Linear Programming-Based Optimization for Robust Data Modeling in a Distributed Sensing Platform

Anurag Umbarkar, Varun Subramanian, and Alex Doboli, *Senior Member, IEEE*

***Abstract*—Creating accurate data models describing the dynamics of physical phenomena in time and space is important in optimized control and decision making. Models highlight various trends and patterns. However, producing accurate models is challenging as different errors are introduced by sampling platforms with limited resources, e.g., insufficient sampling rates, data loss due to buffer overwriting, reduced communication bandwidth, and long communication delays. Furthermore, the dynamics of the environment, like mobile energy sources and sinks, might further increase errors as resources must be shared between the sampling and communication activities. This paper presents a procedure to systematically construct robust data models using samples acquired through a grid network of embedded sensing devices with limited resources, like bandwidth and buffer memory. Models are in the form of ordinary differential equations. The procedure constructs local data models by lumping state variables. Local models are then collected centrally to produce global models. The proposed modeling scheme uses a linear programming formulation to compute the lumping level at each node, and the parameters of the networked sensing platform, i.e., best data communication paths and bandwidths. Two algorithms are described to predict the trajectories of mobile energy sources/sinks as predictions can further reduce data loss and delays during communication. The computed parameters and trajectory predictions are used to configure the local decision making routines of the networked sampling nodes. Even though the procedure can be used to model a broader set of phenomena, experiments discuss the effectiveness of the method for thermal modeling of ULTRASPARC Niagara T1 architecture. Experiments show that the presented method reduces the overall error between 58.29% and 76.91% with an average of 68.87%, and communication delay between −11.49% and 57.62% with an average of 21.85%.**

***Index Terms*—Design for space exploration, mixed-mode, modeling, symbolic techniques.**

I. Introduction

CYBER-PHYSICAL systems (CPS) are expected to integrate data acquisition, networking, and control in an effort to produce effective decisions while operating in complex physical environments [19]. The acquired data is utilized to build data models, which are then used to devise (or synthesize) optimized control (decision) strategies by estimating, predicting, and identifying trends and patterns of the physical environment as expressed by data models. Data model accuracy is critical as unaccounted modeling errors might produce unpredicted decisions, hence reduce the robustness of the systems.

This paper focuses on data models expressed as ordinary differential equations (ODEs) in time and space and describing the attributes of the observed environment. ODEs have been used to present the energy conservation laws for a variety of physical environments and phenomena [3]. For example, the temperature and density of substances in dissipative or nondissipative 3-D volumes (e.g., the liquid inside a computer server cooling system, or the gases forming the atmosphere in a city or a room) is modeled by instantiating the flow equations in physics based on the sampled signals, energy sources, and energy flow properties of the monitored physical environment. Recent work focuses on techniques to automatically create data models for physical environments, including the causal origin of the observed variations and correlations [24], [29].

The ODEs formulated for a physical environment are solved by discretizing the equations through backward (or forward) Euler integration formula, e.g., 7-point finite difference discretizing [15], [27]. Equation discretizing is described as a network of energy injection and removal elements connected through transfer elements (i.e., resistors and capacitors) that propagate energy in time and space [see Fig. 2(b)]. Depending on the nature of the physical phenomenon, the energy injection/removal and transfer elements are described as linear, nonlinear, or stochastic expressions over the parameters of the phenomenon. In circuit design, temperature maps are examples of data models. A multigrid thermal modeling method is proposed in [15] to address the heterogeneous areas of 3-D ICs. A similar approach, based on adaptive discretizing grids, is discussed in [9].

Data model construction in CPS applications is challenging because the dynamic of physical environments is hard to track accurately and in real-time through a distributed sensing network with limited resources, like bandwidth and memory. Continuously streaming large volumes of data samples can result in important data losses due to low bandwidth and many data overwrites in the memory buffers. Also, low bandwidths and long communication paths increase the time delays of data samples, thus adding further errors to time-sensitive data models. Fig. 1 shows that, even for a smaller, 25-node network, there is significant difference between the real thermal map (left) and the map (right) based on distributed sensing. As the network size grows, the streaming volume increases and consequently the difference in the thermal maps further

Manuscript received January 19, 2014; revised April 9, 2014; accepted May 26, 2014. Date of current version September 16, 2014. This paper was recommended by Associate Editor Y. Wang.

The authors are with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794-2350 USA (e-mail: adoboli@ece.sunysb.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2014.2334295

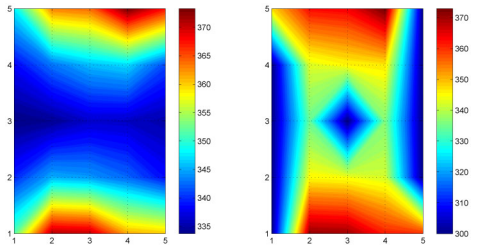


Fig. 1. (Left) Complete thermal map. (Right) Thermal map for distributed sensing with data loss. The map is significantly distorted due to data loss.

increases. The modeling error reduces the optimality of the devised control decisions of CPS applications.

The impact of data loss on model accuracy can be mitigated by building local data models, and then streaming the local models instead of the samples used to create the models. For example, for wireless sensor networks (WSNs), data aggregation has been proposed to improve the network performance, like throughput, bandwidth, and energy consumption [11], [12], [16]. A regression-based framework for modeling sensor data is presented in [10] and [13]. Data communications are reduced by exploiting the correlations between data sensed by neighboring nodes [32] or by fitting generic functions that approximate the transmitted values [2].

Building accurate, global data models from streamed local models raises two main issues. First, the decision on which local models to produce (e.g., which state variables to express) is made only based on local information even though the decision impacts the accuracy of the global model. The embedded sensing nodes lack global knowledge about the environment's dynamics, including moving energy sources and sinks, and changing sensitivities of its state variables. This issue introduces significant differences from methods like [9], [15], and [20], which assume statically known environments in which all state variables are available to construct the model. Second, due to data loss, model accuracy depends on the resource characteristics of the distributed sensing platform. Resource utilization must be optimized such that the resulting data loss and time delays generate a minimum modeling error. This issue has been less explored by related work.

This paper presents a procedure to systematically construct robust data models based on samples acquired through a grid network of embedded sensing devices with limited resources, like bandwidth and buffer memory. The data models are in the form of ODEs. The procedure constructs local data models by lumping state variables, so that the errors due to data loss and time delays are minimized. The model error is captured by mathematical bounds for four types of errors: due to data loss, time delays and clock nonsynchronization, loss of correlation information, and variable lumping (loss of state variables). These errors describe the types of inaccuracies that occur during data model construction using a distributed sensing network. The proposed bounds are then used to compute using linear programming (LP) the lumping level of the local models (e.g., number of lumped variables) and the parameters of the networked sensing platform, including data communication paths and bandwidth rates. The computed parameters are utilized to set threshold values used by local schemes (at the embedded sensing nodes) to decide the specific modeling actions (see Fig. 10).

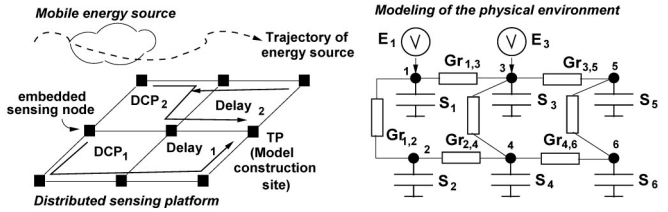
The novelty of the work is in that it emphasizes the robustness of the created data models (ODEs) by tackling the connection between modeling error and model characteristics, physical environment dynamics, and sensing platform resources. Traditional aggregation methods [8], [13], [32] focus on reducing communication traffic to lower power and energy consumption. Feedback-based adaptation addresses the changing traffic conditions and time requirements of data aggregates sent by parent nodes to their children over a tree network [11]. Aggregation also avoids transmission of redundant data [12]. The method in [33] defines a centralized threshold-OR fusing rule for combining sensor samples under normally distributed, independent additive noise conditions. Xue *et al*. [31] propose a locally weighted fusion function for improving model accuracy. A cluster-based technique is proposed in [18] to perform structural health monitoring. Data aggregation is at the cluster-level by filtering spatial and semantic correlations to improve energy efficiency and reduce data storage. Particle swarm optimization (PSO) is used in [25] for data aggregation to minimize network cost and communication delay. The correlations between data sensed by neighboring nodes have been used to reduce the amount of data transmitted in a network [32]. The work in this paper complements these results by adding model accuracy as a main requirement of distributed model construction.

The experimental section uses the proposed data modeling procedure for thermal modeling of ULTRASPARC Niagara T1 architecture [14]. However, the method is not limited to thermal modeling and can be used for modeling other physical phenomena, like the characteristics of ocean water, e.g., salinity, temperature, and pH. There have been extensive studies on thermal modeling in 3-D integrated circuits (3-D ICs). Current techniques [17], [27] observe the thermal behavior of 3-D IC by simulating the chip as a part of the prefabrication validation stage. Once this behavior is characterized, thermal management through voltage and frequency scaling and task scheduling is performed [5]–[7]. Although sensors are distributed over the entire area of the chip, most thermal management policies use a centralized repository of the real-time thermal data. However, policies must include models for data losses and delays associated with the real data collection networks in order to address inevitable modeling errors. The proposed work presents a distributed approach to physical phenomena modeling while minimizing the effect of various model error sources.

The paper has the following structure. Section II presents an overview of physical phenomena modeling, its associated modeling errors, and the proposed modeling methodology. Section III discusses the model error bounds, trajectory prediction, and the related optimization formulation. Section IV details the experiments. Conclusions end the paper.

II. Data Modeling and Error Minimization

This paper considers data models that are ODEs over time and space for the parameters of physical environments.



(a) (b)

Fig. 2. (a) Distributed sensing. (b) Data model structure.

Parameters are strongly coupled with each other, e.g., pressure and temperature, wave propagation speed and gas density, and many more [3]. In addition to environmental characteristics, the dynamic of physical parameters is decided by mobile energy sources and sinks with variable properties. Some energy sources and sinks move along unknown trajectories. Fig. 2(a) illustrates the concept. This section details the characterization of physical phenomena, the types of modeling errors, and the proposed modeling procedure to reduce modeling error.

