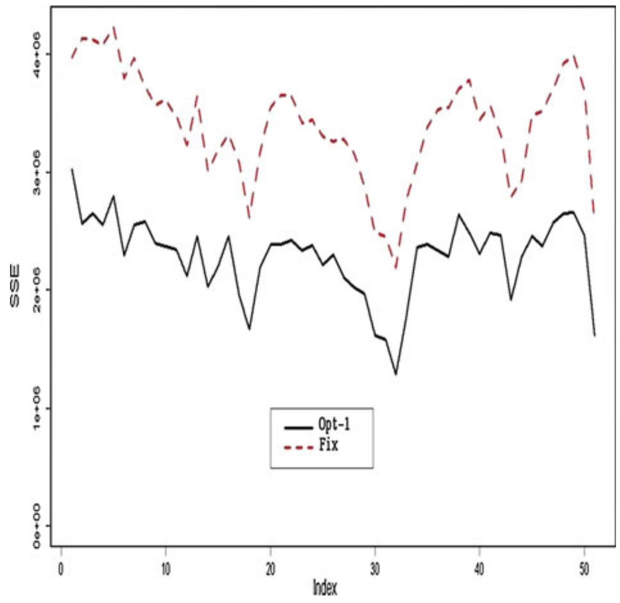

Fig. 3 Evolution of SSEs during one year. Opt-1: optimization at week $w-1$ and sampling at week w. Fix: fixed regular sampling

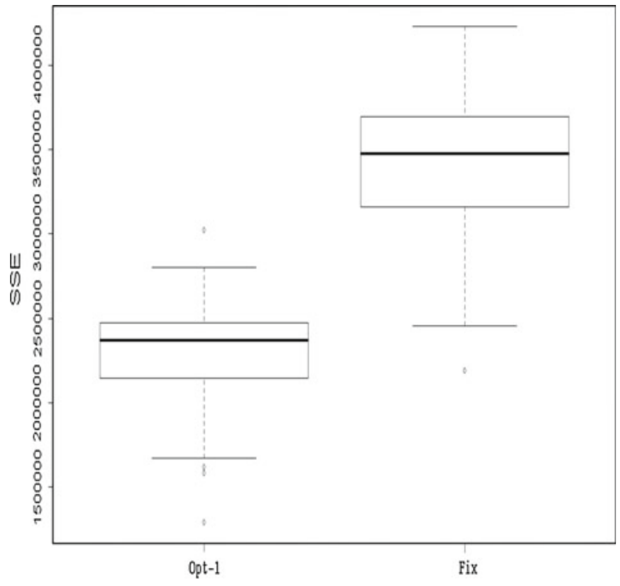

Fig. 4 The distribution of SSEs ($s = 9,408$).opt-1: Optimization at week w-1 and regular sampling (LI) at week w. fix.: fixed regular sampling

We consider a finite population (the N sensors). Each sensor can be identified by a label. Let $\{1, 2, \ldots, r, \ldots, N\}$ be the set of these labels. The aim of survey sampling is to study a variable of interest which is the curve in our context. The curve takes the value $C_r(t)$ for a sensor r at instant t. The purpose is more specifically to estimate a function of interest of

Table 3 Student's T test for $d-1$ (364 days) and $w-1$ (51 weeks), sampling method = regular sampling with linear interpolation, confidence interval = 95%

Test set	Set size	p value	95% Confidence interval	Mean SSE opt	Mean SSE fix
d-1_LI_s2100	364	p value < 2.2e-16	[−12,99,315, −11,44,825.0]	497,349.2	1,719,419.1
d-1_LI_s700	364	p value < 2.2e-16	[−4,83,642.8, −4,50,992.0]	17,708.33	485,025.7
d-1_LI_s280	364	p value < 2.2e-16	[−1,51,702.5, −1,43,742.5]	544.2718	148,266.8162
w-1_LI_s2100	51	p value < 2.2e-16	[−1,10,132,96, −79,75,031]	42,304,101	3,724,574
w-1_LI_s700	51	p value < 2.2e-16	[−41,25,256, −34,83,674]	190,075.5	3,994,540.6
w-1_LI_s280	51	p value < 2.2e-16	[−12,01,194, −10,37,923]	7,439.395	1,126,998.033

the C_r's for all or a subset of sensors. Note that we concentrate on SUM aggregate queries in this article.

