2015 IEEE 35th International Conference on Distributed Computing Systems

An online method for minimizing network monitoring overhead

Simone Silvestri*, Rahul Urgaonkar†, Murtaza Zafer‡ and Bong Jun Ko†
*Department of Computer Science, Missouri University of Science and Technology, Email: silvestris@mst.edu
†IBM T.J. Watson Research Center, Yorktown Heights, NY, USA, Email: {rurgaon, bongjun_ko}@us.ibm.com
‡Nyansa Inc., Palo Alto, CA, USA, Email: murtaza.zafer.us@ieee.org

***Abstract*—Network monitoring is an essential component of network operation and, as the network size increases, it usually generates a significant overhead in large scale networks such as sensor and data center networks. In this paper, we show that measurement correlation often exhibited in real networks can be successfully exploited to reduce the network monitoring overhead. In particular, we propose an online adaptive measurement technique with which a subset of nodes are dynamically chosen as monitors while the measurements of the remaining nodes are estimated using the computed correlations. We propose an estimation framework based on jointly Gaussian distributed random variables, and formulate an optimization problem to select the monitors which minimize the estimation error under a total cost constraint. We show that the problem is NP-Hard and propose three efficient heuristics. In order to apply our framework to real-world networks, in which measurement distribution and correlation may significantly change over time, we also develop a learning based approach that automatically switches between learning and estimation phases using a change detection algorithm. Simulations carried out on two real traces from sensor networks and data centers show that our algorithms outperforms previous solutions based on compressed sensing and it is able to reduce the monitoring overhead by 50% while incurring a low estimation error. The results further demonstrate that applying the change detection algorithm reduces the estimation error up to two orders of magnitude.**

I. Introduction

Network monitoring is a key operation at the basis of several network management tasks such as performance diagnosis [1], overlay network design [2], scheduling [3], resource allocation and selective activation [4]. Acquiring accurate knowledge on the current network state requires nodes to collect local measurements and periodically forward these measurements to a central network management entity for further analysis and appropriate actions.

The monitoring activity inevitably incurs an overhead which may significantly affect the performance of large scale networks such as sensor networks and data centers. As an example, sensing environmental data with all nodes in a sensor network and sending this data to a sink may easily create network congestion and also drain sensor batteries. Similarly, monitoring the resource utilization of all nodes in a data center may cause performance degradation and waste system resources.

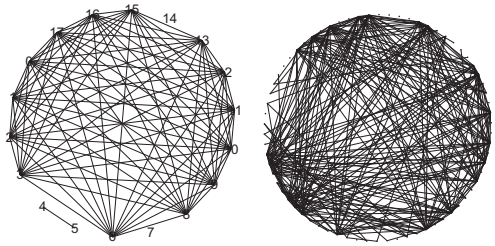

Fig. 1. Measurement correlation: (a) sensor network, and (b) and data center.

Network measurements, such as the ones mentioned above, are often correlated. To support this claim, we consider measurement data from real networks and show in Figure 1 (a-b) the correlations that exist between nodes. More specifically, we consider two data-sets, one collected from an outdoor sensor network measuring the sunlight intensity over time and the second collected from a real world data center in which servers periodically measure their CPU utilization[1]. We represent sensors and servers as nodes in a graph and we add an edge between a pair of nodes if measurements have a correlation coefficient higher than a threshold[2]. We set the threshold to 0.75 for the sensor network data set and to 0.65 for the data center case. Figures 1 (a) and (b) show the results for the respective data-sets. As the figures point out, in both cases the measurements show a high degree of correlation.

The above insight can be used to reduce the monitoring overhead of the network. In particular, only a subset of nodes can be selected to act as *monitors* and collect measurements, which are used to estimate the measurements of the other nodes in the network using correlation. Intuitively, some nodes can provide better estimation than others, and a larger number of monitors would reduce the estimation error but would also increase the monitoring overhead. As a result, the design of a framework to minimize the monitoring overhead and accurately estimate unobserved measurements requires to address the following main problems: i) the selection of the best set of monitors given a budget on the monitoring overhead, ii) the estimation of unobserved measurements given those collected by the selected monitors. In addition, the framework should be able to handle dynamics that may occur and result in a change

This research was sponsored in part by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

[1]Further details on these networks are provided in Section VII.

[2]The correlation coefficient of two populations of samples is defined as the covariance between the populations divided by the product of their standard deviations. A correlation coefficient equal to 1 represents perfect correlation.

1063-6927/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDCS.2015.35

of the measurement correlation over time.

Most previous works in this area are either based on *geometric models* or *compressed sensing*. Geometric models [4], [5] are generally considered in geographically deployed networks, such as wireless sensor networks, and exploit the spacial correlation of measurements. However, in practice location information may not be available or accurate, and, more importantly, these techniques cannot be applied to virtualized environments such as data centers, where there is no notion of distance. Compressed sensing based approaches [6]–[12] reconstruct the signal, i.e. the full set of measures, from a reduced representation in a vectorial space with less dimensions. These approaches mainly focus on the reconstruction of the original signal, and only marginally address the monitor selection problem.

In this paper, we propose an online adaptive measurement framework to jointly address the problems mentioned above. Our framework can be applied to virtual environments such as data centers and to geographically deployed networks with no location information. In addition, it adapts to dynamic changes in the measurement correlation thanks to a change detection mechanism. According to our method, time is divided in *training intervals* and *operation intervals*. During a training interval all nodes in the network collect measurements. On the basis of this information the measurement correlation is calculated and a subset of nodes is selected as monitors. During operational intervals, measurements of the other nodes are estimated using those collected by monitors.

We address the problem of estimating unobserved measurements by modeling nodes in the network as jointly Gaussian distributed random variables, whose realizations represent the collected measurements. We formulate an optimization problem to select monitors under a budget constraint, where the budget represents the overhead generated by the monitoring activity. We show that the problem is NP-Hard and solve it optimally for the special cases of single and pair monitor selection. We use this analysis for the design of three heuristics, with different computational complexity, to solve the general hard case. The three heuristics provide a tradeoff between computation complexity and estimation performance.

We propose an online change detection mechanism based on Welch's t-test in order to enable our framework to dynamically adapt to changes in the measurement correlation. The test allows to timely detect when the information gathered in the previous training interval does not represent the current variable distribution and correlation, and hence a new training interval needs to be performed.

We evaluate the performance of the three heuristics on synthetic and real traces against a recently proposed compressed sensing approach. Results show that our heuristics outperform the previous approach and in real networks are able to reduce the monitoring overhead by up to 50% while incurring a low estimation error. In addition, the incurred error is close to an optimal monitor selection strategy obtained through brute force computation. We also show that our change detection mechanism is able to quickly detect changes in the measurements and maintain the estimation error two orders of magnitude lower than the error that would have been incurred without detecting changes.

In summary, the main contributions of this paper are:

- We show that measurement correlation can be effectively used in real networks to reduce the monitoring overhead, and propose an online adaptive measurement framework.
- We formalize an optimization problem to select monitors in the network and show that this problem is NP-Hard. We optimally solve the problem for two special cases and use these to design three efficient heuristics.
- We address dynamic changes in real-world networks that would affect network measurement by designing an online change detection mechanism.
- We perform experiments on synthetic and real traces. Results show that our approach outperforms a previous approach based on compressed sensing, and can reduce the monitoring overhead up to 50% while incurring a low estimation error. Additionally, we show that the change detection mechanism enables a reduction of the estimation error up to two orders of magnitude in real networks.

