A statistical process control approach for data collection in sensor networks

#Ilka A. Reis[1,2], Gilberto Câmara[1], Renato Assunção[2],Antônio Miguel V. Monteiro[1]
[1]*National Institute for Space Research (INPE)*
Av. dos Astronautas, 1768 – São José dos Campos – Brazil, {ilka,gilberto,miguel}@dpi.inpe.br
[2] *Departamento de Estatística, Universidade Federal de Minas Gerais (UFMG)*
Av. Antônio Carlos, 6627 – Belo Horizonte – Brazil, assuncao@est.ufmg.br

Abstract

The main goal of a data collection protocol for sensor networks is to keep the network's database updated while saving the nodes' energy as much as possible. To achieve this goal without continuous reporting, the data suppression is a key strategy. In this paper, we propose to use a technique for statistical process control in a temporal suppression scheme. We have inserted one of these techniques (the Shiryayev-Roberts scheme) in a data collection protocol named TS-SPC (Temporal Suppression via Statistical Process Control). TS-SPC detects changes in the mean of the monitored variables and communicates just these changes to the base station. Experiments with real and simulated data have shown the TS-SPC protocol has suppression rates comparable and even greater than the rates of temporal suppression schemes proposed in the literature. Furthermore, it keeps the prediction errors at acceptable levels, if we consider the complexity of the data collection using a sensor network.

1. INTRODUCTION

Sensor networks are a powerful instrument for data collection, especially for applications like habitat and environmental monitoring. These applications often require continuous updates of the database at the network's root. However, sending continuous reports would quickly run out the limited energy of the nodes. A solution for continuous updating without continuous reporting is to use data suppression [1].

To define a data suppression scheme, nodes and base station have to agree on an expected behavior for the nodes' readings. Thus, nodes only send reports to the base station when their values do not fit to the expected behavior, which is used to predict the suppressed data.

A temporal suppression scheme uses the correlation among the readings of a same node to build the expected behavior for the nodes' readings [2]. A spatio-temporal suppression scheme also considers the correlation among the observations of neighboring nodes [1].

Model-driven data collection [3] defines the mean of a node's observations as their expected behavior and models this mean using temporal or spatio-temporal correlations.

In this paper, we consider the sequence of data collected by a node as observations of a temporal process and the mean of this process as its expected behavior. However, instead of modeling this mean, we propose to monitor its changes along the time and only send data to the base station when the value of this mean has a relevant change. To monitor the process mean, we employ the Shiryayev-Roberts scheme, a technique for statistical process control (SPC).

We have inserted a variation of the Shiryayev-Roberts scheme [4] as part of the TS-SPC (*Temporal Suppression via Statistical Process Control*) protocol for data collection in sensor networks.

The main goal of this paper is to define the temporal suppression scheme of the TS-SPC protocol and compare its performance with the existing alternatives to temporal suppression. Since the data transmission is the most important energy consumer, we use the suppression rates as a proxy for the energy consumption. The prediction error measures the quality of data sent to the base station.

The remainder of this paper is organized as follows. Section 2 describes the change-point problem and the Shiryayev-Roberts scheme. In section 3, we present the TS-SPC protocol. Section 4 describes the related work. In Section 5, we present the results of experiments with simulated and real data. Finally, section 6 draws some concluding remarks.

2. THE CHANGE-POINT PROBLEM AND THE SHIRYAYEV-ROBERTS SCHEMES

The problem of detecting changes in the parameters of a temporal process is known as the change-point problem [5] and is often found in areas such as industrial statistics and epidemiology.

Setting the mean as the parameter of interest, we summarize the change-point problem as follows. Let $[X_n] = X_1, X_2, \ldots$ be a sequence of random variables. Before time of change v, the mean of $[X_n]$ is μ_0. After the change-point v, $[X_n]$ has mean $\mu(\Delta)=\mu_0+\Delta\cdot\sigma, -\infty<\Delta<\infty$, where σ is the standard deviation of X_i, i=1,2,3…,n.

The time of change, v, is unknown but we suppose for the moment that μ_0 is known. The goal of a change-point detection scheme is to raise an alarm as soon as possible if a change has occurred, constrained to a predefined frequency of false alarms. Thus, we have a stopping time N on the sequence of observations $[X_n]$. After the stopping time, a new sequence of variables is observed. The expected value for N when there is no change ($v=\infty$), $E_\infty[N]$, is called the *average run length* (ARL). Its value represents the number of observations in the sequence before a false alarm and is often referred as ARL^0.