A. Characterizing Physical Phenomena

Let us assume that $Y_i(t)$ are the state variables of the monitored physical environment, such as state variable Y_i corresponds to the nodes of the discretization grid used in modeling. Note that only some $Y_i(t)$ are observed (e.g., sampled) through sensors. The well-known mathematical expressions of state variables Y are as follows [15]:

$$S_i\dot{Y}_i(t) = \dot{E}_i(t) + \sum_{k\in K}\frac{Y_k(t) - Y_i(t)}{Gr_{k,i}} + \sum_{k\in K}\frac{Y_i(t) - Y_k(t)}{Gr_{i,k}}. \quad (1)$$

K is the set of neighbors of node i. $Gr_{i,k}$ and $Gr_{k,i}$ are the gradient coefficients that define the in- and out-going energy flows between two neighboring points i and k. The coefficients S_i represent the energy stored at node i, and the terms $\dot{E}_i$ are the added or removed energy at node i. The model parameters are dynamic because physical parameters, like temperature, density, pressure, and humidity, change in space and time [3]. Fig. 2(b) illustrates the above mathematical expressions.

Example 1: Section IV present a case study for thermal modeling of integrated circuits (ICs). State variable Y is temperature (T) while coefficients S and Gr represent heat storage and heat transfer coefficients, respectively [3], [7]. The energy injection due to power dissipation of the IC cores is denoted by variable E. Depending on the workload of the cores, the power dissipation varies causing variation in the values of variable E over time and space. Then, (1) corresponds to the following heat transfer equation:

$$S_i\dot{T}_i(t) = \dot{E}_i(t) + \sum_{k\in K}\frac{T_k(t) - T_i(t)}{Gr_{k,i}} + \sum_{k\in K}\frac{T_i(t) - T_k(t)}{Gr_{i,k}}. \quad (2)$$

The left-hand side of the equation denotes the change in heat energy at node i in time dt. The right side indicates the two causes of this change, the energy injection (E) due to power dissipation and the energy transferred from neighboring nodes k to node i. Coefficients S vary according to the heat capacity, and coefficients Gr indicate the thermal conductivity of the material.

The energy sources and sinks in (1) can be static or mobile. Then, energy source (sink) E_i is the sum of all energy sources (sinks) that are located at node i at time moment t. The expression of E_i is as follows:

$$E_i(t) = \sum_{p\in ES} E_p(t)\delta_i\left(\int_t v_p(\tau)d\tau\right). \quad (3)$$

Set ES is the set of all energy sources (sinks) and v_p is the speed of source p. Function $\delta_i(x)$ is one, if $x = i$ at time moment t, otherwise it is zero.

For example, in ICs, energy sources and sinks correspond to thermal energy injection (due to power dissipation) and removal (via liquid-cooling microchannels or heat sinks), respectively. Depending on the variable workload of memories, CPU cores, buses, and other devices, these spots are observed in time at different locations on the chip and are therefore considered mobile.

Finding physical models means, conceptually, that the set of (1) and (3) formulated at each node i of the discretization grid is solved symbolically. For example, if all energy sources and sinks are static ($v_p(t) = 0$), the identified data model are the following solutions of the equation set:

$$Y_i(t) = \sum_k \alpha_{i,k}e^{\lambda_k t} + \sum_l \beta_{i,l}t^l. \quad (4)$$

Parameters $\alpha_{i,k}$, λ_k, and $\beta_{i,l}$ depend on the parameters S_i, $Gr_{i,k}$, $Gr_{k,i}$, and $\dot{E}_{i,p}$ of the modeled process [(1) and (3)]. The solutions indicate explicitly how cause variables influence the parameters of the model as well as the importance (sensitivities) of the variables in deciding the model.

Note that solving symbolically the differential (1) and (3) is difficult. For example, in the general case, the expressions of $\alpha_{i,k}$, λ_k, and $\beta_{i,l}$ are hard to compute. There are closed form solutions only for specific situations. Also, the expressions of the energy sources E_i at points i are usually unknown and their values are not directly sampled by the sensing devices. Also, the expressions of terms S_i, $Gr_{k,i}$, and $Gr_{i,k}$ might be unknown. Hence, the expressions and behavior of these unknowns must be found during model construction, e.g., through profiling and/or identification.

Another difficulty in precise data modeling stems from the correlations between the trajectories of energy sources (sinks) (e.g., sound, heat, etc.) and the communication paths used to transmit the sampled data. As shown in Fig. 2(a), data acquired by the sensor nodes is sent along various data communication paths (DCPs) with minimum loss and within the needed timing constraints. DCPs are defined from the sensing nodes to target points (TPs), where the streamed data is saved. The selected DCPs influence the modeling error as they determine the experienced data loss and delays while forwarding data to TPs. Data loss occurs when data stored in local buffers is overwritten before it is forwarded either because of an ongoing data sampling or data reception. This situation also increases the delay of transmitting data to TPs.

Example 2: Fig. 3(a) shows two different DCPs, and the same mobile energy source trajectory (highlighted in bold)

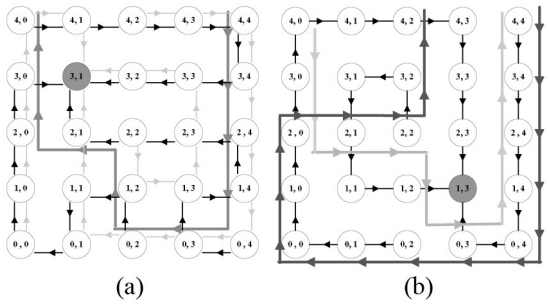


Fig. 3. Data path configurations (DCPs) used in tracking trajectories: (a) Ditrerent DCPs for same trajectory, (b) Same DCP for ditrerent trajectories.

moves through the network for those configurations. The TP is the black bubble. The experimental model for the networked nodes is based on PSoC processor [1]. The first DCP does not experience any data loss for the considered trajectory. The average delay for the nodes is 1434.62 msec and the maximum and minimum delays are 2040 and 1010 msec, respectively. Six nodes in the second DCP experience data loss for the same trajectory. The average delay for the nodes that did not experience any data loss is 1962.86 msec and the maximum and minimum delays are 5660 and 1010 msec, respectively. Hence, nodes have an average delay of 36.82% more than path configuration one. Different DCPs can yield different levels of performance for the same trajectory. Moreover, Fig. 3(b) shows two different trajectories running through the network, which uses the same path configurations. Trajectory one is shown with black line and trajectory two with gray line. Trajectory one has data loss at seven nodes compared to trajectory two which has no data loss. The nodes in trajectory one experience an average delay of 73.43% higher than trajectory two. Hence, the same DCP can yield different levels of performance for different trajectories.

B. Errors During Distributed Data Modeling

Main challenges in constructing precise data models based on samples from a distributed sensing network include minimizing the errors introduced during sensing, estimation and communication of the parameters in (1) and (3). The four types of errors express the inaccuracies that occur when computing locally (at each network node) the parameters S, Gr, and E in (1) based on the values sampled by a node and the data received using DCPs. Errors are grouped into the following four categories.

1) *Errors Due to Data Losses*: Data losses are due to buffer overwriting when streaming from sensors to the collection site for model construction (e.g., TPs). Data losses can be reduced by increasing the buffer sizes but this requires more resources. Such resources are not available for basic sensing nodes. Data loss also occurs when sensor data cannot be acquired due to hardware constraints of the sensing frontends, e.g., insufficient sampling frequency or inaccurate mixed-signal frontends [4].
2) *Errors Due to Time Delays and Clock Nonsynchronization:* The accuracy of the models depends on the delays with which the sampled data becomes available at the TPs [see Fig. 2(a)]. The communication delays are incorporated to some degree into the data models, hence further reducing their accuracy. Transmitting more data samples to construct better models increases the delay of the communication paths. Also, there is no common clock signal available to the distributed nodes, hence there is jitter noise added to the data samples because of nonsynchronization of the local clocks at each sensing node.
3) *Path-Induced Errors:* If the path configuration shown in Fig. 4(a) is used for data collection, certain correlations are missed as the nodes on a DCP do not have access to the samples sent on another DCP. Missed correlations are represented in the figure by dotted lines. The error introduced due to missed correlations is called correlation error. It can be minimized by selecting a path configuration such as in Fig. 4(b).

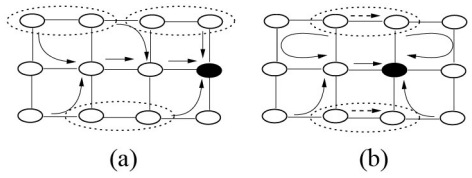


Fig. 4. (a) Error due to missed correlation (dotted lines). (b) Path configuration with minimized error.

Data modeling can use state variable lumping to reduce data loss and delays. Variable lumping eliminates the less significant state variables and simplifies the structure of the discrete representation of the state equations. Variable lumping might eliminate one or several consecutive variables. Lumping introduces the following modeling error.

1) *Errors Due to Lumping:* During lumping, some intermediate state variables are removed. At the TP, the original data needs to be extracted from this lumped information. The inaccuracies observed in the extracted data represent the error due to lumping of state variables while forming the local models.

Assuming an additive error model, the local modeling error at node i is equal to

$$Err_i = Err_i^{(Loss)} + Err_i^{(Delay)} + Err_i^{(Corr)} + Err_i^{(Lump)}. \quad (5)$$

The total modeling error over all nodes i ($\sum_i Err_i$) should be minimized by an optimized modeling scheme.