II. Network model and problem formulation

We consider a general network composed by N elements. As an example, these elements may correspond to sensor devices or servers in a data center. We assume that elements are loosely synchronized and that in each time slot an element produces a *reading*. Readings may represent the data sensed by a sensor or the resource utilization (CPU, memory, etc.) of a server. We introduce a random variable x_i for each element in the network, with $i = 1, \ldots, N$ and we model readings as realizations of these variables. We denote by $X = [x_1 \ldots x_N]^T$ the N dimensional random column vector of the random variables. Let $\Sigma_X = \{\sigma_{ij}\}, i = 1, \ldots, N, j = 1, \ldots, N$ be the covariance matrix. i.e., $\Sigma_X = E[(X - \bar{X})(X - \bar{X})^T]$. Similarly, let $\mu_X = [\mu_1, \ldots, \mu_N]$ denote the vector of expected values of the variables in X. Note that, the true values of Σ_X and μ_X are unknown and may change over time.

We divide time in epochs, each of which is in turn divided into a *training interval* followed by an *operational interval*. During training intervals, the realization of all variables are observed over T_{tr} time slots. On the basis of the gathered samples, Σ_X and μ_X are estimated. At the end of a training interval, we select a subset S of variables in X. We call these variables *observed* or *selected* while we refer to the remaining variables as *unobserved* or *unselected*. We assume that observing a variable x_i incurs a cost $c_i > 0$. This cost may represent the consumed energy of a sensor or the overhead incurred in gathering the data from a server.

During operational intervals, observed variables act as monitors and their realizations are collected. Differently, the realizations of the unobserved variables in $Y = X \setminus S$ are estimated by using the realizations of the observed variables and the information given by Σ_X and μ_X calculated in the training interval. Operational intervals last T_{op} time slots. At each slot t, a realization vector $\mathbf{s}_t$ of the variables in S is observed. On the basis of this vector, for each variable $y \in Y$, we estimate its realization $\hat{y}_t$. We denote by $\hat{\mathbf{y}}_\mathbf{t}(\mathbf{s_t})$ the vector of estimate realizations at time slot t.

The choice of the set of observed variables S influences the error incurred in the estimation process. We use the *Mean Square Error* (MSE) as a measure of the incurred error. In particular, given the estimation $\hat{\mathbf{y}}_\mathbf{t}(\mathbf{s_t})$ and the true realizations $\mathbf{y_t}$, the MSE is calculated as:

$$MSE(\hat{\mathbf{y}}_\mathbf{t}(\mathbf{s_t}), \mathbf{y_t}) = \sum_{y \in Y} (y_t - \hat{y_t})^2 \quad (1)$$

Where y_t is the true realization of the unobserved variable y at time t.

Given Σ_X and μ_X, we are concerned with optimally selecting the set of variables $S^* \subseteq X$ that minimize the overall MSE in the operational interval and incur a cost less than the budget. Formally,

$$S^* = arg \min_{S \subseteq X} \Big(\frac{1}{T_{op}} \sum_{t=1}^{T_{op}} MSE(\hat{\mathbf{y}}_\mathbf{t}(\mathbf{s_t^*}), \mathbf{y_t}) \Big),$$

$$\text{s.t.} \sum_{x_i \in S} c_i \leq B$$

Depending on the network specifically considered, B may represent a budget for monitor placement, the number of sensors to be kept active or a measure of the allowed overhead for measurement collection.

III. Estimation framework and NP-Hardness

In this section we describe the framework we adopt to estimate the unobserved variables given the realization of the selected variables. In addition, we show that our optimization problem is NP-Hard.

A. Estimation framework

In our approach we assume that the variables in X can be approximated by Jointly Gaussian Distributed (JGD) random variables [13]. The probability density function of X can be then approximated as:

$$f(X) \approx \frac{1}{\sqrt{2\pi \ det(\Sigma_X)}} \exp\left(-\frac{1}{2}(x - \bar{x})^T \Sigma^{-1} (x - \bar{x}) \right) \quad (2)$$

As our results show, the above assumption holds in real scenarios and enables us to derive an efficient monitor selection and estimation framework. We address changes that may occur over time in the distribution of X through the change detection framework described in Section V.

Given a set S of selected variables, we can estimate the remaining variables on the basis of the observed realization of the variables in S as follows. For a JGD random vector, the minimum mean square error estimate is the conditional mean [13]. Given a realization vector $\mathbf{s}$, the conditional mean $\mu_{Y|S}$ of the variables in Y can be calculated as follows:

$$\mu_{Y|S} = \mu_Y + \Sigma_{YS} \Sigma_{SS}^{-1} (\mathbf{s} - \mu_S) \quad (3)$$

where Σ_{YS} is the submatrix of the covariance matrix Σ_X that express the correlation of the variables in Y and S, Σ_{SS} is the variance matrix of the variables in S, and μ_Y and μ_S are the average vectors of the variables in Y and S, respectively. Under the assumption of JGD random variables, given the variables in S, the estimation $\mu_{Y|S}$ minimizes the MSE.

We show in Section VII that in real network scenarios this estimation framework, coupled with our change detection method described in Section V, enables an accurate estimation of the unobserved variables.

B. NP-Hardness

Although assuming JGD variables enables an efficient estimation, our optimization problem of selecting the best set of variables S^* to minimize the MSE in an operational interval remains hard, as shown in the following Theorem.

Theorem 1. *The optimization problem is NP-Hard.*

Proof: We provide a reduction from knapsack. Let us consider a general instance of knapsack: a set of elements X, each element $x \in X$ has a value v_x a cost c_x, and a budget B. The goal is to find a set of elements $S^* \subseteq X$ such that the elements in S^* provide maximum value and incur a cost within the budget B. We prove that the general instance of knapsack can be translated to an instance of our problem.

Given the general knapsack instance, we create an instance of our problem as follows. We create a random variable x for each element in X. We set the average as $\mu_x = v_x$, the variance to $\sigma_{xx} = v_x$ and the cost equal to c_x. We make these variables independent on each other, i.e. $\sigma_{x,y} = 0$ if $x \neq y$.

Since variables are independent, given any set of selected variables S and unobserved variables Y, Equation 3 shows that the estimation of the variables in Y that minimizes the error is the mean μ_Y. The overall MSE incurred by this estimation is given by the trace of the conditional covariance matrix $\Sigma_{Y|S}$. Considering that variables are independent we obtain:

$$tr(\Sigma_{Y|S}) = tr(\Sigma_{YY} - \Sigma_{YS}\Sigma_{SS}^{-1}\Sigma_{SY}) = tr(\Sigma_{YY}) = \sum_{y \in Y} \sigma_{yy}$$

As a result, in this setting the objective function of our problem becomes linear:

$$\frac{1}{T_{op}} \sum_{t=1}^{T_{op}} MSE(\hat{\mathbf{y}}_\mathbf{t}(\mathbf{s_t^*}), \mathbf{y_t}) = \frac{1}{T_{op}} \sum_{t=1}^{T_{op}} \sum_{y \in Y} \sigma_{yy} = \sum_{y \in Y} \sigma_{yy}$$

Our optimization problem selects the set S^* such that, under the given budget B, the MSE incurred in estimating the variables in $Y = X \setminus S^*$ is minimized. Since in this setting our optimization function is linear, S^* is the set that minimizes the sum of the elements in Y. Therefore, S^* is necessarily the subset of X that also maximizes the sum of its elements under the budget constraint, because the sum across all elements in X is fixed and $Y = X \setminus S^*$.

