A fundamental challenge in cyber-physical computing is to accurately capture the end-to-end behavior of large systems in which software interacts with a complex physical environment. This paper presents a new hierarchical modeling technique that is a result of an interdisciplinary effort to combine the best of estimation theory and data mining techniques to enable modeling such systems reliably at multiple degrees of abstraction. A reliable model is the one that remains sufficiently accurate over the whole input range. We use green transportation as a running example, where software routing optimizations use physical models of cars, streets, and traffic conditions to enable energy savings. In this case the “system” is the collection of cars traveling on different roads under different traffic conditions. We show that using our new modeling techniques, we are able to significantly improve the accuracy of fuel consumption predictions, while quantifying prediction accuracy, and hence the quality of green routes.

We are especially interested in modeling open systems where some components, interactions, processes, or constraints are not well-understood or not measured. For example, predicting the fuel consumption of a vehicle depends not only on fixed factors such as weight, frontal area, and engine type, but also on variables such as vehicle speed, acceleration, congestion patterns, and idle time, which are hard to predict accurately in advance. A single MPG rating (miles per gallon in highway and city) is quite inadequate. For instance, it cannot help decide which of two alternative city routes will consume less fuel. Building first principle models from scratch is not always practical, as too many parameters are involved. In contrast, using regression to estimate model coefficients is challenging because reliable estimation suffers the curse of dimensionality. The state space grows exponentially in the number of parameters, making sampling of that space sparse. As the number of parameters increases, estimated models become less reliable.

This paper proposes the Sparse Regression Cube modeling technique. It jointly (i) partitions sparse, high-dimensional data into subspaces within which reliable linear regression models apply and (ii) determines the best reliable model for each partition, quantifying uncertainty in output prediction. Evaluation results show that the framework significantly improves modeling accuracy compared to previous approaches and correctly quantifies prediction error, while maintaining high efficiency and scalability.
In estimation theory and statistical learning, numerous regression modeling techniques are well-known, from least squares error estimators to singular value decomposition and support vector regression techniques [12]. While regression modeling is concerned with accurate estimation of regression parameters, it does so by trying to fit the model, as best it can, to all data. Instead, we are concerned with dividing the multi-dimensional data into subspaces and determining which data points and attributes to use for regression in each subspace such that models are reliable. Therefore, the sparse regression cube can be thought of as a high-dimensional equivalent of linearization techniques. It computes the set of “linearized” models directly from sparse observations, and derives estimates of accuracy of these models.

To decide how to break down the multi-dimensional data space into appropriate subspaces, we borrow tools from Online Analytical Processing (OLAP), called data cubes [8]. Data cubes have been widely used for modeling and analysis of multi-dimensional data. Conceptually speaking, they efficiently try to group the data by all possible combinations of parameters, called cuboids, and determine properties of data in each cuboid. Those cuboids where data are particularly “homogeneous” are then used as the basis for data partitioning. Previous techniques have not addressed reliable modeling of sparse data. For example, regression cubes [5] and prediction cubes [4] simply assume that a sufficient number of data points is always present. In contrast, sampling cubes [13] focus on grouping sparse data appropriately into larger homogeneous categories to overcome sparseness problems, but compute only primitive measures such as sum and average within each category. Sparse regression cubes, proposed in this paper, are novel in that they combine the ideas of sampling and regression cubes to enable, for the first time, reliable and efficient modeling of sparse data using linear regression in each resulting data subspace.

The remainder of this paper is organized as follows: Section II motivates the modeling problem in open cyber-physical systems. The problem formulation and the structure of our data cube framework are discussed in Section III. In Section IV, we present our proposed mechanisms that enable reliable model construction for various generalizations over the data space. Section V evaluates our approach using the data collected from a real GreenGPS testbed as well as synthetic simulation data. We present the related work in Section VI and finally conclude the paper in Section VII.

II. Motivation

Our work is motivated by a new category of cyber-physical systems, that are distributed and that interact with an environment that is not entirely predictable. An example is a green navigation service called GreenGPS [7] that finds the most fuel efficient routes between user-defined points. Fuel-efficiency measurements obtained from users of the system are used to construct accurate predictive models of fuel consumption for a larger community. Particularly, users submit tuples of data indicating the fuel consumption of their vehicles on various road segments (e.g., see Table I). Each tuple consists of the amount of fuel consumed per mile, vehicle make, model, year, and other car and trip parameters such as the average velocity and the number of traffic lights encountered. When a driver needs to find the most fuel efficient route to a destination, this service predicts the amount of fuel consumed on different route options using the constructed models.

GreenGPS is an example of a complex cyber-physical system with many components (cars) and sparse measurements (not all cars contribute data on their fuel efficiency). Different types of cars behave differently on different roads and traffic conditions when it comes to the fuel consumption. To represent the behavior of system components, we build a hierarchy of models that could describe the system in different input subspaces. For example, one linear regression model can predict the fuel consumption for all Honda cars while another one predicts Toyota cars. Some may model a more general class of cars such as all sedans or all SUVs. A challenge is to find generalizations that lead to accurate prediction models for the corresponding categories.