The largest ARL to detection (ARL^1) express the expected delay and it is calculated as $\sup_{1\le v<\infty} E_v[N-v|N>v]$. There is a trade-off between the number of observations before a false alarm (ARL^0) and the expected delay to detection (ARL^1): higher the false alarms control (high ARL^0), smaller the speed to detect a change (high ARL^1). An optimal change-point detection procedure minimizes the value of ARL^1 while keeps the ARL^0 at an acceptable level.

Based on the sequence $[X_n]$ and for any $1 \leqslant k < n$, we want to test $(H_0: \nu=\infty)$ against $(H_1: \nu=k, \Delta=\delta)$. The value δ is the *target change*.

Among the several proposals to deal with the change-point problem [5], Pollak and Siegmund [4] has proposed a variation of the Shiryayev-Roberts (SR) scheme to deal with the unknown μ_0. We call this proposal as *SR-invariant*. Considering an original sequence of Gaussian variables with unit variance, the SR-invariant has the following test statistic for a two-sided alternative ($|\Delta| = \delta$)

$$R_n(\delta) = \sum_{k=1}^{n-1} \cosh\left[\delta\left(kS_n/n - S_k\right)\right] \Big/ \exp\left[\delta^2 k\left(1-k/n\right)/2\right], \quad (1)$$

where $S_n = \sum_{i=1}^{n} X_i$ and $S_k = \sum_{i=1}^{k} X_i$.

To complete the specification of an SR scheme, it is necessary to define a stopping time N_B so that $N_B = \inf\{n: n>1, R_n(\delta) \geq B\}$. The value for the threshold B depends on the value of δ and on the minimum value for ARL^0, which is determined by the user. So, we can write B_δ.

The size and the frequency of the change to be detected influence the expected delay to detection (ARL^1) [4, 6]. Small changes are harder to be detected than large changes. For instance, to detect a small change (Δ=0.5) and a large change (Δ=2.0) in a very dynamic time series (changes in the mean occur to each 10 observations, in average), the expected delays of the SR(δ=0.5) scheme are 62.3 and 5.4 time periods, respectively. The scheme SR(δ=2.0) takes, in average, 291.5 and 3.4 time periods, respectively.

This simple numerical example is helpful to illustrate another characteristic of the SR schemes: to detect large changes, SR schemes with small δ values are as fast as SR schemes with large δ values (from the example, ARL^1 = 5.4 and 3.4 time periods, respectively). However, to detect small changes, schemes with large δ values are not so fast as schemes with small δ values (ARL^1 = 291.5 and 62.3 time periods, respectively).

Lower the changes frequency, lower the expected delay to detection. For instance, in a process with an interval between changes (ν) equal to 10 time periods, a SR(δ=1.0) scheme detects a real change Δ=1.0 with an expected delay of 6.85 time periods after the change. If the process is less dynamic (e.g. ν=100), the same scheme detects the same change with an expected delay of 4.67 time periods after the change. The influence of the changes frequency on the expected delay decreases as the size of the change rises. To a real change of Δ=2.0, for instance, an SR scheme with δ=2.0 has the expected delay to detection of 2.20 and 1.90 time periods for process with ν=10 and ν=100, respectively.

The SR scheme has optimal properties and its performance is comparable to the performance of popular methods as CUSUM [4, 6, 7]. Moreover, the SR scheme has no assumption about the independence of the variables in the sequence $[X_n]$.

In the next section, we describe our proposal to use the SR-invariant procedure in the TS-SPC data collection protocol.

3. THE TS-SPC PROTOCOL

We consider a relevant change in a node's value occurs if the mean of monitored variable changes. Otherwise, the difference between two sequential values is due only to the random variability of the measures. We propose to use the SR-invariant scheme to detect changes in the mean of the nodes' readings. Once a change is detected, the node sends its most recent value to the base station.

Let *t'* the last time a node has sent its value to the base station. At each time *t*, the node collects the data and computes $R_n(\delta)$, where $n = t - t'$. To compute $R_n(\delta)$, the node needs the *n* most recent readings since the last transmission, $X_{(t'+1)}, X_{(t'+2)}, \ldots, X_{(t'+n-1)}, X_{(t'+n)}$. To optimize the calculations, the node stores the sums S_i, i = 1,2,…,*n*, instead of the values X_i. At the next time *n+1*, the node computes S_{n+1} just adding the recent reading, X_{n+1}, to S_n.