There are many strategies of sampling in literature: simple random sampling, stratified sampling, sampling with unequal probabilities, balanced sampling, etc. (for details, see [24]). We use the simple random sampling in our framework. Let S be a sample containing n sensors, π_r be the inclusion probability, i.e., the probability that sensor r is in the sample S, π_{rq} be the second-order inclusion probability, i.e., the probability that both sensors r and q belong to the sample S. We use the simple random sampling in our framework where the same inclusion probability π_r is assigned to each sensor. An unbiased estimator often used is the Horvitz-Thompson estimator. We can calculate the SUM estimate of curves as follows:

$$\hat{C}_{HT}(t) = \sum_{r \in S} \frac{C_r(t)}{\pi_r}.$$

The variance of this SUM can be estimated by:

$$\hat{\mathrm{Var}}(\hat{C}_{HT}(t)) = \sum_{r \in S} \sum_{q \in S} \left(\frac{C_r(t)}{\pi_r} \frac{C_q(t)}{\pi_q} \frac{\pi_{rq} - \pi_r \pi_q}{\pi_{rq}} \right)$$

In the case of simple random sampling, the inclusion probabilities are $\pi_r = \pi_q = \frac{n}{N}$ and $\pi_{rq} = \frac{n(n-1)}{N(N-1)}$.
The confidence intervals with a confidence level of $(1-\alpha)$ are estimated as follows:

$$CI_{(1-\alpha)}(t) = \Big[C_{HT}(t) - z_{1-\alpha/2}\sqrt{\hat{\mathrm{Var}}(\hat{C}_{HT}(t))}, \\ C_{HT}(t) + z_{1-\alpha/2}\sqrt{\hat{\mathrm{Var}}(\hat{C}_{HT}(t))} \Big]$$

where $z_{1-\alpha/2}$ is the $(1-\alpha/2)$-quantile of a standard normal random variable.

6 MASTER algorithm

In both sampling techniques above (temporal sampling and spatial sampling), summarized estimates are used to answer the SUM aggregate queries. Interpolation in the first case estimates affected by sampling error in the second case. We propose a method that combines these two approaches in order to reduce summarizing errors as much as possible. We call this "Method for Adaptive Spatio-Temporal summaRies" or MASTER.

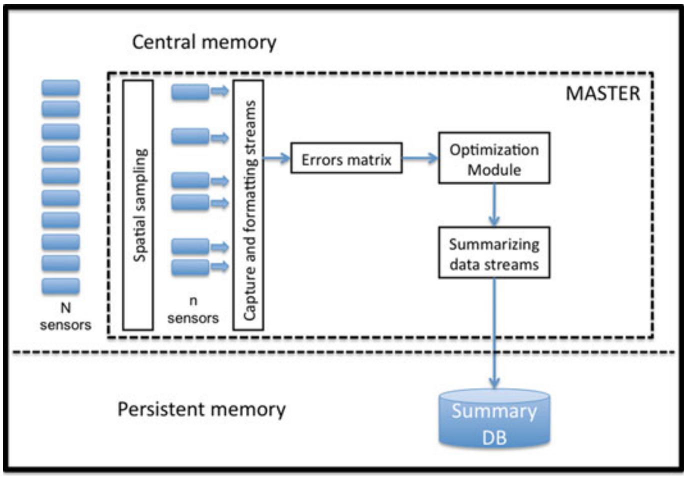

Fig. 5 Global architecture of MASTER

The principle of the spatio-temporal approach is represented in Fig. 5 and is as follows: A survey technique is integrated into the temporal sampling approach to make up a spatio-temporal summary. As with traditional approaches, a survey plan is set up to select a sample with a size of n sensors from the whole population, i.e., the N sensors. Then, we apply a sampling method to collect measurements at variable granularities in time for each sensor in the spatial sample and store them in a database. For the needs of our application, we concentrate on one method: regular sampling.

For the three sampling methods presented in Sect. 4, the worst case complexity of computing the errors for different summarizing levels is $O(mp)$ (curve segmentation). However, we can reduce computational overhead for some summarizing methods. In the case of regular sampling (complexity $O(1)$), errors can be computed incrementally. Indeed, to update the matrix of errors in the case of regular sampling, each sensor should send m SSEs (we recall that maxStep is the lower limit of the sampling step). However, it is not necessary to send all maxStep SSEs. Indeed, for a curve sampled at a step j, the sensor sends maxStep $- M_j$ SSEs, M_j is the number of multiples of j that are lower than maxStep. The DSMS collect sampled data and can therefore compute the missing SSES to update the matrix. For example, if a curve is sampled at steps from $j = 2$ to maxStep $= 20$, the sensor sends at the end of the sampling period maxStep $- M_2 = 20 - 10 = 10$ SSEs as $M_2 = \text{card}(\{2, 4, 6, \ldots, 20\}) = 10$. In our experiments, for a threshold $s = 14{,}400$ and maxStep $= 20$, each sensor sends an average of 9 SSEs instead of 20. Even if the segmentation is the most efficient summarizing method for power consumption curves, we apply regular sampling in MASTER because of its low complexity.

The sensors that are not part of the spatial sample are not observed. The curves in the temporal sample are reconstructed by linear interpolation. The remaining curves are estimated with the average curve of the sample. Algorithm 6.1 describes MASTER steps.