Since we set the variance of our variables equal to the values of the corresponding element in the knapsack problem, S^* is the set that maximizes the values within the budget B, i.e. it is the optimal solution of the knapsack problem.

The above reasoning shows that any instance of knapsack can be translated into an instance of our problem. As a consequence, our problem is at least as difficult as knapsack and thus it is NP-Hard. ■

It is worth noting that, although we assumed JGD variables, the MSE, objective function of the optimization problem, *is not*

submodular. As an example, let us consider the set of variables $X = \{x_1, x_2, x_3, x_4\}$ with the following covariance matrix:

$$\Sigma_X = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \\ \begin{pmatrix} 1.8485 & 0.8515 & 0.7463 & 1.1311 \\ 0.8515 & 0.6408 & 0.3679 & 0.9206 \\ 0.7463 & 0.3679 & 0.5307 & 0.4066 \\ 1.1311 & 0.9206 & 0.4066 & 1.3816 \end{pmatrix} \end{array}$$

It can be shown that $\text{MSE}_{\{1\}} - \text{MSE}_{\{1,3\}} < \text{MSE}_{\{1,2\}} - \text{MSE}_{\{1,2,3\}}$, hence MSE is not submodular. As a consequence, the standard optimization theory of submodular functions [14]–[16] cannot be applied to our case.

IV. Efficient Selection of Monitors

In this section we introduce three heuristic solutions to our optimization problem. We first optimally solve the problem for two special cases, then we use this analysis as a building block to develop the heuristics for the general case.

A. Optimal solution of special cases

In the following, we consider the case in which, for each pair of variables $x_i, x_j \in X$, the costs are such that $c_i + c_j > B$, hence a single variable can be selected. The following theorem provides an efficient way to select the best single variable that minimizes the MSE. For ease of exposition, we implicitly do not consider variables whose individual cost exceeds the budget in the proof.

Theorem 2. *Let X be a set of JGD random variables, with average μ_X and correlation matrix Σ_X, the optimal single variable x_j^* that minimizes the MSE is given by:*

$$x_j^* = \arg\max_{x_l \in X} \left(\frac{\sum_{i=1}^N \sigma_{il}^2}{\sigma_{ll}} \right)$$

Proof: Let us consider a generic variable x_j and let $Y = X \setminus \{x_j\}$. Given x_j, the conditional distribution of Y is also JGD [13]. Following the estimation framework described in Section III-A, the estimation $\hat{y_i}$ of the variable y_i is:

$$\hat{y_i} = \mu_i + \frac{\sigma_{ij}}{\sigma_{jj}}(x_j - \mu_j), \quad i \neq j \tag{4}$$

where x_j is the observed realization of the selected variable.

The mean square estimation error $E[||Y - \hat{Y}||^2]$ (where we represent Y as a vector and $\hat{Y}$ is the vector of values $\hat{y_i}$) is given as the trace of the conditional covariance matrix:

$$E[||Y - \hat{Y}||^2] = \sum_{i \neq j} \left(\sigma_{ii} - \frac{\sigma_{ij}^2}{\sigma_{jj}} \right) = \sum_{i=1}^N \sigma_{ii} - \frac{\sum_{i=1}^N \sigma_{ij}^2}{\sigma_{jj}} \tag{5}$$

In order to optimize the single measurement selection, we must choose the variable x_j such that the second term above is maximized (this will minimize the error). As a result, the best variable x_j^* that minimizes the error is:

$$x_j^* = \arg\max_{x_l \in X} \left(\frac{\sum_{i=1}^N \sigma_{il}^2}{\sigma_{ll}} \right) \tag{6}$$

■

We further extend this analysis considering the case in which the budget allows to select two variables. The following theorem shows how to optimally solve the optimization problem in this case.

Theorem 3. *Let X be a set of JGD random variables, with average μ_X and correlation matrix Σ_X, the optimal pair of variables $\{x_{j_1}^*, x_{j_2}^*\}$ that minimizes the MSE is given by:*

$$\{x_{j_1}^*, x_{j_2}^*\} = \arg\max_{x_k, x_l \in X} \left(\frac{\sum_{i=1}^N (\sigma_{ik}^2 \sigma_{ll} + \sigma_{il}^2 \sigma_{kk} - 2\sigma_{ik}\sigma_{il}\sigma_{kl})}{\sigma_{kk}\sigma_{ll} - \sigma_{kl}^2} \right)$$

Proof: The proof is provided in the Appendix. ■

According to Theorem 2, the problem of selecting the optimal single variable can be solved in $O(N^2)$. Similarly, Theorem 3 shows that the optimal pair selection can be solved in $O(N^3)$. In order to reduce the complexity of our solutions, we designed our heuristics using the optimal single variable selection as a building block. However, these heuristics can be easily extended to adopt the optimal pair selection.

B. Heuristics

In this section we describe three heuristics, named Top-W, Top-W-Update and Batch Selection, to select the observed variables. These heuristics have different computational complexity. As we show in Section VII, there is a tradeoff between computation complexity and estimation performance.

1) **Top-W***:* In this heuristic we use the optimal single variable selection analysis provided in Theorem 2 to rank the N variables. In particular, we assign to each variable x_j a weight w_j defined as:

$$w_j = \frac{1}{c_j} \left(\frac{\sum_{i=1}^N \sigma_{ij}^2}{\sigma_{jj}} \right) \tag{7}$$

Intuitively, the weight represents the gain provided by a variable when it is individually selected, normalized by its cost. We rank variables according to their weights in a decreasing order. The heuristic selects variables following the order until the budget allows.

Top-W is optimal for the case of homogeneous costs and a budget that enables to select at most a single variable. However, as our experiments in Section VII confirm, it may perform poorly in some scenarios because it does not take into account the contribution to the estimation error of the already selected variables. As a result, it may select variables which have individually a high weight, but are not the best selection when considered together.

Top-W requires $O(N^2)$ to calculate the weights, $O(N \log N)$ to sort them and $O(N)$ to select the variables. Hence, the overall complexity is $O(N^2)$.

2) **Top-W-Update***:* Top-W-Update works in iterations. At every iteration, it uses the weights similar to Top-W, but it updates such weights every time a variable is selected. At the first iteration, the weights are calculated according to Eq. 7. The variable x_1^* with the highest weight among those whose costs do not exceed the budget is selected, and the conditional covariance matrix of the remaining $N - 1$ variables given

Algorithm: Top-W-Update

Input: Set of variables X, covariance matrix Σ_X, budget B.
Output: Set S of selected variables.

