Sleeping Coordination for Comprehensive Sensing Using Isotonic Regression and Domatic Partitions

Farinaz Koushanfar[†‡]
[†]Rice Univ., [‡]Univ. of Illinois, Urbana-Champaign

Nina Taft[§]
[§]Intel Research, Berkeley

Miodrag Potkonjak[♮]
[♮]Univ. of California, Los Angeles

Abstract— **We address the problem of energy efficient sensing by adaptively coordinating the sleep schedules of sensor nodes while guaranteeing that values of sleeping nodes can be recovered from the awake nodes within a user's specified error bound. Our approach has two phases. First, development of models for predicting measurement of one sensor using data from other sensors. Second, creation of the maximal number of subgroups of disjoint nodes, each of whose data is sufficient to recover the measurements of the entire sensor network. For prediction of the sensor measurements, we introduce a new optimal non-parametric polynomial time isotonic regression. Utilizing the prediction models, the sleeping coordination problem is abstracted to a domatic number problem and is optimally solved using an ILP solver. To capture evolving dynamics of the instrumented environment, we monitor the prediction errors occasionally to trigger adaptation of the models and domatic partitions as needed. Experimental evaluations on traces of a medium size network with temperature and humidity sensors indicate that the method can extend the lifetime of the network by a factor of 4 or higher even for a strict error target.**

I. Introduction

Energy is the single most important resource and constraint in wireless sensor networks. Numerous techniques for reducing energy consumption and, therefore, enhancing the lifetime of the network have been proposed and examined, including multihop communication, adaptive sampling, compression, and energy-aware routing. Sleeping has been demonstrated to be an exceptionally effective strategy for prolonging the lifetime of the network [1][2][3][4][5][6][7]. Our goal is to develop a new sleeping coordination strategy that would enable nodes to enter the sleep mode in such a way that the overall sensing capabilities of the network are maintained. Maintaining sensing quality is ensured by strategically placing a subset of nodes in sleep mode in such a way that, from the remaining small set of awakened nodes, one can recover the data at the sleeping nodes to within a user specified target error rate.

Our approach has two main components. First, we develop a model, using a new *isotonic regression* approach, for all pairs of nodes such that one node can be used to predict another. Using this predictive model, we build a graph, called a *prediction graph*, in which a directed edge from sensor node i to node j exists only if sensor node i can predict the value that node j senses (e.g., a temperature reading) to within a target error rate. Second, we seek to find subgroups (or partitions) of nodes such that each subgroup can accurately predict the sensed values for the entire network. We propose the idea of choosing these groups to be disjoint dominating sets that are extracted from the prediction graph using an ILP-based procedure. This procedure yields mutually disjoint groups of nodes called *domatic partitions*. The energy saving is achieved by having only the nodes in one domatic set be awake at any moment in time. The different partitions can be scheduled in a simple round robin fashion. If the partitions are mutually disjoint and we find K of them, then the network lifetime can be extended by a factor of K. Because the underlying phenomenon being sensed and the inter-node relationships will evolve over time, we extend our system to monitor for changes. Intermittent periods of monitoring allow us to check our current prediction error and thus initiate an adaptation (recalibration of models and domatic partitions) when necessary.

Given a time series of data measurements from two sensors, it is natural to ask whether the values sensed (e.g., temperature) by one sensor can be predicted by the other, i.e., can sensor Y be predicted via some function of sensor X's data, $Y = f(X)$. Regression analysis uses data samples from both X and Y to find the function f. There are many forms of regression analysis in the statistics literature (discussed in Section IV). In this work we propose the use of isotonic regression. Our regression is isotonic because we impose an isotonicity constraint that says if two sensor readings at sensor X are $x1 \leq x2$, then we require $f(x1) \leq f(x2)$. We find that adding the isotonic constraint controls the search for a good predictor function f in a successful way. This constraint has a very natural interpretation in sensor networks. It is intuitive that if the phenomenon being sensed (e.g., temperature, humidity, light) increases, then both sensors will experience an increase in their measurements.

There are several efficient algorithms for isotonic regression in the statistics literature [8][9][10]. In this paper we propose a new optimal polynomial time algorithm. There are two key advantages to our algorithm. First, previous isotonic regression methods use an unlimited number of parameters (e.g., the number of piece-wise linear components). Methods that limit the number of parameters, could no longer maintain optimality [9][10][11]. Our solution allows the number of parameters to be specified as an input (and hence limited to a small number) while still finding an optimal solution. Limiting the number of parameters is attractive because it avoids overfitting the data and because it requires less memory for model storage. This is important for sensor nodes that have limited storage. The second advantage of the new approach is that it allows the use of any error function to be minimized. Most regression methods work for a specific error function and cannot easily be modified for other error

1-4244-0222-0/06/$20.00 (c)2006 IEEE

functions. Our solution maps the isotonic regression problem to a combinatorial domain. We solve the combinatorial problem using a dynamic programming approach. It is because of this style of solution that our method allows limiting the number of parameters and an arbitrary form of error function.

Our contributions are multiple. First, we propose the idea of using isotonic regression for having one sensor node predict another, and provide a new solution for isotonic regression. Second, we propose the idea of using domatic partitions for defining a sleeping schedule. Third, we link the isotonic regression and domatic partitions together in an overall methodology through the use of a prediction graph. The prediction graph is specified using the output of the isotonic regression, and constitutes the input to the domatic partition problem. Fourth, we formulate the domatic partition problem as a simple ILP. Fifth, we incorporate adaptivity into our method to allow our models to evolve over time. Lastly, we evaluate our method on a real dataset and compare it to two other methods.

We find that using isotonic regression, a sensor node can be predicted by a number of other nodes to within 1% error for temperature and humidity. Since the ability of nodes to predict other nodes is so good in this particular network, there is no need to consider multivariate prediction models in which multiple nodes are used to predict a single node. With a target error rate of 2%, we can extend the lifetime of our network by 5 times using our method without adaptive updates. If larger errors can be tolerated, the network lifetime can be extended a good deal more. By adding in the adaptivity feature, we introduce a tradeoff. The periods of monitoring for change reduce the total network lifetime, but yield more accurate sensing and prediction. Our formulation of the domatic partitioning problem has two major advantages in terms of flexibility and short run time. For example, the ILP formulation can be used for solving the case when each node has several sensors of different modalities and when each node has a different amount of energy. Runtime for all instances that we evaluated, even when the number of nodes was several hundred, were less than a minute on a standard 1GHz PC.

Our techniques are applicable to sensor networks whose sensors are located close enough to each other so that there is overlap among subsets of nodes of the phenomenon being sensed. Examples include environmental monitoring networks such as temperature, light, humidity, etc. monitoring in vineyards, forests, deserts and indoor buildings. Our focus is thus on networks having subsets of sensors with spatial correlations. It is clear that phenomena such as temperature or light vary little over ranges of 10's or 100's of feet (in many environments). We focus on sensor network applications targeting either continuous or periodic monitoring, rather than event-driven (i.e., query based) sensor networks as in [12]. Our methods could also handle bursty events if they are are short in time, as long as the burst was spread over a set of sensors. Bursty events that only affect one sensor would not benefit from a system such as ours. In this paper we use a dataset from an indoor building environment where the sensors have 1-hop communication to a centralized server. We propose an extension of our work for multi-hop and localized environments in [13].

The remainder of this paper is organized as follows. Section II outlines our overall methodology. Section III surveys the related literature. In Section IV we describe our isotonic regression solution, evaluate its performance and compare it to two other regression approaches. Our formulation of the domatic number problem as an ILP, along with an evaluation of the domatic partition solutions is given in Section V. We present our approach for dynamically updating the models and partitions in Section VI. We conclude our paper in Section VII.

II. Methodology

A. Testbed Specification and Assumptions

We use the traces from the sensors deployed in an indoor office environment. There are 54 sensors that are attached to the ceiling spaced anywhere from 6 to 15 feet apart. The nodes are composed of Crossbow MICA-2 motes [14] that implement the processor/radio board. All modules provide a processor that runs TinyOS based code [15], a two-way ISM band radio transceiver, and have memory that can store up to 100,000 measurements. The radio is capable of transmitting at 38.4 kbps. The sensor and data acquisition cards that plug into the Mote Processor Radio boards collect light, temperature and humidity measurements, sampled once every 30 seconds.

The radios on the MICA-2 motes have an outdoor transmission range of around 300m. Even though the radio range decreases in the indoor environment due to the presence of obstacles and interferences, the transmission range of the radios are still more than the distances of the nodes deployed in our lab and their distances to the server. For the purposes in this paper we assume that all sensor nodes can directly communicate to servers. The important assumption is that the energy consumption of the nodes is dominated by the energy consumption of the radio unit. Several studies have shown that the radio consumes a significant amount of energy even when it is in the idle mode [2]. Thus, the most efficient way to save power on the nodes is to completely shut the radio down and put the CPU in power save mode. The CPU wakes up periodically to read the samples from sensors. The sampling task takes around 15ms and the rest of the time nodes can stay in the power save mode with their radios down.