Algorithm 6.1 MASTER

```
Input: $T, maxStep, s$
Output: $Sample$
```

Step 1: Apply the spatial sampling to the whole population (the N sensors) and select n sensors
Step 2: Define sensor curves by reading all measurements sent by the n selected sensors during sampling period T
Step 3: Compute SSE matrix ($W_{n*maxStep}$) from curves of the n selected sensors
Step 4: Compute the best sampling rates respecting threshold s for each sensor by applying the linear programming solver to the SSE matrix
Step 5: Apply the temporal sampling to the n selected sensors on sampling period $T+1$ with the sampling rates from period T

7 CLUSMASTER

Clustering techniques group individuals based only on information found in the data describing the individuals and their relationship. The goal is that the individuals in a cluster be similar to one another and different from the individuals in other clusters. The greater the difference between groups and the similarity (or homogeneity) within a group, the better or more distinct the clustering is.

In this section, we present CLUSMASTER (CLUStering on MASTER). The goal of our approach is to apply a clustering technique on the SSE matrix before computing the best sampling rates for each sensor in a network. This is done in order to reduce the execution time and the physical resources (e.g., CPU power) allocation required by the MASTER algorithm. In the SSE matrix $W_{N*\mathrm{maxStep}}$, each row represents a sensor, each column represents a sampling rate in the interval [1, maxStep], and each cell w_{ij} represents the SSE value obtained when the sampling rate j is applied to the sensor i. In our approach, we have analyzed two clustering strategies described as follows.

7.1 Random clustering

In this strategy, all sensors are randomly assigned to k clusters. The sensors are equally distributed among the clusters. The best sample rates are then computed within each cluster by means of the linear programming solver (cf. Fig. 6). The total number of sensors in each cluster is represented by n_c, where $\sum_{c=1}^{k} n_c = N$. The allowed bandwidth for each cluster C is proportional to n_c.

In order to avoid unstable solutions introduced by the random nature of this strategy, the clustering process is repeated a predefined number of times. Notice that when $k = 1$, the solution found by this strategy corresponds to the one found by the MASTER algorithm with no clustering.

7.2 Clustering based on a dissimilarity criterion

In this strategy, we apply to the SSE matrix a clustering method which minimizes a dissimilarity criterion. The dissimilarity criterion corresponds to a distance computed among all the sensors. In our experiments, we investigate three different dissimilarity criterions: Manhattan distance, Euclidean distance and Maximum distance, detailed in Table 4. The clustering method applied is the Hierarchical Ascendant Clustering (HAC). The HAC was chosen because it is independent of the order in which data are presented, and besides that, it can provide a clustering order depicted in a dendrogram which can be useful for interpretation purposes. Furthermore, unlike partitioning clustering methods, HAC is determinist and

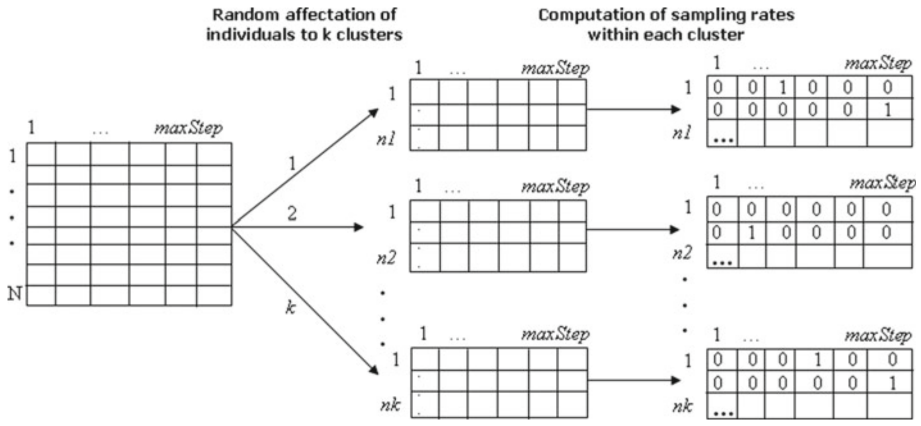
Random affectation of individuals to k clusters
Computation of sampling rates within each cluster
1 ... maxStep
1
N
1
2
k
n1
n2
nk
0 0 1 0 0 0
0 0 0 0 0 1
0 0 0 0 0 0
0 1 0 0 0 0
0 0 0 1 0 0
0 0 0 0 0 1

Fig. 6 Schema of the random clustering

Table 4 Dissimilarity criterions

Dissimilarity criterion	Formula
Manhattan distance	$\mathrm{dist}(w_a, w_b) = \sum_{j=1}^{\mathrm{maxStep}} \lvert w_{aj} - w_{bj} \rvert$
Euclidean distance	$\mathrm{dist}(w_a, w_b) = \sqrt{\sum_{j=1}^{\mathrm{maxStep}} (w_{aj} - w_{bj})^2}$
Maximum distance	$\mathrm{dist}(w_a, w_b) = \max_{j\in\{1,\ldots,\mathrm{maxStep}\}} \lvert w_{aj} - w_{bj} \rvert$

independent of local minima. HAC works in a bottom-up fashion grouping small clusters into larger ones. The procedure finishes when all observations belong to a same cluster (cf. Algorithm 7.1). The final number of clusters (k) is defined by the level at which the dendrogram is cut.