1 $S = \emptyset$;
2 $Y = X$;
3 $\Sigma_Y = \Sigma_X$;
4 $i = 1$;
5 **while** $Y \neq \emptyset$ **do**
  // Weight update
6 **for** $x_l \in Y$ **do**
7 $w_l = \frac{1}{c_l}\left(\frac{\sum_{i=1}^{N}\sigma_{il}^2}{\sigma_{ll}}\right)$
  // Select variable with max weight
8 $x_i^* = \arg\max_{x_l \in Y} w_l$
9 **if** $c_{x_i^*} < B$ **then**
10 $S = S \cup \{x_i^*\}$;
11 $Y = Y \setminus \{x_i^*\}$;
12 $B = B - c_{x_i^*}$;
  // Covariance matrix update
13 $\Sigma_Y = \Sigma_{YY} - \Sigma_{Yx_i^*}\Sigma^{-1}_{x_i^*x_i^*}\Sigma_{x_i^*Y}$;
14 **else**
15 $Y = Y \setminus \{x_i^*\}$;
16 $i++$;
17 **return** S

x_1^* is calculated. At the next iteration, the weights of the unselected variables are recalculated by using the conditional covariance matrix. We then select the variable x_2^* with the highest weight if the budget allows, as in the previous iteration. If x_2^* is selected, the new conditional distribution of the remaining variables is calculated. Iterations continue until no more variables can be selected.

The pseudo-code of Top-W-Update is shown in Algorithm Top-W-Update. The algorithm takes as input the set of variables X, the covariance matrix Σ_X and the budget B. The set S contains the variables selected up to the current iteration, while the set Y contains the variables not already selected. The matrix Σ_Y represents the conditional covariance matrix of the unselected variables in Y given the variables in S. Initially, it is equal to the matrix Σ_X (line 3).

At the i-th iteration, we calculate the weight considering the current covariance matrix (lines 6-7) and select the variable x_i^* with the maximum weight (line 8). If the cost of x_i^* is within the available budget (line 9), x_i^* is selected (line 10-11) and the available budget is updated (line 12). Next, we calculate the distribution of the remaining variables in Y, given the selected variable x_i^* (line 13). The conditional covariance matrix Σ_Y is updated given the new selected variable x_i^* as follows:

$$\Sigma_Y = \Sigma_{YY} - \Sigma_{Yx_i^*}\Sigma^{-1}_{x_i^*x_i^*}\Sigma_{x_i^*Y} \tag{8}$$

where Σ_{YY} is the covariance matrix of the variables in Y, $\Sigma_{x_i^*x_i^*}$ is the variance of x_i^*, $\Sigma_{Yx_i^*} = \Sigma^T_{x_i^*Y}$ is the vector of the covariance between the variables in Y and x_i^*. The updated matrix Σ_Y is used at the next iteration to select the next variable.

The selection process continues until no more variable can be selected. Top-W-Update performs N iterations, at each iteration it recalculates the weights ($O(N^2)$), selects the variable with the highest weight ($O(N)$) and updates the covariance matrix ($O(N^2)$). As a result, the overall complexity is $O(N^3)$.

Algorithm: Batch Selection

Input: Set of variables X, covariance matrix Σ_X, budget B.
Output: Set S of selected variables.

1 $S = \emptyset$;
2 $Y = X$;
3 $i = 1$;
4 **while** $Y \neq \emptyset$ **do**
5 $x_i^* = \arg\min_{x_l \in Y} tr(\Sigma_{Y\setminus\{x_l\}|S\cup\{x_l\}}) \times c_l$
6 **if** $c_{x_i^*} < B$ **then**
7 $S = S \cup \{x_i^*\}$;
8 $Y = Y \setminus \{x_i^*\}$;
9 $B = B - c_{x_i^*}$;
10 **else**
11 $Y = Y \setminus \{x_i^*\}$;
12 **return** S

3) **Batch Selection***:* The Batch Selection heuristic is also based on iterations and makes use of weights. Differently from Top-W-Update, in Batch Selection the weight of a variable x_l represents the error incurred in estimating the unobserved variables by including x_l in the set of observed variables, multiplied by its cost c_l. At the i-th iteration, the variable x_i^* with minimum cost is selected. Note that this approach differs from Top-W-Update as the variables in S and the current new variable are considered as a whole.

The pseudo-code of the algorithm is shown in Algorithm Batch Selection. The algorithm takes similar input than Top-W-Update and returns the set of selected variables. At each iteration, the algorithm determines the variable that minimizes the weight (line 4). In order to do so, it calculates, for each unselected variable x_l, the trace of the conditional covariance matrix $\Sigma_{Y\setminus\{x_l\}|S\cup\{x_l\}}$ as follows:

$$\Sigma_{Y\setminus\{x_l\}|S\cup\{x_l\}} = \Sigma_{Y_lY_l} - \Sigma_{Y_lS_l}\Sigma^{-1}_{S_lS_l}\Sigma_{S_lY_l} \tag{9}$$

where $Y_l = Y \setminus \{x_l\}$ and $S_l = S \cup \{x_l\}$. At the i-th iteration, Batch Selection considers the variable x_i^* that minimizes the trace of the corresponding conditional matrix, multiplied by the cost of selecting that variable. The variable is selected if its cost is within the available budget (line 6).

This heuristic performs at most N iterations. At each iteration it calculates the conditional covariance matrix for each variable x_l, which requires to invert the matrix $\Sigma_{S_lS_l}$ and has a complexity $O(N^3)$ for each inversion. The overall complexity is then $O(N^5)$.

V. Change Detection Mechanism

In real scenarios the statistical distributions of the random variables may change over time. As an example, in an outdoor deployed sensor network measuring the light intensity, the position of the sun and the clouds may substantially change the distribution of the collected measurements. In addition, the approximation of these variables as JGD, with the statistics measured in the training interval, may not hold for long time windows. If such changes are not detected, and proper actions are not taken, the performance of the estimation framework

may significantly worsen over time, as we show with real datasets in Section VII.

In this Section we introduce an online change detection mechanism based on the Welch's t-test [17]. The test is performed on the variables selected by our heuristics. The goal is to verify if the distribution of these variables changes during the operational interval. If a change is detected, a new training interval is started after which new variables are selected[3].

During a training interval of size T_{tr}, we calculate the *training distributions* of the variables in X. In particular, for each $x_i \in X$, we calculate the average μ_i^{TD} and the variance σ_i^{TD} over T_{tr} samples. On the basis of this information, we select the variables using one of the heuristics described in Section IV-B. As soon as the training interval ends, we keep track of the *observed distribution* of each of the selected variables. In particular, we calculate the observed mean μ_i^{OD} and the observed variance σ_i^{OD}. We refer to n^{OD} as the number of time slots during which the observed distribution is measured. μ_i^{OD}, σ_i^{OD} and n^{OD} are updated at each time slot on the basis of the observed realization of the selected variables.

The idea of the test is to verify, for each selected variable x_i, weather the distribution $< \mu_i^{\text{TD}}, \sigma_i^{\text{TD}} >$ and $< \mu_i^{\text{OD}}, \sigma_i^{\text{OD}} >$ belong to the same population (*null hypothesis*) or not (*alternative hypothesis*).