B. Approach

In Figure 1, we depict the overall flow of our proposed methodology and indicate how the various components of our approach are organized. We now explain this figure step by step. We assume there is an initial phase in which all nodes are awake. The initial phase is used to gather data so that we can build models. We have chosen our initial phase to last for 48 hours; the reason is because our testbed network (unsurprisingly) exhibits strong diurnal patterns and thus it is important to examine, at a minimum, one full day's worth of behavior. Our second day is used to validate the model. Using isotonic regression, we build models for all pairs of nodes, in which one node predicts the other. Given the models, we can know all the nodes that can be used to predict any single node. The output of the modeling phase is a prediction graph in which a directed edge between a pair of nodes (i, j) exists if node i can predict node j well.

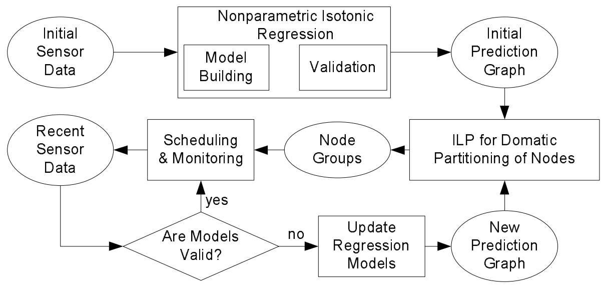


Fig. 1. Overall flow of adaptive modelling and sleep coordination approach.

The prediction graph is the input to our ILP step in which we find as many disjoint dominating sets as possible. The idea is that one dominating set can be used at a time to predict the values of all the sensor nodes. Energy can be saved because while the nodes in one dominating set are awake, all other nodes are put to sleep. Our ILP solution outputs these node groups, that are then organized into a round robin schedule by the scheduling step. In our evaluations, we ran each dominating set for a period of one day, and then cycled to the next one. Since we do not assume any extra cost for transitioning the node from sleep to active, the amount of energy saving is independent of the length of the round robin periods. If the sleep-active transition cost is not insignificant, then the active period for each dominating set can be extended to longer times such as a week or even longer.

The phenomenon that these networks are sensing are, by their very nature, going to undergo change. Temperature and humidity are clearly affected by seasonal changes. Also, the inter-node relationships can change if, for example, a heater or cooler is installed in an indoor environment. There can also be micro-effects that cause the inter-node relationships to change. Thus a system such as ours requires a monitoring mechanism to detect change and launch a readjustment of the underlying models.

In our method, the monitoring box randomly schedules short intermittent period when all the nodes are turned on. The "recent" data obtained during a monitoring period from a given node i, is compared to the prediction generated by the node assigned to predict node i during the evaluation period. If the match is good (i.e. the prediction error does not exceed the target error), we decide the current models are valid and return to the original schedule. If the match is poor, then we update our models by leaving all the nodes on for a *collection period* whose length depends upon when enough new data has been collected to bring the prediction error under the target. This new data produces a new prediction graph. If the new graph is different from the old one, the new graph is fed back into the ILP step to recompute a new collection of dominating sets.

III. Related Work

Statistical modeling has been used to enrich data acquisition and query processing applications [12][16]. Even though the query processing optimization problems are very different from our sleeping coordination problem, the underlying models are relevant to our work. Deshpande et al. [12] propose an interactive sensor querying system that is based upon statistical models. Deshpande et al. further introduce a prototype for model-based querying called BBQ that uses a specific model based on time-varying multivariate Gaussian (MVG). BBQ uses historical data to construct the initial representation of the Gaussian probability density function (PDF). The authors illustrate examples of applications of MVG modeling on static sensor networks, to perform range queries, value queries and average aggregates. Guestrin et al [16] propose a distributed regression framework in sensor networks. The algorithm is based upon Kernel linear regression, where the model takes the form of a weighted sum of local basis functions. They define basis functions that model the phenomena within each local spatial region and smooth the overlaps of the local models by a Gaussian elimination algorithm.

Aside from the obvious difference in application scenario, there are other important differences between our modeling approach and those of [16], [12]: (1) We do not impose strong prior model assumptions, such as a Gaussian distribution on data. Rather, we use data-driven non-parametric statistical modeling techniques to capture the precise relationships among the sensors. (2) We identify and utilize the isotonicity constraint as a means to capture the hidden covariates (dependencies) that influence the relationships between sensors. (3) We do not use models that incorporate distance since we observed in our application data, a substantial amount of inconsistency between correlations across sensors and their inter-sensor distance. (4) Both BBQ and distributed regression utilize models based on Gaussian distribution that minimizes the L_2 error norms. Our modeling approach can handle many types of errors, including L_2 and others. This is beneficial since the L_2 error norm is known to be very sensitive to outliers common in sensor data.

The early work in energy efficient sensing has focused on sensor deployment and fault tolerance [6]. More recently, there have been a number of proposals for placing a subset of nodes into a sleep mode while maintaining a suitably defined objective function that captures sensing coverage. The proposed sleeping approaches cover a broad range: [3], [7] use an asymptotic theoretical analysis that assumes stationary Poisson point processes for deployment in two-dimensional field, [17], [5] emphasize localized sleeping algorithms for distributed detection, [2] focuses on system issues and prototyping, while [1] targets sleeping while detecting rare events. In addition, there have been efforts to simultaneously address communication coverage and sensing surveillance [4]. The common denominator for all the previous efforts is: (1) they use variants of disk sensing coverage models, or (2) they emphasis ensuring that a movable target is detected. In contrast, our approach is data-driven and does not place any prior assumptions on the sensing coverage. We define complete coverage as the maximum coverage attainable while all sensors are awake. Our goal is to ensure that within a user's specified error range, the complete coverage is maintained while a significant number of nodes are placed in the sleep state.

In many scientific modeling scenarios, physical considerations suggest that the response variable is an isotonic (monotonic) function of explanatory variables [9][10][11]. The earliest and most widely used nonparametric algorithm for isotonic regression is pool adjacent violators (PAV) [8]. The approach is designed for univariate regression and works by sorting the explanatory and then averaging the responses that violate the isotonicity requirements. The PAV algorithm is prone to misadjusting the models to the outliers. To alleviate this problem, several heuristic smoothing methods have been proposed [9][10][11]. We have developed a new isotonic regression approach that solves the problem by mapping it into the combinatorial domain. By addressing the problem this way, we are able to find in polynomial time, a model that is provably minimal for an arbitrary error norm. One of the most effective ways to smooth a function is to restrict the number of parameters used. We can find the optimal smooth isotonic regression in polynomial time, by forming a model that is minimal in error for a limited number of parameters.

IV. Inter-sensor modeling

An attractive way to capture the complex relationships between distributed sensor measurements is to use data-driven statistical modeling techniques. The readings at a sensor s_Y could be related to numerous other elements of the networks, including multiple sensor nodes, spatial coordinates (denoted by a generic x, y, z), and time τ; hence the goal of a data-driven model is to uncover the mapping function $s_Y = f(s_{X1}, s_{X2}, x, y, z, \tau)$. We use the term *response* variable to refer to the measurements at sensor s_Y, and the term *explanatory* variables to refer to each of the arguments inside the mapping function f.

There are two broad classes of statistical modeling techniques that could be used: (i) parametric, and (ii) non-parametric models. Parametric statistical models assume that the form of mapping function f is known with the exception of a limited number of parameters. Nonparametric models differ from parametric models in that the mapping function is not specified a priori, but is instead determined from data. The term nonparametric is not meant to imply that such models completely lack parameters; rather, the number and nature of the parameters is flexible and not fixed in advance. Nonparametric models are also called distribution free.

Parametric statistical modeling techniques have the advantage of being simple and of having a limited number of parameters. However the performance achieved is based on specific assumptions about the form of the mapping function and the distribution of the variables. If these strong modeling assumptions do not hold, the performance of such methods could be quite poor. The advantage of nonparametric models is that they place much milder assumptions on the variables. However their potential drawback is that they can potentially explode in terms of the number of parameters used to describe the model. A large number of parameters is not desirable as there is a danger of over-fitting the data.

We build a nonparametric model that finds a piece-wise linear fit to model the response variable. Our regression method is isotonic because we impose isotonicity constraints, and it is univariate because we will build models that rely on a single explanatory variable, thus keeping things simple. Unlike previous methods, our solution explicitly allows the number of linear components to be limited (specified in advance), thereby avoiding the problems of an excessive number of parameters in nonparametric methods.

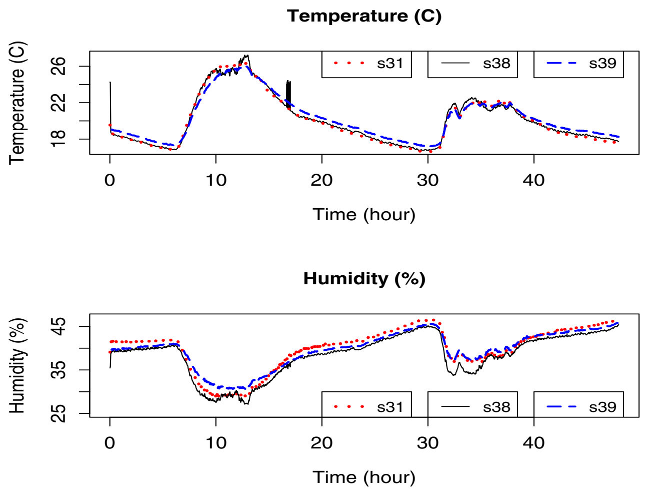


Fig. 2. Similar temperature and humidity readings for 3 sample sensors (31, 39 and 38) hint of correlation and predictability.