Algorithm 7.1 Hierarchical Ascendant Clustering (HAC)

```
Step 1: Start with each point in a cluster of its own
Step 2: Until there is only one cluster
    – Find the closest pair of clusters
    – Merge them
```

In addition to choosing a dissimilarity criterion to use when comparing two sensors, it is also necessary to define what should be assessed when two clusters containing more than one sensor are compared. In our approach, this is done by the Ward's method which joins the two clusters that minimize the Sum Square Error (SSE)—defined as the sum of the squared distances of sensors from the center of gravity of the cluster to which they have been assigned [cf. Eq. (1) where p_c is the center of cluster c, and n_c is the number of observations in it. Δ is called the merging cost of combining the clusters A and B].

$$\Delta(A, B) = \frac{n_a n_b}{n_a + n_b}\mathrm{dist}(p_a, p_b)^2 \tag{1}$$

Initially, the SSE starts out at zero (because every point is in its own cluster) and then grows as clusters are merged. Ward's method keeps this growth (Δ) as small as possible.

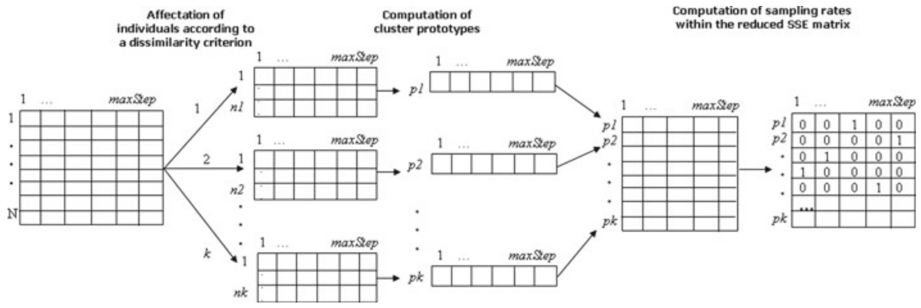
Affectation of individuals according to a dissimilarity criterion
Computation of cluster prototypes
Computation of sampling rates within the reduced SSE matrix
1 ... maxStep
N
p1
p2
pk
n1
n2
nk

Fig. 7 Schema of the clustering based on a dissimilarity criterion

Once the execution of HAC finishes, the representative (center or prototype) of each final cluster is computed by summing up the SSE of all sensors belonging to a same cluster. The prototype of a cluster C is represented by a maxStep-tuple $p_c = (p_c^1, \dots, p_c^j, \dots, p_c^{\text{maxStep}})$, where $p_c^j = \sum_{i \in C} w_{ij}$. All tuples belonging to a same cluster in the SSE matrix are then replaced by the prototype of the corresponding cluster. The linear programming solver is applied to the reduced SSE matrix containing k rows and maxStep columns (cf. Fig. 7). The sampling rate corresponding to a particular cluster is assigned to each sensor belonging to the cluster.

Unlike the random clustering strategy which finds the optimal solution for the sampling rates within each cluster, the solution found by means of the clustering strategy based on a dissimilarity criterion is approximate. In this strategy, as sensors belonging to a same cluster have similar SSE values, it is expected that the same sampling rate will fit all sensors in a cluster.

8 Experimental framework

The parameters used in the experiments are specified as follows. The number of clusters k is in the range $[1, 50]$, maxStep is equal to 10, the bandwidth is equal to 50% which means that the concentrator accepts half of total measurements at maximum, the number of repetitions of the random clustering is equal to 50, and parameter T is equal to 1, i.e., the time window has length equal to one day.

In our experiments, we applied three strategies described as follows:

1. MASTER algorithm with no clustering;
2. MASTER algorithm with random clustering;
3. MASTER algorithm with clustering based on a dissimilarity criterion.

One of the advantages of clustering strategy 2 is that it can be solved in a framework that applies parallelized jobs, which can significantly reduce the execution time of this strategy. This characteristic was however not explored in our experiments. In strategy 3, we apply three different distances (cf. Table 4) within raw and normalized data. For each strategy, we calculate the maximum, the minimum and the average execution time as well as the SSE on sampling period T and the SSE on sampling period $T + 1$.

8.1 Data description

In the experiments described in this section, we analyze data streams produced by a network composed of 1,000 m each producing one measurement every 30 min (48 measurements per day) during one year (365 days). For experimental purposes, all the measurements sent by the sensors are funnelled into a single input file which is then treated by the DSMS Esper from EsperTech.[1] In real situations, the multi-source data streams can be directly read from the sensors by Esper. Esper is an open-source event stream processing (ESP) and event correlation engine (CEP) providing a rich event processing language (EPL) to express filtering, aggregation, and joins, possibly over sliding windows of multiple event streams. Esper can be easily embedded in an existing Java application or middleware to add real-time event-driven capabilities to existing platforms without paying high serialization cost or network latency for every message received and action triggered.

For the data streams analyzed, each event is represented by a unique identifier, a single measurement (power) and a timestamp. Events are encapsulated by Esper in a time window and then sent to our engine. Once the sampling rates are computed, samples for each sensor are stored in a PostgreSQL database.

