A Genetic Programming Approach to Cost-Sensitive Control in Wireless Sensor Networks

Afsoon Yousefi Zowj, Josh C Bongard, and Christian Skalka

Department of Computer Science, University of Vermont
{ayousef1,jbongard,ceskalka}@uvm.edu

Abstract. In some wireless sensor network applications, multiple sensors can be used to measure the same variable, while differing in their sampling *cost*, for example in their power requirements. This raises the problem of automatically controlling heterogeneous sensor suites in wireless sensor network applications, in a manner that balances cost and accuracy of sensors. We apply genetic programming (GP) to this problem, considering two basic approaches. First, we construct a hierarchy of models, where increasing levels in the hierarchy use sensors of increasing cost. If a model that polls low cost sensors exhibits too much prediction uncertainty, the burden of prediction is automatically transferred to a higher level model using more expensive sensors. Second, we train models with cost as an optimization objective, called non-hierarchical models, that use conditionals to automatically select sensors based on both cost and accuracy. We compare these approaches in a setting where the available budget for sampling is considered to remain constant, and in a setting where the system is sensitive to a fluctuating budget, for example available battery power. We show that in both settings, for increasingly challenging datasets, hierarchical models makes predictions with equivalent accuracy yet lower cost than non-hierarchical models.

1 Introduction

Wireless Sensor Networks (WSNs) have revolutionized environmental monitoring by combining low cost with flexibility in sensor capabilities [31]. They have been used in diverse environmental monitoring applications and continue to be adapted in new fields. Because WSNs are often, even typically, deployed in remote locations, and thus rely on combinations of battery power and energy harvesting, a major challenge in WSN design is to minimize system power consumption.

Minimizing power consumption can be accomplished in a variety of ways, in particular by adapting sensor control strategies that optimize the balance between measurement accuracy and the cost of powering sensors [30]. In this paper, we propose new sensor control algorithms for WSNs with heterogeneous sensor suites that balance cost and accuracy, obtained using genetic programming (GP) techniques.

By "heterogeneous sensor suite", we mean WSNs equipped with multiple types of sensors for prediction of the same phenomena. Each of these sensors is characterized by its accuracy in relation to the phenomena, and a cost of use which is often measured by its power consumption. Such systems support multi-modal sensor fusion, a well-studied technique where data from multiple sensor modalities (types) is combined to predict a single variable [30]. The contribution of our work is a consideration of cost in multi-modal sensor fusion, and the development and testing of associated control algorithms. These algorithms will call upon particular sensors only when needed, and otherwise rely on the cheapest available sensors at any given time. Our problem is distinguished from adaptive sampling [30] in that the latter is concerned with optimally modulating sampling frequency of a given sensor, not choosing between a suite of possible sensors.

While various multi-modal sensor fusion applications exist, we are especially interested in the Snowcloud system which combines snow density telemetry with snow depth and air temperature sensors to predict areal snow water equivalent (SWE) [24]. We envision extending Snowcloud to incorporate ground based light detection and ranging (LIDAR) scanning [5] to be used for SWE estimation as part of its sensor suite. However, while LIDAR yields more accurate data than existing Snowcloud telemetry, it does so at significant additional power cost. Thus, the challenge is to commit these resources only at optimal times. It is also a refinement of multi-modal sensor fusion, since we are mainly interested in settings where available data gathering techniques differ in accuracy, with less accurate sensors being cheaper than more accurate ones.

A fundamental component of our approach is the use of prediction *uncertainty* to drive sensor usage. We propose a scheme whereby predictions are attempted using lower-cost sensors at first. If uncertainty is below an acceptable threshold, then the prediction is used. Otherwise we switch to higher-cost sensors, make a new prediction based on those inputs, evaluate uncertainty again, and continue to move the burden of prediction to more accurate and costly sensors as needed. This scheme is discussed in detail in Section 2.4 and described graphically in Figure 2. Note that while the Snowcloud system is an intended application of this scheme, it can be generalized to any WSN application using heterogeneous sensor suites comprising sensors with varying cost and accuracy.

To quantify uncertainty we are aided by machine learning ensemble methods– we use entropy in ensemble predictions as a proxy for uncertainty [23]. To obtain predictive models themselves, in this work we use genetic programming (GP) [14]. This is largely due to characteristics of our intended application space. Previous work has demonstrated that the relationships between snow cover and the topographic and meteorological factors that influence it include non-linearities [26], while the spatial distribution of SWE is nonlinear because it is influenced simultaneously by various forcing effects [27]. Nonlinear predictors are therefore desirable. Furthermore, recent results [7] show that GP has advantages over other approaches (such as decision trees) due to associated techniques for preventing overfitting, e.g. treating model size minimization as an objective [12]. Although C4.5 only supports classification, sufficiently fine classification granularity can

achieve competitive performance on regression problems, and this approach is popular in the environmental science community [7]. Finally, GP is appealing due to its white-box nature: it can potentially provide physical insights into modeled phenomena.

An alternative approach to our problem is to not rely on external measures of entropy to switch between sensors, but to treat cost as an additional objective in a multi-objective optimization problem. We explore this option in our work, in direct comparison to the hierarchical approach. However, due to the "curse of dimensionality", adding another optimization dimension may have deleterious effects on prediction performance, especially since selection for size to avoid overfitting already imposes a multi-objective optimization regime [6]. We therefore hypothesize that a hierarchical approach will outperform a non-hierarchical approach in settings with multiple sensors of differing predictive abilities, and we explore this comparison in our experiments.

In our initial comparison of these two approaches– hierarchical and non-hierarchical– our regime is not concerned with the available budget. However, in real deployments, budget levels can have significant impacts on what sensors are chosen. For example, if battery levels are low, expensive sensors should probably be avoided regardless of prediction uncertainty, both to reduce system downtime and sensor noise. Therefore, we also consider a comparison of the hierarchical and non-hierarchical approaches in a setting where models are sensitive to dynamic budget fluctuations. As for the basic setting, we hypothesize that the hierarchical approach will perform better than the non-hierarchical.

1.1 Related Work

Previous work on adaptive sampling [30] has aimed to reduce sampling rates in Resource Constrained Sensor Systems (RCSS) applications to balance sensor cost and accuracy. In particular, Alippi *et al.* [4] have tried to find the optimal adaptive frequency of sampling for avalanche monitoring. It has further been claimed that compressed sensing — sending aggregated data instead of raw data — performs better in conjunction with reducing sampling rates, rather than just reducing the sampling rate alone [17]. A variety of methods for compressed sensing [8] have been proposed. Although these methods have achieved cost reduction in monitoring, they are not applicable to our problem since we intend not to change the rate of sampling of one sensor type, but rather to reduce sampling cost by switching between available sensors of different type and accuracy.

Another line of work focuses on finding the optimal location for sensors in distributed deployments, in order to maximize accuracy while minimizing deployment densities. Krause *et al.* [15] have used a probabilistic method to predict the communication cost for a given deployment topology. Papadimitriou *et al.* [19] have employed GP and a Bayesian statistical method to minimize entropy over a set of sensor locations. In contrast, our work is concerned with reducing the cost of sampling from an available set of sensors at any given time, not with reducing the densities of sensor topologies.