The Welch's t-test defines a parameter t, which depends on the two distributions. We perform the test for each observed variable x_i and calculate the value of t_i as follows:

$$t_i = \frac{\mu_i^{\text{TD}} - \mu_i^{\text{OD}}}{\sqrt{\frac{\sigma_i^{\text{TD}}}{T_{tr}} + \frac{\sigma_i^{\text{OD}}}{n^{\text{OD}}}}} \tag{10}$$

According to the Welch's test, for each t_i we can estimate the degree of freedom ν_i as follows:

$$\nu_i \approx \frac{(\frac{\sigma_i^{\text{TD}}}{T_{tr}} + \frac{\sigma_i^{\text{OD}}}{n^{\text{OD}}})^2}{\frac{(\sigma_i^{\text{TD}})^2}{T_{tr}^2(T_{tr}-1)} + \frac{(\sigma_i^{\text{OD}})^2}{(n^{\text{OD}})^2(n^{\text{OD}}-1)}} \tag{11}$$

We want to determine if the alternative hypothesis is verified with a given probability α. For each t_i and ν_i we can determine, by using Student's t distribution tables, the value β_i such that if $t_i > \beta_i$ then the alternative hypothesis is true with probability α.

The above described test is used to detect changes in operational intervals as explained in the following. At each time slot, we observe the realization of the selected variables and we estimate the values of the remaining variables. For each selected variable, we update its observed distribution $< \mu_i^{\text{OD}}, \sigma_i^{\text{OD}} >$ and perform the test. If the test verifies the alternative hypothesis for at least one selected variable, i.e. the distribution calculated in the training interval is different than the observed distribution, we consider that a change has occurred. Hence the current information is stale and needs to be recalculated by means of a new training interval.

The tests assumes that variables are normally distributed. In Section VII we show using real datasets how this test is robust even when this assumption is not perfectly met.

[3]We selected the Welch's t-test since it performs well in the considered scenarios. Similar tests, such as the Chi-square and the Student's t-test, could also be used. Our approach can be easily extended to other tests as well.

VI. A RECENT COMPRESSED SENSING APPROACH

In this section we describe the approach proposed in [6] which we use for performance comparison to our heuristics. We select this approach since it is closely related to our work and it shows particularly good performance. The approach is based on a combination of principal component analysis and compressed sensing. Similar to our work, time is divided into training intervals and operational intervals, and nodes are represented in a set X.

During a training interval, the covariance matrix Σ_X and the average vector μ_X are calculated. Σ_X is used to calculate the matrix U, defined as the orthonormal matrix whose columns are the unitary eigenvectors of Σ_X placed according to the decreasing order of the corresponding eigenvalues. Let us denote by $\mathbf{x}$ the vector of variable realizations, we can define $\mathbf{q} = U^T(\mathbf{x} - \mu_X)$. Based on compressed sensing theory, if measurements are correlated then the vector $\mathbf{q}$ is sparse, i.e. has few non-zero coefficients.

During operational intervals, at each time slot a *random* set of nodes is selected as monitors. Let $\mathbf{s}$ be the vector of readings provided by such monitors. The approach calculates an estimation $\hat{\mathbf{q}}$ of $\mathbf{q}$ given $\mathbf{s}$. This is used to calculate an estimation $\hat{\mathbf{x}}$ of the original signal $\mathbf{x}$ as follows:

$$\hat{\mathbf{x}} = \mu_X + U\hat{\mathbf{q}}$$

We refer the reader to [6] for further details.

VII. EVALUATIONS

In this section we evaluate the performance of our framework. To this purpose we consider two synthetic datasets and two real datasets. We generate the first synthetic dataset by using 18 JGD random variables. For each variable, we randomly select the mean in the interval [600,1000], while the elements of the covariance matrix are randomly chosen in the interval [3000,15000]. The second synthetic dataset is generated using 18 multivariate *exponentially distributed* random variables. We we use a parameter $\lambda = 5$ and scale the values by a factor 200 to get a similar range than the previous dataset. We generated correlated exponential variables with the same correlation matrix of the first dataset using the technique described in [18].

The first real dataset is obtained from the sensor network testbed developed in [19]. The authors deployed 18 solar sensors on five buildings in the University of California Merced Campus. Sensors recorded the light intensity every 5 seconds for several days. We hereafter refer to this dataset as the *sensor dataset*. The second dataset is collected from a real world data center in a production environment that hosts business analytics applications. The trace records the CPU utilization of 74 servers where measurements were taken once every 15 minutes over a period of 83 days[4]. In the following, we refer to this dataset as the *data center dataset*.

In the experiments, we assume a homogeneous unitary cost model, i.e. each variable has cost 1. In this setting, the budget is expressed as the maximum number B of variables that can be selected.

[4]We cannot provide additional information on this dataset for privacy reasons.

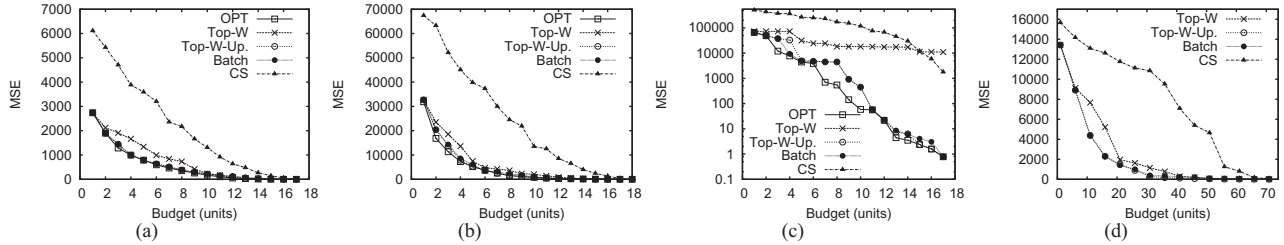

Fig. 2. MSE varying the available budget. Synthetic traces: Gaussian distribution (a) and exponential distribution (b). Real traces: sensor dataset (c) and data center dataset (d).

A. Evaluation of variable selection heuristics

In this section we compare the performance of Top-W, Top-W-Update and Batch selection against the compressed sensing approach described in Section VI (referred to as CS). For all datasets we use a training interval of length $T_{tr} = 500$ time slots[5]. During the training interval, we calculate the average vector μ_X and the covariance matrix Σ_X.

In order to better focus on the performance of the variable selection methods, in these experiments we fix the length of the operation interval to $T_{op} = 500$ and we disable the change detection mechanism. We use the heuristics to select B variables on the basis of the information gathered in the training interval. We use these variables to estimate the unobserved variables for the entire length of the operational interval, using the framework explained in Section III-A.

The methods are compared in terms of the MSE incurred in estimating the unobserved variables. We also consider the optimal strategy (OPT in the figures) which selects the set of B variables that minimize the MSE. The optimal set is obtained through a brute force computation.

Figure 2 (a) shows the results obtained from the first synthetic dataset with JGD random variables, where MSE's are shown on the y-axix as we increase the available budget in the x-axis. CS has the worst performance. Since this approach randomly selects monitors at each round, it may often select nodes which do not allow to calculate a good estimation of the vector $\mathbf{q}$ and consequently a good estimation of the unobserved measures. As expected, all our heuristics are optimal for $B = 1$, since costs are homogeneous and they make use of the single optimal measurement analysis provided in Section IV-A. Top-W performs worse than the other heuristics since at each iteration it selects the new variable without considering the impact of the variables already selected. As a result, the selected variable may not provide significant improvement. On the contrary, Top-W-Update and Batch Selection show performance close to the optimum since at each iteration they both take into account the effects of the previously selected variables to select the next one.