TS-SPC protocol uses the statistic $R_n(\delta)$ as follows. Suppose we define a relevant change as Δ = 2 and set a proper value for B_δ. Let `SR.rel` be a function so that

$$\texttt{SR.rel} = \begin{cases} 1, & \text{if } R_n(\delta=2) \geq B_{(\delta=2)} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

At each time *t*, the node evaluates the function `SR.rel`. If `SR.rel`= 1, it sends data to base station. When the base station receives a message from the node with a new value, it updates its database using this value. Otherwise, the base station uses the last stored value.

A. *Dealing with outliers*

Even if the mean of a time series is constant, discrepant values can occur sometimes. These values are outliers and appear as the "peaks" or "spikes" of the time series plot.

The presence of outliers does not signify the mean has changed. This is clear if we can compare the time series values before and after the outlier, examining the time series plot, for instance.

However, an on-line change-point detection scheme can interpret an outlier as a change in the mean, since the scheme only knows the values before the outlier, not after it. Furthermore, a sequence of increasing (or decreasing) values can mimic a change in the mean, since the sequence can reach a "peak" and decreases (or increases) towards the mean value again.

The only way to distinguish a change-point from an outlier is to know what happens after the change-point candidate. Then, the solution is to use a post-monitoring window. Once the scheme detects a change, this value is declared as a change-point candidate and some sensor readings are monitored after this point. These readings compose the post-monitoring window. We use these monitoring readings to compare the time series before and after the change-point candidate and decide if it is a change-point or not.

Once the node evaluates the function `SR.rel` in (2) and its output is 1, it calls the post-monitoring algorithm. For the next *A* time periods (the size of monitoring window), the node only collects the data. At the end of the monitoring interval, it calculates the average of the collected values and compares the result with the average of the (*n*-1) values collected before the change-point candidate, `serie.avg`. If the change-point candidate is an outlier, the average values before and after it should be similar. Otherwise, the average values will be different, pointing to a change-point. This comparison is standardized by the standard deviation estimate (`sigma.est`). If this standardized difference is greater than a `limit` value (e.g. 1 or 2), the algorithm declares the change-point candidate as a change-point (`chg=1`). The post-monitoring algorithm returns the result (`chg`), the values read during the monitoring interval (`serie.real`), the time count updated and the average of the data collected during the monitoring window (`serie.real.avg`).

To classify the change-point candidate as a change-point, we define

the function non.outlier as follows

$$\texttt{non.outlier} = \begin{cases} 1, & \text{if } \texttt{chg} = 1 \\ 0, & \text{otherwise} \end{cases}, \quad (3)$$

where chg is one of the outputs of the post-monitoring algorithm.

As we discuss in the section 5, the post-monitoring window has an important role to decrease the prediction error, especially when the sequence of sensor readings is very "spiky".

B. The SR-invariant scheme in the TS-SPC protocol

The TS-SPC has two phases: the learning phase and the operation phase. In the learning phase, TS-SPC estimates the variance of the monitored variable, which is essential to the next phase, the TS-SPC operation.

1) Learning phase

To use the SR-invariant procedure, it is necessary to know the standard deviation of the observations (σ). Before beginning its operation, the node collects values during a short time window and uses them to estimate σ. That is the learning phase.

This approach assumes the mean is constant inside this time window, i.e., there is no change-point. This assumption can be realistic if, during the learning window, we use a sampling rate greater than the sampling rate set to the nodes operation. For instance, if the regular sampling rate is one reading per minute, the learning window can use three readings per minute during few minutes (ten or fifteen).

If a change-point occurs during the learning window, the variance will be overestimated. This would increase the suppression rates, but it can increase the difference between the real value and the value the base station stores. Discrepant values can also affect the estimative of the standard deviation. Then, the learning algorithm filters these outliers before calculating the estimative for σ.

Until completing N.L observations, the node collects and stores values every t.s time units, the user set sampling rate. The outliers limits are calculated according to the rules for building boxplots [8]. First, we calculate P_{25} and P_{75}, the 25th and the 75th percentiles of the observations, respectively. To calculate the percentiles, the algorithm has to sort the data, which can be done during the values storage. The difference IQ=(P_{75}-P_{25}) is called interquartile range. The upper and lower limits are defined as outlier.upper = P_{75} + 1.5 IQ and outlier.lower = P_{25} - 1.5 IQ. Values outside these limits are considered to be outliers.