In work on so-called multi-modal sensor fusion, data from multiple sensors in a potentially heterogeneous suite are aggregated to monitor a specific measurement application [28, 9]. This method has been widely used, for example in visual monitoring [18, 20] and target tracking [21, 25]. Data fusion focuses on sensor applications that need to compute the correlation between multiple sensor modules and cannot be measured by a single sensor. However, these works do not consider the cost of using different sensors, or minimizing cost.

Cost sensitive multi-modal sensor fusion methods have been developed to balance cost against accuracy, with an eye towards providing fault tolerance [13]. However, we are not concerned with fault tolerance, but strictly between selecting sensors from heterogeneous suites. Willett *et al.* [30] use a small number of sensors to send their readings to a fusion center, and based on the correlation among the sensed data, the fusion center decides which additional sensors should be activated. The same concept has also been tried in a distributed fashion [16]. However, sensing costs in these cases are a function of the number of sensors sampled, not their type.

Perhaps most related to our work is that of Wang *et al.* [29]. They propose a method to find the optimal set of sensors to be polled, using a hybrid tree, where non-leaf nodes act as a decision tree and leaves are standard regression models using a subset of sensors. However, these trees support decision making based on external constraints, i.e., which sensors to use depending on an organization's goals and resources. In contrast, our models are intended to support automated sensor control in WSNs during deployments.

Outside of the adaptive sampling and sensor fusion fields, multi-objective optimization has been used for cost-sensitive modeling. For example Kim [12] sets error as one objective and tree size as another, as we do here. Zhao [32] sets the false negative rate and false positive rate as the two objectives. However, these works do not consider the hierarchical approach that we do.

1.2 Organization of the Chapter

The remaining text is organized as follows. In Section 2 we formalize our basic problem description, and explain how hierarchical and non-hierarchical models are constructed. In Section 3 we describe the experiments we perform to compare these two approaches, and the quantitative results from those experiments. In Section 4 we describe an extension where dynamically changing budget information can be taken into account, and reformulate a problem formalization, as well as a description of methods, experiments, and quantitative results in this extended setting. In Section 5 we discuss and reflect on our quantitative results for all experiments. In Section 6 we conclude with remarks on future work.

2 Methods

This section provides a formalization of the problem, how genetic programming is applied to solve it, and the two variants of genetic programming that we

compare in this work. All of the material for replicating the work described here is available online [1].

2.1 Problem Formalization

Let us assume that t values of some environmental phenomenon $\mathbf{g}$ (the ground truth) are known at time steps $1, \dots t$. These values are stored in $\mathbf{g} = g_1, \dots g_t$. Let us further assume there are k sensors $s_1, \dots s_k$ available that can be used to predict $\mathbf{g}$. Let $r_i^{(t)}$ denote the reading of sensor i taken at time t. Moreover, let $s^{(t)}$ and $r^{(t)}$ denote a subset of sensors, and readings taken from them, at time t. We denote the amount of variance of $\mathbf{g}$ explained by sensor i as $v_{r_i}^{(\mathbf{g})}$. This value is determined by linearly regressing only r_i against g. Finally, let $e_i = 100(1 - v_{r_i}^{(\mathbf{g})})$ and c_i represent the prediction error and cost of using sensor i respectively. Using this formulation, e_i represents the percentage of prediction error incurred by just using sensor i to predict $\mathbf{g}$.

The cost of a sensor c_i is usually inversely proportional to its error e_i, so for the work reported below, we set $c_i = v_{\mathbf{r}_i}^{(\mathbf{g})}$ for each sensor. In certain sensor deployments there may be other factors that affect c_i such as power consumption, market price, effort required to collect a sensor's reading, proprietary issues, and so on. In any case it is important to clarify that in this work we only consider costs of sensor sampling, not operational costs of the platform, e.g. the cost of post-sampling data processing.

We suppose that an ordering of sensors exists such that s_1 is the least expensive sensor with the highest error and s_k is the most expensive sensor with the lowest error. Formally,

$$\forall i, j \,.\, 1 \leq i < j \leq k \rightarrow e_i > e_j \,\wedge c_i < c_j.$$

Let us denote the prediction of a model using a subset of sensors at time t by $p^{(t)}$, i.e., $p^{(t)}$ is a function on $\mathbf{r}^{(t)}$. Then, the error of each sampling $e^{(t)}$ would be

$$e^{(t)} \triangleq |p^{(t)} - g^{(t)}|.$$

The cost of each sampling, $c^{(t)}$ is the cumulated cost of all sensors $s_i \in \mathbf{s}^{(t)}$ that were polled for that sampling:

$$c^{(t)} \triangleq \sum_{j \in \{i | s_i \in \mathbf{s}^{(t)}\}} c_j.$$

It is desired that each sampling $\mathbf{s}^{(t)}$ entails low error and cost. That is, the following equality is desirable:

$$\underset{\mathbf{s}^{(t)}}{\operatorname{argmin}}\, e^{(t)} = \underset{\mathbf{s}^{(t)}}{\operatorname{argmin}}\, c^{(t)}.$$

Our goal is to design models which combine and transform sensor readings to accurately predict the outcome measure, but can also intelligently determine which sensors to poll when cheap, less accurate sensors exhibit uncertainty about the current prediction.

2.2 General Genetic Programming approach

Genetic programming has widely been employed for regression tasks in which the functional form of the equations relating inputs to outputs is unknown [14]. Here, inputs are sensor values and the output is a prediction for a given outcome measurement.

Although many recent improvements have been proposed for GP, here we have kept the genetic programming algorithm simple and instead focused on comparing GP-generated hierarchical and non-hierarchical models. Thus, GP is restricted to the four simple algebraic operators, and each evolutionary trial is initialized with a fixed-sized population of 100 randomly-generated solutions containing three nodes. Maximum tree depth is not set since the tree size is considered as an objective in multi-objective optimization. The crossover rate is set to 0.2 and no fitness stall is considered. If the number of non-dominated solutions reaches 50% of the population size, the training restarts. At the conclusion of each generation, four values are computed for each solution: (1) *error* on training data as defined below, (2) the combined *cost* of the sensors used to make the prediction, (3) the *size* of the solution, and (4) the *age* of the solution. We now discuss each in turn.

Error: Let n be the population size and j range over $\{1, \cdots, n\}$. Let t_j be some solution tree. We represent the error of sampling at time t using solution t_j with $e_{t_j}^{(t)}$. Moreover, $d^{(\text{train})}$ and $d^{(\text{test})}$ denote the training dataset and testing dataset, respectively. Then, we define the error on training data using solution t_j by $e_{t_j}^{(\text{train})}$ and as the average of $e_{t_j}^{(t)}$ on all samples in $d^{(\text{train})}$, i.e.,

$$e_{t_j}^{(\text{train})} \triangleq \sum_{g^{(t)} \in d^{(\text{train})}} \frac{e_{t_j}^{(t)}}{|d^{(\text{train})}|}. \tag{1}$$

Each solution t_j was allowed to use a subset (possibly empty) of available sensors. The cost of each solution depends on the sensors that are employed and the sampling.