A. Exploratory Data Analysis

We begin our development of statistical models for pairs of sensor nodes (called inter-sensor models) by doing some exploratory data analysis. This initial phase of examining the data is used to try to uncover correlations (or lack thereof) that can be exploited in modeling, to identify which variables can act as explanatory variables (to predict the response variable), as well as to expose hidden covariates (correlations or dependencies). We observed the following.

• *Spatial and Temporal Correlations* Figure 2 shows temperature and humidity measurements at sensors 31, 39, and 38, for the first two days of the experiment. These plots reveal the existance of spatial correlations across nodes in that different nodes behavior similarly. The readings at individual sensors are correlated in time because the phenomenon being sensed is a slowly changing phenomenon.

• *Spatial correlation and distance.* It is natural to postulate that there could be a correlation between the distance between two sensor nodes and their ability to predict each other's measurements. To check whether a clear relationship between inter-sensor correlation and distance exists, generally across all of our sensors, we generated scatter plots of correlation versus distance for all the nodes in the network (Figure 3). Each dot corresponds to a single sensor. It is clear from this figure that there is no consistent relationship between inter-sensor correlation and distance.

Another way to examine this distance hypothesis is to make the following consistency check. Let σ_{xy} denote the correlation between nodes x and y, and $d(x, y)$ denote the physical distance between x and y. If two pairs of nodes (x, y) and (x, z) have $\sigma_{xy} > \sigma_{xz}$ and $d(x, y) > d(x, z)$ then a distance hypothesis is consistent for these two pairs of nodes. We count the number of times such node pair comparisons are consistent. We found that for temperature readings, our data exhibited 68% consistency, and for the humidity readings, our data had 66% consistency. This is considered a low level of consistency and thus again indicates that distance would not be good enough to use as an explanatory

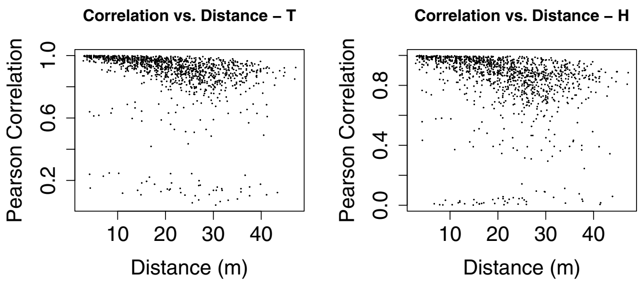


Fig. 3. Pearson's correlation vs. distance for all node pairs in the lab. The linear fit line is also shown on the plots: Temperature (left) and humidity (right).

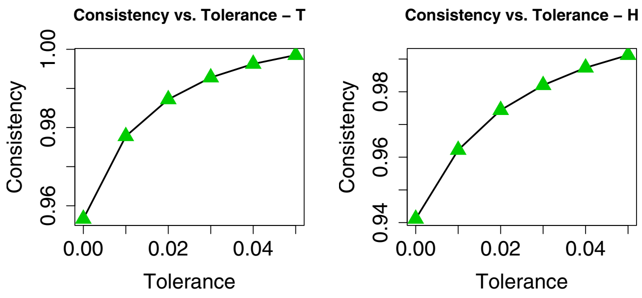


Fig. 4. Consistency of monotonicity of readings at sensors 1 and 2 for temperature (left) and humidity (right).

variable for sensor node prediction.

We believe the reason why distance is not a good indicator of correlation in our network is because even though two sensors may be physically close and exposed to similar sources of temperature and humidity, there can be other things in indoor environments that affect the sensor readings such as walls, air vents, and so on. We thus elect not to include distance as an explanatory variable in our model.

• *Isotonicity.* If two sensors s_X and s_Y are exposed to the same sources of stimuli, then intuitively increasing the value of the stimuli should result in higher readings at both s_X and s_Y. We incorporate this into our prediction model by adding an isotonicity (non-decreasing monotonicity) requirement that when the values of sensor s_X increase, so should the predicted values for sensor s_Y increase. Such constraints expose physical phenomena that would otherwise be hidden covariates in our model. We checked the consistency of our isotonicity hypothesis as follows. If $x(t_2) > x(t_1)$ and $y(t_2) > y(t_1)$ then our count of consistency increases by one. We calculated the consistency by checking this over all time slots and over all node pairs. In order to allow for small amounts of noise in the measurement process, we also consider a pair of readings to be consistent if $x(t_2) > x(t_1)$ and $y(t_2) > y(t_1)+\delta$ where δ is a small noise tolerance parameter. In Figure 4 we show the consistency of monotonicity between two sensors for a range of tolerance values ($\delta \in [0, 0.05]$). Even with zero noise tolerance, the consistency of monotonicity is very high, greater than 96% (94%) for temperature (humidity) respectively. This very strong consistency of the isotonicity constraint indicates that it would be a very useful constraint to include in regression analysis, limiting the search for a solution to the right portion of the search space.

B. *CIR: Combinatorial Isotonic Regression*

In Section III, we discussed our motivations for proposing a new method for solving isotonic regression. We present a four step method in which the first two steps are used to convert the isotonic regression into a combinatorial problem. We then rely on a dynamic programming paradigm to solve the problem. We call our solution *combinatorial isotonic regression* (CIR). Our method finds the univariate isotonic regression of the data in polynomial time. The advantage of converting this problem to the combinatorial domain is that this yields additional flexibility inside the solution. In particular, we will see that there is a natural way to limit the number of parameters of the resulting model.

Let the time series of readings from sensor s_Y be denoted by the data stream $Y = \{y(t)\}$ with time index $t = 1, 2, ..., T$ where T denotes the total number of measurements. Similarly, the time series data measured at sensor node s_X forms data stream $X = \{x(t)\}$. Our data comes in the form of pairs $(x(t), y(t))$ where $x(t)$ denotes the value of sensor s_X at time t and $y(t)$ indicates the value of sensor s_Y at time t. We can have pairs such as $(18, 19)$ and $(18, 20)$ since sensor s_X could read 18 degrees at two different times while sensor s_Y has two different readings at these two time slots.

We order the values (temperature measurements) in X according to $x_1 \leq ... \leq x_N$ where $x_1 = \min x(t)$ and $x_N = \max x(t)$, hence the range of values the sensor measures falls within is $[x_1, x_N]$ and the cardinality of the values in stream X (the number of measurements) is given by $|X| = N$ [1]. There will be redundancies in this ordering since at many times the sensor obtains the same reading. To reduce this ordering to a strict order, we eliminate redundancies by grouping the same values together, and thus generate an order $x_{(1)} < ...x_{(i)} < ...x_{(I)}$. This specifies the distinct I values that sensor s_X may read. Thus the notation $x(t) = x_{(i)}$ indicates that sensor s_x at time t measured the value $x_{(i)}$ where $x_{(i)}$ denotes the i-th value in our ordered set of temperature readings. We produce the same strict order for the data stream Y yielding $y_{(1)} < ...y_{(j)} < ...y_{(J)}$ where J denotes the number of distinct values for sensor s_Y. Note that, the sensors in our experiment are quantized and have less than a 1000 distinct values for humidity and temperature readings.

Our goal is to find a mapping $X \rightarrow \hat{Y}$ denoted by $\hat{Y} = f(X)$. Regression analysis selects a mapping that minimizes a specified error metric. Let ϵ_p denote a function for measuring the prediction error, i.e., $\epsilon_p = g(Y - \hat{Y})$. The objective of the isotonic regression is to find the mapping f to predict Y, $\hat{Y} = f(X)$, that minimizes the error metric ϵ_p subject to the non-decreasing isotonic constraint $\hat{y}_1 \leq \hat{y}_2 .... \leq \hat{y}_N$.

One advantage of our approach is that it is independent of the form of the error function ϵ_p. Commonly used forms of ϵ_p are the L_p norms of error that are shown in Equation 1.

$$\begin{aligned} &(\sum_{j=1}^{J} w_j |y_j - \hat{y}_j|^p)^{1/p} \quad if \quad 1 \leq p < \infty; \qquad (1) \\ &max_{j=1}^{J} w_j |y_j - \hat{y}_j| \quad if \quad p = \infty. \end{aligned}$$

[1]Although $N = T$ we use different notation here to emphasize that the index N corresponds to the size ordering rather than a temporal ordering

where w_j could be weights, or penalties, of making specific errors.

Our method consists of four steps that are executed for each pair of sensor nodes. We state them now and then explain each one in detail below. We will use the example in Figure 5 to help illustrate each of these steps.

1) Build the relative importance matrix R.
2) Build the error matrix E.
3) Build a cumulative error matrix C.
4) Starting at the minimum value in the last column of C, trace back a path to the first column that minimizes the cumulative error.