After eliminate the possible outliers, the algorithm calculates the variance of the observations (sigma2.est).

If the characteristics of the monitored variable are well-know, the user can set upper and lower thresholds to the variance estimative. If this estimative is not inside the thresholds, the node starts a new learning phase.

2) The operation phase

After the learning phase, the node has all the parameters it needs to start the operation phase: the user-set values (B, δ and limit) and the estimative for σ (sigma.est). Fig. 1 presents the pseudo-code for TS-SPC operation phase. We use the notation [i] to represent the *i-th* element of a queue.

After initializing the time counts, global and local, the node collects the first value and sends it to the base station (lines1-4). This is the first data transmission. Then, the node standardizes the collected value, stores it into the queue serie.Sn and updates the global time count (lines 5-7).

The algorithm proceeds while the node's battery has a non-critical level of energy (energy.OK=1). The node reads the sensed value, standardizes it and updates the local count n (lines 9-11).

The algorithm proceeds while the node's battery has a non-critical level of energy (energy.OK=1). The node reads the sensed value, standardizes it and updates the local count n (lines 9-11).

```
TS-SPC operation.phase()

Input     B, delta.tg, A, limit, sigma.est
Output    values sent to base station

1)  t = 1 ;          # the global time count
2)  n = 1 ;          # the local time count
3)  read value.t ;
4)  send value.t ;
5)  value.t = value.t / sigma.est ;
6)  serie.Sn = value.t ;
7)  t = t + 1;
8)  while (energy.OK = 1) do
9)    read value.t ;
10)   value.t = value.t / sigma.est ;
11)   n = n + 1 ;
12)   calculate
        serie.avg = serie.Sn/(n-1) ;
        S.n = serie.Sn[n-1] + value.t ;
13)   enqueue S.n into serie.Sn ;
14)   calculate
        R.n = Rn.delta(serie.Sn,delta.tg) ;
15)   if (SR.rel = 1)
16)    if (A≠0)
17)      post-monitoring(serie.avg, limit,
                         sigma.est, A, t) ;
18)      t = tk ;
19)      if (non.outlier = 1)
20)        send serie.real.avg * sigma.est;
21)        serie.Sn = serie.real.avg ;
22)        n = 1 ;
23)      else
24)        for j=1 to A do
25)            S.n = S.n + serie.real[j] ;
26)            enqueue S.n into serie.Sn ;
27)        n = n + A ;
28)    else           #if A=0
29)      calculate
           value.t=serie.Sn[n]-serie.Sn[n-1];
30)      send  value.t * sigma.est;  #End of if (line 15)
31)   t = t + 1 ;     #End of while
32) calculate
33)   value.t = serie.Sn[n] - serie.Sn[n-1];
34)   value.t = value.t * sigma.est  ;
35) send  ( value.t , end.flag ). #End of node's operation
```

Fig. 1. Pseudo code for the TS-SPC operation phase algorithm

Using the last sum queued into serie.Sn (S_{n-1}) and the last read value (value.t), the algorithm calculates the average of the last (n-1) read values and the value of S_n (line 12). Then, it updates the queue serie.Sn. Using this strategy, the node needs to store only n+1 values (S_1, S_2, …, S_n and the average) and quicken the calculations, doing only two calculations to update them.

At next step (lines 14-15), the algorithm calculates the value for $R_n(\delta)$

using (1) and evaluates this value in the function `SR.rel` in (2). If the `SR.rel` output is equal to 1, the `value.t` is a change-point candidate. Then, if the size of the monitoring window (`A`) is different from zero (line 16), the algorithm calls the post-monitoring algorithm. After the end of the monitoring window, the global time count is updated to take account the `A` time periods of the window (line 18).

If the `non.outlier` function in (3) has declared the change-point candidate as a change-point, the node "unstandardizes" the mean of the values read inside the monitoring window (`serie.real.avg`) and sends it to base station (lines 19-20). Then, it starts a new sums queue with the mean value sent to base station and reset the local count `n` to one (lines 21-22). If the change-point candidate is not declared as a change-point (line 23), the algorithm calculates the `A` sums using the values in `serie.real` and updates the sums queue `serie.Sn` and its size count `n` (lines 24-27).