There is previous work in data mining and machine learning communities that builds such generalization hierarchies. Most notably, regression trees [3] use a tree of regression models where the root node is the most general model that applies to all of the input space, while children nodes are specialized to subspaces separated by hyper-planes. Regression Cubes [5] are online analytical processing frameworks that do the same by exploring all different ways of generalizing data and using mostly categorical attributes to divide the input spaces into smaller subspaces (e.g., make of the car).

A major challenge in GreenGPS and similar sparsely-sampled cyber-physical systems is that the measurements provided by the users may be limited and cover only a small fraction of roads and car models. Observe that factory measurements by manufacturers only describe new cars and hence do not apply for used cars, whose characteristics may have deteriorated with age. When the samples used to build a regression model are sparse, the model overfits the samples and performs poor predictions. We define the term Reliability of a regression model to denote whether the model is able to predict future queries accurately or not. In the face of sparse data, traditional methods such as regression cubes and regression trees build unreliable models that fail to predict accurately.

Modeling this type of systems needs a new capability: a framework to efficiently construct statistically reliable models at different levels of generalization over the data space. Different levels of abstraction or generalization can be compared using the modeling error and appropriate data subspaces can be identified. Sparse regression cubes also provide reliability guarantees and model confidence in each subspace.

To achieve this capability, our framework employs a set of mechanisms:

• Algebraic Representation of Regression Models and Error: Hierarchical structures such as regression trees or regression cubes tend to construct a large (even ex-
TABLE I
GREENGPS APPLICATION SAMPLES.

<table>
<thead>
<tr>
<th>Fuel cons.</th>
<th>Street Attributes</th>
<th>Vehicle Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.030 gal/m</td>
<td>Speed Lim. ($v_l$)</td>
<td>Weight (m)</td>
</tr>
<tr>
<td>0.037 gal/m</td>
<td>Lights ($LT$)</td>
<td>Avg. Speed ($v_{avg}$)</td>
</tr>
<tr>
<td>0.036 gal/m</td>
<td>Stops ($ST$)</td>
<td></td>
</tr>
<tr>
<td>0.027 gal/m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.067 gal/m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.033 gal/m</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We use $y_i$ to denote the value of the output and $x_{ij}$ to denote the value of input $x_j$ in the $i$-th sample. Note that some $x_{ij}$s may be constants or functions of other input variables. The modeling framework aims to represent a complex function that relates $y_i$ to $x_{ij}$'s using a set of linear regression models.

We assume that a subset of input variables, namely, $x_1, \ldots, x_k$, $k \leq d$ is used for regression modeling. Those are called regression attributes and $k$ is called the number of regression dimensions. Also, we assume that another subset of inputs, $x_{d-k+1}, \ldots, x_d$, is used to group data together in the data cube. These attributes are called group_by attributes and $l$ is the number of group_by dimensions. Using this notation, we emphasize that an input can be a regression attribute and a group_by attribute at the same time.

As an example, suppose that we are given the samples in Table I where there are 8 input variables in total ($d = 8$). A reasonable choice is to pick $x_1 = v_l$, $x_2 = LT$, $x_3 = ST$, $x_4 = m$, and $x_5 = v_a$ as regression attributes ($k = 5$) and use $x_6 = make$, $x_7 = model$, and $x_8 = year$ as group_by attributes ($l = 3$).

A data cube characterizes each group of data by a Data Cell. A data cell $c$ is defined by a particular assignment of values to some of group_by attributes. For example, \{make=Honda\} is a data cell that represents all Honda cars while \{(make,year) = (Honda, 2000)\} identifies all Honda cars built in year 2000. In the latter case, make ($x_6$) and year ($x_8$) are the set of group_by attributes. A cuboid is defined as the collection of all data cells which share the same set of group_by attributes. For instance, \{make=Honda\} and \{make=Toyota\} belong to the same cuboid (make) while the data cell \{(make,year) = (Honda, 2000)\} belongs to the cuboid (make, year).

Cuboids are arranged in a hyper-cube formation. Let $A_1$ and $A_2$ be the set of group_by attributes of two cells $c_1$ and $c_2$. $c_1$ is an ancestor of $c_2$ if $A_1 \subset A_2$ and is parent of $c_2$ if $A_1 \subset A_2$ and |$A_2$| = |$A_1$| + 1 [13]. Based on our choice of group_by attributes, cuboids in the sparse regression cube are arranged as in Figure 1.