Figure 2 (b) shows the results for the second synthetic dataset with exponentially distributed random variables. The purpose of these experiments is to test our framework in a scenario where variables explicitly violate the jointly Gaussian distribution assumption. Also in this case, our heuristics significantly outperform CS. Additionally, the results show that, even with exponentially distributed traces, our approach incurs a low MSE when the budget allows to select at least one third of the variables.

Figure 2 (c) shows the results for the sensor dataset. Differently from the synthetic traces, in which the variable correlation is relatively uniformly distributed, in this dataset variables have skewed correlation as a consequence of the sensors' geographical distribution. As a result, more sensors are required by all approaches to achieve a low MSE. CS is strongly penalized by the skewed correlation distribution (note the log-scale on the y-axis), since poor monitor selections result in an even higher estimation error than in the synthetic case. Top-W shows a step wise behavior. This is due to cluster of sensors located on close buildings which are highly correlated and provide similar performance in estimating other variables. Since this heuristic does not update variables weights, it selects most of the sensors in a cluster before moving to other sensors on different buildings. As a result, until B is not sufficiently large to force the algorithm to select sensors on other buildings, it keeps selecting sensors on the same cluster, not improving the overall MSE. On the contrary, Top-W-Update and Batch Selection select a proper set of variables and achieve performance close to the optimal policy. Batch Selection has slightly better performance than Top-W-Update in this scenario. Since Batch Selection considers the overall decrease in the estimation error for each variable, it is more effective when measurement correlation are particularly skewed, although it incurs a higher computational complexity.

Let us now focus on the particular case $B = 4$ for the sensor dataset. As Figure 1 (a) suggests, this datasets has 4 main clusters composed by the following sensors IDs: $\{7\}, \{4, 5\}, \{14\}, \{0, 1, 2, 3, 6, 8, 9, 10, 11, 12, 13, 15, 16, 17\}$. This structure reflects the sensor geographical distribution [19]. When $B = 4$ it would be desirable to select a sensor in each cluster. The random selection of CS is likely to select most sensors from the largest cluster, thus resulting in a high estimation error, as show in Figure 2 (c). Top-W selects the sensors 10, 11, 8 and 9. These are all located in the largest cluster, hence they provide a poor estimation of the other sensors readings. Nevertheless, Top-W still outperforms CS, highlighting the superiority of our estimation framework. Top-W-Update, instead, selects 10, 4, 7 and 0. It spreads the selected sensors on 3 different clusters, achieving a significantly lower MSE than Top-W and CS. Finally, Batch Selection selects 0, 4, 7 and 14. It successfully selects one

[5]The setting $T_{tr} = 500$ is suitable for all scenarios and datasets considered in this paper. In practice, the length of the training interval may be application dependent and may require tuning. The development of a self-tuning strategy will be addressed in our future work.

sensor per cluster, resulting in an MSE close to the optimum at the expense of an increased computational complexity. Note that, Top-W-Update successfully covers all clusters for the case $B = 5$.

Figure 2 (d) shows the results for the data center dataset. In this dataset, measurement correlations are more uniformly spread than in the sensor case. As a result CS and Top-W achieve better performance than with the sensor dataset. Top-W-Update and Batch selection achieves similar performance and incur in a low error as the budget increases. We do not show the optimal policy for this dataset due to the excessive running time required by the brute force computation.

Overall, Figures 2 (c) and (d) show that in real networks our approach can reduce the network monitoring overhead by more than 50% during operational intervals. In fact, for the sensor dataset, 9 out of 18 sensors are sufficient to incur in a MSE of few units. Similarly, for the datacenter dataset, only 30 out of 74 servers are needed for a good estimation of the remaining resources.

B. Evaluation of the change detection mechanism

In this section we evaluate our change detection mechanism. We fist consider synthetic traces, for which changes can be artificially controlled. Then, we show that our method is effective also with real measurement data.

We use a training interval of length $T_{tr} = 200$ time slots, while the length of the operational interval is determined by our change detection mechanism. In particular, an operative interval is terminated, and another training interval of length T_{tr} is started, as soon as a change is detected. We set $\alpha = 0.95$, hence we detect a change when the training distribution of at least one observed variable does not match with the observed distribution with 0.95 probability.

In the experiments we set $B = 8$ for the synthetic and sensor dataset, and $B = 30$ for the data center dataset. We use the Top-W-Update heuristic to select variables. We observed similar results with Batch Selection and with the optimal selection. In order to highlight the time variation of the error due to occurring changes, in the results we calculate the MSE over a time window of 20 time slots.

We first study the stability of our online change detection mechanism on synthetic traces with JGD random variables in a steady state, i.e. when no change occurs. The results are shown in Figure 3 (a). The mechanism successfully detects no change: after the initial training interval no other training is required and the same set of variables is used to estimate the remaining variables. It is worth noting that, since the variables are actually JGD, the actual MSE is close to the expected MSE which is given by the trace of the correlation matrix Σ_X, calculated during the training interval.

Next, we investigate the time needed to detect a change. To this aim, we use synthetic traces where we generate a new distribution for each variable and a new covariance matrix every 500 time slots. Results are shown in Figure 3 (b). As the figure shows, the MSE suddenly increases as a change occurs, due to the stale information used for the estimation. However, our mechanism successfully detects the change in few time slots and initiates another training interval. At the end of the training interval, a new set of variables is selected by our heuristics for the next operational interval.

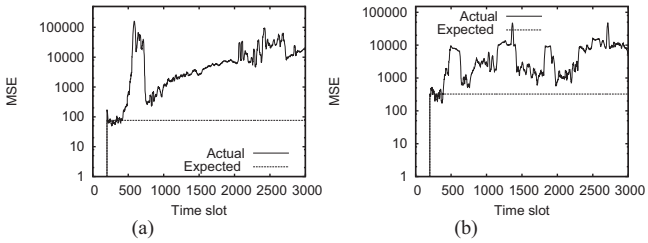

Fig. 4. MSE over time - Real traces with no update of the selected variables: sensor dataset (a) and data center dataset (b).

We performed similar experiments for synthetic traces with exponentially distributed random variables. Figure 3 (c) shows the steady state when no change occurs in the variable distributions. Although the incurred MSE is higher than with JGD variables, the change detection mechanism is robust and correctly identifies no changes. Figure 3 (d) shows the case of periodic changes every 500 time slots. Also in this case the mechanism rapidly detects a change as it occurs, and triggers the start a new training interval.

In order to highlight the benefit of our change detection mechanism in real scenarios, we first show the MSE over time that would be incurred by not updating the information with the two real datasets. Figure 4 (a) and (b) shows the results. In both cases, the MSE is initially close to the expected value, as the variable distributions can be successfully approximated as JGD and their distribution remains stable for a period of time. However, several factors may change these conditions. As an example, for the sensor dataset these factors include the position of the sun and the movements of the clouds, while for the datacenter dataset the workload distribution and the job allocation. As a result, in both cases the estimation performance of the prediction mechanism significantly worsen over time (note the logarithmic scale on the y-axis).