If a monitoring window is set to zero (line 28), the algorithm must transmit the last collected value. Since the algorithm stores the sums S_i and not the values themselves, it has to calculate the last read value based on the difference of the last two stored sums (line 29). After "unstandardizing" this value, the algorithm sends it to the base station (line 30).

The global count is updated (line 31) and the cycle restarts (lines 8-31). When the nodes' battery is running out (`energy.OK=0`), the algorithm must transmit the last collected value. Then, it calculates the last read value based on the difference of the last two stored sums, "unstandardizes" this value and sends it to the base station together a flag indicating the end of the nodes' operation (lines 32-35).

C. *Costs*

1) Data and parameters storage: at each time period *t*, the node has to store $n = t - t'$ sums, where *t'* the last time a node has sent its value to the base station. If a post-monitoring window is open, the node also has to store the *A* values read inside that window. During the learning phase, the node has to store N_{LS} values. The TS-SPC protocol needs to store six parameters: *A*, δ, B, `sigma.est`, `limit` and N_{LS}.

2) Calculations: the TS-SPC protocol operation involves mainly simple calculations, as additions and multiplications. The most sophisticated calculations are two exponentiations of the $R_n(\delta)$ expression and the square root of the standard deviation expression, which is used only in the learning phase.

3) Sent messages: the message the node sends to the base station contains only one value. Supposing each value in a message costs 1 energy unit, the cost of a TS-SPC message is 1 energy unit.

4. RELATED WORK

Recently, some protocols for data collection in sensor networks have proposed to use statistical models to predict the nodes' data at the base station reducing the amount of communication inside the network (model-driven approach to data collection [3]).

The main idea in [3] and correlated works is to keep synchronized two probabilistic models: one at base station and other at the nodes. The model parameters are estimated in a learning phase. Based on these identical models, nodes and base station make the same predictions on the data to be collected. Then, the node collects the actual data and compares them to its prediction. If the difference between the real and predicted values is greater than a user-defined error bound, the node sends its data to the base station. Otherwise, the node suppresses the data.

A similar idea appears in [2]. The PAQ protocol makes predictions based on a time series model, the third-order autoregressive model, AR(3). Given a time period t, the predicted value in t is written as a linear combination of the last three observations before t. When the real and the predicted values differ by an amount greater than pre-defined error bounds, PAQ uses an algorithm to monitor outliers and re-learn the four model parameters, sending their new values (or the outliers) to base station. A variation of PAQ, called in [1] as exponential regression (EXP), uses the observation in the time period (t-1) in a simple linear regression to predict the observation in t. Thus, EXP has to estimate two model parameters. We return to PAQ and EXP in sections V and VIII.

We classify our TS-SPC proposal as a model-driven approach for temporal suppression [1]. As the protocols we have described, we use the mean of the monitored variable to predict its value. However, differently from them, we do not model this mean. The TS-SPC protocol just monitors the mean changes along the time.

The messages of the TS-SPC protocol are cheaper than the messages of the earlier described protocols. The TS-SPC messages contains only one value, whereas EXP and PAQ messages, for instance, can contain two or four values, respectively, if they send the new values for their models parameters.

It is worth to note that our TS-SPC proposal is not constrained to applications whose interest is to monitor the mean of the nodes measurements. We use a method for mean change detection as a strategy to decide whether the node must suppress its data, adopting the mean as an estimate of the node's value, likewise the other model-driven approaches for temporal suppression.

To the best of our knowledge, our proposal for data collection protocol in sensor networks is the first one using an SPC technique as a basis for a data suppression scheme.

In this paper, we have focused on temporal suppression. We defer the discussion on spatio-temporal suppression strategy to a full version.