Employing continuous variables like velocity to group data may not result in meaningful cells, because each sample may have its own unique value of that variable. In this case, the variable can be discretized. For example, velocity can be divided into 10 mph bins (i.e., 10–20 mph, 20–30 mph, and so on). We assume that appropriate discretization intervals are provided by the user.
Let vector $Y = \{y_i : 1 \leq i \leq n\}$ and matrix $X = \{x_{ij} : 1 \leq i \leq n, 1 \leq j \leq k\}$ be outputs and inputs to our modeling framework. Let $X_c$ be the matrix formed by gathering rows of matrix $X$ which match the attribute list of data cell $c$ (e.g., all Honda Accords). $Y_c$ can be defined similarly by gathering corresponding values from vector $Y$. Assuming a local linear behavior (i.e., linear behavior at each cell) yields:

$$Y_c = X_c\eta_c + \epsilon_c$$

where $\epsilon_c$ is a zero mean noise with variance $\sigma^2$ that is not correlated with $X$. The regression model at cell $c$ can be estimated using the ordinary least-squares method:

$$\hat{Y}_c = X_c\hat{\eta}_c$$

$$\hat{\eta}_c = (X_c^T X_c)^{-1} X_c^T Y_c$$

where $\hat{\eta}_c$ is the vector of estimated regression coefficients for cell $c$. Let $c_1, \ldots, c_m$ be children of cell $c$. Based on the group-by nature of data cube dimensions, we can write:

$$X_c = [X_{c_1} \cdots X_{c_m}]^T, \quad Y_c = [Y_{c_1} \cdots Y_{c_m}]^T$$

Similar to traditional regression methods, we use the mean square error of a regression model as the measure of accuracy of the model [10], [12]. However, we propose a particular reliability or confidence measure for the calculated mean square error later in Section IV. Formally, we use the residual sum of squared errors for a cell $c$, defined as follows:

$$Err_c = \sum_{i \in c} (y_i - \hat{y}_i)^2 = (Y_c - X_c\hat{\eta}_c)^T (Y_c - X_c\hat{\eta}_c)$$

The following example illustrates regression coefficients and error of a data cell.

**Example 1 (Regression Coefficients and Error):** Given the data in Table I and cell $c : \{\text{make} = \text{Toyota}\}$, we have:

$$Y_c = \begin{bmatrix} 0.030 \\ 0.027 \\ 0.031 \\ 0.067 \\ 0.033 \\ 0.03 \end{bmatrix}, \quad X_c = \begin{bmatrix} 40 & 0 & 1 & 3.086 & 34.3 \\ 40 & 1 & 0 & 2.723 & 32.9 \\ 30 & 0 & 2 & 2.467 & 25.3 \\ 45 & 0 & 1 & 5.390 & 38.3 \\ 40 & 1 & 2 & 3.490 & 31.5 \\ 35 & 0 & 1 & 3.086 & 11.4 \end{bmatrix}$$

According to (1) and (2), the estimated regression parameters and residual sum of squares for cell $c$ are given by:

$$\hat{\eta}_c = \begin{bmatrix} -0.0005 \\ -0.0034 \\ -0.0002 \\ 0.0149 \\ 0.0003 \end{bmatrix}^T$$

$$Err_c = (Y_c - X_c\hat{\eta}_c)^T (Y_c - X_c\hat{\eta}_c) = 3.9342 \times 10^{-5}$$

The values of regression coefficients should be calculated for each data cell. This calls for an efficient computation method using an algebraic representation of a regression model. This mechanism is described in the next section.

**IV. RELIABILITY MECHANISMS**

This section presents the mechanisms that are used in sparse regression cubes to construct a reliable hierarchy of regression models for different levels of abstraction over the input space. We first present an algebraic representation of regression models that enables efficient calculation of regression coefficients and error at each data cell. Next, we describe the mechanism used to determine if a cell is reliable or not and to provide a confidence interval with every prediction. We propose our reliable modeling technique that uses the algebraic representation and the reliability measure to select appropriate predictors for each data cell. Finally, we summarize how the cube is constructed and predictions are performed.

**A. Algebraic Representation of Regression Models and Error**

The size of the data cube described in Section III can be exponential with respect to the number of group-by attributes, $d$. Although later in this section we discuss how to avoid building the exponentially large structure, it is essential to compute regression model coefficients and error ($\hat{\eta}_c$ and $Err_c$) for each cell efficiently and without accessing the raw data. It was shown previously that regression coefficients, $\hat{\eta}_c$, could in fact be efficiently computed in a hierarchical manner [5]. Here, we show that regression error can be computed in a hierarchical manner as well.

We propose a set of matrices to be stored in each cell that can be used to derive regression models while supporting distributive aggregation:

- $\rho_c = Y_c^T Y_c$: A scalar value
- $\nu_c = X_c^T Y_c$: Vector of size $k$
- $\Theta_c = X_c^T X_c$: A $k \times k$ matrix
- $n_c$: Number of tuples in cell $c$

This is called the Algebraic Representation of the cell regression model. Note that, none of the matrices used in this representation depend on $n$. Therefore, their sizes are constant with respect to the number of data tuples. An example of such algebraic representation is presented next.