C. Methodology for Creating Distributed Models

Every sensing node of the network makes local decisions in an attempt to minimize the local error described by (5). The decisions include selection of a certain DCP, the communication rate of the DCP, and performing variable lumping or not. Let us assume that the following probabilities express these decisions: $prob^{(DCP_p)}$ is the probability of using path p (from the set of available paths), $prob^{(r_{j,p})}$ is the probability of using rate j (from the set of available communication rates of DCP p), and $prob^{(lump)}$ is the node's probability to perform variable lumping. Then, finding the optimized implementation scheme requires computing the probabilities for each sensing

TABLE I
SYMBOLS USED IN THE MODEL AND THEIR DEFINITIONS

Symbol	Definition	Symbol	Definition	Symbol	Definition	Symbol	Definition	Symbol	Definition
Y (C)	State variable	E (C)	Energy source/sink	Gr (C)	Transfer coef.	S (C)	Storage coef.	Err (O)	Total error
$Err^{(Loss)}$ (O)	Err.loss	$E^{(Delay)}$ (O)	Err.delay	$E^{(Corr)}$ (O)	Path-ind.err.	$Err^{(Lump)}$ (O)	Err.lump.	$Err^{(Buff)}$ (O)	Err.buff.loss
$Err^{(Coll)}$ (O)	Coll.err.	$error_{(k)}$ (O)	Err.discard.data	γ (P)	Coef.bounds	κ (P)	Coef.bounds	BW (R)	Comm. bandwidth
DCP (R)	Data Comm. Path	t (T)	Time	δ (R)	Time disc.step	p (O)	Lumping level	q_{sens} (R)	Sampling rate of sensors
α (O)	Rate disc.sensor data	β (O)	Rate sel.bandw.	λ (O)	Rate sel.lump.	buff (R)	buff.size	$DATA^{OUT}$ (R)	pack.size
lump_level$_j$ (R)	Lump.level	Loss (O)	Data loss	In_rate (R)	Req.resol.	NET^{IN} (O)	Input data	NET^{OUT} (O)	out. data
$Delay_p$ (O)	Delay DCP	$prob^{(DCP)}$ (O)	Prob.sel.DCP	$prob^{(r)}$ (O)	Prob.sel.BW	$prob^{(Lump)}$ (O)	Prob.lump	x (y)	Cartesian coord.
$N_{alt\ traj}$	Number altern. traj.	l	Length traj.	V	Velocity agent	Grad	Grad. traj.	θ	Angle traj.

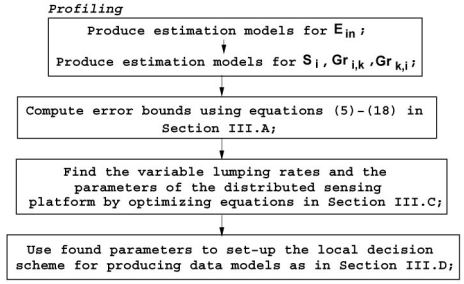


Fig. 5. Data modeling methodology.

node in the network. At run time, every sensing node uses its specific probabilities to dynamically select the most effective DCP depending on the trajectories of the energy sources (sink), configures the parameters of the DCP (e.g., baud rate), and decides whether to lump or not.

Fig. 5 presents the methodology to construct mathematical data models for distributed sensing platforms. The first step, profiling, uses raw data samples of the observed state variables Y (a sub-set of all state variables Y) to produce the initial estimates of parameters E_i, S_i, $Gr_{k,i}$, and $Gr_{i,k}$ in (1) and (3). The parameters are found for a large set of scenarios for the dynamics of the monitored physical entities. Table I summarizes the model variables estimated during profiling [labeled as (P)]. The second step uses the profiling information to find the characteristics of the dynamics, e.g., the bounds for first and second order derivatives of Y and energy sources E_i. This insight is then utilized to find the maximum error bounds that intervene during distributed data sensing. The error types are discussed in Section III-A and captured in (5)–(18). The next step finds the variable lumping scheme and the parameters of the distributed sampling architecture that minimizes the errors of the data models, hence maximizes the model robustness. The parameters are computed by solving the optimization model described by the equations in Section III-C. Finally, the computed parameters are used to set-up the local data modeling decision making scheme of each embedded node. Section III-D presents the local routines.

III. OPTIMIZATION OF DISTRIBUTED DATA MODELING

This section introduces the algorithms and error models that are part of the data modeling methodology in Fig. 5. Section III-A details error modeling. Section III-B discusses trajectory prediction for mobile energy sources (sinks). Section III-C describes the computation of the probabilities used in local decision making: probability $prob^{(DCP_p)}$ of using path p (from the set of available paths), probability $prob^{(r_{j,p})}$ of using rate j (from the set of available communication rates of DCP p), and probability $prob^{(lump)}$ to perform variable lumping. Table I summarizes the variables of the modeling and their meaning.

A. Detailed Error Modeling

The four error types introduced in Section II-A are modeled as follows.

1) Errors Due to Data Loss: Data loss error is the sum of the errors due to buffer loss and collection loss. Buffer loss $Err^{(Buff)}$ represents the error due to buffer overwriting and collection loss $Err^{(Coll)}$ is the error due to the limited sensing (sampling) capabilities of a node

$$Err_i^{(Loss)} = Err_i^{(Buff)} + Err_i^{(Coll)}. \tag{6}$$

Lemma 1: The loss of n consecutive data values due to buffer overwriting at node i is described by the following expression:

$$Err_i^{(Buff)} \leq \frac{n}{BW}||\dot{Y}_{\max}| - |Y_{n+1} - Y_0|| + \frac{n}{BW^2}||\ddot{Y}_{\max}| - |\dot{Y}_{n+1} - \dot{Y}_0||. \tag{7}$$

BW is the data communication bandwidth at node i's input.

Proof: Using the first three terms of the Taylor series (Y_0 is the starting value), $Y(t) \approx Y_0 + \dot{Y}t + \ddot{Y}t^2$. Losing n consecutive data results in estimating the first two derivatives as $\dot{Y} \approx BW((Y_{n+1} - Y_0)/n)$ and $\ddot{Y} \approx BW((\dot{Y}_{n+1} - \dot{Y}_0)/n)$. The total error due to the miss-prediction of the first derivative is $(1/BW)\sum_{i=0}^{n}||Y_{i+1} - Y_i| - |Y_{n+1} - Y_0|| \leq (n/BW)||\dot{Y}_{\max}| - |Y_{n+1} - Y_0||$. Similarly, the error due to the miss-prediction of the second derivative is $(1/BW^2)\sum_{i=0}^{n}||\dot{Y}_{i+1} - \dot{Y}_i| - |\dot{Y}_{n+1} - \dot{Y}_0|| \leq (n/BW^2)||\ddot{Y}_{\max}| - |\dot{Y}_{n+1} - \dot{Y}_0||$.

Collection errors are introduced if data cannot be acquired fast enough due to the hardware constraints of the sensing frontends. For example, the analog-to-digital converters are too slow, or not all sampled values in the input buffers can be processed [4]. Let us assume that error Err_{sens_j} is due to discarding one physical value of sensor $sens_j$. The total error introduced by all discarded data is as follows:

$$Err_i^{(Coll)} = \sum_{\forall sens_j} \alpha_j\ q_{sens_j}\ Err_{sens_j} = \sum_{\forall sens_j} (1sec)\alpha_j\ q_{sens_j}\left(\sum_{\forall k} error_{(k)}\right). \tag{8}$$

q_{sens_j} is the sampling rate of the sensor (number of samples in one second). α_j is the rate of discarding values at sensor $sens_j$ ($\alpha_j < q_{sens_j}$). $error_{(k)}$ is the error introduced for tuple

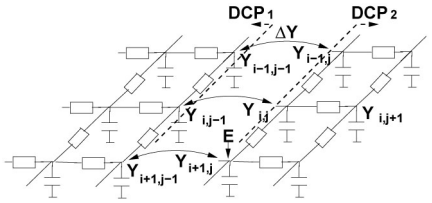


Fig. 6. Path-induced error estimation.

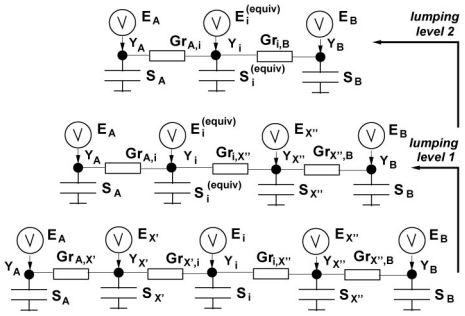


Fig. 7. Multilevel variable lumping.

k by the discarded data. Term 1 sec was added to correctly represent the unit of the error.

In (6), buffer and collection errors represent conceptually the same type of errors, which are due to data packets loss (data is overwritten during communication or not sampled due to low sensor sampling rates). The unit for the errors in (6) is *BU*, the basic unit of the state-variable Y, e.g., degree Kelvin if variable Y is temperature. ■

2) Errors Due to Time Delays and Clock Nonsynchronizations: These errors are introduced by the delays at which the sampled data reaches the target point (TP). Errors change (1) to $S_i\dot{Y}_i(t) = \sum_p \dot{E}_{i,p} + \sum_k ((Y_k(t - T_1^{Delay}) - Y_i(t - T_2^{Delay}))/Gr_{k,i})$, where T_i^{Delay} is the delay of the related data communication paths between nodes i and k. The errors change the fundamental matrix (ϕ) of the equation set (1), which changes the symbolic expressions of the state variables Y (3).

As already explained, it is difficult to compute closed form expressions for state variables Y and then estimate the error based on the differences in the fundamental matrix due to time related errors. Instead, the local errors introduced by time delays are characterized by the differences in the time delays of any data communication paths (DCPs) s and t that converge at node i [see Fig. 2(a)]

$$Err_i^{(Delay)} \sim \sum_{\forall s,t \in DCP_i} \left| T_s^{Delay} - T_t^{Delay} \right|. \quad (9)$$

DCP_i is the set of all communication paths converging at node i.

3) Path-Induced Errors: These errors are introduced by the missed data correlations due to the sampled data being sent using different DCPs. The error is estimated as follows:

$$Err_{b,path} = \sum_{\forall b} \sum_{\forall p}^{NL} \sum_{c \in p} error_{(p)}. \quad (10)$$

b is a pair of neighboring nodes i and k that belong to different paths. $error_{(p)}$ is the error introduced due to the missing tuples c at all lumping levels p of node k (assuming that data is transmitted from node i to node k).

The path configuration used for data collection eliminates the coupling gradient coefficients for some of the nodes in the network. Fig. 6 shows a network of transfer and storage elements modeling state variable Y expressed by (1) (the resistive elements represent parameters Gr and the capacitive elements correspond to parameters S). The Y values sampled along column $j-1$ are communicated using DCP_1, while the values sampled along column j are sent using DCP_2. As the values sent using DCP_2 are not available to the nodes on DCP_1, any partial models built during data communication do not include the correlations between the neighboring state variables, e.g., values $Y_{i,j-1}$ and $Y_{i,j}$. Hence, any decision made using the partial models does not consider the coupling gradients between columns j and $j-1$. For example, if there is a large energy source at node $Y_{i+1,j}$, without the correlation information, the partial models would suggest that there is an energy source at node $Y_{i+1,j-1}$, therefore some action (e.g., cooling) should be targeted at that point. As shown in Fig. 6, correlation errors are modeled by disconnecting columns j and $j-1$ in the network model.