Cost: As described in the following sub-sections, current sensor readings may trigger readings from additional sensors. Thus, different $r_i^{(t)}$ may cause t_j to need different $\mathbf{s}^{(t)}$. The average cost of a tree on training data $c_{t_j}^{(\text{train})}$ is thus defined as the cost of all of the sensors that have been used to predict the outcome for each training instance, averaged over all instances in the training dataset:

$$c_{t_j}^{(\text{train})} \triangleq \sum_{\mathbf{r}^{(t)} \in d^{(\text{train})}} \sum_{l \in \{i | s_i \in \mathbf{s}^{(t)}\}} \frac{c_l}{|d^{(\text{train})}|}. \tag{2}$$

If a solution uses a sensor more than once, no extra cost is incurred: because the sensor has already been polled, its output is already available and can thus be re-used as often as required.

Size: To avoid bloat, solution size, defined as the number of nodes in the tree, was incorporated into the fitness objectives during the optimization process [11].

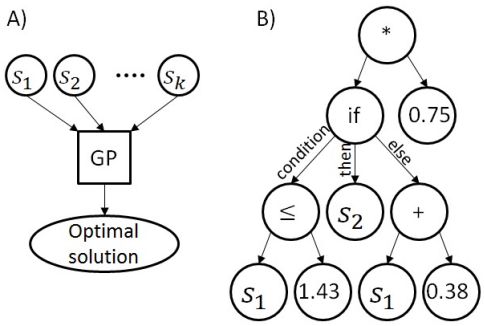

Fig. 1. A) Non-hierarchical framework. B) A non-hierarchical sample solution.

Age: We employed the Age-Fitness Pareto Optimization (AFPO) method [22], which injects a new randomly-generated solution into the population at each generation and compares the solutions with same age in an effort to guard against convergence. Each solution's age is defined as the number of generations since its oldest ancestor was injected into the population. A new solution produced by mutating an existing solution inherits the same age as its parent. If two existing parents are crossed to produce two new offspring, the offspring inherit the age of the older of the two parents. AFPO is a multiobjective optimization method as solution age is used as an additional fitness objective during optimization.

Optimization. At the end of each generation, the Pareto front is computed according to the objectives used, and the dominated solutions are discarded. Multi-objective optimization with all four objectives described above could easily lead to population collapse in the sense that all members of the population could become non-dominated. To guard against this eventuality, one possibility is to restart the evolutionary run with new solutions if no dominated solutions are detected in the population at the end of a given generation. Alternatively, a very large population size can be employed. However, both of these solutions greatly increase the computational effort required to obtain satisfactory solutions to the given problem. To avoid this situation, different multi-objective optimization approaches has been proposed. One of the simplest non-parametric approaches is to reduce the number of objectives by multiplying objectives together and using the result in the optimization process [10]. In this experiment, since error is the most important outcome, error is used for the primary objective and the second objective is the result of multiplying cost, size and age together.

Once the dominated solutions are deleted, the empty slots in the population are then filled by mutating and crossing copies of the non-dominated solutions. Tournament selection is used to select parents from the front for these operations. After the last generation, age is discarded when computing members of the Pareto front, since the goal is to use only small, accurate and cost-effective solutions for prediction, regardless of their age.

2.3 Non-hierarchical GP

A naive approach to cost-sensitive modeling using GP would be to evolve individual trees that add conditional and comparative operators to the base set of operators, and allow the tree to poll the values of all sensors if desired, as shown in Figure 1.A. In this way, different parts of the solution tree will be visited depending on the current values of the sensors. If less expensive sensors report a certain combination of values which in the current circumstances is unlikely to provide a good prediction, successful solutions may evolve that visit nodes containing references to expensive sensors.

Figure 1.B shows an hypothetical example of a GP solution t_j that has evolved to encode a useful conditional. In this example, an inexpensive sensor s_1 is first polled. If its reported value $r_1^{(t)}$ is below some threshold, the reading of a more expensive sensor s_2 will be used. It is assumed here that s_1 leads to making poor predictions of the outcome if its reading is below 1.43. If this threshold is exceeded, $r_1^{(t)}$ is then used to predict the outcome.

Conditional operators should, indirectly, encode the differential effects on the available sensors, and the relative costs of those sensors. Note that this is possible even if GP does not have direct access to these differential effects and costs, as they are indirectly reflected in the errors and costs incurred when each solution is evaluated. This issue is worth mentioning in that these effects are complex, non-linear and noisy, and even field experts cannot define them precisely.

2.4 Hierarchical GP

An alternative approach to reconciling prediction error and prediction cost is to build a hierarchy of models: models in the lower layers only have access to inexpensive sensors, while models in the upper layers have access to a greater subset of the sensors, including more expensive ones. When deployed, the overall model returns a prediction from a lower layer if the inexpensive sensors are confident of their combined prediction. If they are not, predictions are drawn from a higher layer.

Briefly, constructing such a model proceeds in two phases:

1. Build a set of k layers, one for each sensor modality. For each layer i, run GP to find a set of accurate and low-cost solutions that use one or more sensors from the set $s_1, s_2, \ldots s_i$.
2. Define conditions which determine which layer should be allowed to provide the prediction, given the current environmental conditions.

Figure 2 illustrates what such a hierarchical model looks like. At the outset of attempting to provide a prediction for the current environmental conditions, the models stored in the lowest layer are evaluated, which only have access to the least expensive sensor s_1. If the certainty of their combined predictions is acceptable, return the combined prediction of these models. Otherwise, evaluate the models at the next layer, which have access to s_1 and the next least expensive

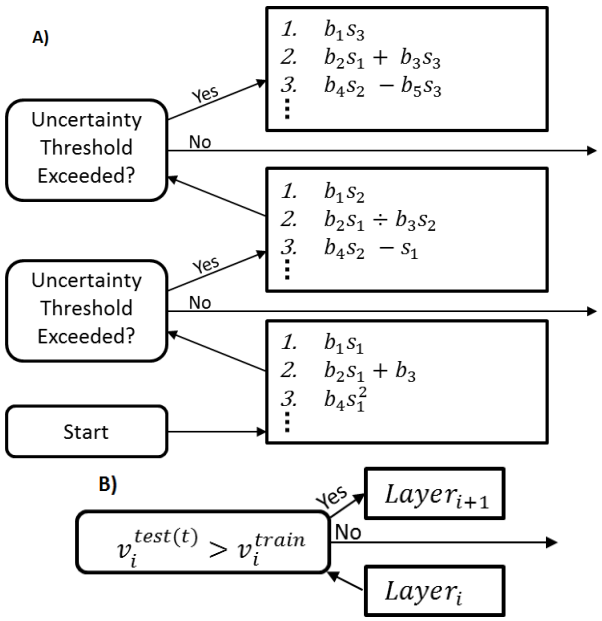

Fig. 2. Hierarchical framework (A). Using the difference between training data prediction variance and test data prediction variance as the condition for switching between model layers (B).

sensor s_2. If these models are acceptably confident in the prediction, return their combined prediction; otherwise, evaluate the solutions at the next layer, and so on. If the top layer is reached, the combined predictions of the models found there are returned as the overall prediction, regardless of their level of certainty. The incremental construction of these models is described next.