**Example 2 (Algebraic Representation):** Consider data cell $c : \{\text{make} = \text{Toyota}\}$ and matrices $X_c$ and $Y_c$ from the previous example. One can derive the algebraic representation matrices as follows:
\[ \rho_c = 0.0091 \]
\[ \nu_c = \begin{bmatrix} 8.595 & 0.06 & 0.255 & 0.8115 & 6.6492 \end{bmatrix}^T \]
\[ \Theta_c = \begin{bmatrix} 8950 & 80 & 260 & 796.53 & 6829.5 \\ 80 & 2 & 2 & 6.21 & 64.4 \\ 260 & 2 & 11 & 23.48 & 197.6 \\ 796.53 & 6.21 & 23.48 & 73.78 & 609.4 \\ 6829.5 & 64.4 & 197.6 & 609.4 & 5488.09 \end{bmatrix} \]
\[ n_c = 6 \]

In order to calculate the regression model from the algebraic representation, we reformulate the derivation of regression coefficients and error for cell \( c \) as follows:

\[ \hat{\eta}_c = (X_c^T X_c)^{-1} X_c^T Y_c = \Theta_c^{-1} \nu_c \]  

(3)

\[ E_{rc} = (Y_c - X_c \hat{\eta}_c)^T (Y_c - X_c \hat{\eta}_c) = Y_c^T Y_c - (X_c^T \hat{\eta}_c)^T Y_c - (X_c^T \hat{\eta}_c) X_c^T \hat{\eta}_c = \rho_c - \hat{\nu}_c \nu_c - \nu_c^T \hat{\nu}_c + \hat{\nu}_c^T \Theta_c \hat{\nu}_c \]  

(4)

So far, we have shown how to derive the regression coefficients and error from the algebraic representation. Next, we show how they can be hierarchically aggregated. Let \( i = 1, \ldots, m \) be the \( m \) cells used to obtain aggregate values for a cell \( c \). It can easily be verified that \( \rho_c, \nu_c, \Theta_c, \) and \( n_c \) are distributive measures and can be accurately aggregated as follows:

\[ \rho_c = Y_c^T Y_c = \begin{bmatrix} Y_{c1}^2 & \cdots & Y_{cm}^2 \end{bmatrix} = \sum_{i=1}^{m} Y_{ci} Y_{ci} = \sum_{i=1}^{m} \rho_{ci} \]

\[ \nu_c = X_c^T Y_c = \begin{bmatrix} X_{c1} Y_{c1} & X_{c2} Y_{c2} & \cdots & X_{cm} Y_{cm} \end{bmatrix} = \sum_{i=1}^{m} X_{ci}^T Y_{ci} = \sum_{i=1}^{m} \nu_{ci} \]

\[ n_c = \sum_{i=1}^{m} n_{ci} \]

The result for \( \Theta_c \) is obtained similarly in previous work [5]:

\[ \Theta_c = \sum_{i=1}^{m} \Theta_{ci} \]

We conclude that the regression error can be derived hierarchically alongside with the regression parameters from the algebraic representation. The space complexity of the algebraic representation is \( O(k^2) \).

### B. Reliability Measure

The second mechanism proposed in our framework is to determine whether or not a specific model can be confidently used for prediction. There are several established statistical tests [12] to evaluate a regression model (e.g., Cross-validation). In all cases, access to the raw samples is required for the test to work. We develop a reliability criterion which only uses the information stored in the data cells (i.e., the algebraic representation). Not requiring further information makes this criterion easy to evaluate and therefore usable in the context of a data cube with a potentially large number of data cells.

The purpose of this criterion is to make sure that the sampled estimator remains the least error estimator for the whole distribution of data corresponding to the data cell. We use the term Reliable Cell to refer to a cell where the above is true with a probability greater than a threshold, \( p \):

**Definition 1 (Reliable Cell):** For a given probability threshold, \( p \), a cell is reliable if it satisfies \( \Pr(\|\hat{\eta}_c - \eta_c\| > \delta) < p \).

Here, \( \hat{\eta}_c \) and \( \eta_c \) are the estimated and actual regression parameters, \( \delta \) is the confidence interval, and \( \|x\| \) denotes the \( \ell_2 \) norm of vector \( x \). Intuitively, we use Markov’s inequality to derive an upper bound on the difference between our estimated coefficients and the actual values:

**Theorem 1:** \( \Pr(\|\hat{\eta}_c - \eta_c\| > \delta) \leq \frac{k\sigma^2}{\lambda_{\text{min}}(X_c^T X_c)} \) where \( \lambda_{\text{min}} \) denotes the minimum eigenvalue and \( \sigma^2 \) is the variance of the inherent model error. To prove the theorem, we observe that \( \hat{\eta}_c \) is the least squared estimator of \( \eta_c \), and hence is unbiased [12]:

\[ E[\hat{\eta}_c] = \eta_c \]