Lemma 2: The expression for path-induced error at lumping level 2 in Fig. 7 is given by the following expression:

$$Err_{i,corr} = \frac{1}{\gamma\ G_{r_{a,i}}} \left[expr_{x,corr\ 1} - expr_{x,corr\ 2} \right] \quad (11)$$

$$expr_{i,corr\ 1} = \gamma\ S_i^{(equiv)}\ G_{r_{a,i}}\ \dot{Y}_i(t) + (\gamma + 1)\ Y_i(t) \quad (12)$$

$$expr_{i,corr\ 2} = \gamma\ Y_A(t) + Y_B(t) + \gamma\ G_{r_{a,i}}\ E_i^{(equiv)}(t). \quad (13)$$

γ is the ratio of the coefficients Gr from neighboring nodes.

Proof: After decoupling the network along the nodes on the communication path, the equivalent coefficient parameters G_i^{equiv} and S_i^{equiv} are measured from the decoupled node to the actuator. The error associated with decoupling the columns $j-1$ and j is proportional to the flux through the removed gradient variables, which is computed by applying Kirchhoff's law at node i. The derivation leads to (11), where $Y_A(t)$ and $Y_B(t)$ are state variables at nodes A and B, respectively, $\dot{Y}_i(t)$ is first order derivative at node i, and $E_i^{(equiv)}(t)$ is the external input energy source incident at the node. ■

Lemma 3: The path-induced error at state variable Y_i over W samples is bounded by the following expression, where $bound_{W,corr} = Err_{x,corr}(t) - Err_{x,corr}(t - W\ \delta)$:

$$bound_{W,corr} \leq \frac{W\ \delta}{\gamma\ G_{r_{a,i}}} \left[bound_{W,corr\ 1} - bound_{W,corr\ 2} \right] \quad (14)$$

$$bound_{W,corr\ 1} = \gamma\ S_i^{(equiv)}\ G_{r_{a,i}}\ \ddot{Y}_{i,\mathrm{MAX}} \quad (15)$$

$$bound_{W,corr\ 2} = \gamma \dot{Y}_{A-i_{\mathrm{MIN}}} + \dot{Y}_{B-i_{\mathrm{MIN}}} + \gamma G_{r_{a,i}} \dot{E}(equiv)_{i,\mathrm{MIN}}. \quad (16)$$

$\ddot{Y}_i^{\mathrm{MAX}}$ is the maximum of the second derivative of the state variable Y, $\dot{Y}_{A-i_{\mathrm{MIN}}}$ and $\dot{Y}_{B-i_{\mathrm{MIN}}}$ are the minimum first order derivative of the difference in sensor readings at i with respect to nodes A and B. These values are estimated using profiling.

4) Errors Due to State Variable Lumping: These errors are generated by removing state-variables during lumping. The bottom part of Fig. 7 shows the variables sampled at five nodes Y_A, Y'_X, Y_i, Y''_X, and Y_B and the corresponding model for (1), including transfer elements (G_r), storage elements (S), and energy injection elements (E). During lumping, at level 1, variable Y'_X is removed and the equivalent parameters $Gr_{A,i}$, $S_i^{(equiv)}$, and $E_i^{(equiv)}$ are computed and connected to node Y_i as shown in the figure. Similarly, at level 2, the procedure is repeated to remove variable Y''_X. The model on top approximates the original model shown at the bottom of the figure. Assuming that data is sent from node Y_A to node Y_B, the described lumping procedure is performed at node Y_B.

Lemma 4: The lumping error at node i is expressed as follows:

$$Err_i^{(Lump)} = \kappa \left| \gamma Y_A(t) + Y_B(t) - (\gamma+1) Y_i(t-\delta) - (\gamma+1)\frac{\dot{E}^{(equiv)}(t)}{S_i} \right| \quad (17)$$

where $\kappa = \gamma S_i Gr_{AB}/((\gamma+1)^2\delta + S_i Gr_{AB})$. $E^{(equiv)}$ models the equivalent external input energy. δ is the time discretizing step. γ is the ratio of $Gr_{i,B}$ and $Gr_{A,i}$.

Proof: The expression for lumping error at a node i is derived by subtracting the expressions for the discretized ODEs of $Y_i(t)$ before lumping from the expression after lumping ($Err_i^{(Lump)} = |\Delta Y_i(t)|$). ■

The following bounds exist for the lumping errors in (17).

Lemma 5: The maximum increase in lumping errors for lumping state variable Y over M samples is as follows:

$$Err^{(Lump)} \leq \kappa M \delta \left(bound_1^{(Lump)} - bound_2^{(Lump)} \right) \quad (18)$$

where, $bound_1^{(Lump)} = (\gamma+1)\delta \ddot{Y}_i^{\mathrm{MAX}} + \gamma \dot{Y}_{A-i}^{\mathrm{MAX}} + \dot{Y}_{B-i}^{\mathrm{MAX}}$ and $bound_2^{(Lump)} = (((\gamma+1)\delta)/S_i)\dot{E}_i^{(equiv),\mathrm{MIN}}$.

$\ddot{Y}_i^{\mathrm{MAX}}$ is the maximum of the second derivative of state variable Y, $\dot{Y}_{A-i}^{\mathrm{MAX}}$ and $\dot{Y}_{B-i}^{\mathrm{MAX}}$ are the maximum first order derivative of the difference in sensor readings at i with respect to nodes A and B. ($Y_{A-i}(t) = Y_A(t) - Y_i(t)$, and $Y_{B-i}(t) = Y_B(t) - Y_i(t)$) $bound_M^{(Lump)} = error_{Y_i}(t) - error_{Y_i}(t - M\delta)$. The values for $\dot{Y}^{\mathrm{MAX}}$, $\ddot{Y}^{\mathrm{MAX}}$, and $\dot{E}_i^{equiv,\mathrm{MIN}}$ are found through profiling.

The computing of the error bounds due to buffer loss, path-induced errors, and lumping errors assumed that the dynamics of state-variable Y in (1) can be expressed with a reasonable accuracy using the first three terms of their Taylor series expansion. Taylor series expansion has been a popular method to approximate systems with weak nonlinearities, like analog circuits [22] and mechanical systems [30]. Other popular techniques for describing parameter dynamics include Volterra series [23] and nonlinear model order reduction [21]. Fig. 12 plots the lumping and correlation error bounds for four different scenarios presented in Section IV-B. The bounds change in time (e.g., iteration) corresponding to the repositioning of energy sources, i.e., moving heat sources in the case of data sets 2 and 4. Moreover, Fig. 13 shows that there is a small error between the actual thermal data from the 3-D ICE simulator [26] and the data map constructed using the method based on the proposed approximation model of the bounds. However, if the modeled parameters have stronger nonlinearities then methods based on Volterra series can be used to compute the error bounds.

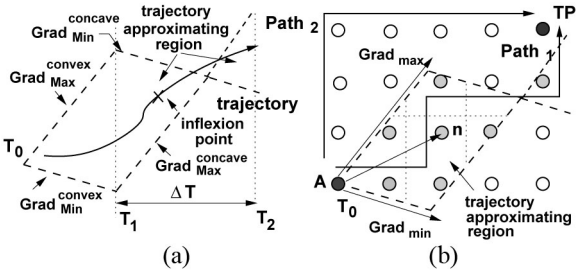


Fig. 8. (a) Trajectory description using bounded trajectory model. (b) Discretization of trajectory approximating region.

B. Mobile Energy Source/Sink Trajectory Prediction

As explained in Section II, model errors also occur due to the correlations between the trajectories of mobile energy sources (sinks) and the data communication paths (DCPs). Trajectory prediction algorithms help in reducing errors by minimizing data loss and delays of data communications through optimized selection of data communication paths (DCPs). In addition to our previous method [28], this paper presents a new algorithm for predicting the trajectory of energy sources (sinks). The two algorithms differ depending on their assumptions on the trajectory characteristics. The first algorithm is for trajectories that pass stochastically through a set of bounded regions. This algorithm was also presented in [28]. The second method assumes trajectories with both quasi-static parts and parts in bounded regions.

1) Stochastically Bounded Trajectories: The algorithm uses the concepts of bounded trajectory and stochastically-bounded trajectory. As explained in [28], a bounded trajectory refers to a mobile energy source's physical trajectory that is located inside a region defined by minimum and maximum gradients. The region is called trajectory approximating region (TAR). Then, the trajectory is approximated as sequences of convex–concave fragments with bounded gradients of known ranges. The point separating each successive convex–concave fragments is called inflexion point. The average time distance ΔT between inflexion points is known.

Example 3: Fig. 8(a) illustrates a trajectory expressed in this way. The gradient of the convex fragment is in range [$Grad_{\mathrm{Min}}^{convex}$, $Grad_{\mathrm{Max}}^{convex}$]. At time T_1, there is a break of the two dashed lines as the trajectory switches to the concave part. The gradients of this part are in range [$Grad_{\mathrm{Min}}^{concave}$, $Grad_{\mathrm{Max}}^{concave}$]. The trajectory in the figure is well approximated by the corresponding TAR.