8.2 Result analysis

The experiments described in this section were carried out in a dual-core Sun machine, ultra24 model running Linux Fedora 11. In order to guarantee representative results, all the one thousand sensors are selected in the *spatial sampling* module of MASTER and thus activate in our experiments. According to the parameters used, the SSE matrix is computed at the end of the each day and has one thousand rows (one per sensor) and 10 columns (one per sampling rate).

Figure 8 shows the execution time spent by each clustering strategy. Figures 9 and 10 show the logarithm of SSE values computed on sampling period T (current day) and on sampling period $T + 1$ (next day), respectively, by each clustering strategy according to the number of clusters $k \in [1, 50]$. All the charts present the minimum, the average and the maximum values obtained by the three strategies to process a day of measurements. In the charts, the horizontal gray lines represent the minimum (min), the average (avg) and the maximum (max) values obtained by the strategy applying the MASTER algorithm with no clustering.

In Fig. 8, the average execution time (represented by the horizontal gray line named avg) of strategy 1 is roughly equal to 300 s (5 min). The execution time of strategy 2 (represented by the red line) is equal to that of strategy 1 when $k = 1$ and tends to zero when the number of clusters increases. This is due to the fact that when $k = 1$, all the individuals are assigned to the same cluster and the linear programming solver is applied within a single cluster. In this case, strategy 2 has no advantages over strategy 1. When the number of clusters increases, the original SSE matrix is split into smaller SSE matrices. As a consequence, the linear programming solver deals with optimization problems of smaller magnitude, which requires shorter execution times. On the other hand, strategy 3 is very fast. Its execution time tends to zero no matter the number of clusters and the dissimilarity function applied (cf. the superimposed lines representing different distances applied within this strategy). This is due to the fact that the original SSE matrix containing $N = 1{,}000$ rows is reduced to a matrix containing k rows (cf. Fig. 7), which represents a significant time execution.

[1] http://www.espertech.com.

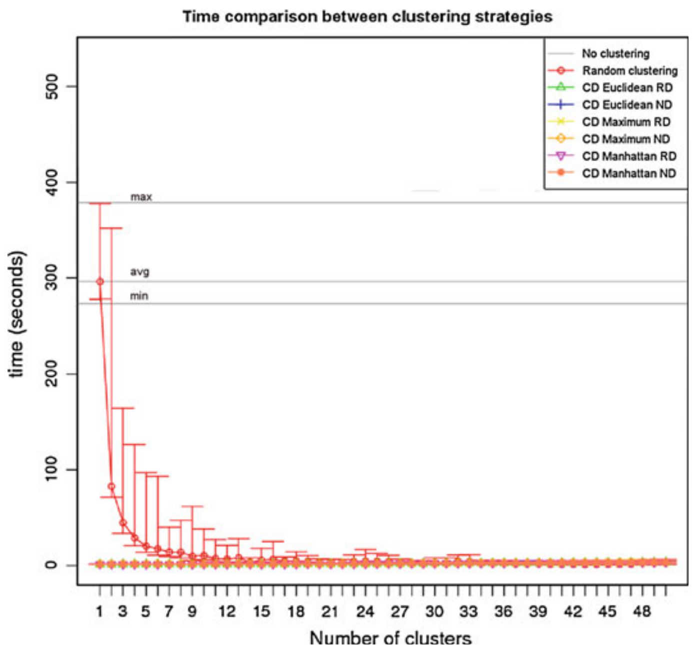

Fig. 8 Execution time for each clustering strategy (where *CD*: clustering with distance, *RD*: raw data, *ND*: normalized data)

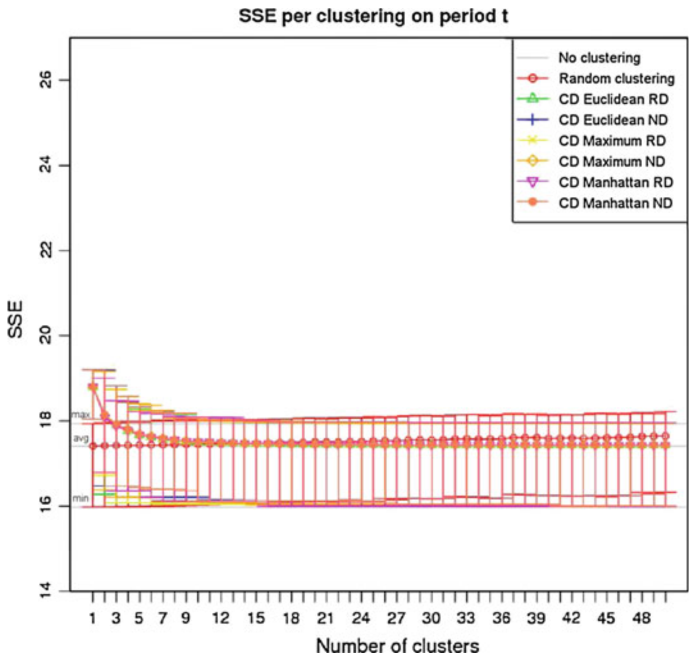

Fig. 9 SSE on sampling period *T* (current day) for each clustering strategy (where *CD*: clustering with distance, *RD*: raw data, *ND*: normalized data)