Using Markov’s inequality, the probability at which the estimator deviates from its mean is shown [11] to be:

\[ \Pr(\|\hat{\eta}_c - \eta_c\| > \delta) \leq \Pr(\|\hat{\eta}_c(i) - \eta_c(i)\| > \frac{\delta}{\sqrt{k}} \text{ for some } i) \]

assuming that all sample errors, \( e_i \), are zero mean and have the same variance of \( \sigma^2 \). This is a common assumption for linear regression models. Consequently, the variance of \( \hat{\eta}_c \) equals \( \sigma^2(X_c^T X_c)^{-1} \). Hence:

\[ \text{var}(\hat{\eta}_c(i)) = \sigma^2 \lambda_{\text{max}}(X_c^T X_c)^{-1} \]

\[ \Pr(\|\hat{\eta}_c - \eta_c\| > \delta) \leq \frac{k\sigma^2}{\lambda_{\text{min}}(X_c^T X_c)} \]

where \( \lambda_{\text{max}} \) and \( \lambda_{\text{min}} \) are the maximum and minimum eigenvalues. Since \( e_i \)'s have zero mean, \( \sigma^2 \) can be estimated from the mean square error of the regression. In particular, \( \sigma^2 \) is obtained by [12]:

\[ \hat{\sigma}^2 = \frac{\sum_{i \text{ belongs to } c} e_i^2}{n_c - k} = \frac{E_{rc}}{n_c - k} \]
The reason behind using $n_c - k$ in the denominator is that $|\eta_c| = k$ degrees of freedom is lost by fixing the mean values in $\eta_c$. To illustrate this, consider the case where $n_c = k$. Since the regression model completely fits the data, the regression error is always zero. This overfitting leads to a wrong estimate for $\sigma^2$ if the degrees of freedom are not adjusted.

In our algorithm, we use a probability threshold of $p = 0.05$ to identify whether or not a cell is reliable. In other words, we want the regression coefficients to be in the confidence interval of $\delta^2$ with at least 95% probability. To capture the effect of scaling in $\eta_c$, we set $\delta$ to be $\ell_2$ norm of $\hat{\eta}_c$, $\delta = ||\hat{\eta}_c||$. Trivially, we cannot build a model when the number of samples is less than the number of dimensions (i.e., $n_c < k$). Therefore, a cell is reliable if it satisfies $n_c > k$ and $\frac{\sigma^2}{\delta^2 \lambda_{\text{min}}(\Theta_c)} < 0.05$.

The benefit of employing this criterion instead of using traditional approaches such as cross-validation is its minimal computational overhead. A cross-validation is very time consuming due to the need to access the raw data. To illustrate this, we compared the average execution time of our reliability condition against leave-one-out cross-validation for an average data cell (which matches 5000 tuples) both implemented in C++. On a Windows desktop computer with 4GB of memory, the cross-validation is $5 \times 10^4$ times slower than evaluating the reliability criterion largely because of access to the raw dataset.

For any given prediction task given to the data cell, the expected prediction error can be derived using:

$$E[(y - \hat{y}_c)^2] = E[(\hat{y}_c + c - \hat{y}_c)^2] = E[(\hat{y}_c - \hat{y}_c + c)^2]$$

where $y$ is the actual value of the attribute being predicted. In case that the regression model in cell $c$ is perfectly accurate ($\hat{y}_c = \eta_c$), we expect the following prediction error according to (5):

$$EPE = E[c^2] = \sigma^2 = \frac{\text{Err}_c}{n_c - k}$$

We can derive a 95%-confidence bound on the prediction error using the reliability criterion. Since $c$ and $x$ are independent and $c$ has a zero mean, expected prediction error equals:

$$EPE = E[(x^T (\hat{\eta}_c - \hat{\eta}_c))^2] + E[c^2] \leq E[x^T \hat{\eta}_c (\hat{\eta}_c - \hat{\eta}_c)^T (\eta_c - \hat{\eta}_c)] + \sigma^2$$

Cauchy-Schwarz inequality is used here to conclude that $E[(\hat{\eta}_c - \hat{\eta}_c)^2] \leq x^T \hat{\eta}_c (\hat{\eta}_c - \hat{\eta}_c)^T (\eta_c - \hat{\eta}_c)$. We already know that with 95% probability, $||\eta_c - \hat{\eta}_c|| \leq \delta$ if the reliability condition is met. Therefore, with 95% probability the prediction error is bounded by:

$$EPE \leq E[x^T \hat{\eta}_c |\delta^2| + \sigma^2$$

Example 3 (Data Cell Confidence): Consider the cell $c$: \{\text{make = Toyota}\} from the previous example. We estimate $\hat{\sigma}^2$.

$$\hat{\sigma}^2 = \frac{\text{Err}_c}{n_c - k} = \frac{3.9342 \times 10^{-5}}{6 - 5} = 3.9342 \times 10^{-5}$$

We have $\delta = ||\hat{\eta}_c|| = 0.0153$. So, the confidence probability is computed as

$$\frac{k\sigma^2}{\delta^2 \lambda_{\text{min}}(\Theta_c)} = \frac{1.9671 \times 10^{-4}}{(2.3387 \times 10^{-4})(0.8172)} = 1.0337$$

This means that the probability at which the estimated parameters deviate more than $\delta$ from the actual values is bounded by 1.0337 (i.e., not bounded at all). Since this is more than the 5% limit in reliability condition, the cell is unreliable.

C. Reliable Modeling

The third mechanism proposed in our framework is to choose the right subset of predictors that ensures the reliability of the model hierarchy by employing the algebraic representation and evaluating the reliability criterion. A smaller model requires fewer samples to achieve the same confidence level. However, for a given sample set, reducing model size theoretically increases the regression error. Hence, our goal is to find a subset of attributes which minimizes the error while the data cell is kept reliable.