Step 1. We start by using the data tuples to form a relative importance matrix R where each r_{ij} captures the number of times when sensor s_X measured value $x_{(i)}$ and sensor s_Y read the value $y_{(j)}$ at the same moment. If we define $\rho_{ij}(t)$ to be:

$$\rho_{ij}(t) = \begin{cases} 1, & \text{if } x(t) = x_{(i)}, \text{ and } y(t) = y_{(j)} \\ 0, & \text{otherwise;} \end{cases}$$

then the elements of the matrix R are obtained from: $r_{ij} = \sum_{t=1}^{T} \rho_{ij}(t)$. In the example in Figure 5, each sensor can take on one of five different values. In the R matrix, we see that whenever sensor s_X read value $x_{(1)}$, sensor s_Y can experience any of the readings $y_{(1)}$ through $y_{(5)}$. A particular entry in R indicates how many times a value of say $y_{(3)}$ was observed when the explanatory variable measured $x_{(1)}$. This is essentially a histogram since it indicates, for example, that when sensor s_X measures the temperature to be $x_{(1)}$, then node s_Y is more likely to measure the temperature $y_{(1)}$ than other values (although others are also possible). The matrix R is a histogram on the entire set $x_{(i)}, y_{(j)}$ for all i and j.

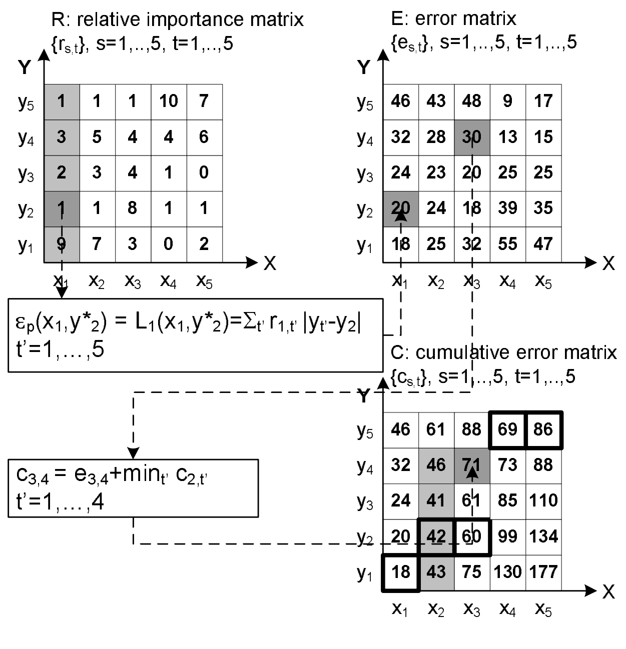


Fig. 5. Example: steps for building a CIR model for a response sensor s_Y from the data at sensor s_X, where each sensor has only 5 possible values.

Step 2. We now populate the error matrix E whose elements $\{e_{i,j}\}$ each describe the error that would be made if the prediction of $\hat{Y}$ was selected to be $y_{(j)}$ whenever sensor s_X observed the reading $x_{(i)}$. We can see, as in the example, that whenever we have $X = x_{(1)}$, the most common observation at Y is $y_{(1)}$, although there are times when sensor x_Y observes the readings $y_{(2)}$ or $y_{(3)}$ and so on. Hence the predictor $y_{(1)}$ isn't always correct, and thus some error arises for this prediction. In our example, we use the error

$$e_{ij} = \sum_k r_{ik} |y_{(j)} - y_{(k)}|$$

which corresponds to the L_1 error norm. The weight r_{ik} captures how often the particular error $y_{(j)} - y_{(k)}$ occurs. We could define any other error metric, based on $e_{ij} = \sum_k \epsilon_p(y_k, y_j, r_{ik})$, but for ease of presentation we limit ourselves here to the L1 norm. Our example in Figure 5 shows how the matrix E is constructed. We highlight e_{12} and illustrate how it is computed using the first column of the R matrix as weights in the error function.

Step 3. We can now interpret the goal of the isotonic regression (i.e. finding $\hat{y} = f(x)$) to be equivalent to finding a non-decreasing function that traverses the graph under matrix E in our example. We want to find a non-decreasing "path" that starts at the point $x_{(1)}$ with some y-intercept, and moves in non-decreasing fashion across the space of x values to $x_{(I)}$ (generally, and to $x_{(5)}$ in our example). This path should accumulate the minimum error and visit each column exactly once. To find this path we construct a cumulative error matrix C in which each element c_{ij} summarizes the errors seen so far when the X takes on values less than or equal to $x_{(i)}$, i.e., $x_{(1)}, ..., x_{(i)}$. In our example we start filling the cumulative error matrix C by copying the first column of E matrix. The C matrix is then filled column by column, from left to right. Each value of $c_{i,j}$ is simply the sum of its corresponding value in the matrix E, namely e_{ij}, and the minimum value in the previous column to the left. This value from the previous column describes the accumulated errors seen so far, for values of $x_{(l)}$ where $l < i$. By building up the i-th column, we are adding to this cumulative error, amounts that occur when predictions are made using sensor value $x_{(i)}$. We are constrained on the minimum value used from the previous column. If we are computing c_{ij} then we can look in the $i-1$st column only for values of Y that are less than y_j. In our example we show how $c_{3,4} = 71$ is computed by adding $e_{3,4} = 30$ and the minimum of its lower or equal Y value in the previous column of C. The candidate values are highlighted; the minimum is $c_{2,3} = 41$. For each value in C matrix, we also keep the index that identifies the minimum value in its previous column that led to the current estimate of the cumulative error.

Step 4. Once the cumulative matrix C is fully constructed, we find the minimum value in the last column of C (index $j = 5$) and extract the associated index (called $mini$) to the previous column (in our example $mini = 5$). Finally, the procedure backtracks over the columns updating the $mini$ value at each step (steps 9-11). For each column, the value of $y_{(mini)}$ is stored as the best predicted value for $\hat{y}_{(j)}$. In our example, the final mapping is shown by thick perimeter boxes on Figure 5.

The procedure used here for steps 3 and 4 corresponds to a dynamic programming solution to the problem of finding

```
Procedure $IsotonicRegression(E, X, Y, \hat{Y})$
{  1. $\forall\ i, \forall\ j,\ c_{i,j} = 0$;
 2. for  $(j = 1$ to $j = J)$ {
 3.       $c_{1,j} = e_{1,j}$; }
 4. for  $(i = 2$ to $i = I)$ {
 5.      for  $(j = 1$ to $j = J)$ {
 6.            $c_{i,j} = e_{i,j} + \min_{1 \le j' \le j} c_{i-1,j'}$;
 7.            $index_{i,j} = \arg\min_{1 \le j' \le j} e_{i-1,j'}$; } }
 8. $mini = \arg\min_{i=I-1, 1 \le j < J} c_{i,j}$;
 9. for  $(i = I - 1$ down to $i = 1)$ {
10.       $\hat{y}_{(i)} = y_{mini}$;
11.       $mini = index_{i,mini}$; }
}
```

Fig. 6. Pseudocode for the dynamic programming algorithm for finding the isotonic model between node pairs.

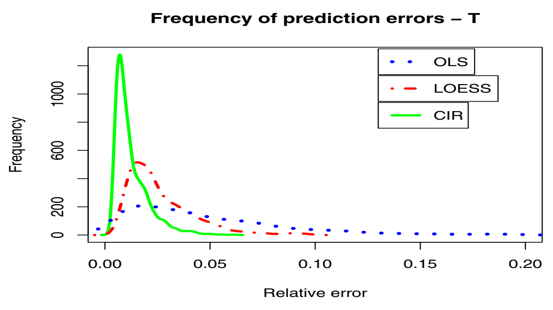
an isotonic regression that minimizes the chosen error. Our pseudocode given in Figure 6 specifies the detailed procedure.

The runtime of the procedure in Figure 6, is dominated by the two nested loops in steps 5 and 6. Hence, it is $(I \times J)$ (the procedure in step 7 can be optimized for execution in a constant runtime). Assuming that the tuples were not sorted, the original ordering of tuples takes an order of $O(NlogN)$ if standard sorting techniques are used. Since the number of values measured by the sensor is often significantly larger than the set of unique values for X and Y (i.e. $N >> I \times J$), the runtime of the procedure is dominated by the original sorting of the data, which can be accomplished in linear time with respect to N by using bucket sort [18]. It is easy to show that the dynamic programming procedure presented calculates the provably optimal model (i.e., minimum error) with respect to the given error norm and with the current set of data points [18].

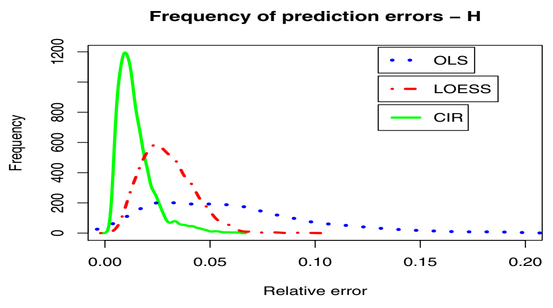
C. Evaluation

To evaluate our approach, we compare prediction error of the combinatorial isotonic regression (CIR) to prediction errors of (a) ordinary least square (OLS) linear regression and (b) robust least-square (LOESS) non-parametric regression. The reasons for selecting these two models as comparison points were (i) to compare the results of isotonic regression to both parametric and non-parametric models, (ii) OLS regression is the most widely used parametric modeling method, and (iii) LOESS regression with proper parameter selection is a widely used robust non-parametric modeling method. The major difference of CIR with respect to LOESS is inclusion of the isotonicity constraint. Our intent is to gain insight as to what extent the prediction error can be reduced by including the isotonicity requirement.