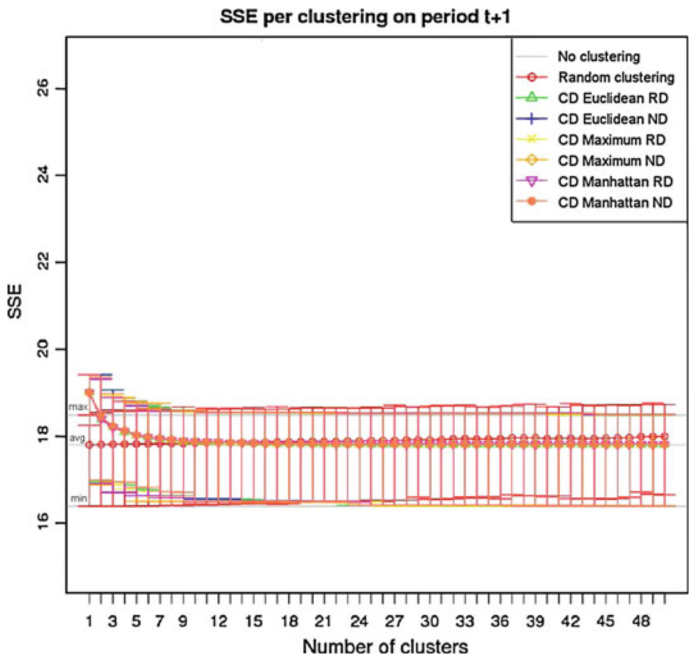

Fig. 10 SSE on sampling period $T + 1$ (next day) for each clustering strategy (where *CD*: clustering with distance, *RD*: raw data, *ND*: normalized data)

In Fig. 9, the logarithmic average value obtained by strategy 1 is approximately equal to 17.5. The logarithmic SSE value obtained by the strategy 2 corresponds to the average value obtained by strategy 1 when $k = 1$ and increases with the number of clusters. This is due to the fact that a same bandwidth is shared by a greater number of clusters which are composed by sensors randomly assigned. On the other hand, strategy 3 assigns similar sensors to a same cluster. This strategy achieves its worst logarithmic SSE value when $k = 1$, i.e., when all the individuals are assigned to a same cluster and have a same sampling rate. This strategy tends to find more homogeneous clusters when the number of clusters increases. Since the sensors assigned to a same cluster are similar, a same sampling rate tends to better fit smaller clusters. As a consequence, the SSE values decrease when the number of clusters increases. If the number of clusters is equal to the number of sensors ($k = N$), i.e., if each cluster has only one sensor, strategy 3 reaches the same SSE values obtained by strategy 1, because in this case the reduced SSE matrix is equal to the original SSE matrix.

Regarding the SSE on sampling period $T+1$, computed when the sampling rates obtained from data on sampling period T are used to sample data on sampling period $T+1$, the results obtained are those shown in Fig. 10. We can notice that the logarithmic SSE values obtained by the strategies on sampling period $T + 1$ are quite similar to those obtained on sampling period T (Fig. 9). The main difference is that the SSE values obtained by the strategies on sampling period $T + 1$ are greater then those on sampling period T. This is explained by the fact that the optimization module of the strategies is computed on sampling period T and has no information about the sensor measurements of sampling period $T + 1$.

Regarding the dissimilarity distance applied in the strategy 3, we did not notice a particular difference between the three distances applied (cf. Table 4). Figure 11 shows a graphical

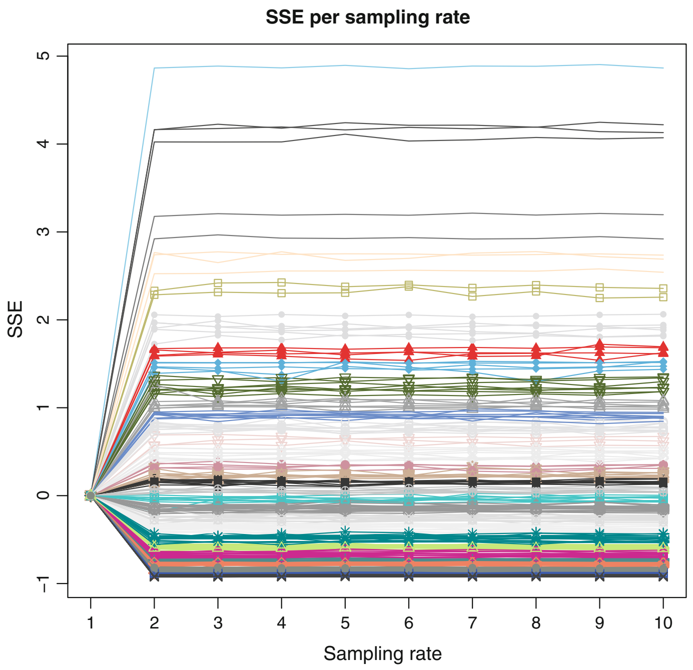

Fig. 11 Graphical representation of the SSE matrix computed with the Euclidean distance within normalized data