Probability p_n that node n, inside TAR, samples the agent's trajectory is estimated as follows. Let us consider a discretization of TAR into sub-regions, as shown with dotted line in Fig. 8(b). Probability p_n depends of the unknown length $l_{trajectory}$ of the trajectory inside the sub-region containing the node, the area $Area_{sub-region}$ of the sub-region, and the number $N_{alt\ traj}$ of alternative trajectories that are inside the

sub-region and pass through the node [28]

$$p_n \propto \frac{l_{trajectory}}{Area_{sub-region}}\ N_{alt\ traj} \approx \frac{\int_{\Delta T}\sqrt{x(t)^2+y(t)^2}dt}{Area_{sub-region}}\ N_{alt\ traj} \tag{19}$$

or

$$p_n \propto \frac{\int_{\Delta T}\sqrt{x_0^2+y_0^2+2(Grad_x+Grad_y)t}\ dt}{Area_{sub-region}}\ N_{alt\ traj}. \tag{20}$$

$x(t)$ and $y(t)$ are the unknown equations describing the agent's trajectory. x_0 and y_0 are the coordinates of the agent at time T_0, and $Grad_x$ and $Grad_y$ are the gradients at time T_0 of the trajectory along the two axes. The number of alternative trajectories $N_{alt\ traj}$ can be estimated based on: 1) the angle defined by vector $\overrightarrow{(A,n)}$ from node A (the node currently considered) to node n and the vector corresponding to $Grad_{\min}$ and 2) the angle defined by vector $\overrightarrow{(A,n)}$ and the vector corresponding to $Grad_{\max}$. The larger the product of the two angles, the higher is $N_{alt\ traj}$

$$N_{alt\ traj} \propto \widehat{\overrightarrow{(A,n)}, \overrightarrow{Grad_{\min}}} \times \widehat{\overrightarrow{(A,n)}, \overrightarrow{Grad_{\max}}}. \tag{21}$$

Two approximations were introduced for expressions (20) and (21) to reduce the effort of computing probability p_n by the sensing nodes. First, we considered that all nodes of a TAR are equally likely to be part of the trajectory. Therefore, expression (20) has the same value for all nodes, and is computed using the area value for the entire TAR without any discretization. Then, N_{alt} is the same for all nodes and can be eliminated from the expression. Second, we assumed that an agent's trajectory can have any gradient inside TAR, hence, in the worst case, it coincides with the data communication path (DCP) inside the region. Then, probability p_n can be estimated as follows:

$$p_n \propto \frac{1}{length_{n\in DP_i}}. \tag{22}$$

DP_i is the DCP containing node n. $length_{n\in DP_i}$ is the length of DP_i inside TAR.

Example 4: In Fig. 8(b), DCP $Path_2$ has no nodes inside the regions, hence the probability of its nodes sampling the trajectory is lesser than for the nodes of $Path_1$.

Counting the number of path nodes inside a TAR requires a low computing effort. For windows of size one, expression (20) is proportional to the cosine of the angle between the trajectory and the DCP at node A.

Moreover, stochastically-bounded trajectory are bounded trajectories in which the change to another fragment follows a stochastic rule. Fig. 9(b) shows a trajectory that is described by four fragments, each being characterized by specific gradient ranges and average time ΔT of switching to another fragment. The switching between different fragments is modeled as a Markovian process. Let us denote x_i the steady-state probability of fragment i, and $t_{i,j}$ the transition rates between the fragments. The values of $t_{i,j}$ are found through observing

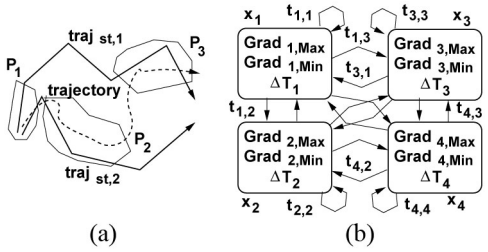


Fig. 9. (a) Trajectory description using stochastically bounded model. (b) Trajectory described by four fragments.

various trajectories through the same region. The steady-state probabilities can be computed by solving the following equations:

$$t_{i,i}x_i - \sum_{j\neq i} t_{j,i}x_j = 0 \tag{23}$$

and

$$\sum_{\forall k} x_k = 1. \tag{24}$$

Using the same reasoning as for bounded trajectories, probability p_n in expression (20) is updated as follows:

$$p_n \propto x_i\ \frac{\int_{\Delta T}\sqrt{x_0^2+y_0^2+2(Grad_x+Grad_y)t}\ dt}{Area_{sub-region}}\ N_{alt\ traj} \tag{25}$$

or the approximation in (22) is changed to expression

$$p_n \propto x_i\ \frac{1}{length_{n\in DP_i}}. \tag{26}$$

2) Adaptive Trajectory Prediction: The second method adapts its prediction based on the current state and the rate of change of trajectory parameters. If the rate of change of velocity or angle changes beyond a range, the trajectory has entered another state. For example, a higher rate of change of angle causes the trajectory to loop. A lower rate of change of angle causes an almost linear trajectory.

Using the above definition of a trajectory state, the prediction method implements the following steps. If the rate of change of angle and velocity over time has been constant or within a small range (e.g., to support noise margins), the trajectory is assumed to remain in the same state. A new state is defined by changes of velocity ΔV and angle $\Delta\theta$ over time. As shown in (27), the probability p_{state} for which a trajectory remains in the same state can be computed based on the time for which it has been in a given state. That depends on the number $n_{traj_{state}}$ of previous events for which the trajectory assumed the same state and the number of deviations of current values of velocity $n_{\delta(\Delta V)}$ and angle $n_{\delta(\Delta\theta)}$ with respect to the current state. Hence

$$p_{state} \propto \frac{n_{traj_{state}}}{n_{\delta(\Delta V)} + n_{\delta(\Delta\theta)}}. \tag{27}$$

For higher probabilities, predictions correspond to longer future trajectory segments, and define time $t_{predict}$ over which the trajectory can be predicted accurately

$$t_{predict} \propto p_{state}. \tag{28}$$

```
make_decision(Err_total, W, λ_j, β_m , Path_k ){
 while (true) {
   if (equation (38) is true) {
     select lumping at level j with rate λ_j;
     update total error up to current time;
   }
   estimate mobile source trajectory using
     method in Section III.B;
   for(DCP which overlap less with the predicted trajectory){
     select DCP k with rate Path_k ;
     select communication BW_m with rate β_m ;
   }
 }
}
```

Fig. 10. Local decision making routine.

$t_{predict}$ depends on the current state of the trajectory e.g., rate of change of velocity and angle, and the current absolute values of angle and velocity.

C. Computing the Parameters of Decision Making Policies

The procedure computes the amount of sensing and lumping at every sensing node and the parameters of the networked sensing platform, so that a cost function including the overall error and propagation delay is minimized. The computed parameters for the networked sensing platform include the path utilization rates and bandwidths of the data communication paths (DCPs), and are used to set-up the local decision making schemes of the nodes (Fig. 10). An earlier version of the method was presented in [8].

Table I summarizes all the parameters of the LP description, including their symbols and description. Also for each symbol we indicated the way in which it is found: (P) indicates that the parameter is found during profiling, (C) means that the parameter is computed by the local decision scheme (Section III-D), (R) means that the parameter has a known value (e.g., performance requirement or resource size), and (O) indicates that the parameter is computed by solving the LP formulation of the optimization scheme. Otherwise, the parameters are part of the mathematical analysis used to compute error bounds or trajectory prediction.

The details of the LP formulation are presented next.

1) Cost Function: The LP equations are solved along with the cost function to compute the utilization rates: α for discarding sensed values (8), λ for different lumping levels, β for bandwidth values, and *Path* for DCPs

$$\min \sum^{\forall p \in S} \left\{ \zeta Err_p^{(Loss)} + \xi Err_p^{(Delay)} + \eta Err_p^{(Corr)} + \theta Err_p^{(Lump)} + \mu Delay_{av_p} \right\} \quad (29)$$

where, S is the set of all DCPs. Parameters ζ, ξ, θ, μ, and η are user-defined weights that reflect different importance assigned to the various modeling error types and delay.

The following equations describe the LP optimization model. The equations are specific to data communication path DCP_j with different path segments ($p_x \in DCP_j$).

2) Errors Due to Data Loss ($Err^{(Loss)}$): Data loss due to buffer overwriting ($Err^{(Buff)}$) occurs at node x when the input rate is higher than the output rate and the difference is more than the available buffer size *buff*. The data loss is equal to

$$Loss_x = \left| \left(In\_rate_x + NET_x^{IN} \right) - NET_x^{OUT} - buff \right|. \quad (30)$$

buff is the local buffer size. The average sensing rate of node x (In_rate_x) is equal to the required sampling resolution. NET_x^{IN} is the amount of data input at node x from other nodes in the path. The amount of data output from node x is NET_x^{OUT}

$$NET_x^{IN} = \sum_{j \in Pred} NET_j^{OUT}. \quad (31)$$

Pred is the set of all nodes that precede immediately node x on the used data communication paths (DCPs).

NET_x^{OUT} depends on the amount of lumping at node x and all previous nodes pv in path segment p_x of the DCPs that lead to node x

$$NET_x^{OUT} = prob_x^{(Lump)} DATA_x^{OUT} + \sum_{\forall pv \in p_x} prob_{pv}^{(Lump)} DATA_{pv}^{OUT}. \quad (32)$$

$DATA_x^{OUT}$ is the size of the packet sent out by node x for the locally lumped data model.

The error due to buffer loss $Err^{(Buff)}$ is expressed using (7) in which n is replaced by variable $Loss_x$. Collection errors ($E^{(Coll)}$) are described as in (8). Equation (6) describes the total error $E^{(Loss)}$ due to data loss at node x.

3) Path Delay ($Delay_{av}$) and Errors Due to Time Delays ($Err^{(Delay)}$): The delay of path p_x is the sum of the average execution time of all nodes x along the path plus the average time for transmitting the output data of each node

$$Delay_{p_x} = \sum_{\forall x \in p_x} \left(Exec_x + Delay_x^{out} \right). \quad (33)$$

$Exec_x$ is the execution time of the primitives at node x.

The average time for transmitting the output data NET_x^{OUT} of node x is expressed as follows:

$$Delay_x^{out} = NET_{x,\mathrm{MAX}}^{OUT} \sum_{\forall j} \beta_j \frac{1}{BW_j}. \quad (34)$$

β_j is the rate of using bandwidth BW_j for the output link and $\sum_{\forall j} \beta_j = 1$. $NET_{x,\mathrm{MAX}}^{OUT}$ is the maximum NET_x^{OUT} at node x found during profiling. The worst-case value for NET^{OUT} had to be considered in order to keep the model linear.

The average delay ($Delay_{av}$) is the average of the delays $Delay_{p_x}$ for all paths p_x used for communicating the sensed data to the data construction site.

The error due to time delay $Err^{(Delay)}$ is described using expression (9) in which T^{Delay} is replaced by variables $Delay_{p_x}$.

4) Correlation Error ($Err^{(Corr)}$): The correlation error is given by

$$Err^{(Corr)} = \sum_{\forall i \in DCP} Path_i \; Err_i^{(Corr)} \quad (35)$$

where $Err_x^{(Corr)}$ are computed using (14).