We briefly summarize OLS and LOESS modeling. If the goal is to model the readings Y of sensor s_Y, from the readings X at sensor s_X, the OLS model would be: $OLS(X) = \hat{Y} = \beta_0 + \beta_1.X + \epsilon_{ols}$, where β_1 and β_2 are the OLS regression coefficients and ϵ_{OLS} is the vector of residuals. The least squares estimates of the unknown coefficients are the values of $\hat{\beta}_0, \hat{\beta}_1$ that minimizes $\sum(Y - [\beta_0 + \beta_1.X])^2$. In applications where the residuals (errors) ϵ_{ols} follow the Gaussian Normal distribution, the result of OLS regression is the same as the Maximum Likelihood (ML) estimate of the response variable and is optimal. Another advantage of OLS model is its simplicity and closed-form of computation. The drawback is that for many real-life data sets, the assumptions about the underlying distributions being Gaussian and the form being linear do not hold. The LOESS non-parametric regression is often referred to as locally weighted polynomial regression [19]. In this method a low-degree polynomial is fit to the data, usually by the method of least squares.



(a) Temperature



(b) Humidity

Fig. 7. Frequency of prediction errors for 3 different prediction models (OLS, LOESS, CIR) on all node pairs for temperature and humidity.

To compare the different models, we use learn and test procedures, resampling, bootstrapping and confidence intervals as our statistical validation and evaluation methods. For the learn and test procedure, we split our N data tuples (x_n, y_n), $n = 1, .., N$ into two disjoint sets: the learning set that contains N_{learn} tuples and a test set with $N_{test} = N - N_{learn}$ tuples. Each model is built using the learning set, and the test set is used to evaluate accuracy. The residual for each predicted value is defined as $r_i = y_i - \hat{y}_i$. Thus, from each model we obtain a vector of residuals $[r_1, r_2, ..., r_{N_{test}}]$. We have used several different norms to obtain the summary statistics of residuals and relative residuals. The L_p error norm for *relative* residuals is defined as: $L_p(relative) = (\sum_{i=1}^{Ntest} |\frac{r_i}{y_i}|^p)^{1/p}$, for $y_i \neq 0$ and $1 \le p < \infty$.

We have experimented with a wide variety of error measures, including L_1 and L_2 norms of both residuals and relative residuals. The resulting prediction error comparisons were very similar across the different error measures. Due to space limitations, the results shown in this section are limited to the average (L_2) error of the relative residuals. We use two days of data for our modeling phase with one day for learning and one for testing. We use the rest of the days from our data for the operational phase with monitoring and scheduling to see how long we can prolong the life of this network. The frequency of L_2 prediction errors over all possible node pairs, for the test data, is shown in Figure 7 for all three methods. It is easy to see small errors are much more likely for CIR than for two other methods and that OLS has significant heavier tail of large errors. Analysis indicates that for both temperature and humidity sensors the unconstrained non-parametric LOESS modeling achieves on average a factor of 2 lower error when compared to OLS modeling.

Isotonic modeling method achieves on average more than 4 times improvement in prediction error when compared to the LOESS modeling approach. For example, according to our results, for both temperature and humidity, the number of the internode models with the relative error rate of less than 1% is around a 1000, while the number of internode models with less than 1% error is less than 250 for the LOESS model and less than a 150 for the OLS model.

We can also interpret the prediction error results in terms of absolute temperature and humidity values. During the test phase, the average temperature in the lab was $18.76°C$ and the average humidity in the lab was 42.17%. For the temperature models, those node pairs with a relative error measure of 2% or less, can predict the response sensor with an absolute error of less than $0.4°C$. Similarly, for humidity the node pairs with relative error less than 2% can predict the response sensor with an absolute error of less than 0.8%. This included over half the node pairs for the CIR modeling. When we set 2% as the target error rate and carry out our sleeping algorithm, we *only* use the node pairs that achieve this relative error. Thus CIR enables many more options for sleeping than the other regression methods that identify fewer node pairs for prediction. Direct comparisons of performance with previous work are difficult because different studies use different error metrics. However, we note that in [16], they achieved errors of roughly $1°C$ using a root mean squared error metric. Our L2 and L1 errors are less than this. Although these are not directly comparable because of the different error metrics, and the differing scenarios for application, it seems clear that both methods are high performing with acceptable error rates in roughly the same range.

We apply resampling and bootstrapping to find the distribution and confidence intervals for the prediction errors. Specifically, we randomly select 65% of the data from the first two days data as the learning set. We use the rest of the data as a test set and calculate the prediction errors. We perform bootstrapping on the prediction error by repeated resampling from the original data set and estimating the sampling distribution of the prediction error. The confidence intervals of the modeling method are directly found from this distribution. An example for the bootstrapping results between two nodes is shown in Table I, where the resampling was repeated 100 times. The distribution quantiles are computed over all pairs of sensors included in the prediction graph. The results indicate that for a 95% confidence, the CIR method is a factor 3 to 4 times better than LOESS, and that CIR not only produces smaller error values, but is a more consistent method than the alternatives.

quantile	5%	25%	50%	75%	95%
OLS	0.0368	0.03375	00.0379	0.0388	0.0392
LOESS	0.0138	0.0139	0.0147	0.0150	0.0153
CIR	0.0039	0.0041	0.0042	0.0044	0.0045

TABLE I
SAMPLE QUANTILES FROM DISTRIBUTION OF TEMPERATURE PREDICTION ERRORS.

D. *Flexibility of the CIR Approach*

A key advantage of our CIR modeling method is its inherent flexibility. Modeling in sensor networks is often a precursor for subsequent optimizations. Because the CIR method solves the isotonic regression problem by mapping the problem into a combinatorial domain, the flexibility of the combinatorial domain can be exploited to impose additional modeling requirements. Examples of additional requirements on the models include imposing specific forms on the regression function such as symmetric or convex, robustness, limiting model discontinuity and limiting the number of parameters in a model. The exact description of the flexibility of CIR is beyond the scope of this paper and can be found in [20]. Here, we only focus on the aspect of limiting the number of parameters which is useful for smoothing the model and avoids overfitting the data. This is particularly important for sensor networks when nodes are constrained in memory size, and distributed versions ([20]) of our methods are used.

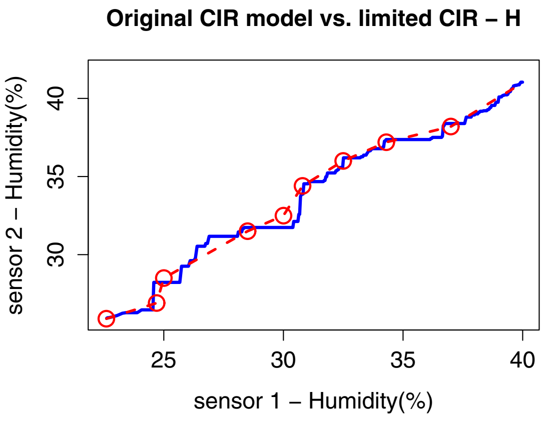


Fig. 8. Fitting of two sample humidity models predicting sensor-2 from sensor-1: unlimited CIR (solid line) model with 73 parameters and limited CIR (dashed line) with 9 breakpoints; break points are shown by circles.

Since we fit piece-wise linear components to our data, we define breakpoints as those points where two components meet. In general, breakpoints are coordinates where a change in the form of the model begins. The number of breakpoints is thus an indicator for the number of model parameters. In Figure 8, we show the fitting for both the unlimited CIR model (shown as a solid line) that selected 73 breakpoints, and the limited CIR model (shown by the dashed line) using 9 breakpoints (shown with circles). Traditional smoothing methods like splines can find a modeling solution that uses fewer parameters [21]. However, blind usage of smoothing methods to restrict the number of parameters affects the optimality of the solution. The goal is to constrain the number of parameters of the model in such a way that we maintain the lowest possible prediction error under imposed constraints.

We now explain how we modify our CIR algorithm presented in subsection IV-B to produce a regression function that has at most D break points. The modified CIR model still guarantees both the isotonicity constraint and the minimum error for the given number of parameters. In order to restrict D, the model should have a notion of its local slope which indicates the value of jump from the current prediction ($f(x_{(i)})$) to the next ($f(x_{(i+1)})$). The maximum possible slope is J, but the local slopes are typically much smaller than that.