Starting with the least expensive sensor s_1, GP is used to find the best models for converting $r_1^{(t)}$ to $g^{(t)}$. When GP terminates, the final non-dominated solutions are then organized as a group named layer L_1. The same process is repeated for s_2, except for the fact that since s_1 is already polled in L_1, it may be incorporated into models during evolution without incurring an extra cost for the solution tree that makes use of it. Similarly, for each sensor s_i, a separate GP run is performed with sensors s_1 to s_i available as input to construct layer L_i. These layers are then organized in a hierarchical fashion. The order of layers is based on the cost of the most expensive sensor they are representing, from L_1 to L_k. Suppose each layer L_i consists of n_i solutions and the jth solution t_j in L_i is denoted as $t_{i,j}$. Let $p_{t_{i,j}}^{(t)}$ denote the prediction of $g^{(t)}$ that $t_{i,j}$ provides.

Then, the final prediction of layer L_i for $g^{(t)}$ is

$$p_{L_i}^{(t)} \triangleq \sum_{j=1}^{n_i} \frac{p_{t_{i,j}}^{(t)}}{n_i}.$$

The error that corresponds to $p_{L_i}^{(t)}$ is

$$e_{L_i}^{(t)} \triangleq |p_{L_i}^{(t)} - g^{(t)}|.$$

In the second phase, a conditional must be formulated to determine whether the current layer should return its prediction, or whether the burden of prediction should be passed up to the next layer. One common method for measuring how confident an ensemble of models is, is to compute the variance in their predictions [23]: if variance is low, and those models are sufficiently independent of one another, there is a greater likelihood that their combined predictions can be trusted. If variance is high, this is likely the result of differing assumptions encoded in the models, which cannot all be true reflections of the hidden relationship being modeled. Note the assumption here that the models are relatively independent: a set of identical models will never exhibit a variance in their predictions, regardless of how accurate the individual models are. We can be somewhat confident of the independence of our models, as they are produced by the AFPO algorithm: models with differing ages are likely to arrive on the final Pareto front used to build each layer, and such differently-aged genomes are likely to be independent because of their different genetic origins.

Formally: Let $p_{L_i}^{\text{train(t)}}$ and $e_{L_i}^{\text{train(t)}}$ denote $p_{L_i}^{(t)}$ and $e_{L_i}^{(t)}$ using $\mathbf{r}^{(t)}$ on d^{train}, respectively. Similarly, $p_{L_i}^{\text{test(t)}}$ and $e_{L_i}^{\text{test(t)}}$ respectively denote $p_{L_i}^{(t)}$ and $e_{L_i}^{(t)}$ using $\mathbf{r}^{(t)}$ on d^{test}. Moreover, assume $v_i^{\text{train(t)}}$ and $v_i^{\text{test(t)}}$ are the variances of all $p_{t_{i,j}}^{(t)}$s on d^{train} and d^{test}. Also, v_i^{train} denotes $v_i^{\text{train(t)}}$ averaged over all the samplings in d^{train}.

To determine whether the burden of prediction should remain with the current layer or passed off to a higher layer, we measure the difference in prediction variance between the models when presented with the training data (v_i^{train}) or with the testing data, i.e. the current environmental conditions ($v_i^{\text{test(t)}}$). When $v_i^{\text{test(t)}}$ is almost the same as v_i^{train}, there is a high probability that $e_{L_i}^{\text{test(t)}}$ is an approximation of $e_{L_i}^{\text{train(t)}}$, and we can be relatively confident that these models will yield a good collective prediction of $g^{(t)}$. When the variance of test data prediction is significantly higher than prediction on the training data, this signals that the solutions in that layer are exhibiting increased disagreement regarding the current environmental conditions. This could be due to the fact that a specific sensor is not physically able to predict under the current conditions, or the solutions have not been trained for the current situation. In such an eventuality it would be advantageous to switch to the next layer, in the hope that its models will exhibit more confidence in their ability to predict the current conditions. In

Table 1. Available sensors and their features.

Name	**Equation Template of $r_i^{(t)}$**	**Cost**
s_3	$g^{(t)}$	0.3
s_2	$b_{2,1}g^{(t)} + b_{2,2}$	0.2
s_1	$b_{1,1}(g^{(t)})^2 + b_{1,2}g^{(t)} + b_{1,3}$	0.1

this paper, the variance is considered as a proxy for entropy, but any other entropy related metric could be used instead. Figure 2 illustrates how this intuition is encoded into the switching condition in the hierarchy of layers.

By considering the amount of difference between prediction variance on training and testing data, we can dynamically tune how conservative or liberal the overall hierarchical model is: if little difference is tolerated, the burden of prediction will often be passed to higher layers, resulting in expensive yet accurate predictions; if much difference is tolerated, lower levels will tend to predict, resulting in less expensive and less accurate predictions. The advantage of this approach is that the amount of tolerance could be dynamically tuned based on the current available budget for sensing.

For example, for larger budgets, more cost could be expended in order to obtain more accurate results. In this regard, the tolerance of the difference between variances could be decreased, transferring the burden of prediction to higher layers. Similarly for small budgets, the tolerance would be increased. Through this adjustment, more disagreement would be tolerated and less accurate predictions would be obtained for lower cost. To implement this dynamic tuning given a fluctuating budget, a tolerance parameter $\tau \in [0, 1]$ is defined, reflecting the tolerance of disagreement between the solutions of a given layer. Equation (3) demonstrates how this parameter is used to determine which level should be activated for prediction.

$$p^{(t)} = \begin{cases} p_{L_i}^{(t)} & \text{if } v_i^{\text{train}} > |1 - \tau| \cdot v_i^{\text{test}(t)} \\ p_{L_{i+1}}^{(t)} & \text{otherwise} \end{cases} \tag{3}$$

It should be noted that in the present work, the same value for τ is used at the interstices between each pair of layers. However, different values for τ could be employed between different layers to enable the model to respond better to changes in the overall available budget. The extreme cases occur when $\tau = 0$ or $\tau = 1$. The former case ensures that the condition in Equation 3 holds when the prediction variance on the testing data is greater than the prediction variance on the training data. This occurs with high probability, so setting $\tau = 0$ tends to extract the predictions from solutions on the uppermost layer. Setting $\tau = 1$ ensures that the first layer always provides the prediction since variance on the testing data will always be finite. Values greater than $\tau = 1$ are not investigated in this work, but are possible. Greater τ value increases the probability of the conditional to be true. $\tau = \infty$ causes the conditional to always be true, thus the method always collects predictions from the last layer.

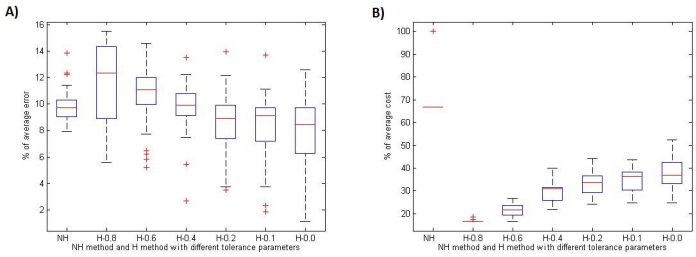

Fig. 3. Average error (A) and average cost (B) on the test data for the non-hierarchical and the hierarchical methods with different tolerance parameters. Statistical significance of these results are reported in Table 2.

3 Results

The proposed methods are evaluated over two set of experiments, using a synthesized dataset and ten actual datasets. This section summarizes these datasets, experimental setups, and quantitative results. These results were also reported in a preliminary version of the work reported here [33].

3.1 Synthesized Data

In these experiment, the proposed methods have been evaluated on a synthetic system monitored by three different sensors. Table 1 shows these three sensors, their readings in relation to $g^{(t)}$, and their cost.