5) Lumping Error ($Err^{(Lump)}$): The lumping error is given by

$$Err^{(Lump)} = \sum_{\forall x \in DCP} \left(prob_x^{(Lump)} Err_x^{(Lump)} \right) \quad (36)$$

where $Err_x^{(Lump)}$ are computed using (18). $prob_x^{(Lump)}$ is the probability that node x is lumped, and is given by the following equation:

$$prob_x^{(Lump)} = \sum_{\forall j} \lambda_j lump\_level_j \tag{37}$$

where $lump\_level_j$ corresponds to the lumping level, which is defined by the percentage of lumped nodes in p_x. λ_j is to the utilization rates of using $lump\_level_j$. $\sum_{\forall j} \lambda_j = 1$.

D. Local Decision Making Routine

The local decision making routine at a node is shown in Fig. 10. It uses as input parameters the computed values for the rates $Path_k$ of using DCP k, β_m of using different bandwidths BW_m for DCP, rate λ_k of employing lumping hierarchy level k at a node i, and the estimated minimum error Err_{total}. Equation (38) is used to decide locally, if node i performs lumping or not. This decision is based on the following lemma.

Lemma 6: Node i performs variable lumping at current time $T^{(curr)}$, if the following constraint is met:

$$\frac{Err_i^{(prev)} + Err_i^{(k)}}{\mathbf{E}[Err_i]} < \frac{W - T^{(curr)}}{W}. \tag{38}$$

Otherwise, there is no lumping at time $T^{(curr)}$. $\mathbf{E}[Err_i]$ is an estimation of the minimum total error at sensing node i over time window W. The estimate is computed by the optimization in Section III-C. $Err_i^{(prev)}$ is the error due to previous lumping. $Err_i^{(curr)}$ is the error introduced by the current state lumping.

Proof: $\mathbf{E}[Err_i]$ represents the lower bound of the error that can be achieved by the sensing node (according to the optimization in Section III-C). Assuming a uniform distribution of the lumping error, the total error in the remaining time of the current time window W (i.e., $W - T^{curr}$) is $\mathbf{E}[Err_i]((W - T^{curr})/W)$. Hence, the decision making procedure should select to lump or not, so that the difference between the bound $\mathbf{E}[Err_i]$ and the sum $Err^{(prev)} + Err_i(k) + \mathbf{E}[Err_i]((W - T^{curr})/W)$ is minimized. This proves the lemma.

Conceptually, the local decision making procedure at node i tracks the lower bound $\mathbf{E}[Err_i]$.

Next the decision making routine estimates the trajectory of mobile energy sources or sinks using one of the algorithms presented in Section III-B. The DCPs with least overlapping with the trajectories are selected as they reduce the data loss along the DCPs (see Section II-A and Fig. 3). From the selected DCPs, the actual DCP k used in communication is chosen based on the computed rates $Path_k$. The bandwidth of the chosen DCP is set to value BW_m based on the rates β_m.

The LP formulation is solved offline, as shown in Fig. 5, while trajectory prediction and lumping level selection are executed online. The energy consumption and time delay due to these computations is small compared to the cost associated with data communication. Apart from generating and sending its own data packets, each node receives data from other nodes, and forwards it toward the target point along the selected DCP. Due to state-variable lumping, data traffic and communication are reduced leading also to reduction in average energy costs. In a sampling-processing-communication iteration for unoptimized case, up to 70.84% of the time is spent on communication by a node in a network of size 25. In the optimized case, the average reduction in traffic compared to unoptimized case up to 25.69%. ■

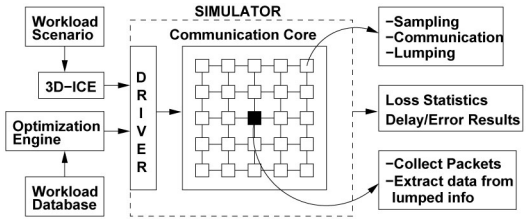


Fig. 11. Experimental framework.

IV. Experiments and Results

Experiments were performed to verify the accuracy and efficiency of the discussed modeling scheme. An overview of the simulation framework is shown in Fig. 11. A temperature-sensing network was modeled in SystemC to describe accurately the timing, resolution, bandwidth, and local memory of reconfigurable mixed-signal PSoC processor [1]. PSoC processor includes an 8-bit processor operating at 24 MHz, 2k SRAM and 64k EPROM on-chip memory, programmable clocks, and serial data communication (SPI and UART) with programmable bandwidths. This Simulator has a communication core, which consists of a grid network of embedded nodes that perform various tasks, such as sampling, communication, and generation of local models. One of the nodes in the network, is designated as the target point (TP), and all other nodes are connected to this TP via predefined path configurations. The TP receives data packets, extracts temperature information from model parameters, and builds the resultant thermal map. Loss, delay, and error statistics are then extracted from the information received by the TP.

A. Trajectory Prediction and Optimization

The two trajectory prediction algorithms were studied using six trajectories defined such that they cover the entire data communication network. The trajectories are presented in [28], and cover different trajectory directions, velocity values, angle ranges, and rates of change of velocity and angle. The data communication paths (DCPs) were defined such that there are three unique path configurations for each of the four target points. Hence, the experiments used a total of twelve path configurations.

Both algorithms, stochastically bounded trajectories method (A1) and adaptive method (A2), offer an improvement in data loss as compared to the best-case static path configurations, in which all nodes in the network are static. For the adaptive algorithm (A2), in most cases, the data loss is zero and it is close to zero for A1. The maximum data loss was four samples for A1, two samples for A2, and 33 out of 74 events for static path configuration. Both algorithms provide improvements in average delay as compared to static path configurations. Improvements are as high as 38% for A1 and 40% for A2, but there are instances in which the resulting delay is higher for the two methods as compared to static paths.

TABLE II
SUMMARY OF THE TWO TRAJECTORY PREDICTION ALGORITHMS

Trajectory	Events	Average Data Loss Improv.		Average Delay Improv.(%)	
		A1	A2	A1	A2
1	20	0.75	0.75	-8.50	26.00
2	21	1.00	1.00	0.75	20.00
3	25	1.00	1.00	2.50	21.75
4	30	4.00	4.50	3.75	18.50
5	26	1.25	1.25	-8.00	18.75
6	74	6.25	7.75	-16.75	12.50

Table II summarizes the improvement and percentage improvement in average data loss and average delay, respectively, for the six trajectories compared to the best-case static path configurations. Trajectories 4 and 6 are the longest trajectories in terms of number of events generated. For both trajectories, algorithms A1 and A2 produce significant reduction in data loss, but only method A2 results in reduced delay. For trajectories 1, 2, 3, and 5, both algorithms achieve small improvement in data loss over the best-case static configuration results. Algorithm A2 causes reduction in average delay for all these trajectories, while A1 manages to do that only for trajectories 2 and 3. The results suggest that algorithm *A*2 is superior to algorithm *A*1.

B. Data Modeling

1) Experiment Set-Up: The improvements in data loss, modeling error, and latency were recorded and compared with the unoptimized case. The scalability of the methods was tested by running simulation using three network sizes: 25, 64, and 100 nodes. Four different data communication path (DCP) configurations between nodes were used for each of the three network sizes.

The simulator selects an appropriate configuration for each iteration based on results of the optimization engine. For the unoptimized case, the path utilization rate was set as 25% for each path configuration. So, each configuration had an equal probability of being selected. The output of the optimization engine was also used to select the lumping thresholds and the communication bandwidth. For the unoptimized case (used as a reference), the bandwidth utilization rate was fixed as 33.33% for each bandwidth value. For the optimized case, the results of the minimum error estimation step were used to set probabilities β_j of the three bandwidth values.

In the optimization engine, weights θ, μ, and η in the cost function were set to reflect different importance assigned to modeling error, delay, and correlation error: OptRatio1 ($\theta/\mu/\eta$) = 0.6/0.2/0.2, OptRatio2 = 0.2/0.6/0.2, and OptRatio3 = 0.2/0.2/0.6. Depending on which ratio is being used, the probability of using a particular communication rate changes. Weights ζ and ξ were set to 0. For example, OptRatio2 assigns more importance to reducing latency, hence increasing the probability of using the fastest communication rate.

Thermal modeling was used a case study for distributed data modeling, even though the technique can be also used for other data modeling situations. The thermal data was generated for the ULTRASPARC Niagara T1 architecture [14]. Experiments were performed using eight datasets, which represent different workload scenarios resulting in stationary, fluctuating, and moving hotspots. The temperature behavior was found for the datasets using the temperature simulator 3-D ICE [26].

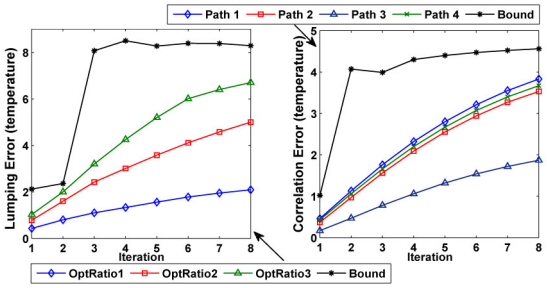


Fig. 12. (Left) Bounds on lumping. (Right) Correlation errors.

2) Model Validation: The bounds for lumping error and correlation error were estimated using (18) and (14), respectively. The thermal data was generated through profiling of various workload scenarios. The bounds were tested for each of the datasets used in the experiments for all DCPs and optimization ratios (OptRatios). The value of correlation error depends on the selected communication path and the dataset. While the magnitude of the bound depends on the dataset, the shape of the curve (over time) depends on the configuration of the DCP. Among all the datasets, Dataset 6 has the highest average correlation error. Fig. 12(right) shows the correlation error for Dataset 6 for all DCPs along with the bounds. The bounds were recomputed at every iteration. Low values of the bounds correspond to state variables with small values, while large bounds reflect large state variables. The slope of the bound plots describes the rate of change of the state variables. Path 1 has the highest error compared to the other DCPs because it has longest path segments. The two main factors that affect the value of lumping error are dataset and OptRatio. Dataset 3 has the highest lumping error compared to other datasets. The lumping error for all OptRatios for Dataset 3 along with the bounds is shown in Fig. 12(left). The shape of the curve for the bound depends on the lumping level while the magnitude depends on the dataset. The error is highest for OptRatio3 since it performs the highest amount of lumping. Note that if the number of iterations increases, the bounds would adjust accordingly by moving up. Hence, the actual errors values would not exceed the bounds.