The basis for the algorithm that calculates CIR with a restricted (specified) number of breakpoints D and a maximum slope of L is a version of the generic CIR dynamic programming-based algorithm that creates for each entry of the cumulative error matrix (C), a separate matrix SM of size $D \times L$. Entry (d, l) of matrix SM for the entry c_{ij} of matrix C stores the cumulative error that corresponds to the cumulative error of the best CIR mapping that ends at values $s_X = x_{(i)}$ and $s_Y = y_{(j)}$ while the mapping uses exactly d parameters, and its slope at that point is l. This value is denoted by a four dimensional entity $SMC(i, j, d, l)$ and is calculated as shown in Equation 2.

$$SMC(i,j,d,l) = \min\{SMC \quad (i-1, j-l, d, l) + E(i,j) \\ , \{\min_{q=1}^{q=l} SMC \quad (i-1, j-q, d-1, q) + E(i,j)\}\} \tag{2}$$

The first term in Equation 2 corresponds to the path that does not induce a need for a new parameter on the considered path and retains its previous slope. This first term only exist if a valid value of mapping function can be defined for the previous column, and for the given slope and number of parameters. The second term corresponds to the path that uses one more breakpoint and changes the slope of the mapping CIR function at $x_{(i)}, y_{(j)}$. Therefore, with respect to the basic CIR dynamic programming algorithm we store $D \times L$ more variables at each step, and we find the optimal solution in overall time that is $D \times L$ longer. Note that, the maximum number of parameters is I and the maximum slopes is at most J. At the limit case where $D = I$ and $L = J$, the runtime becomes the square of the original CIR runtime. In practice, D and L are small constants in order to ensure creation of a smooth model with few parameters. Therefore, the runtime of the algorithm is most often just increased by a small constant and stays linear.

The goodness of fit of a model and comparison between different models on the same data set can be quantified through the use of statistical information criteria, such as AIC (Akaike information criteria), BIC (Bayesian information criteria), or DIC (deviance information criterion). We have used AIC [22] as the goodness-of-fit measure of choice for our models. AIC uses the logarithm of the likelihood of the model and also penalizes for the number of parameters in the model. AIC is formally defined as, AIC = $-2log(likelihood) + 2D$, where $2D$ is the number of parameters in the model. Using the AIC criteria, we compare the original isotonic model to models where we restrict the

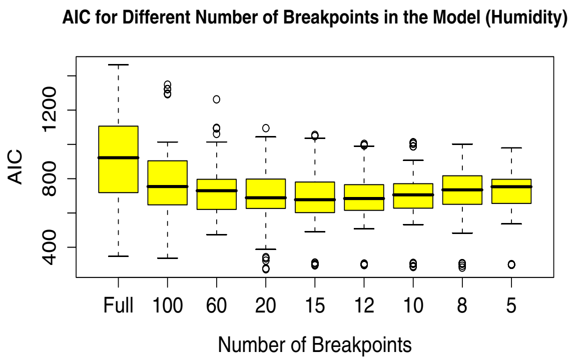


Fig. 9. AIC (y-axis) for different number of breakpoints (x-axis) in the model.

number of breakpoints. The results, shown in Figure 9 depict the AIC of the best predictor of each node for humidity. (The results for temperature, not included here, are very similar.) Each boxplot shows the values for a defined number of breakpoints in the model. The sensors have around a 1000 discrete values and the unrestricted isotonic model has in average more than 300 breakpoints. The results show that limiting the number of parameters in general provides a large improvement for the popular AIC goodness-of-fit metric. Moreover, it shows that with a small number of parameters, around 10 or so, our limited CIR method achieves a good fit.

As a second example of the flexibility of our methods, we now extend them to produce models that rely on more than one sensor as an explanatory variable. For some applications it may be beneficial to build such multivariate models. It is already clear that our method, relying on only one sensor to predict another, achieves very low error rates and thus intuitively there is no need for additional explanatory variables. Nevertheless, we consider this case here because of the often espoused view that using more sensors to predict a single one is a good idea. Increasing the dimensions make the CIR modeling problem NP-complete and finding a solution in polynomial time is not possible. We introduce an ILP formulation of the multivariate modeling problem that finds a practical solution to the small modeling instances on our sensor data.

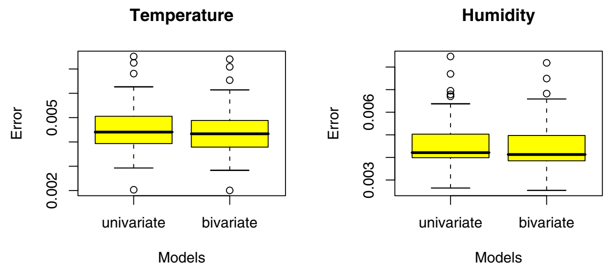


Fig. 10. Comparison of prediction errors between the best univariate model and the bivariate model built from the 2 best univariate models.

The use of multivariate models can only be justified if they decrease the prediction error for the response variables. In our experiments, we developed bivariate models for each of the response sensors using its two *best univariate predictors* (i.e. predictors with the least amount of prediction error). The resulting errors are presented in Figure 10, where boxplots show the prediction errors for the best univariate model and the best bivariate model for each of the nodes. By best univariate model we mean the univariate model with the predictor that has least amount of prediction error. We can see that the bivariate models do not achieve a significant reduction in modeling error when compared to the best univariate models. This can be explained by the fact that the modeling errors for the best univariate models are so small that adding more variables could not significantly reduce the already small errors. Therefore, we decided not to use multivariate models on our data; the increase in complexity does not warrant the tiny improvements in prediction errors. We do not describe the details of our multivariate modeling method due to space limitations and refer the reader to [20] for the exact ILP formulation and solutions for multivariate modeling.

V. Centralized Sleeping Coordination

Once the intersensor models are available on the server, we can formally define the problem of sleeping coordination. The problem is to find multiple subsets of the sensor nodes such that each subset can achieve *complete sensing coverage*, over the entire field spanned by the network, for multiple modalities simultaneously. Complete sensing coverage is defined as the maximum coverage attainable by the network while all sensor nodes are awake. We guarantee that within a user specified error range ε, the complete coverage is maintained, while a significant number of nodes are placed in the sleep state. The sleeping coordination method works by selecting the awake nodes in such a way that the data from the sleeping nodes is predictable from the data at the awake nodes, within a $\pm\varepsilon$ error bound.

The starting point for the problem specification is to abstract the sensor network into a graph $G(S, E)$, where $|S| = N_n$ and N_n is the number of nodes in graph. There is an edge (or hyperedge) from the sensors $s_{X1}, s_{X2}, ...$ to a sensor s_Y, if and only if the $\hat{s_Y} = f(s_{X1}, s_{X2}, ...)$ predicts the value at sensor s_i within a user's specified error tolerance $\pm\varepsilon$. Edges indicate a relationship between two nodes and therefore correspond to univariate models. Hyperedges indicate a relationship between several nodes and thus correspond to multivariate models.

Before we instantiate a graph, we have to answer the following questions: (i) do we need hyperedges or regular edges? (ii) Should edges be directed or not? (iii) Should edges be dynamic or static? The results of the prediction studies in the previous section directly answer the first two questions: (1) we do not need data from multiple nodes (or hyperedges); (2) although not presented here due to space limitations, we found in our data that prediction accuracy is *not* symmetric, and thus the graph should be a directed one; and (3) the edges in the graph are considered valid and static as long as the isotonic prediction errors are within the specified error bound. During time periods when the models are unchanging, our graph remains static. To accomodate changes, we incorporate a monitoring capability that updates the models and the schedule as needed (Section VI). Note that there are multiple graphs (m) that each correspond to different sensor modalities. For a given set of m prediction graphs, we adopt the following problem formulation.

Problem: Domatic Partition on Multiple Directed Graphs.

Instance: m directed graphs $G_m = (S, E_m)$, where all the graphs have the same set of nodes.

Question: *Is there a partition of nodes in the graphs to K disjoint sets, S'_1, S'_2, ..., S'_K, such that for each set S'_k, the subset $S'_k \subseteq S$ is such that all nodes in each graph G_m that are not in S'_k have at least one incoming edge from a node in S'_k?*

The decision problem can be mapped to a maximization problem using a binary search. The problem of finding the maximum K on one graph is a special case of our graph (where there is only one modality), is called the domatic number problem and is one of the classical NP-complete problems [23].

We now formulate the sleeping coordination problem as an instance of an integer linear program. Even though the problem is NP-complete, for our scenarios of networks with less than 100 nodes, we are able to find the solutions quickly on a standard PC. We believe that even networks with a few hundreds of nodes could be solved on a standard server. Even though attractive approximation algorithms are available [24], we decided to develop an ILP formulation to retain optimality so as to be able to assess the maximum extent of the gain of our approach. We solve our ILP using the commercial CPLEX package. We first introduce the constants and variables for our ILP formulation, and then formulate the objective function and constraints.

Given: A number $K \leq (\delta + 1)$ and m prediction matrices $P_{m\{N_n \times N_n\}}$ (one for each modality) with elements p_{mXY}, s.t.

$$p_{mXY} = \begin{cases} 1, & \text{If } |\epsilon(\hat{s_Y} = f(s_X))| \leq |\varepsilon| \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Where $|\epsilon(\hat{s_Y} = f(s_X))|$ is the error in predicting the value at sensor s_Y given the data at s_X, and ε is the user's specified error tolerance and δ is the minimum degree of the nodes in the graph.