To create the training and testing datasets, at first coefficients in the equations of the sensor relations, i.e., $b_{i,j}$, were randomly selected in the range $[0, 1]$. Then, random numbers were generated for $g^{(t)}$ in the range $[0, 3]$, and used to calculate the sensor readings based on the given template and selected coefficients. The training and testing dataset sizes were 150 and 50, respectively, and each experiment was repeated 40 times.

Table 2. A) P-values considering error of the non-hierarchical and the hierarchical methods with different tolerance parameters. B) P-values considering cost of the non-hierarchical and the hierarchical methods with different tolerance parameters.

A	P-values	B	P-values
$\tau = 0.8$	0.013414	$\tau = 0.8$	$\ll 0.001$
$\tau = 0.6$	0.046626	$\tau = 0.6$	$\ll 0.001$
$\tau = 0.4$	0.635566	$\tau = 0.4$	$\ll 0.001$
$\tau = 0.2$	0.001309	$\tau = 0.2$	$\ll 0.001$
$\tau = 0.1$	$\ll 0.001$	$\tau = 0.1$	$\ll 0.001$
$\tau = 0.0$	$\ll 0.001$	$\tau = 0.0$	$\ll 0.001$

Non-hierarchical setup. The population size is 100 and is trained for 300 generations. The optimization process during the last generation does not consider

age as an objective and the Pareto front is selected using *error* and *cost* $\times$ *size* as two separate objectives. After training, the knee of the non-dominated solutions is selected and tested using the testing dataset. In order to select the knee, the euclidean distance of each solution on the Pareto front is calculated from the ideal model. The ideal model is a solution with no error and zero cost. This is defined as follows:

$$t_{\text{knee}} = \underset{t_j \in \text{Pareto front}}{\operatorname{argmin}} \sqrt{(e_{t_j} - 0)^2 + (c_{t_j} - 0)^2}.$$

Hierarchical setup. The population size for each layer is 100 and each layer was trained for 100 generations to equalize the total computational effort applied in both methods. Similarly to the non-hierarchical setup, during the last generation, *age* is not considered in the Pareto optimization process, and non-dominated solutions are selected based on *error* and *cost* $\times$ *size* as two separate objectives. After training, for each layer L_i, the variance of the solutions output on training data v_i^{train} is computed and stored as the threshold of switching to the next layer L_{i+1}. This variance is not computed for the layer corresponding to the most expensive sensor, i.e., L_3, since there are no more sensors to be called. The experiment was repeated 40 times for each of the different tolerance parameters $\tau = 0.0, 0.1, 0.2, 0.4, 0.6, 0.8$.

Results on Synthesized Data We now consider average error and cost of the different modeling approaches on synthesized data and report P-values for two tailed t-tests where $\alpha = 0.5$.

Average error. The average error of the non-hierarchical method is $e_{t_j}^{\text{test}}$, where t_j is the final selected solution. The average error of the hierarchical method is the average of $e_{L_i}^{\text{test}}$, where L_i is the last layer reached in the hierarchy, during the sampling. As can be seen in Figure 3, the largest difference in error occurs at maximum tolerance i.e. $\tau = 0.8$ where the error of the hierarchical method is 1.34% higher than the non-hierarchical method. The hierarchical method tends to achieve lower average error when the tolerance parameter is $\tau < 0.4$. P-values obtained for different tolerance parameters are represented in Table 2.A and show that $\tau = 0.4$ is the boundary where the hierarchical method begins to outperform the non-hierarchical method.

Average cost. By considering t_j as the final selected solution in the non-hierarchical method, the average cost is $c_{t_j}^{\text{test}}$. The average cost of the hierarchical method is the average of $\sum_{j=1}^{i} c_{L_j}^{\text{test}}$, where the last layer reached during the sampling is L_i. In order to compare both methods and understand how much of the potential cost each method uses, the cost of each method is represented as the percentage of cost of using all available sensors. Figure 3 shows that the average cost of the hierarchical method is significantly lower than the non-hierarchical method (at most 54.88% and at least 33.81% lower cost). Table 2.B) summarizes the p-values to show how significantly the cost of the hierarchical method is lower than the non-hierarchical method.

Table 3. Used UCI datasets.

DS No.	DS Name	No. of Instances	No. of sensors	$g^{(t)}$ Average
DS_1	Auto MPG	398	7	23.51457
DS_2	Housing	506	13	22.53281
DS_3	Forest Fires	517	12	0.031663
DS_4	Energy Efficiency	768	8	22.3072
DS_5	Concrete Compressive Strength	1030	8	35.81796
DS_6	Solar Flare	1389	9	0.300188
DS_7	Airfoil Self-Noise	1503	5	124.8359
DS_8	SkilCraft1 Master Table Dataset	3395	19	4.184094
DS_9	Wine Quality	4898	11	5.877909
DS_{10}	Parkinson's Telemonitoring	5875	17	29.01894

Table 4. Value of $v_{\mathbf{r}_i}^{(\mathbf{g})}$ for all of the sensors of Auto MPG dataset.

DS No.	s_1	s_2	s_3	s_4	s_5	s_6	s_7
Auto MPG	0.1766	0.3175	0.3356	0.5951	0.6012	0.6467	0.6918

Table 5. Minimum and maximum amount of variance a sensor accounts for and the order of their difference $\frac{\max v_{\mathbf{r}_i}^{(\mathbf{g})}}{\min v_{\mathbf{r}_i}^{(\mathbf{g})}}$, in each dataset.

DS No.	$\min v_{\mathbf{r}_i}^{(\mathbf{g})}$	$\max v_{\mathbf{r}_i}^{(\mathbf{g})}$	Difference ratio
DS_1	0.1766	0.6918	3.92
DS_2	0.0307	0.5441	17.72
DS_3	0.0002	0.2578	1289
DS_4	0.0076	0.7911	104.10
DS_5	0.0112	0.2478	22.13
DS_6	0.000	0.096	96
DS_7	0.0157	0.1527	9.73
DS_8	0.0005	0.4542	908.40
DS_9	0.0001	0.1897	1897
DS_{10}	0.0037	0.0263	7.11

3.2 Actual Data

In this experiment, ten datasets are selected from the UCI database repository [2] based on the number of instances and features from the regression section. Table 3 summarizes these datasets and their features. For these datasets, we partition each into two halves for training and testing, and treat the individual features as individual sensors. Each experiment in this section was repeated 30 times.

In order to determine the accuracy of each sensor s_i in predicting $g^{(t)}$, the value of $v_{\mathbf{r}_i}^{(\mathbf{g})}$ is calculated for each available sensor of each dataset, using linear regression. The greater $v_{\mathbf{r}_i}^{(\mathbf{g})}$ is, the better that sensor can predict $g^{(t)}$.Table 4 summarizes the values of $v_{\mathbf{r}_i}^{(\mathbf{g})}$ for all of the sensors of the Auto MPG dataset, as an example. We define the cost of each sensor in these datasets as $v_{\mathbf{r}_i}^{(\mathbf{g})}$.

Non-hierarchical setup. The population size is 200 and for each dataset with k features, it is trained for $200 \times k$ generations.