The problems of feature selection and model shrinkage have been widely studied [10]. Mechanisms such as forward selection or backward elimination [17] are widely used to reduce the number of predictors used in the regression models and can prevent the over-fitting of a model. Our reliable modeling mechanism differs from traditional feature selection in the sense that it is based on our reliability measure. Hence, it can build models with quantifiable confidence values. Furthermore, it only uses the algebraic representation matrices to permit efficient modeling. We first define the reduced model, a model that only uses a subset of predictors for prediction and then propose a simple approach based on the forward selection method to derive the reduced model from the original model without accessing raw data.

Let $L = \{a_{r_1}, \ldots, a_{r_j}\}$ be the set of $j$ attributes used for regression. We define $X^L$ to be sub-matrix of $X$ given by column indices in $L$. The following equation represents a regression carried out only on the attributes included in $L$:

$$Y_c = X^L_c \hat{\eta}^L_c$$

$\hat{\eta}^L_c$ is the Reduced Regression Parameters based on attribute list $L$. $\text{Err}^L_c$ is the corresponding regression error.

Computing $\hat{\eta}^L_c$ and $\text{Err}^L_c$ for every $L$ should be fast and efficient. To this end, we propose a method based on transforming the algebraic representation matrices. The transformed matrices are then used as before to derive $\hat{\eta}^L_c$ and $\text{Err}^L_c$. Formally, the reduced model based on $L$ on cell $c$ is described by the following algebraic representation:

- $\rho^L_c = Y^T Y = \rho_c$
Algorithm 1 Model Reduction

1: \( L \leftarrow \{\}\)
2: while \( |L| < k \) do
3: \( \text{Candidates} \leftarrow \{\}\)
4: for every \( 1 \leq a \leq k, a \notin L \) do
5: Compute the transformed matrices for \( L \cup \{a\} \)
6: if corresponding model is reliable then
7: \( \text{Candidates} \leftarrow \text{Candidates} \cup \{a\} \)
8: end if
9: end for
10: if \( \text{Candidates} = \{\}\) then
11: \( L \leftarrow L \cup \{a^*\} \)
12: end if
13: end while

- \( \nu_c^L \): a sub-vector of \( \nu_c \) obtained by selecting dimensions in \( L \).
- \( \Theta_c^L \): a sub-matrix of \( \Theta_c \) obtained by choosing row and column indices in \( L \).
- \( n_c^L = n_c \)

It is clear that the elements in algebraic representation matrices are not modified at all. In fact, a reduced model is obtained just by excluding some of the rows and columns of the original measures from matrix operations. This means that no extra storage or computation is required on top of the original algebraic representation.

Given such an approach for reducing model dimensions, the naive model reduction technique is to search over all possible subsets of regression attributes and find the one which conforms to the reliability condition and has the minimum regression error. This search space is indeed exponential in size and so its growth is 

\[ \Theta_c^L \left( \begin{array}{c} X_c^L \end{array} \right)^T \left( \begin{array}{c} X_c^L \end{array} \right) = \text{rows}_L \left( \begin{array}{c} X_c^L \end{array} \right)^T \text{rows}_L \left( \begin{array}{c} \nu_c^L \end{array} \right) \]

or

\[ n_c^L = n_c \]

The pseudo-code for model reduction is presented in Algorithm 1. The running time of this method is \( O(k^2) \). The following gives an example of the model reduction algorithm:

**Example 4 (Model Reduction):** Considering cell \( c \) in Example 2, we run the model reduction algorithm to find the appropriate set of predictors. We choose \( L = \{\}\) at the beginning and add all potential attributes to \( L \). The left section of Table II shows 5 different sets of single attributes. Then, the error is computed for each set according to (2). After the first step, only the models based on \( L = \{v_1\} \) and \( L = \{v_4\} \) are reliable while \( L = \{m\} \) has the least mean squared error. Therefore, we fix \( m \) as one of the predictors and try all attribute combinations involving \( m \) in the next step.

After the second step, \( L = \{v_1, m\} \) and \( L = \{m, v_4\} \) both generate reliable models while \( L = \{v_1, m\} \) has the minimum error. Trying combinations of size three while fixing \( m \) and \( v_1 \) as two of the predictors yields one reliable sub-model: \( L = \{v_1, ST, m\} \). After this, we also add \( ST \) to the list of reliable models. The model reduction algorithm outputs \( L = \{v_1, ST, m\} \) as the final set of predictors. The final regression error is 1.3486 × 10⁻⁴.

**D. Summary**

A complete data cube has an exponential number of cuboids (and data cells) in terms of the number of group-by dimensions, \( d \). Calculating even simple measures over all data cells can be intractable when facing such an exponential explosion. Hence, data cubes are often partially constructed or materialized [5], [9], [13]. In other words, only measures for a small subset of cuboids, called *Cube Shell*, are calculated to improve the efficiency. We aim to limit the number of cells that are being materialized to the ones which provide the most useful predictive information similar to the Cube Shell Construction algorithm [13].