representation of the SSE matrix computed with the Euclidean distance within normalized data. Each curve in this chart represents a row of the original SSE matrix. Curves that belong to a same cluster are painted with a same color. Notice that similar curves are assigned to a same cluster. It is worth emphasizing an important characteristic of the SSE matrix related to the fact that SSE increases with the sampling rate. Notice that in Fig. 11 the SSE curves are all increasing and never cross each other. This is due to the fact that the SSE computed on a non-linear curve sampled with rate z is smaller than the SSE computed for this same curve sampled with rate $z + 1$. In other words, the greater the sampling rate is, the smaller the number of points sampled is. As a consequence, the interpolated curve is less accurate and the SSE values are greater.

In conclusion, we can say that clustering strategies 2 and 3 perform very well over the strategy 1, which applied the MASTER algorithm with no clustering. In the strategies with clustering, special attention should be paid to the choice of the number of clusters. In our experiments, $14 < k < 24$ seems to be a good choice for strategy 2 and $k > 14$ for strategy 3 since the trade-off between the execution time and SSE values is acceptable. If the goal of the application is to rapidly determine the best sampling rate for all sensors in a network, strategy 2 is suitable. If besides the sampling task, the application has to identify groups of similar individuals (in our study case, a cluster could represent a group of individuals with similar electrical consumption behavior), strategy 3 should be applied.

9 Conclusion

In this article, we presented CLUSMASTER, a new approach for combining clustering and sampling techniques on data streams. The goal is to assign the best sampling rate to each individual sensor in a network in a rapid and efficient way by minimizing the sum of square errors (SSE). In order to evaluate the efficiency of our approach, experiments have been done on real data streams describing measurements of electric power consumption generated by power meters installed in French households. As reported in the result analysis (cf. Sect. 8.2), the proposed approach presents a good trade-off between the execution time and the SSE values. Moreover, our approach overcomes the main limitations of the well-known data stream clustering methods (cf. Sect. 2.2). Compared to BIRCH and STREAM methods, CLUSMASTER proposes a dynamic optimization stage which continuously adapts with the content of the stream. Unlike CluStream, CLUSMASTER proposes a generic approach whose parameters are independent of the nature of the data streams. Differently from StreamSamp, the accuracy of CLUSMASTER is constant and does not degrade over time.

Concerning the electrical power and automation industries, new challenges will arise from the development of the energy Web. The energy Web envisions a day when intelligent devices all across the value chain—generation, delivery, end use—communicate and interact to optimize the system. The energy Web refers to the integration of the utility electrical system, telecommunications system and the energy market to optimize loads on the electrical network, reduce costs to consumers, facilitate the integration of renewable resources, increase electrical system reliability and reduce environmental impacts of load growth. In a smart home equipped with intelligent electronic devices (e.g., lighting, temperature control, multimedia, security and window and door operations) emitting instantaneous measurements, the volume of data streams thus generated can reach colossal magnitudes. The proposal of efficient and fast sampling techniques able to deal with this new environment of sensor networks will certainly yield future research challenges.

The analysis of results obtained by our approach clearly testifies the power and benefits introduced by combining clustering and sampling techniques. Clustering strategies as the ones presented in this work are recommended for dealing with the problem of sampling and storing massive data streams generated by multiple sources. Future work on CLUSMASTER may include the following: experiments on larger data sets generated by a larger number of sensors on a longer sampling period, extension of the approach to sensors producing multidimensional numerical data, investigation and test of other clustering techniques.

Acknowledgments This work is supported by the French national research agency—the ANR (Agence Nationale de la Recherche)—under grant n∘ ANR-07-MDCO-008 (the ANR MIDAS Project).

References

1. Aggarwal CC (2010) A segment-based framework for modeling and mining data streams. In: Knowledge and information systems, pp 1–29. Springer
2. Aggarwal CC, Han J, Wang J, Yu PS (2003) A framework for clustering evolving data streams. In: Proceedings of the 29th international conference on very large data bases (VLDB'2003), pp 81–92
3. Bash BA, Byers JW, Considine J (2004) Approximately uniform random sampling in sensor networks. In: Proceeedings of the 1st international workshop on Data management for sensor networks (DMSN'04), pp 32–39 (Toronto)
4. Bellman RE (1961) On the approximation of curves by line segments using dynamic programming. Commun ACM 6(4):284
5. Broder AZ, Charikar M, Frieze AM, Mitzenmacher M (1998) Min-wise independent permutations. In: Proceedings of the 30th annual ACM symposium on theory of computing (STOC'98), pp 327–336. Dallas
6. Chiky R, Hebrail G (2008) Summarizing distributed data streams for storage in datawarehouses. In: Proceedings of the 10th international conference on data warehousing and knowledge discovery (DaWaK 2008), pp 65–74