5. EXPERIMENTS

We have run experiments using real data, which have been collected at the weather station of the University of Washington (USA) [1]. We have chosen four variables with different behaviors: smooth changes but often (solar irradiance); smooth changes but not so often (air temperature), "spiky" with abrupt changes (wind speed); "spiky" with very abrupt changes (relative humidity). The temporal resolution is one measurement per minute and the time series have nearly 4000 observations (about 3 days). In the solar irradiance time series, we have not considered the night periods (readings equal to zero), since the nodes should turn off this kind of sensor to save energy. To each chosen time series, we have evaluated the performance of the following suppression schemes: value-based (VB) [1], exponential regression (EXP), PAQ and TS-SPC. VB scheme is the most simple temporal suppression scheme. It calculates the absolute difference between two sequential values x_t and x_{t-1}, divides the resulting value for x_{t-1} and compares the final result to an error threshold ε_{VB}. The values for the parameter of the value-based scheme have been ε_{VB}=(0.03, 0.05, 0.10). For the TS-SPC scheme, the parameters for the SR-

[1] http://www-k12.atmos.washington.edu/k12/grayskies/nw_weather.html

invariant scheme and the post-monitoring window have been δ=(0.5,1.0,2.0,3.0) and *A*=(0,5,10,15), respectively. We have set ARL^0=500, the post-monitoring `limit`=1.5 and calculated the values for B_δ according to the Table 1 in [9].

For EXP and PAQ parameters, we have initially set N_{LS}=60, *A*=15, a=8, the model re-learn threshold ε_δ=(1.8, 3.0) and the outlier threshold ε_v=6.0, which are the values cited in [2] as good choices. During the schemes evaluation, we have observed other choices for the values of A, ε_δ and ε_v could improve the performance of the EXP and PAQ schemes. Whenever this occurred, we have adopted the parameters choice that results in the best performance and indicate this choice in the results presentation.

We have calculated the average message cost of PAQ and EXP schemes as the weighted average of the costs to send an outlier and the model parameters. The weights are the number of outlier messages and parameters messages. The models of PAQ and EXP have four and two parameters, respectively. Thus, the costs to send a message with these parameters are four energy units for PAQ and two energy units for EXP. The cost to send an outlier is one energy unit, which is also the costs of the TS-SPC and valued-based schemes.

We have evaluated the performance of TS-SPC protocol using two measures: the *suppression rate* and the *median absolute relative error* (MARE). The suppression rate is calculated as the proportion of suppressed data

$$\text{Suppression rate} = 1 - \frac{(\text{number of sent messages})}{(N_{TS} - A \times n_A)}, \quad (4)$$

where n_A is the number of times the post-monitoring window *A* have been used. During the post-monitoring window, there are no transmissions. Then, we have to discount these time periods (*A* x n_A) of the time series size (N_{TS}).

The median absolute relative error (MARE) measures the prediction error and is calculated as

$$MARE = median_{(t=1,2,\ldots,N_{TS})}\left[\frac{|x_t - \hat{x}_t|}{x_t} \times I_{(0,\infty)}\left(|x_t - \hat{x}_t|\right)\right] \quad (5)$$

where $I_{(0,\infty)}(.)$ is the indicator function. In the MARE expression, the indicator function works to exclude the zeros. A zero value occurs when the node sends its value to the base station. In the TS-SPC schemes, this occurs only if *A*=0. To avoid accounting for the number of sent messages into both performance measures, we have excluded the values equal to zero.

There is a trade-off between the suppression rate and MARE. In general, we expect schemes with low suppression rates have low MARE values, since they update the base station database more often. However, this is not a rule, as we will see later.

The points in the figures 2 to 5 present the summaries of the performance measures for the best parameters choices of each scheme. Points nearer to the upper-left corner represent the schemes with the best performances.

The TS-SPC schemes have had performances comparable and even superior to the other temporal suppression schemes, especially when the time series are "spiky" and have abrupt changes (Fig. 4 and 5). Although the other schemes have got suppression rates greater than the TS-SPC rates (EXP and PAQ in Fig. 4; VB in Fig. 5), the size of their errors has degraded their global performance. Furthermore, for the EXP and PAQ schemes, the average message costs have been 2.0 and 3.93 energy units, respectively (wind speed time series). Unfortunately, neither [1] nor [2] has enough details on its evaluation experiments to allow us for comparing our results with its results.