Variables: m matrices $Z_{m\{K \times N_n\}}$ with elements z_{mXk}, and a vector $U_{\{K\}}$ with elements u_k s.t.

$$z_{mXk} = \begin{cases} 1, & \text{If node } s_X \text{ is in set } S'_k \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$u_k = \begin{cases} 1, & \text{If the set } S'_k \text{ was selected} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Objective Function: The objective function is to maximize the number of disjoint dominating sets, i.e., $max \sum_k u_k$.

If we find, say 5, dominating sets, then we can extend the life of the network 5 times as compared to a scenario in which all nodes are left on all the time. This is because each dominating set achieves complete sensing coverage. Since our goal is to extend the network lifetime as much as possible, clearly, this corresponds to finding the largest number of dominating sets possible.

Constraints: The problem has three sets of constraints. The first set of constraints (C_1) ensures that if a set S'_k exist (i.e. $u_k = 1$), for each G_m, all nodes in G_m that are not in S'_k have an incoming edge from a node in S'_k. For $m = 1, ..., M$, $X = 1, ..., N_n$, $k = 1, ..., K$,

$$C_1 : \quad z_{mXk} + \sum_Y P_{mXY} z_{mYk} \geq u_k \tag{6}$$

The second set of constraints (C_2) is that if a node is selected in one group, it cannot be selected for any other group. Since the assignment of nodes to the groups is done for all m graphs, we introduce an auxiliary variable w_{Xk} that indicates that a node s_X is assigned to a group S'_k in at least one of the graphs. In other words, the variable w_{Xk} is the OR function over the variables z_{mXk}. We write the OR function in the language of linear constraints as shown in C_{2a}. The relationship C_{2b} ensures that each node is only selected for one group. For $m = 1, ..., M$, $X = 1, ..., N_n$, $k = 1, ..., K$,

$$C_{2a}: \quad w_{Xk} \geq z_{mYk}, \quad w_{Xk} \leq \sum_m z_{mYk}, \quad 0 \leq w_{Xk} \leq 1$$

$$C_{2b}: \quad \sum_k w_{ik} \leq 1 \tag{7}$$

The last set of constraints (C_3) ensures that the variables u_k and z_{mXk} are within the [0,1] range. For $m = 1, ..., M$, $X = 1, ..., N_n$, $k = 1, ..., K$,

$$C_3: \quad 0 \leq u_k \leq 1, \quad 0 \leq z_{mXk} \leq 1 \tag{8}$$

Since we have two modalities (temperature and humidity), we have $m = 2$. We extract the P_m matrices and constant $K = (\delta + 1)$ from our modeling studies.

A. *Evaluation of Centralized Sleeping Coordination*

In this subsection, we show the results of applying our ILP solver to domatic partitioning problem. Naturally, we desire to have the maximum number of possible domatic sets, so we can coordinate more groups to prolong the lifetime of the network. The number of domatic sets are dictated by the topology of internode constraints (models) that are defining the edges on the graph. An important factor in determining the edges on the graph is the user's specified error bound (ε).

Table II shows the number of domatic partitions obtained for different target errors. Each partition found can be used to predict both temperature and humidity. The first column indicates the target ε allowed for the prediction. The next three columns indicate the number of partitions resulting from different prediction models. The last column shows the average runtime of the solver on a 1GHz PC. We observe a number of interesting things from this table. First, if the target error is less than 1%, all the methods fail to find more than one domatic partition. However with a target error of 2%, all methods can find multiple domatic partitions, with our isotonic regression finding the most (5). It is a substantial gain to be able to extend the life of a network by a factor of 5 just by tolerating a mere 2% average relative error. If we consider larger target errors, such as 5%, we can extend the lifetime between 6 to 12 times depending on the prediction model used. Second, we find that the isotonic fit (with a limited number of parameters) is significantly more effective than the linear and non-parametric LOESS fits. Third, although we used an optimal ILP, we see that the average runtimes are low. We were also able to solve randomly generated instances with more than 1000 nodes in less than 10 minutes, indicating that our technique is scalable, for many realistic network scenarios, even when we insist on optimality.

Ave	Prediction Model			CPLEX
Err	Linear	LOESS	Isotonic (D=15)	Time
0.01	1	1	1	0.07
0.02	3	3	5	0.31
0.03	3	4	6	1.16
0.04	4	4	9	1.16
0.05	6	8	12	0.98
0.06	7	12	17	3.22
0.07	10	14	24	1.58

TABLE II
NUMBER OF DOMATIC PARTITIONS FOR 3 PREDICTION MODELS AND VARIOUS ERROR BOUNDS. AVERAGE RUNTIME (SEC) ALSO SHOWN.

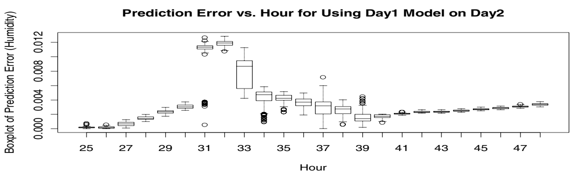
B. *Distributed Coordination Algorithm*

The centralized coordination algorithm proposed so far is well-suited for our in-building application, where the size of the network and the radio ranges of the nodes permits direct communication to a server. In a number of other sensor network settings, nodes might not be able to communicate to the server in a single hop. In order to generalize our approach to multi-hop networks, we have designed two different localized algorithms for the sleeping coordination problem. We have also evaluated the quality of the solutions and computed the communication energy consumption for each of these localized schemes. Because of space considerations, we do not provide the details of the localized sleeping coordination here and refer the interested readers to our technical report [13].

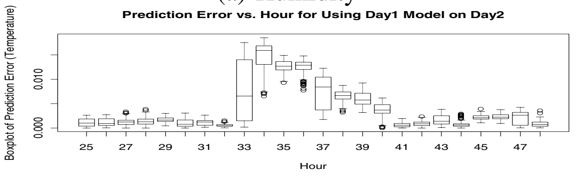
VI. SCHEDULING, MONITORING AND ADAPTIVITY

The domatic partitions-based approach for optimization of the lifetime of a network provides a sleeping decision procedure that implicitly assumes a scheduling strategy that places the nodes in each of the partitions to sleep for equal amounts of time. The approach so far has assumed that the observed phenomena is such that prediction models are static. However, intuition suggests that due to changes in weather patterns, seasons, environment (e.g. workday vs. holiday or weekend) and other factors, the models are not time invariant. There is a need to develop scheduling, monitoring, and adaptations procedures that address the dynamics of the observed signals. The *scheduling procedure* decides the length of each round of the round robin procedure. The *monitoring procedure* indicates at which points in time one will collect information to check the validity of models among sensors. Finally, the *adaptation procedure* determines how long one has to collect measurements from all sensors in order to build a new set of models and new domatic partitions.

To determine a strategy for these three procedures, four key questions should be answered: (1) Is there indeed a need for updating the dominating sets due to dynamic changes in the observed phenomena? (2) How long should a single round of the round robin scheduling be? (3) How often and for how long should monitoring be conducted in order to detect whether or not the models and dominating sets need to be updated? (4) What should the conditions be that initiate the creation of new models and domatic partitions?



(a) Humidity



(b) Temperature

Fig. 11. Prediction error vs. hour for two nodes in the experiment. We use the model built from the first day of data to predict the second day of experiment.

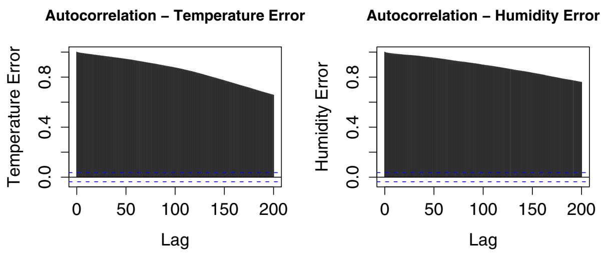


Fig. 12. The autocorrelation of the residuals of the model built on the first day's data and tested on the second day's data: temperature (left), humidity (right).

Our approach to answering these questions is data driven. Figure 11 shows time behavior of a typical prediction error. We summarize error behavior for each hour using box-plots. The figure shows only data for one day in order to maximize visibility of the important information. Three observations are valid over the range of all pairs of nodes: (i) all high errors appear between 6 am and 6 pm; (ii) high errors in humidity lead to high errors in temperature by a lag that is between an hour and three hours; and (iii) error rates are slow changing and very consistent within intervals as long as one hour. Figure 12 shows autocorrelation of prediction error for unit time lag of 30 seconds. We see that correlation is very high (above 90%) even for rather long periods of time (longer than 10 minutes).

Figure 13 presents a histogram of the number of high error predictions (above 2%) for each 100 consecutive samples (sampling rate is 30s) in the second day of the experiment. For example, value 0 indicates that there was no error in a 100 consecutive samples. We see that most often the erroneous values in a 100 sequential samples are consistent.