We now consider the performance of our approach by enabling the change detection mechanisms in the same scenarios. Figures 5 (a) and (b) show the MSE over time for the sensor dataset and the datacenter dataset, respectively. In both cases, changes are successfully detected and the incurred error remains relatively stable until a new training interval is started. It is noteworthy that the MSE is up to two orders of magnitude lower than the error incurred when no update is performed.

VIII. Related Work

The development of a framework for the reduction of the monitoring overhead through the estimation of unobserved measurements has been mainly investigated using *geometric models* or *compressed sensing*. Geometric models [4], [5] are usually considered in the context of wireless sensor networks and exploit the spacial correlation of measurements in the estimation process. In practice location network information may not be available or accurate, and more importantly these techniques cannot be applied to virtual environments such as data centers where there is no notion of distance.

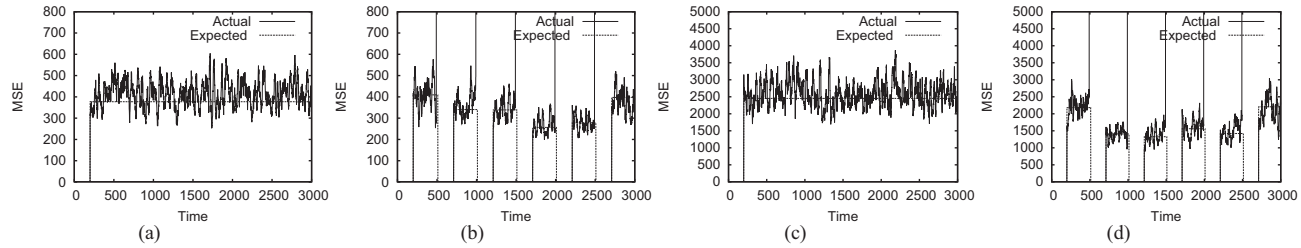

Fig. 3. MSE over time - Synthetic traces. Gaussian distribution: steady state (a) and periodic changes every 500 time slots (b). Exponential distribution: steady state (a) and periodic changes every 500 time slots (b).

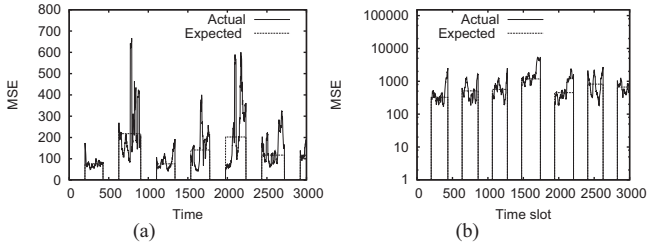

Fig. 5. MSE over time - Real traces with change detection enabled: sensor dataset (a) and data center dataset (b).

Compressed sensing based approaches [6]–[12] reconstruct the signal, i.e. the full set of measures, from a reduced representation in a vectorial space with less dimensions. Recent advances of this technique focus on the estimation of multi-dimensional signals [9]–[11]. These approaches mainly focus on the reconstruction of the original signal, and only marginally address the monitor selection problem. We use the approach proposed in [6] for comparisons with our solutions in this work.

Other works related to ours consider the problem of selecting the optimal subset of monitors from a given set in order to minimize an error metric. Amongst these, the closest to our work are ones that consider the multivariate jointly Gaussian framework [15], [16], [20]–[23]. The work in [15], [16] studies the problem of optimizing sensor placements to monitor Gaussian Processes, where the objective is to maximize the *mutual information* between the chosen locations and the locations which are not selected. This problem is shown to be NP-Hard and a greedy selection algorithm is shown as $(1-1/e)$-approximate. Similar $(1-1/e)$ approximation results are obtained for alternate objectives of minimizing the log volume of confidence ellipsoid for a sensor selection problem in [20] and for the V-optimality criterion in the context of batch active learning in [21]. All of these results are obtained by showing the *submodularity* of these objective functions and then leveraging the celebrated result by Nemhauser et al. [14] that shows the universal $(1 - 1/e)$ approximation bound for a greedy selection algorithm on *any* monotone submodular set function. As we show in Section III-B, the MSE function considered in this work is not submodular, hence the above cited results cannot be applied to our problem.

The problem of selecting a set of monitors has been considered also in the context of *network tomography* [24], [25]. These approaches aim at estimating the performance of the internal part of the network given the end-to-end measurements provided by monitors. Our approach is complementary to theirs, and could be used to further reduce the overhead of network tomography.

The work in [22], [23] considers the MSE metric for the subset selection problem and shows that under certain structural properties of the covariance matrix, the performance of a greedy selection algorithm can be characterized in terms of an approximate notion of submodularity called submodularity ratio. These works focus on scenarios where the correlation matrix is assumed to be stationary. However, as shown by our experiments, this is far from true in real-world datasets. In this work, we design an online learning based approach that automatically switches between learning and estimation phases using a change detection algorithm.

Other works in the context of sensor networks make use of measurement correlation [26], [27]. Differently from our work, these approaches focus on minimizing the network energy consumption of the network and do not specifically target the estimation error.

IX. Conclusions

In this paper we propose an online method to reduce the network monitoring overhead by exploiting measurement correlation. We model measurements as realizations of JGD random variables and propose an estimation framework to estimate the unobserved variables given the correlation and the realization of the observed variables. We show that selecting the best set of variables is NP-Hard and propose three heuristics based on a special case for which the problem can be solved efficiently. We also propose an online change detection method in order to deal with time-varying measurement distributions and correlation. We validate our approach on synthetic and real traces against a previous solution based on compressed sensing. Results show that our method outperforms the previous solution and it reduces the monitoring overhead up to 50% while incurring in a low estimation error. In addition, we show that the change detection algorithm reduces the estimation error up to two orders of magnitude compared to the case in which changes are not detected.

Appendix

Theorem 3. *Let X be a set of JGD random variables, with average μ_X and correlation matrix Σ_X, the optimal pair of variables $\{x^*_{j_1}, x^*_{j_2}\}$ that minimizes the MSE is given by:*

$$\{x^*_{j_1}, x^*_{j_2}\} = \arg\max_{x_k, x_l \in X} \left(\frac{\sum_{i=1}^{N} (\sigma_{ik}^2 \sigma_{ll} + \sigma_{il}^2 \sigma_{kk} - 2\sigma_{ik}\sigma_{il}\sigma_{kl})}{\sigma_{kk}\sigma_{ll} - \sigma_{kl}^2} \right)$$