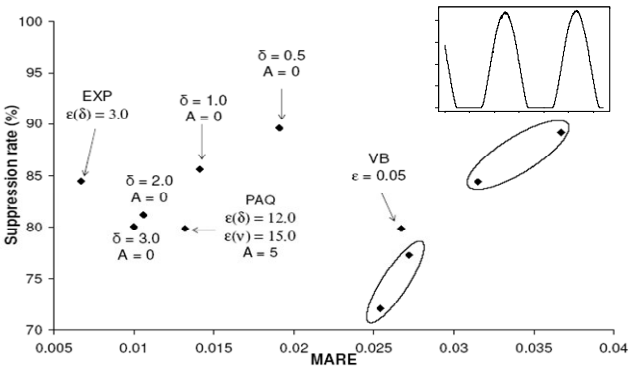

Fig. 2. Performance of the suppression schemes in the solar irradiance time series (upper-right corner). We have excluded the TS-SPC schemes with MARE values greater than 0.04 to improve the visualization.

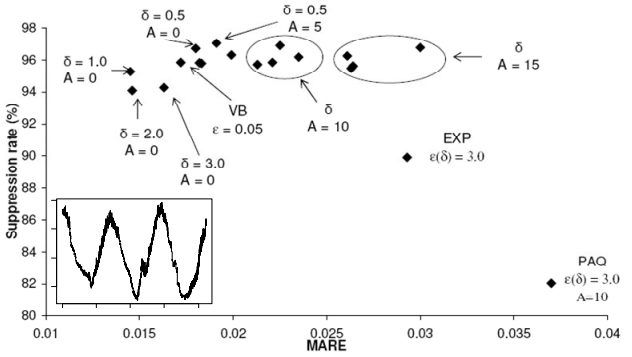

Fig. 3. Performance of the suppression schemes in the air temperature time series (bottom-left corner).

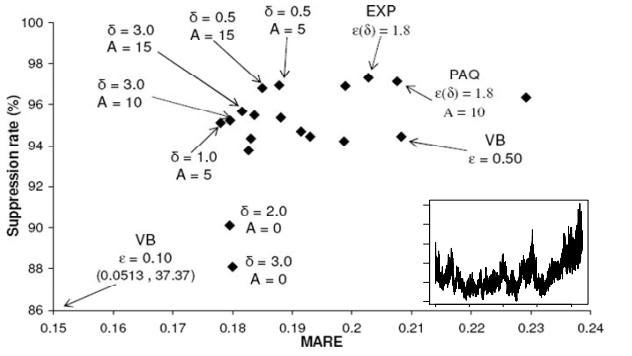

Fig. 4. Performance of the suppression schemes in the wind speed time series (bottom-right corner).

When the changes in the time series have smooth but often changes (e.g., solar irradiance), we expect smaller suppression rates, since the schemes force the nodes to send data to the base station more often. However, even in this adverse condition, the TS-SPC schemes have got the highest suppression rates while have kept the error in acceptable levels (Fig. 2, TS-SPC schemes [δ=0.5,A=0] and [δ=1.0,A=0]).

The δ parameter helps the TS-SPC scheme to adapt to the size and the frequency of the changes. When changes were smooth and often, the

TS-SPC schemes with the best performances have had small δ values (Fig. 3). If the smooth changes are not so often, the schemes with the greatest δ (2.0 and 3.0) values have the best performances (Fig. 4).

The greatest δ values also leads to the best performances of the TS-SPC schemes when the time series is "spiky" and changes are abrupt (Fig. 4). The difference is the value for the monitoring window (*A*). The monitoring window does not provide any improvements on the TS-SPC performance if the time series has smooth changes (Fig. 3). However, this outlier control is important for "spiky" time series, especially if the changes are not very abrupt (Fig. 4).

When the time series has two kinds of changes, small and abrupt (e.g. relative humidity), TS-SPC schemes with small values of δ (0.5 and 1.0) have had the best performances (Fig. 5). As we have discussed in the later section, SR schemes with small target changes (δ) can detect small real changes as fast as large real changes. The versatility of these SR schemes explains their better performance. Furthermore, the absence of a monitoring window (A=0) has also contributed to improve the performances, since the often changes cannot afford to additional delays.

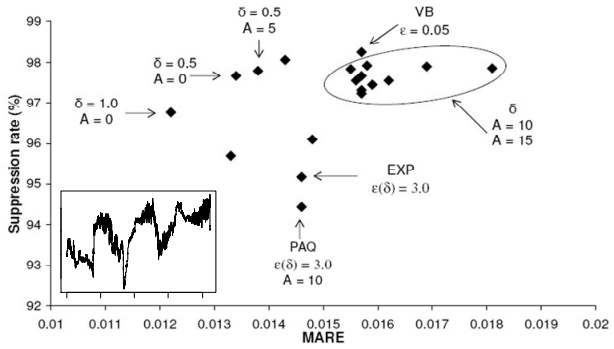

Fig. 5. Performance of the suppression schemes in the relative humidity time series (bottom-left corner).