Hierarchical setup. The population size for each layer is 200. Similar to synthesized data experiments, in order to equalize search effort in both methods,

Table 6. Average error (L) and cost (R) percentages and the corresponding p-values for the hierarchical and the non-hierarchical methods.

DS No.	NH: error %	H: error %	P-value	NH: cost %	H: cost %	P-value
DS_1	20.85	25.81	$\ll 0.001$	38.89	12.33	0.022
DS_2	25.90	28.93	$\ll 0.001$	23.18	1.26	$\ll 0.001$
DS_3	126.49	202.12	0.565	6.83	15.81	$\ll 0.001$
DS_4	29.19	36.70	$\ll 0.001$	32.90	4.18	$\ll 0.001$
DS_5	35.29	39.68	0.393	53.63	28.63	0.004
DS_6	110.63	111.08	0.223	0.00	0.98	0.040
DS_7	0.00	0.00	0.082	11.58	7.35	0.005
DS_8	37.59	28.65	0.194	2.55	0.00	$\ll 0.001$
DS_9	10.79	10.67	0.197	0.02	0.00	$\ll 0.001$
DS_{10}	32.11	29.88	0.423	20.62	3.88	0.009

each layer was trained for 200 generations. After training, a subset of the non-dominated solutions with least error are selected and organized in the corresponding layer. The cardinality of this subset is 2% of the population size. This experiment was conducted for tolerance parameter $\tau = 0.1$. This value is selected based on the results in 3.1 and will be discussed in more detail in Section 5.1.

Results on Actual Data We now consider average error and cost of the different modeling approaches on actual data obtained from UCI data repository.

Average error. The average error for the non-hierarchical and the hierarchical methods are $e^{\text{test}}_{t_j}$ and $e^{\text{test}}_{L_i}$ respectively, where t_j is the final selected solution in the non-hierarchical method and L_i is the last layer reached during the sampling in the hierarchical method. Table 6 summarizes the average error of both methods on all of the datasets as a percentage of error. It can be seen that for 6 datasets, the average error of the hierarchical method is higher than the average error of the non-hierarchical method. However, the p-value for the two-tailed t-test shows that for 3 datasets, this difference is not significant. There are three cases where the difference is significant i.e., DS_1, DS_2 and DS_4.

Average cost. Similar to Section 3.1, the average cost is represented as the percentage of the maximum possible cost. Table 6 summarizes the percentage of the average cost each method uses for prediction. The cost of the hierarchical method is significantly lower in all cases except for DS_3 and DS_6.

4 Adapting to Dynamic Budgets

In remote sensor deployments, the cost associated with sensor sampling may have an effect on the *budget* available. Budget fluctuations can be due to various reasons, depending on the network and the particular definition of the budget. For example, if the budget is defined to be the capacity of a solar rechargeable battery powering the sensor system, the budget may increase on a sunny day, regardless of sampling frequencies, and may decrease on a cloudy day or at night due to sensor usage and battery draw-down. In fact, battery power levels in systems with solar recharging often exhibit a consistently diurnal pattern.

Since it is possible for budgets to fluctuate, a cost-sensitive approach to sensor sampling will ideally *adapt* to changing budget levels, in order to extend deployment lifetimes. In particular, as budgets decrease, models should be biased more towards use of less-costly sensors, to preserve the existing budget and prevent using the entire budget. In the case where the budget is taken to be the battery power level, complete use of the budget corresponds to complete battery drawdown– a potentially catastrophic situation that generally should be avoided.

In this section we reconsider the hierarchical and non-hierarchical methods described previously, with modifications to adapt to fluctuating budgets. In the case of the hierarchical method, we adapt models by allowing the threshold τ to be dynamically tuned in proportion to the remaining budget. In the case of the non-hierarchical method, we add the remaining budget as an input parameter to training and testing. Our main goal is to explore the relative performance of models generated by these respective methods. All of the material for replicating the work described here is also available online [1].

A crucial element of this investigation is the concept of *noise*. Active sensors typically have thresholds for reliable use, and as power levels drop near and then below these levels, sensor noise increases. We observe that this phenomena actually benefits adaptation to budget levels in model training, since increased noise increases error and hence discourages sampling. We consider in particular the scenario where more expensive sensors experience more noise as sensor levels drop– this scenario has an empirical basis in the experience of the authors [3], and has the added benefit (as we will show) of greater bias towards less expensive sensors as budget levels decrease.

Summary of Training and Testing Regimes. To encourage adaptation to fluctuating budgets, during training each model is exposed to two different environments: one with a "high" budget and the other with a "low" budget, relative to a posited lower threshold for sensor inputs. Note that only the non-hierarchical models will use the budget level as an input parameter, but predictions of models generated by both methods experience noise proportional to the budget level and the cost of sensors used in the prediction. The optimization objectives for both the hierarchical and non-hierarchical regimes remain the same as in the preceding experiments.

After training, the resultant models are tested on four conditions: each condition takes one of two different initial budgets—high and low—and one of two different budget behaviors: one that stays constant until drawn down by sensor use, and another that has an underlying sinusoidal pattern that simulates diurnal replenishing from solar recharging. Models that exhibit low error and cost in all four situations are considered most desirable.

4.1 Problem Formalization

Let $B^{(t)}$ be a real number defining the amount of the available budget at time t if none of the sensors is polled from the first sampling $\mathbf{S}^{(1)}$ to the last sampling

before now $\mathbf{S}^{(t-1)}$. Then, $\mathbf{B}$ is the vector of the available budget for all of the sampling times without any sensor being polled.

Let $\mathbf{B}_H$ and $\mathbf{B}_L$ be the vectors reporting the currently available budget if none of the sensors are polled, where the budget was initially 'high' or 'low'. We denote individual budget values in these vectors as $B_H^{(t)}$ or $B_L^{(t)}$, respectively. The initial budget is considered to be high if the model has enough of a budget to poll two thirds of the available sensors for each sampling,

$$B_H^{(1)} \geqslant |d^{train}|(\frac{2}{3}\sum_{i=1}^{|\mathbf{S}|} c_i)$$

The average cost of the hierarchical and the non-hierarchical models reported in Section 3 are all less than $\mathbf{B}_H$. That is the reason we believe this is a good threshold for the high budget level. If the budget is not enough to poll at least one third of the available sensors for each sampling, then it is considered to be low:

$$B_L^{(1)} \leqslant |d^{train}|(\frac{1}{3}\sum_{i=1}^{|\mathbf{S}|} c_i)$$

Let $c_{t_j,\mathbf{B}}^{(t)}$ denote the cost of evaluating solution tree t_j at time t, considering the available budget $\mathbf{B}$, which could in turn be drawn from $\mathbf{B}_L$ or $\mathbf{B}_H$. Note that $c_{t_j,\mathbf{B}}^{(t)}$ depends on the particular solution tree t_j and the sensors used by that model, as explained in Section 4. This cost should be deducted from the currently available budget.

Let $R_{t_j,\mathbf{B}}^{(t)}$ then denote the amount of remaining budget at time t, considering budget $\mathbf{B}$ for each solution tree t_j. Then, $R_{t_j,\mathbf{B}}^{(t)}$ can be defined as

$$R_{t_j,\mathbf{B}_b}^{(t)} = B^{(t)} - \sum_{l=1}^{t-1} c_{t_j,\mathbf{B}_b}^{(l)}, b \in \{H, L, \epsilon\}$$

We define $R_{t_j,\mathbf{B}_H}^{(t)}$ and $R_{t_j,\mathbf{B}_L}^{(t)}$ as the remaining budgets when the budget $\mathbf{B}$ being used is either the high budget $\mathbf{B}_H$ or the low budget $\mathbf{B}_L$.