Proof: Let us consider two selected variables x_k and x_l, let $S = \{x_k, x_l\}$ and $Y = X \setminus S$. Using the JGD estimation framework the conditional covariance matrix $\Sigma_{Y|X}$ of the remaining $(N-2)$ variables is given by $\Sigma_{Y|S} = \Sigma_{YY} - \Sigma_{YS}\Sigma_{SS}^{-1}\Sigma_{SY}$, where Σ_{YY} is the $(N-2)\times(N-2)$ covariance matrix between the unobserved $(N-2)$ variables, $\Sigma_{YS} = \Sigma_{SY}^T$ is the $(N-2)\times 2$ covariance matrix between the $(N-2)$ unobserved and the 2 selected variables, and Σ_{SS} is the 2×2 covariance matrix between the 2 selected variables. Σ_{SS}^{-1} can be expressed as:

$$\Sigma_{SS}^{-1} = \frac{1}{\sigma_{kk}\sigma_{ll} - \sigma_{kl}^2} \begin{bmatrix} \sigma_{kk} & -\sigma_{kl} \\ -\sigma_{kl} & \sigma_{ll} \end{bmatrix}$$

By expanding the product term $\Sigma_{YS}\Sigma_{SS}^{-1}\Sigma_{SY}$, the diagonal entries d_{ii} are of the form:

$$d_{ii} = \frac{\sigma_{ik}^2\sigma_{ll} + \sigma_{il}^2\sigma_{kk} - 2\sigma_{ik}\sigma_{il}\sigma_{kl}}{\sigma_{kk}\sigma_{ll} - \sigma_{kl}^2} \quad \forall i \in \{1, \ldots, N\}, i \neq k, l$$

Let MSE_S represent the estimation error incurred using the variables in S, x_k and x_l. Since the MSE can be expressed as the trace of the matrix $\Sigma_{YY} - \Sigma_{YS}\Sigma_{SS}^{-1}\Sigma_{SY}$, we have

$$\begin{aligned} \text{MSE}_S &= \sum_{\substack{i=1 \\ x_i \notin S}}^{N} \left(\sigma_{ii} - \frac{\sigma_{ik}^2\sigma_{ll} + \sigma_{il}^2\sigma_{kk} - 2\sigma_{ik}\sigma_{il}\sigma_{kl}}{\sigma_{kk}\sigma_{ll} - \sigma_{kl}^2} \right) \\ &= \sum_{i=1}^{N} \sigma_{ii} - \frac{\sum_{i=1}^{N} \sigma_{ik}^2\sigma_{ll} + \sigma_{il}^2\sigma_{kk} - 2\sigma_{ik}\sigma_{il}\sigma_{kl}}{\sigma_{kk}\sigma_{ll} - \sigma_{kl}^2} \end{aligned}$$

Thus, the optimal pair of variables $\{x^*_{j_1}, x^*_{j_2}\}$ that minimizes the MSE is given as:

$$\{x^*_{j_1}, x^*_{j_2}\} = \arg\max_{x_k, x_l \in X} \left(\frac{\sum_{i=1}^{N} (\sigma_{ik}^2 \sigma_{ll} + \sigma_{il}^2 \sigma_{kk} - 2\sigma_{ik}\sigma_{il}\sigma_{kl})}{\sigma_{kk}\sigma_{ll} - \sigma_{kl}^2} \right)$$

■

REFERENCES

[1] A. A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao, "Towards automated performance diagnosis in a large iptv network," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 231–242, 2009.

[2] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes." *IEEE Communications Surveys and Tutorials*, vol. 7, no. 1-4, pp. 72–93, 2005.

[3] G. Wang and T. E. Ng, "The impact of virtualization on network performance of amazon ec2 data center," *IEEE INFOCOM*, 2010.

[4] N. Bartolini, T. Calamoneri, T. La Porta, C. Petrioli, and S. Silvestri, "Sensor activation and radius adaptation (sara) in heterogeneous sensor networks," *ACM Transactions on Sensor Networks*, vol. 8, no. 3, p. 24, 2012.

[5] M. Umer, L. Kulik, and E. Tanin, "Spatial interpolation in wireless sensor networks: localized algorithms for variogram modeling and kriging," *Springer Geoinformatica*, vol. 14, no. 1, pp. 101–134, 2010.

[6] G. Quer, R. Masiero, G. Pillonetto, M. Rossi, and M. Zorzi, "Sensing, compression, and recovery for wsns: Sparse signal modeling and monitoring framework," *IEEE Transactions on Wireless Communications*, vol. 11, no. 10, pp. 3447–3461, 2012.

[7] G. Coluccia, E. Magli, A. Roumy, V. Toto-Zarasoa *et al.*, "Lossy compression of distributed sparse sources: a practical scheme," *EUSIPCO*, 2011.

[8] J. E. Barceló-Lladó, A. M. Pérez, and G. Seco-Granados, "Enhanced correlation estimators for distributed source coding in large wireless sensor networks," *IEEE Sensors Journal*, vol. 12, no. 9, pp. 2799–2806, 2012.

[9] M. Leinonen, M. Codreanu, and M. Juntti, "Compressed acquisition and progressive reconstruction of multi-dimensional correlated data in wireless sensor networks," *IEEE ICASSP*, 2014.

[10] C. Anagnostopoulos and S. Hadjiefthymiades, "Advanced principal component-based compression schemes for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 11, no. 1, p. 7, 2014.

[11] C. Caione, D. Brunelli, and L. Benini, "Compressive sensing optimization for signal ensembles in wsns," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 382–392, 2014.

[12] G. Coluccia, S. K. Kuiteing, A. Abrardo, M. Barni, and E. Magli, "Progressive compressed sensing and reconstruction of multidimensional signals using hybrid transform/prediction sparsity model," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 3, pp. 340–352, 2012.

[13] A. Gut, *An intermediate course in probability*. Springer, 2009.

[14] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functions," *Springer Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[15] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Near-optimal sensor placements: Maximizing information while minimizing communication cost," *ACM IPSN*, 2006.

[16] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, pp. 235–284, 2008.

[17] B. L. Welch, "The generalization of student's' problem when several different population variances are involved," *JSTOR Biometrika*, vol. 34, no. 1/2, pp. 28–35, 1947.

[18] I. Yahav and G. Shmueli, "On generating multivariate poisson data in management science applications," *Wiley Applied Stochastic Models in Business and Industry*, vol. 28, no. 1, pp. 91–102, 2012.

[19] S. Achleitner, A. U. Kamthe, T. Liu, and A. E. Cerpa, "SIPS: Solar irradiance prediction system," *ACM/IEEE IPSN*, 2014.

[20] M. Shamaiah, S. Banerjee, and H. Vikalo, "Greedy sensor selection: Leveraging submodularity," *IEEE CDC*, 2010.

[21] Y. Ma, R. Garnett, and J. G. Schneider, "Submodularity in batch active learning and survey problems on gaussian random fields," *CoRR*, vol. abs/1209.3694, 2012.

[22] A. Das and D. Kempe, "Algorithms for subset selection in linear regression," *ACM STOC*, 2008.

[23] ——, "Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection," *ACM ICML*, 2011.

[24] L. Ma, T. He, K. K. Leung, D. Towsley, and A. Swami, "Efficient identification of additive link metrics via network tomography," *IEEE ICDCS*, 2013.

[25] S. Tati, S. Silvestri, T. He, and T. La Porta, "Robust network tomography in the presence of failures," *IEEE ICDCS*, 2014.

[26] E. K. Lee, H. Viswanathan, and D. Pompili, "Silence: distributed adaptive sampling for sensor-based autonomic systems," *ACM ICAC*, 2011.

[27] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," *VLDB*, 2004.