The validation of *S* and *Gr* coefficients was performed for the unoptimized case as well as optimized case for all OptRatios and all DCPs using a large data set for simulated workloads. First, the sensed temperature data is converted to equivalent storage (*S*) and gradient coefficients (*Gr*) at each node. At the target point, the original thermal information is extracted back from these coefficients. For example, for Dataset 8, the average error associated with these conversions is 2.5% for the un-optimized case and 3.0% for the optimized case. Fig. 13 shows a comparison between the actual thermal map sensed by the sensor nodes versus the thermal map extracted from the *S* and *Gr* coefficients. The two maps are very similar.

3) Results and Analysis: The results for Dataset 2 (two moving heat sources) and Dataset 6 (six stationary heat sources) for different network sizes are detailed. Then an overview of results for all datasets and network sizes is summarized.

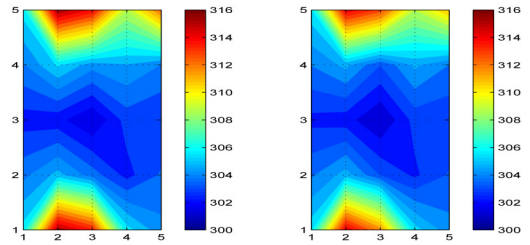


Fig. 13. (Left) Complete thermal map. (Right) Thermal map constructed using the data model.

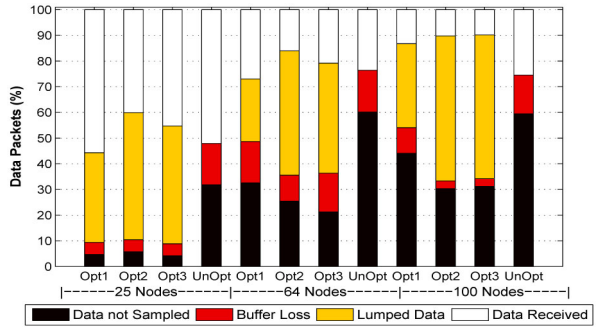


Fig. 14. Dataset 2: data loss for different network sizes.

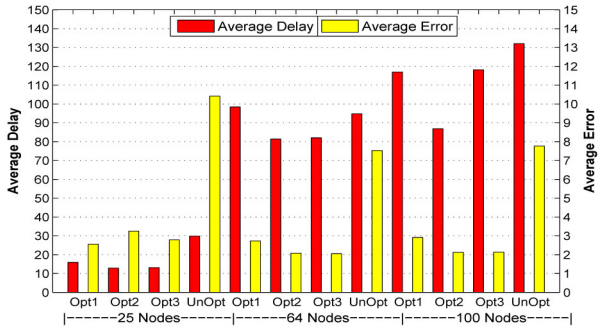


Fig. 15. Dataset 2: delay and error for different network sizes.

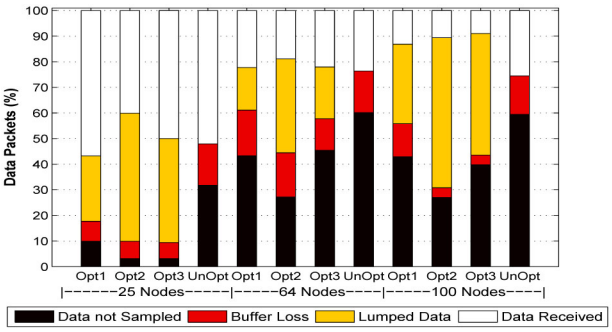


Fig. 16. Dataset 6: data loss for different network sizes.

The following parameters were extracted from the simulation results.

1) *Data not sampled* gives the number of instances when the node failed to sense the temperature data because it was busy forwarding packets from previous nodes on the path.
2) *Buffer loss* is the loss of data because the packet got overwritten in the buffer due to incoming packets.
3) *Lumped data* gives the loss of packets because the node was lumped.
4) *Data received* is the number of packets received at the target point.
5) *Avg. delay* is the average delay associated with packets that were received at the model construction node.
6) *Avg. error* is the average value of lumping error for the packets that could not reach the model construction node.

4) Dataset 2: The experimental results for Dataset 2 are as shown in Figs. 14 and 15. The x-axis in Fig. 14 gives the network size. The y-axis shows the number of packets in terms of percentages. The data not sampled and buffer loss represent undesirable losses as they cause uncontrollable errors for the data models. Lumped data represents controllable loss as its impact is factored in the final error of the data models.

For the 25 node network, the unoptimized case has the highest collection loss and buffer loss as compared to any of the optimized cases. Also, the delay and error is much larger for the unoptimized case. The amount of lumping increases if the cost function assigns higher priority on reducing latency and lower importance to modeling error and correlation error. The increase in lumped data leads to lower communication traffic but causes a higher modeling error. Also, faster bandwidth rates are used. The average delay increases as the network size increases. Since a larger volume of packets is generated in the larger networks, but the buffer size of the nodes remains fixed, there is higher collection loss and buffer loss. The values of average error are comparable only because there is higher loss leading to fewer packets reaching the TP.

A couple of anomalies were noted in the results for network sizes 64 and 100 nodes. Firstly, the requirement to increase lumping to improve communication (at the penalty of higher error) was less than if the objective was to reduce error by allowing less local lumping. Less lumping combined with slower communication rates causes increase in delay and losses due to buffer overflow. Therefore, fewer packets reach the TP, leading to increase in error. Secondly, especially for 64 node network, the delay for unoptimized case is less than the delay for the case that minimizes lumping. This is because delay can only be estimated for packets that actually reach the TP. However, since very few packets actually make it to the TP in the unoptimized case (hence the large error) and since most of these packets are from nodes close to the TP, the average delay is minimized. Therefore, it is important to combine the loss and the delay/error statistics to get a more accurate understanding of the results.

5) Dataset 6: The experimental results for Dataset 6 are as shown in Figs. 16 and 17. Most of the observations noted for the results for Dataset 2 are also valid for Dataset 6. An important difference is in the amount of lumped data as there is less lumping for Dataset 6. The basic purpose of lumping is to reduce communication traffic by lumping some of the less important data. Since Dataset 6 has more thermal activity (hotspots), there are fewer regions where data can be lumped. This causes an increase in communication traffic leading to higher delays and buffer loss.

6) Summary of Results: Table III summarizes the percentage improvement in loss, model error, and delay for the

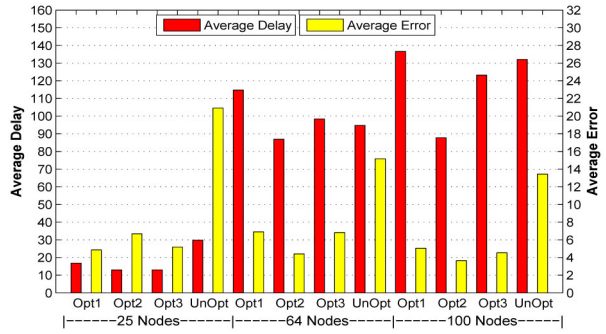


Fig. 17. Dataset 6: delay and error for different network sizes.

TABLE III
SUMMARY OF RESULTS

Net Size	Opt Ratio	% Improvement vs UnOpt Case							
		Coll Loss	TrPr Coll Loss	Buff Loss	TrPr Buff Loss	Error	TrPr Error	Delay	TrPr Delay
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
25	1	79.10	9.43	62.50	14.52	76.91	1.05	45.69	2.16
	2	83.61	6.35	67.34	10.48	68.21	1.89	57.62	1.54
	3	84.43	13.93	69.35	18.95	72.67	2.82	56.57	0.80
64	1	36.26	14.27	-14.02	4.54	58.29	9.35	-11.49	0.81
	2	57.96	19.27	16.31	4.88	73.49	5.21	5.83	7.04
	3	44.80	20.30	13.11	9.97	65.97	9.74	-2.14	6.96
100	1	25.72	8.07	24.89	5.04	62.66	3.47	0.40	1.24
	2	50.50	14.97	76.16	1.37	72.99	3.56	34.95	3.60
	3	37.02	12.37	76.79	0.63	68.68	3.91	9.25	1.29

optimized cases versus the unoptimized case. Columns 4, 6, 8, and 10 in the table present the amount of performance improvement due to trajectory prediction (TrPr) using algorithm A2. The results are averaged over the eight datasets. The collection loss (data not sampled) is reduced by up to 84.43% for 25 nodes, up to 57.86% for 64 nodes, and up to 50.50% for the 100 node network. Similarly, for buffer loss, there is significant reduction of up to 67.34% for 25 nodes and up to 76.16% for 100 node network. The improvement is smaller for 64 nodes up to 16.31%. This is because, for the unoptimized case, more than 60% of the data is lost as collection loss. So, there is already less traffic in the network leading to less buffer loss. This causes the misperception that the buffer loss is comparable for the optimized and unoptimized cases. It is also interesting to note that the percentage improvement for both, collection and buffer loss, is higher for OptRatios 2 and 3 as compared to OptRatio 1. The reduction in overall error is up to 76.91% for 25 nodes, between 58.29% to 73.49% for 64 nodes and up to 72.99% for 100 nodes. The highest improvement is shown either by OptRatio 1 or 2, for any network size. There is a significant reduction in the average delay for 25 nodes (up to 57.62%), especially for OptRatio2 which tries to improve latency. For the other network sizes as well, the improvement in latency is highest for OptRatio2 but the percentage is smaller for 64 and 100 nodes. Again, the low values for 64 nodes are caused by the fact that more than 60% of the packets in the unoptimized case are not generated at all (collection loss) leading to less traffic and lower delays for the remaining packets. Hence, it is important to combine the information over the four columns to get useful insights into the results of these experiments.

For the 25 node network, trajectory prediction provides up to 13.93% reduction in collection loss and up to 18.95% reduction in buffer loss. It also contributes to a small improvement in overall error and delay. For the 64 node network, the prediction results in up to 9.74% reduction in error and up to 7.04% reduction in delay. The improvement is loss is also high, up to 20.30% for collection loss and up to 24.54% for buffer loss. Similarly for the network size of 100 nodes, there is a small contribution to reduction in loss, delay and up to 12.37% reduction in collection loss.