It is notable that, as explained in Section 4, the accuracy of sensors is affected by the level of the remaining budget. By decreasing the amount of the available budget, the error of a sensor S_i and the amount of noise in its reading r_i will increase. Moreover, we consider a noise model in which different sensors may be affected by a reading differently: we assume that less expensive sensors become less noisy as the level of the remaining budget drops, since they are less costly and thus reduce the budget less than expensive sensors. This behaviour is modelled as follows, where $U(\min(r_i), \max(r_i))$ is a uniform random number from the r_i domain:

$$r_i'^{(t)}(t_j, \mathbf{B}) = (\frac{R_{t_j,\mathbf{B}}^{(t)}}{B^{(t)}})^{c_i} r_i^{(t)} + (1 - (\frac{R_{t_j,\mathbf{B}}^{(t)}}{B^{(t)}})^{c_i}) U(\min(r_i), \max(r_i))$$

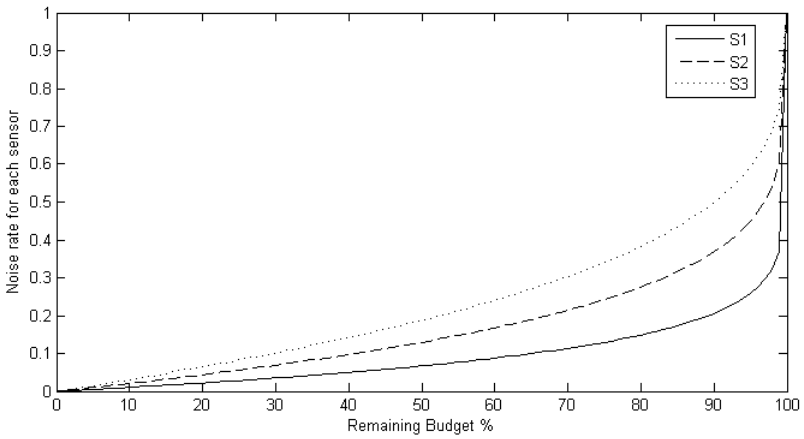

Fig. 4. The noise rate for different sensors with different cost from synthesized dataset.

In this manner, by decreasing the available budget, the accuracy of the sensor readings will decrease and the noise in the sensor readings will increase. This effect is proportionate to the cost. Figure 4 shows the rate of noise as the available budget level drops for three sensors in the synthesized dataset introduced in Section 3.1. As shown there, the cheapest sensor S_1 becomes less noisy with decreasing budget, whereas the most expensive sensor S_3 suffers a greater noise increase as the budget decreases.

This noise model should encourage the selection of models that make use of less expensive sensors for predictions for two reasons. First, less expensive sensors become less noisy when the available budget level drops, compared to more expensive sensors. Second, using less expensive sensors keeps the cost of each prediction low. Since the prediction cost has to be paid for from the available budget, low cost models cause a slower decrease in the budget and thus retain more accurate sensor readings.

When the accuracy of a sensor s_i is affected by the available budget, then the accuracy of the solution tree t_j that makes use of s_i also suffers. Let $e^{(t)}_{t_j,\mathbf{B}}$ denote the error of solution tree t_j at time t, and where the available budget is $\mathbf{B}$.

4.2 Methods

In order for the models to adapt to a changing budget, both the hierarchical and non-hierarchical methods described in Section 2 should be able to alter their prediction strategies, given the current budget.

Non-Hierarchical GP In order to enable the non-hierarchical models to modify their prediction strategy given the current budget, we include the currently remaining budget $R^{(t)}_{t_j,\mathbf{B}}$ as an additional 'sensor' that can be incorporated into solution trees during model training. This is realistic since budget information such as power level data is frequently accessible in WSN systems. If this sensor is incorporated into a model, it can read the level of the remaining budget at no cost, when it is needed.

The solution trees are trained in the same way described in Section 2.3, except for how the trees are evaluated. Each solution tree t_j is evaluated twice, once using $\mathbf{B}_H$ and once using $\mathbf{B}_L$, to encourage model robustness. These two budget distributions are considered to be flat, without any budget harvesting: that is, the overall budget does not increase or decrease over time if no sensors are polled.

Let $e^{(t)}_{t_j,\mathbf{B}_H}$ and $e^{(t)}_{t_j,\mathbf{B}_L}$ denote the error of a solution tree t_j at time t when the budget level was either high or low. The error of a solution tree t_j at time t, denoted as $e^{(t)}_{t_j}$, can then be computed as

$$e^{(t)}_{t_j} = \frac{e^{(t)}_{t_j,\mathbf{B}_H} + e^{(t)}_{t_j,\mathbf{B}_L}}{2} \tag{4}$$

Similarly, the cost of a solution tree at time t, $c^{(t)}_{t_j}$, is computed as

$$c^{(t)}_{t_j} = \frac{c^{(t)}_{t_j,\mathbf{B}_H} + c^{(t)}_{t_j,\mathbf{B}_L}}{2} \tag{5}$$

With this formulation, the average error of a solution tree $e^{\text{train}}_{t_j}$, and the overall cost of a tree $e^{\text{train}}_{t_j}$, can be calculated based on Equations 1 and 2 given in Section 2.2.

Hierarchical GP The hierarchical method is trained the same as described in Section 2.4 except that, like the non-hierarchical method, models in the hierarchical method are trained on both $\mathbf{B}_H$ and $\mathbf{B}_L$, and their respective costs and errors are computed as the average cost and error incurred in these two budget regimes (Equations. 4 and 5). In this manner, each layer L_i consists of models with high robustness over different budget distributions.

In Section 2.4 the tolerance parameter τ is statically defined and does not change during model execution. Here however, as the budget is dynamic, the tolerance parameter should change accordingly. Therefore, τ is defined as

$$\tau' = 1 - \frac{R^{(t)}_{t_j,\mathbf{B}}}{B^{(t)}}$$

This balances which layer of the model hierarchy provides predictions, given the currently remaining budget. When the remaining budget is high, the threshold for disagreement between models of a given layer is low, so predictions tend

to be drawn from higher layers which have high accuracy. A low remaining budget means that the threshold for disagreement between models of a given layer is high, thus relegating predictions to lower levels of the model hierarchy. This has the effect of causing the overall hierarchical model to become increasingly conservative in its use of sensors as the budget decreases.

Substituting the tolerance parameter τ with this new τ' in Equation 3 given in Section 2.4 thus results in a new condition for switching between layers:

$$p^{(t)} = \begin{cases} p_{L_i}^{(t)} & \text{if } v_i^{\text{train}} > \dfrac{R_{t_j,\mathbf{B}}^{(t)}}{B^{(t)}} \cdot v_i^{\text{test}(t)} \\ p_{L_{i+1}}^{(t)} & \text{otherwise} \end{cases}$$

4.3 Results

These altered methods for training models were evaluated against two sets of data: a synthesized dataset and ten actual datasets, as described in Section 3. This section summarizes the results from training with these datasets.