Using these observations, we concluded that the minimum period of a round robin sleeping schedule should be once every $12/k$ hours, where k is the number of disjoints sets in the domatic partition. We use equal periods so that all nodes are awake for equal amount of time and therefore have the same rate of energy consumption. This is reasonable for networks with homogeneous sensors, all starting with the same energy store. Note that, one can select other periods for the round robin schedule. Since we do not assume a high cost for the sleep to active transitions, our

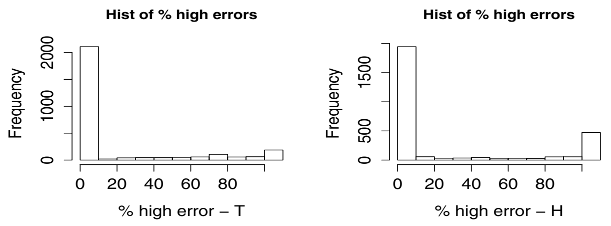


Fig. 13. Histogram of percentage (%) of high error measurements in 100 consecutive error values for temperature (left) and humidity (right) sensors. The error is for the model built on the first day and tested on the second day's data.

results would be the same for different round robin periods. In situations where the wake up cost is not insignificant, the rounds can be set to longer periods (such as a week or even more).

We also concluded that from 6:00PM to 6:00AM there is no need for monitoring since there was no single instance when the initial model were not accurate. Monitoring from 6:00AM to 6:00PM should be guided solely by acceptable delay in detecting the first instance of unacceptable error prediction and it is sufficient to monitor only humidity. Recall that the error autocorrelation is high and the histogram indicates that once the first error is observed, the probability of consecutive errors is very high. Thus, once high errors are detected, data collection has to be conducted at all nodes, in the same period (30 seconds), at the initial sampling rate, to ensure completeness of the collected data. The collection should continue until the first simultaneous occurrence of a very few (no more than ten) correctly predicted samples. The correctness of the sample predictions are evaluated using the initial models.

Lastly, we examine the overall performance of the system for two cases. First, we perform initial modeling and use a target error of 2% to find the prediction graph for the two modalities. We run the ILP-based partitioning procedure on this graph to find the domatic sets. We schedule the nodes in each set to be active for one day and then report the mean of the prediction error over all predicted sensors that are not awake for around 10 days of the network operation after the initial phase. We do not perform any monitoring or adaptive scheduling for this case. In the second case, we add the monitoring, remodeling and adaptive scheduling to our system. During the 10 days of operation, we found we only needed to update our models once, on day 9. We found that this removed all errors greater than 3% that occurred on the 10th day when no adaptivity was done.

A portion of the network's overall energy is going to be consumed by the overhead due to the initial modeling, monitoring and updating phases. We have used the graphs with different target errors to study the net extension to the network's lifetime with all the components of our methodology active (i.e., including monitoring). Table III shows the overhead and the extension results for different target errors shown on the first row. The duration of the monitoring and collection phases are denoted by T_{moni} and T_{col} respectively. The overheads of the initial modeling phase, monitoring and collecting durations are calculated in hours and are shown on the second, third and fourth row respectively. The fifth row shows N_S, the number of domatic sets found. The total overhead time is the sum of the second, third and fourth rows of the table and is shown by T_{OH}. If the original life time was T_{Orig}, the extension to the lifetime after considering the overhead is simply calculated as, $\frac{N_S(T_{Orig}-T_{OH})+T_{OH}}{T_{Orig}}$ and is shown at the last row of the table. We see that we achieve a large saving in network lifetime, even after considering the monitoring and remodeling overhead. Because little remodeling was needed for our dataset, we did not have enough data to fully examine the three-way tradeoff between overhead, error performance and life extension. We leave this for future work that will examine datasets of longer duration.

Target Error	.02	.03	.04	.05	.06	.07
Initial Phase	48h	48h	48h	48h	48h	48h
$\sum T_{moni}$	12h	10h	9h	9h	9h	9h
$\sum T_{col}$	57h	15h	9h	8h	7h	7h
T_{OH}	117h	73h	66h	65h	64h	64h
Number of Sets	6	8	10	13	21	29
x Life Extension	3.6	5.9	7.5	9.8	15.7	21.4

TABLE III

AMOUNT OF LIFETIME EXTENSION CONSIDERING ALL MODELING, PARTITIONING, SCHEDULING, & MONITORING OVERHEAD.

VII. CONCLUSIONS

We have developed a methodology for extending the lifetime of a sensor network by exploiting the redundancy in sensor readings. Through the use of an isotonic regression method we discover which nodes can be used to predict other nodes well. Based on our prediction models, we organize the nodes into domatic partitions such that each partition (subset of nodes) can predict the sensing capabilities of the entire network. This combination of methods works very well. Through exploratory analysis of our indoor network data, we discovered that the data obeys an isotonic constraint with extreme consistency. We thus selected isotonic regression as our regression method of choice for prediction. We illustrated that including this constraint brings significant gains because the resulting prediction models are very accurate. We compared our method to a common form of robust regression (LOESS) and showed that our method achieves better performance by roughly a factor of 4 over robust regression for a few different performance metrics related to prediction errors (e.g., when performance is measured by the number of pairs of sensors whose prediction model achieves less than a specified error target, and when performance is measured by the 95% confidence interval on errors). Our isotonic regression also outperforms linear and LOESS regression in that it provides superior inputs to the domatic number problem. They are superior in the sense that the ILP solution finds more domatic partitions when isotonic regression is used for inter-sensor modeling.

Our isotonic regression solution is new in that we convert the regression problem to one that can be solved using dynamic programming. Solving the problem in the combinatorial domain enables us to support any modeling error function, and allows us to limit the number of regression parameters. We address the practical issues of adapting our models and sleeping schedules to evolving data by extending our method to monitor for changes and to update the models and domatic partitions as necessary. Putting all the components of our method together, and accounting for all overheads this incurs (as it consumes energy), we found that if average relative errors of 2% can be tolerated, then our method can extend the life of our type of sensor network by over 3 times (depending upon the modeling method). If higher error rates (say 5%) are acceptable, then the network's lifetime can be extended even more; our isotonic method achieved a factor of 12 extension (Table II) and the simpler LOESS method combined with our domatic partitions achieved a factor of 8 improvement. It is encouraging to know that this is possible.

REFERENCES

[1] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare event detection," in *IPSN*, 2005, pp. 20–27.

[2] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, "Energy-efficient surveillance system using wireless sensor networks," in *ACM Mobisys*, 2004, pp. 270–283.

[3] C. Hsin and M. Liu, "Network coverage using low duty-cycled sensors: random & coordinated sleep algorithms," in *IPSN*, 2004, pp. 433–442.

[4] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Ple, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *ACM Sensys*, 2003, pp. 28–39.

[5] T. Yan, T. He, and J. Stankovic, "Differential surveillance for sensor networks," in *ACM Sensys*, 2003, pp. 51–62.

[6] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang, "Peas: A robust energy conserving protocol for long-lived sensor networks," in *ICDCS*, 2003, pp. 28–37.

[7] H. Zhang and J. Hou, "On deriving the upper bound of α-lifetime for large sensor networks," in *ACM MobiHoc*, 2004, pp. 121–132.

[8] M. Ayer, H. Ewing, W. Reid, and E. Silverman, "An empirical distribution function for sampling with incomplete information," *Annals of Mathematical Statistics*, 1955.

[9] J. Friedman and R. Tibshirani, "The monotone smoothing of scatter plots," *Technometrics*, , no. 26, pp. 243–250, 1984.

[10] P. Hall and L. Huang, "Nonparametric kernel regression subject to monotonicity constraints," *Annals of Statistics*, vol. 29, pp. 624–647, 2001.

[11] E. Mammen, "Estimating a smooth monotone regression function," *Annals of Statistics*, vol. 19, pp. 724–740, 1991.

[12] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *VLDB*, 2004, pp. 588–599.

[13] F. Koushanfar, N. Taft, and M. Potkonjak, "Sleeping coordination for comprehensive sensing: Isotonic regression and domatic partitions," Tech. Rep. IR-IT-2006-4, Intel Research, 2006.

[14] "http://www.xbow.com/products," .

[15] J. Hill, R. Szewczyk, A. Woo, S.Hollar, D. E. Culler, and K. S. J. Pister, "System architecture directions for networked sensors," in *ASPLOS*, 2000, pp. 93–104.

[16] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: an efficient framework for modeling sensor network data," in *IPSN*, 2004, pp. 1–10.

[17] G. Xing, C. Lu, R. Pless, and J. O'Sullivan, "Co-grid: an efficient coverage maintenance protocol for distributed sensor networks," in *IPSN*, 2004, pp. 414–423.

[18] T. Corman, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, 2001.

[19] W. S. Cleveland, "Robust locally weighted regression and smoothing of scatterplots," *Americal Statistical Association*, vol. 74, pp. 829–836, 1979.

[20] F. Koushanfar, "Statistical modeling and recovery of intermittent sensor data streams," Masters thesis, UC Berkeley Statistics Department, 2005.

[21] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, Springer, New York, 2001.

[22] H. Akaike, *Breakthroughs in Statistics*, vol. 1, chapter Information theory and an extension of the maximum likelihood principle., pp. 610–624, Springer-Verlag, 1992.

[23] M. R. Garey and D. S. Johnson, *Computers and intractability. A Guide to the theory of NP-completeness*, W. H. Freeman and Company, 1979.

[24] U. Feige, M. M. Halldorsson, G. Kortsarz, and A. Srinivasan, "Approximating the domatic number," *SIAM Journal of Computing*, vol. 32, no. 1, pp. 172–195, 2002.