TABLE IV
OPTIMIZATION RESULTS FOR SINGLE VERSUS MULTIPLE BANDWIDTHS

BW	OptRatio	% Improvement vs UnOpt Case			
		Coll Loss	Buff Loss	Error	Delay
Multiple BW_j	1	79.10	62.50	76.91	45.69
	2	83.61	67.34	68.21	57.62
	3	84.43	69.35	72.67	56.57
Fixed BW_1	1	52.66	52.78	63.30	-18.76
	2	66.22	72.69	59.09	0.06
	3	61.17	66.67	61.67	-6.42
Fixed BW_3	1	87.90	52.78	79.81	67.65
	2	86.16	59.72	74.59	70.74
	3	87.23	56.94	76.68	69.89

In a platform with constant bandwidth, $j = 1$ and variable β_j is one in (34). Optimization of the selection of DCPs and lumping levels is still performed to minimize data loss and delay. Table IV compares the optimization results when three bandwidths BW_j are used versus constant bandwidth. When fixed BW_1 (slowest) is used, as expected, the improvements in loss, error, and delay are less than for the fully optimized case (multiple BW_j) for all optimization ratios. For fixed BW_3 (highest), the results for collection loss and error are similar to the fully optimized case (multiple BW_j). The improvement in delay is higher and in buffer loss is less. But, operating only at maximum bandwidth increases power consumption.

V. CONCLUSION

This paper presents a procedure to construct robust data models using samples acquired through a grid network of embedded sensing devices with limited resources, like bandwidth and buffer memory. The procedure constructs local data models by lumping state variables, and then collects centrally the local models to produce global models. The modeling procedure uses a linear programming formulation to compute the lumping level at each node, and the parameters of the networked sensing platform, like data communication paths and bandwidths. Two algorithms are described to predict the trajectories of mobile energy sources/sinks as predictions can further reduce data loss and delays during communication. The computed parameters and trajectory predictions are used to set-up the local decision making routines of the networked sampling nodes. Experiments discuss the method's efficiency for thermal modeling of ULTRASPARC Niagara T1 architecture.

Experiments show that variable lumping reduces the overall error by up to 76.91% and delay by up to 57.62%, as compared to no lumping being used. The error is smallest if latency reduction has high priority. The attempt to minimize local error performs less lumping, however, results is larger data loss, and hence in more overall error. The attempt to reduce the overall error by minimizing the correlation error results in increased latency. As the network size increases from 25 nodes to 64 and 100 nodes, the larger communication traffic leads to further losses and delays. Therefore, accuracy-centered optimization becomes critical for performing reliable

data extraction. Trajectory prediction using adaptive method (A2) reduces modeling error by up to about 10%.

REFERENCES

[1] *Cypress Semiconductor Corporation, PSoC Mixed Signal Array*, Document no. PSoC TRM 1.21, 2005.

[2] T. Banerjee, K. Chowdhury, and D. Agrawal, "Distributed data aggregation in sensor networks by regression based compression," in *Proc. Int. Conf. Mobile Adhoc Sensor Syst. (MASS)*, Washington, DC, USA, Nov. 2005.

[3] D. Basmadjian, *The Art of Modeling in Science and Engineering*. Boca Raton, FL, USA: Chapman & Hall, 1999.

[4] A. Doboli and E. Currie, *Introduction to Mixed-Signal, Embedded Design*. New York, NY, USA: Springer-Verlag, 2010.

[5] A. K. Coskun, T. S. Rosing, and K. Whisnant, "Temperature aware task scheduling in MPSoCs," in *Proc. Design Autom. Test Europe Conf. Exhibit.*, Nice, France, 2007, pp. 1–6.

[6] K. Skadron *et al.*, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 94–125, Mar. 2004.

[7] F. Zanini, M. M. Sabry, D. Atienza, and G. De Micheli, "Hierarchical thermal management policy for high-performance 3D systems with liquid cooling," *IEEE J. Emerg. Sel. Topic Circuits Syst.*, vol. 1, no. 2, pp. 88–101, Jun. 2011.

[8] C. Ferent, V. Subramanian, M. Gilberti, and A. Doboli, "Linear programming approach for performance-driven data aggregation in networks of embedded sensors," in *Proc. Design Autom. Test Eur. (DATE)*, Dresden, Germany, Mar. 2010, pp. 1456–1461.

[9] A. Fourmigue, G. Beltrame, G. Nicolescu, and E. M. Aboulhamid, "A linear-time approach for the transient thermal simulation of liquid-cooled 3D ICs," in *Proc. 9th Int. Conf. Hardware/Softw. Codesign Syst. Syn. (CODES+ISSS)*, Taipei, Taiwan, Oct. 2011, pp. 197–205.

[10] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: An efficient framework for modeling sensor network data," in *Proc. 3rd Int. Symp. Inf. Process. Sensor Netw. (IPSN)*, Berkeley, CA, USA, Apr. 2004, pp. 1–10.

[11] T. He, B. Blum, J. Stankovic, and T. Abdelzaher, "AIDA: Adaptive application-independent data aggregation in wireless sensor networks," *ACM Trans. Embedded Comput. Syst.*, vol. 3, no. 2, pp. 426–457, May 2004.

[12] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proc. 5th Annu. ACM/IEEE Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, 1999, pp. 174–185.

[13] M. Ketel, N. S. Dogan, and A. Ugur, "Distributed wireless sensor networks with power efficient data modeling," in *Proc. 37th Southeastern Symp. Syst. Theory (SSST)*, Mar. 2005, pp. 415–418.

[14] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-way multithreaded sparc processor," *IEEE Micro*, vol. 25, no. 2, pp. 21–29, Mar./Apr. 2005.

[15] P. Li, L. Pileggi, M. Asheghi, and R. Chandra, "Efficient full-chip thermal modeling and analysis," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2004, pp. 319–326.

[16] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He, "RAP: A real-time communication architecture for large-scale wireless sensor networks," in *Proc. 8th IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, 2002, pp. 55–66.

[17] H. Mizunuma, C.-L. Yang, and Y.-C. Lu, "Thermal modeling for 3D-ICs with integrated microchannel cooling," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design Dig. Tech. Papers (ICCAD)*, San Jose, CA, USA, Nov. 2009, pp. 256–263.

[18] P. Nie and B. Li, "A cluster-based data aggregation architecture in WSN for structural health monitoring," in *Proc. 7th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Istanbul, Turkey, Jul. 2011, pp. 546–552.

[19] L. Parolini, N. Tolia, B. Sinopoli, and B. Krogh, "A cyber-physical systems approach to energy management in data centers," in *Proc. 1st ACM/IEEE Int. Conf. Cyber-Phys. Syst. (ICCPS)*, New York, NY, USA, 2010, pp. 168–177.

[20] J. Ranieri, A. Vincenzi, A. Chebira, D. Atienza, and M. Vetterli, "EigenMaps: Algorithms for optimal thermal maps extraction and sensor placement on multicore processors," in *Proc. 49th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2012, pp. 636–641.

[21] M. Rewienski and J. White, "Model order reduction for nonlinear dynamical systems based on trajectory piecewise-linear approximations," *Linear Algebra Appl.*, vol. 415, nos. 2–3, pp. 426–454, Jun. 2006.

[22] R. Rutenbar, G. Gielen, and J. Roychowdhury, "Hierarchical modeling, optimization, and synthesis for system-level analog and RF design," *Proc. IEEE*, vol. 95, no. 3, pp. 640–669, Mar. 2007.

[23] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. New York, NY, USA: Wiley, 1980.

[24] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, 2009.

[25] M. Shoaib and W. Song, "Data aggregation for vehicular ad-hoc network using particle swarm optimization," in *Proc. 14th Asia-Pacific Netw. Oper. Manag. Symp. (APNOMS)*, Seoul, Korea, 2012.

[26] A. Sridhar *et al.* (2012). *3D-ICE* [Online]. Available: http://esl.epfl.ch/3D-ICE

[27] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza, "3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2010.

[28] V. Subramanian, A. Umbarkar, and A. Doboli, "Maximizing the accuracy of sound based tracking via a low-cost network of reconfigurable embedded nodes," in *Proc. Adapt. Hardware Syst. (AHS)*, San Diego, CA, USA, 2011.

[29] E. Vladislavleva, G. Smits, and D. Den Hertog, "Order of nonlinearity as a complexity measure for models generated by symbolic regression via Pareto genetic programming," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 333–349, Apr. 2009.

[30] S. C. Wong and A. Barhorst, "Polynomial interpolated Taylor series method for parameter identification of nonlinear dynamic system," *J. Comput. Nonlinear Dyn.*, vol. 3, no. 1, pp. 248–256, Mar. 2006.

[31] F. Xue, R. Subbu, and P. Bonissone, "Locally weighted fusion of multiple predictive models," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Vancouver, BC, Canada, 2006, pp. 2137–2143.

[32] K. Yuen, B. Liang, and B. Li, "A distributed framework for correlated data gathering in sensor networks," *IEEE Trans. Veh. Technol.*, vol. 57, no. 1, pp. 578–593, Jan. 2008.

[33] M. Zhu *et al.*, "Fusion of threshold rules for target detection in sensor networks," *ACM Trans. Sensor Netw.*, vol. 6, no. 2, p. 18, Feb. 2010.

Anurag Umbarkar received the B.Tech. degree in electronics and telecommunication from the College of Engineering Pune, Pune, India, in 2008, and the M.S. degree in 2010. He is currently pursuing the Ph.D. degree in computer engineering at Stony Brook University, Stony Brook, NY, USA.

His current research interests include performance-optimized detection, tracking, and modeling of physical phenomena in distributed sensing environments.

Varun Subramanian received the B.E. degree from Mumbai University, Mumbai, India, in 2007, the M.S. degree in 2009, and the Ph.D. degree in computer engineering from Stony Brook University, Stony Brook, NY, USA, in 2012.

He is currently a Corporate Applications Engineer with Carbon Design Systems. His Ph.D. research was focused on building distributed data models in a goal-oriented optimization framework for cyber-physical systems.

Alex Doboli (S'99–M'01–SM'07) received the M.S. and Ph.D. degrees in computer science from "Politehnica" University, Timisoara, Romania, in 1990 and 1997, respectively, and the Ph.D. degree in computer engineering from the University of Cincinnati, Cincinnati, OH, USA, in 2000.

He is an Associate Professor with the Department of ECE, Stony Brook University, Stony Brook, NY, USA. His current research interests include electronic design automation, including analog and mixed-signal CAD, cyber-physical systems, and methodologies for design innovation.

Dr. Doboli is a member of Sigma Xi.