7. Cormode G, Garofalakis M (2008) Approximate continuous querying over distributed streams. ACM Trans Database Syst 33(2):1–39
8. Csernel B, Clerot F, Hebrail G (2006) Streamsamp: datastream clustering over tilted windows through sampling. In: ECML PKDD 2006 Workshop on knowledge discovery from data streams, Berlin
9. Grossi V, Turini F (2011) Stream mining: a novel architecture for ensemble-based classification. In: Knowledge and information systems, pp 1–35. Springer
10. Hartl G, Li B (2005) Infer: a Bayesian inference approach towards energy efficient data collection in dense sensor networks. In: International conference on distributed computing systems, pp 371–380
11. Hugueney B (2003) Reprsentation symbolique de courbes numriques. PhD thesis, University of Paris VI
12. Jain A, Chang EY (2004) Adaptive sampling for sensor networks. In: DMSN'04: Proceeedings of the 1st international workshop on data management for sensor networks. Toronto, pp 10–16
13. Karabulut K, Alkan A, Yilmaz AS (2008) Long term energy consumption forecasting using genetic programming. Math Comput Appl 13(2):71–80
14. Keogh E, Chu S, Hart D, Pazzani M (2001) An online algorithm for segmenting time series. In: Proceedings of IEEE international conference on data mining, pp 289–296
15. Khalilian M, Mustapha N (2010) Data stream clustering: challenges and issues. In: The 2010 IAENG international conference on data mining and applications, Hong Kong
16. Kranen P, Assent I, Baldauf C, Seidl T (2010) The ClusTree: indexing micro-clusters for anytime stream mining. In: Knowledge and information systems, pp 1–24. Springer
17. Land A, Doig AG (1960) An automatic method for solving discrete programming problems. Econometrica 28(3):497–520
18. Liu C, Wu K, Tsao M (2005) Energy efficient information collection with the arima model in wireless sensor networks. In: IEEE global telecommunications conference (GLOBECOM-05), vol 5, pp 2470–2474
19. Mahdiraji AR (2009) Clustering data stream: a survey of algorithms. Int J Knowl Based Intell Eng Syst 13(2):39–44
20. Marbini AD, Sacks LE (2003) Adaptive sampling mechanisms in sensor networks. London Communications Symposium, London
21. Nath S, Gibbons PB, Seshan S, Anderson Z (2008) Synopsis diffusion for robust aggregation in sensor networks. ACM Trans Sens Netw 4(2):1–40
22. O'Callaghan L, Mishra N, Meyerson A, Guha S, Motwani R (2001) Streaming-data algorithms for high-quality clustering. In: Proceedings of IEEE international conference on data engineering, pp 685–694
23. Tatbul N (2010) Streaming data integration: challenges and opportunities. In: IEEE ICDE international workshop on new trends in information integration (NTII'10), Long Beach, pp 155–158
24. Tille Y (2006) Sampling algorithms. Springer series in statistics
25. Willett R, Martin A, Nowak R (2004) Backcasting: adaptive sampling for sensor networks. In: Proceedings of the 3rd international symposium on Information processing in sensor networks (IPSN'04). Houston, pp 124–133
26. Zhang T, Ramakrishnan R, Livny M (1997) Birch: a new data clustering algorithm and its applications. Data Min Knowl Discov 1(2):141–182

Author Biographies

Alzennyr da Silva received a B.Sc. from UFPB (Federal University of Paraiba, Brazil) in 2003, a M.Sc. degree from UFPE (Federal University of Pernambuco, Brazil) in 2005 and a PhD in 2009 from University of Paris IX Dauphine in France. From 2005 to 2009, she was a member of the AxIS research team at I.N.R.I.A (National Institute of Research in Computer Science and Control) in France. She is currently a postdoctoral fellow at Télécom ParisTech in France and also a member of the Business Intelligence common laboratory between Télécom ParisTech and EDF (the biggest electric power supplier in France). Her research interests include cluster analysis, cloud computing, data mining, biomedical engineering, classical and no-SQL databases, Hadoop ecosystem and smart grids.

Raja Chiky is currently Associate Professor at ISEP where she is responsible for database and data mining courses. She holds a PhD in Computer Science from Télécom ParisTech obtained after a Master degree in data mining and an engineering degree in computer science. Before joining ISEP, she taught statistics, databases and language programming at the University of Paris Dauphine, University Paris 12 and Telecom ParisTech. She worked closely with EDF R&D on research projects related to data stream mining. Her research interests include statistics, data mining, data warehousing and data stream management.

Georges Hébrail joined Télécom ParisTech (Paris, France) in 2002 as a Professor of Computer Science, after almost 20 years spent as a Researcher at EDF R&D. Since 2007, he is also Head of the BILab, a joint laboratory with EDF R&D on Business Intelligence. His research interests cover the different domains of business intelligence: databases, data warehouses, statistics and data mining. He is currently working on the use of data stream approaches to handle very large volumes of data in two application domains: telecommunications and electric power consumption.