Given the pieces that enable reliable hierarchical modeling, we summarize the cube construction procedure as follows: We start by the apex cuboid (which groups all data together and builds a single model) and compute the reliable regression model for all of the cells in that cuboid and its children cuboids. We then find the cuboid with the lowest prediction error compared to its parent. In this process, cuboids that contain an unreliable cell are discarded. At each step, we add the cuboid with the maximum reduction in average error and consider its children as the next-step candidates. This is done until the there is no error reduction above some threshold or all new cells are unreliable.

The intuition behind this algorithm is that drilling down to child cuboids should reasonably improve the prediction accuracy. If the child cuboid does not have a significantly lower regression error than the parent, it is unlikely to be more accurate since it is more sparse comparing to the parent.
When it is desired to predict an output (e.g., fuel consumption of a Ford Taurus 2005 on a given route), we start from the bottom of the cube. If the matching base cell is materialized (i.e., we have a cell for a Ford Taurus 2005), the regression model from that cell is used to derive the output value. Otherwise, ancestor cells are considered (e.g., “Ford Taurus”, “Ford” or “All cars”). To this end, we perform a tree search from the base cell until we find a materialized parent cell and use it for prediction. We emphasize that any data cell that is materialized is reliable according to our criterion.

V. EVALUATION

This section evaluates the performance of our modeling framework when used for GreenGPS [7] predictions. The service finds the most-fuel efficient route between any two given locations by predicting the fuel consumption of cars on different road segments. The data set used for this evaluation has been obtained by instrumenting 20 cars with scanning devices to record various parameters including vehicle location, vehicle speed, instant and average fuel consumption rate, and engine RPM (revolutions per minute) [7]. These data are then processed using city maps to derive attribute values for each street segment. The data collected from the total of 1000 miles of driving. Accompanied with vehicle information such as make, model, year, weight, and the frontal area, the total number of attributes to predict the consumption of a vehicle on a road segment adds up to 14.

In addition to the data collected from the tested, we generate larger synthetic data sets based on the collected distribution of values to evaluate the scalability of our approach. We implemented the sparse regression cube in C++. The experiments are performed on a Windows desktop computer with 2.4 GHz Intel Core 2 processor and 4GB of RAM using a 80GB Hitachi (7200RPM) hard disk drive. In this section, we first evaluate the prediction accuracy of our approach. Next, we evaluate the reliability provided by our sparse regression cube and verify that the prediction error is indeed within the confidence interval. Finally, we evaluate the scalability and efficiency of sparse regression cubes.

A. Prediction Accuracy

We compare the prediction accuracy of the sparse regression cube against three other approaches: i) a single multi-dimensional regression model obtained by using support vector regression (SVR) [10]. ii) regression cubes [5], and iii) sampling cubes [13]. The sampling cube only uses the average value of a data cell as the model. It is analogous to using the MPG rating of the car. We choose SVR since it is one of the most accurate single-model regression tools [10] while regression and sampling cubes are the most relevant hierarchical modeling methods. Other hierarchical modeling techniques such as prediction cubes and logistic regression cubes are more appropriate for classification problems.

Figure 2 presents the prediction accuracy of various techniques. The accuracy is computed using leave-one-out cross validation (i.e., the fuel consumption is predicted for each road segment from model learned from the other segments). The average error over all evaluated samples is then reported. First, we observe that the sampling cube performs a poor prediction. This simply emphasizes on the need for multi-dimensional regression in these cases instead of merely using the average values such as MPG ratings. It is also interesting that using regression cubes for predictive regression without appropriate techniques for sparse data cells (such as model reduction) does worse than a one-size-fits-all model. The reason is that previous regression cubes (that do not consider reliability), build models for individual cells that end up unreliable when data is sparse. The sparse regression cube on the other hand shows a significant reduction in prediction error.

We repeat the comparison for 6 fixed trips in the GreenGPS data set, where the total fuel consumption for each trip is estimated. The percentage of prediction error with respect to the actual fuel consumption is reported in Figure 3. Again, our new technique significantly improves prediction accuracy.

B. Reliability Measure

As discussed before, our sparse regression cube provides a 95% confidence bound on the prediction error of each prediction query. The confidence bound is the error value that is guaranteed to be larger than the actual prediction error 95% of the time. In the next experiment, we verify that the estimated prediction error and the 95% confidence bound match the actual prediction error. To this end, we randomly select a road segment driven by a given car and predict its fuel consumption using a model trained from data on other cars and segments. We change the size of training set and measure the mean squared prediction error observed, then compare it.
against the analytically estimated prediction error from our Equation 6 as well as the 95% confidence bound given by Equation 7. The results, depicted in Figure 4, show that the estimated error is very close to the actual observed prediction error and is indeed less than the confidence bound. To make error values meaningful, we have normalized fuel consumption values to be zero mean and between −1 and 1.