We have run experiments with simulated data to study the effect of the changes size, the autocorrelation and variance of the time series on the performance of the TS-SPC suppression scheme. We consider the TS-SPC schemes have got an excellent performance in the simulated experiments. Even working in an adverse scenario (time series with low autocorrelation and small changes in the mean), TS-SPC schemes have got suppression rates larger than 91% and prediction errors smaller than 5.5%. In their best performances, TS-SPC schemes have got suppression rates close to 99% (time series have had high autocorrelation). For briefness, we defer the complete discussion about these results to a full version of this paper.

C. A note on modeling the mean

The model-driven approach is an efficient solution to data collection in sensor networks if the monitored variable has a well-known behavior so that reliable models can be defined [1]. Thus, in these scenarios, the lack of structure of the TS-SPC model for the mean could degrade its performance if we compare it to the performance of a scheme with a structured model for the mean. To evaluate this hypothesis, we have simulated time series according to the AR(3) model, which is the model PAQ uses. Even in a scenario clearly favorable to PAQ, the TS-SPC schemes have got performances comparable to the PAQ performances, in addition to a smaller average message cost.

6. CONCLUDING REMARKS

To define a TS-SPC suppression scheme, the user has to choose the values for three parameters: the frequency of false alarms (ARL^0), the size of the relevant change in the mean (δ) and the size of the window for outliers control (*A*). The value for ARL^0 depends on the user tolerance to false alarms. Although a low frequency of false alarms is welcome, it is worth to remember a high value for ARL^0 increases the expected delay to detection (ARL^1).

As we have discussed earlier, small values of δ are also efficient to detect larger changes. Thus, setting a small δ should be useful to capture the small changes just as the large ones. The side effect could be a greater number of false alarms. However, using a monitoring window could help to decrease these events.

The main role of the monitoring window (*A*>0) is to filter possible outliers and decrease the number of false alarms. Then, using a monitoring window always improves the suppression rate, although it can have the side effect of increasing the prediction error in some situations. However, the increasing of the prediction errors is not too dramatic, except for time series having a quite predictable behavior (no outliers) as the sequence of solar irradiance measurements.

Our future work includes a spatio-temporal version of the TS-SPC scheme, which explores the spatial homogeneity of the data in some areas of the sensors field and localizes the most part of the communication among the nodes [10].

REFERENCES

[1] A. Silberstein, R. Braynard, G. Filpus, G. Puggioni, A. Gelfand, K. Munagala, and J. Yang, "DataDriven Processing in Sensor Networks," 3rd Biennial Conference on Innovative Data Systems Research 2007.

[2] D. Tulone and S. Madden, "PAQ: Time Series Forecasting For Approximate Query Answering In Sensor Networks," Lecture Notes in Computer Science, pp. 21--37, 2006.

[3] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-Driven Data Acquisition in Sensor Network," 30th Very Large DataBases Conference, 2004.

[4] M. Pollak and D. Siegmund, "Sequential detection of a change in a normal mean when the initial value is unknown," The Annals of Statistics, vol. 19, pp. 394--416, 1991.

[5] M. Frisén, "Statistical Surveillance. Optimality and Methods," International Statistical Review vol. 71, pp. 403–434, 2003.

[6] M. Pollak and D. Siegmund, "A diffusion process and its applications to detecting a change in the drift of Brownian motion," Biometrika, vol. 72, pp. 267--280, 1985.

[7] R. S. Kennet and M. Pollak, "Data-analytic aspects of the Shiryayev-Roberts control chart: surveillance of a non-homogeneous Poisson process," Journal of Applied Statistics, vol. 23, pp. 125--137, 1996.

[8] J. W. Tukey, Exploratory Data Analysis: Addison-Wesley, Reading, MA, 1977.

[9] M. Pollak, "Average run lengths of an optimal method of detecting a change in distribution," Annals of Statistics, vol. 15, pp. 749--779, 1987.

[10] I. A. Reis, G. Câmara, R. M. Assunção, and A. M. V. Monteiro, "Data-Aware Clustering for Geosensor Networks Data Collection," XIII Brazilian Remote Sensing Symposium, 2007.