C. Efficiency and Scalability

We use $500K$ synthetic samples to evaluate the scalability of our framework. Each tuple consists of a fuel consumption value as well as 14 attributes. The data used for this evaluation are synthesized using the following method: We first identify a subset of the attributes to be independently generated. These attributes are similar to the real data set and include the attributes of the road. Each of these attributes are generated using a discrete uniform distribution. We also uniformly generate make, model, and year values as well as other vehicle parameters such as weight and frontal area. Finally, vehicle speed is derived from a normal distribution where its mean and variance are determined by the road parameters. Our data generation scheme assigns about 1000 samples to each vehicle type (make, model, and year). Until otherwise specified, the number of regression dimensions, $k$, for synthetic data is 14.

Our scalability and efficiency evaluation is based on two performance metrics: i) materialization time, and ii) memory usage. The materialization time is the interval between the time the first access to disk is performed until the cube shell construction algorithm is complete. Memory usage is derived both using a probe subprogram to evaluate run-time C++ structure sizes and the total amount of memory used by the program. We compare the sparse regression cube with full functionalities against two other approaches. The first approach completely materializes the cube using bottom-up computation (BUC) [2] while still using the algebraic representation for cell measure aggregation. The second approach employs the cube shell construction algorithm but instead of algebraic representation, reprocesses raw data when materializing each cell.

In the first experiment, we evaluate the effect of data cube dimensions on the materialization time. We change the number of group by dimensions, $l$, from 5 to 14 and calculate the running time for each approach (Figure 5(a)). Since full materialization and raw reprocessing approaches cannot scale with the number of tuples, we use only $50K$ tuples in this experiment to illustrate the difference. Full materialization exposes an exponential increase in running time in the same trend as with the cube size. The raw reprocessing approach maintains the same gap against the sparse regression cube since the number of data tuples are not changing. Memory usage of different approaches is compared in Figure 6(a). Bottom-up computation reveals the same exponential explosion.

Next, we study the effect of regression dimensions, $k$, on the performance metrics. Again we use $50K$ tuples but fix the group by dimension to 10. Figure 5(b) depicts the materialization time for the three approaches. The number of regression dimensions does change the cube size and hence does not result in an exponential explosion of the full materialization approach. This uniformly increases the computation time for all approaches as the number of data cells in high dimensional cuboids (i.e. close to the base cuboid) increases. We also plot the memory usage in Figure 6(b). The number of regression dimensions have a more significant impact on the memory usage since it increases the algebraic representation matrix sizes accordingly. The size of the cube shell is approximately 30 cuboids and independent of $k$.

In the final scalability experiment, we change the number of tuples used for modeling between $10K$ and $500K$ while fixing $l = 10$ and $k = 14$. The materialization time and storage requirement of the cube is presented in Figures 5(c) and 6(c). Since the number of cuboids that shell construction methods materialize is constant, the running time and memory time increases linearly with respect to the number of tuples. BUC on the other hand shows an exponential relation.

VI. RELATED WORK

Modeling complex non-linear systems has been a challenge in a variety of different areas from control theory to data mining. Statistical estimators such as Kalman filters, or Hidden Markov Models (HMMs) are frequently used to model systems. Regression analysis through statistical learning is widely used for prediction. Multi-variate regression analysis techniques are very mature and have been studied for decades [1], [10], [12], [15]–[17]. In addition to least-squares minimization estimators, several techniques such as Support Vector Regression [6] or Ridge regression [10] provide robust predictors.

Many mechanisms enable choosing right set of regression predictors or reduce the dimensionality of the regression problem. Attribute (feature) selection techniques such as forward stepwise selection and backward elimination [17], or least-angle regression have been proposed to choose the right prediction attributes. Principal component regression based on singular value decomposition [10] is another major technique to avoid inaccurate modeling in high-dimensions or when using complex models.

Hierarchical regression techniques that construct several linear regression model have been studied in various contexts. Regression trees [3] build a hierarchy of regression models. In
data mining and data base contexts, data cubes are considered powerful tools to explore high dimensional spaces as well as to perform predictive modeling [14]. Regression Cubes [5] and Logistic Regression Cubes [18] construct linear and logistic regression models for each data cell in the cube.

Other data mining constructs try to perform prediction using methods other than regression. Prediction Cubes [4] materialize various machine learning prediction tools at different data abstraction levels. The Sampling Cube [13] focuses on how to generalize queries to compensate for sparse data cells. None of the previous work considers reliable modeling in a hierarchical settings which is essential for open cyber-physical systems.

VII. CONCLUSIONS

This paper presented a novel modeling technique, called the Sparse Regression Cube, to perform reliable hierarchical modeling for open cyber-physical systems. Our framework jointly partitions sparse, high-dimensional measurement data into subspaces within which the output can be predicted reliably, and determines the best reliable model for each subspace. Uncertainty in prediction is quantified. Three mechanisms achieve this capability: (i) an algebraic representation for efficient calculation of regression models and errors, (ii) a reliability condition and confidence interval for each local model, and (iii) an algorithm to construct the reliable models. We applied our technique to a green transportation service called GreenGPS, and demonstrated improved accuracy and reliability in prediction.

REFERENCES