Each model is trained with given fixed budgets $\mathbf{B}_H$ and $\mathbf{B}_L$. These budget distributions are defined such that for all times t we have:

$$B_b^{(t)} = |d^{train}|(\alpha \sum_{i=1}^{|\mathbf{S}|} c_i) \qquad \text{where } \begin{cases} \alpha = 1/3 & \text{if } b = L \\ \alpha = 2/3 & \text{if } b = H. \end{cases}$$

After training models for each dataset, they are tested on four budget distributions $\mathbf{B}_H$, $\mathbf{B}_L$, $\mathbf{B}_{H,\text{sin}}$ and $\mathbf{B}_{L,\text{sin}}$. Budget distributions $\mathbf{B}_H$ and $\mathbf{B}_L$ are flat and the same as training datasets. Budget distributions $\mathbf{B}_{H,\text{sin}}$ and $\mathbf{B}_{L,\text{sin}}$ were constructed to simulate diurnal replenishing of solar powered sensors. This was accomplished by adding a sinusoidal pattern to $\mathbf{B}_H$ and $\mathbf{B}_L$. The amplitude of the sine wave is set to 2% of the high budget level. We let $B_{b,\text{sin}}^{(t)}$ for $b \in \{H, L\}$ denote the budget value at time t in a given sinusoidal distribution $\mathbf{B}_{H,\text{sin}}$ and $\mathbf{B}_{L,\text{sin}}$. We define the latter such that for any time t we have:

$$B_{b,\text{sin}}^{(t)} = B_b^{(t)} + \sin(\frac{t}{\lceil B_H^{(t)} * 0.02 \rceil}) \qquad b \in \{\epsilon, H, L\}$$

Synthesized Data In these experiments, the synthesized data described in Section 3.1 is used to evaluate the proposed methods. The training and testing datasets are the same, except that budget is also included as an extra sensor. The training and testing datasets both contain 150 samples, and each experiment is repeated 30 times for each budget distribution.

Non-hierarchical model training. The population size is set to 100, and models are trained for 600 generations, or more precisely 200 generations multiplied by the number of available sensors, 3 in this case. The budget sensor is available to the non-hierarchical model. The rest of the settings are the same as described

Table 7. (A) p-values considering error of the non-hierarchical and the hierarchical methods with different budget distribution. (B) p-values considering cost of the non-hierarchical and the hierarchical methods with different budget distribution.

(a)	p-values	(b)	p-values
$\mathbf{B}_H$	$\ll 0.001$	$\mathbf{B}_H$	$\ll 0.001$
$\mathbf{B}_{H,\text{sin}}$	$\ll 0.001$	$\mathbf{B}_{H,\text{sin}}$	$\ll 0.001$
$\mathbf{B}_L$	0.612857	$\mathbf{B}_L$	$\ll 0.001$
$\mathbf{B}_{L,\text{sin}}$	0.590440	$\mathbf{B}_{L,\text{sin}}$	$\ll 0.001$

in Section 3.1. After training, the selected model is tested on all four different budget distributions.

Hierarchical model training. The population size for each layer is set to 100, and each of the three layers are trained for 200 generations. For training each layer L_i, the corresponding sensor S_i and the less expensive ones $\{S_j | j < i\}$ are provided as input. Models do not have access to the remaining budget level as an extra feature in training. The rest of model training is as described in Section 3.1. During testing, the dynamic tolerance parameter τ' is used.

Results on Synthesized Data If the non-hierarchical model selected for testing is t_j, the average cost of the non-hierarchical model is equal to $c^{(t)}_{t_j,\mathbf{B}}$ averaged over the testing dataset d^{test}. The average error of the non-hierarchical model is also equal to the error of each sampling $e^{(t)}_{t_j,\mathbf{B}}$ averaged over the testing dataset d^{test}. The budget distribution $\mathbf{B}$ could be one of the four given distributions. The sensor readings are denoted as r'_i, which reflect the noise considering the remaining budget level as described in Section 4.1.

Let $L^{(t)}$ denote the layer in the hierarchical method that the prediction is drawn from at time t. Then, the average cost of the hierarchical model at time t is equal to $c^{(t)}_{L^{(t)},\mathbf{B}}$, averaged over the testing dataset d^{test}. The average error of the hierarchical method is equal to the error of layer $L^{(t)}$ averaged over the testing dataset. The error of a layer L_i is equal to

$$e^{(t)}_{t_{i,j},\mathbf{B}} = \frac{1}{|L_i|} \sum_{j=1}^{|L_i|} e^{(t)}_{i,j}$$

where $|L_i|$ defines the number of solution trees in layer L_i and $e^{(t)}_{i,j}$ is the error of the jth solution in layer L_i when sensor readings are r'_i.

As can be seen in Figure 5, the error rates between both methods over all four budget distributions are not significantly different (at most 0.32%). Table 7.A makes clear that there are no statistically significant differences in errors across methods when the initial budget is low. Figure 5 also shows that the average cost of hierarchical models is significantly lower than non-hierarchical models (at most 7.3%). Table 7.B summarizes the statistical significance of these differences.

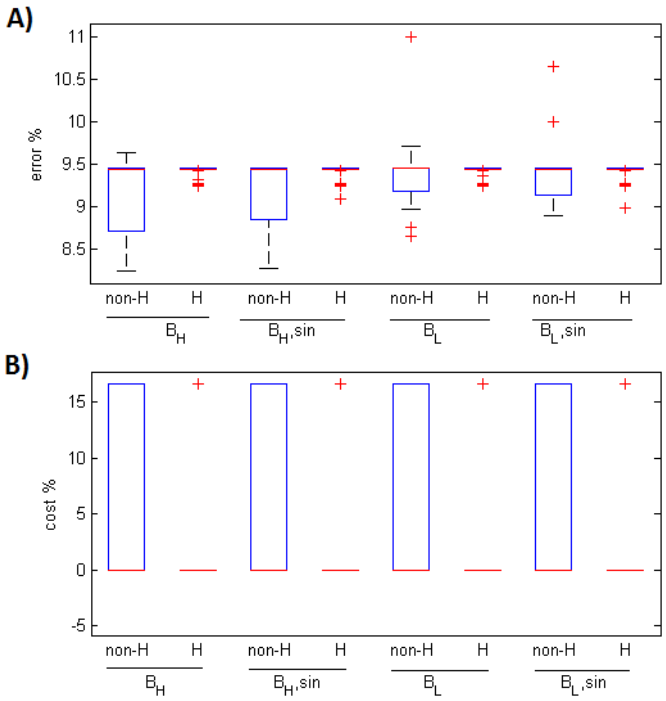

Fig. 5. Average error (A) and cost (B) on the test data for the non-hierarchical and the hierarchical models with different budget distributions. Statistical significance of these results are reported in Table 7.

Actual Data In these experiments, datasets from the UCI repository, as described in Section 3.2, are used to evaluate the two new proposed methods. As in Section 3.2, each dataset is divided into two equal training and testing portions. The budget feature is also included in the datasets when training the non-hierarchical models. Each experiment is repeated 30 times. For each iteration, a model is selected and tested on the test data four times, each time with a different budget distribution.

Non-hierarchical model training. The population size is set to 100, and for each datasets with k available sensors, models are trained for $200 \times k$ generations. The non-hierarchical models have access to the **B** feature during training. The rest of the settings are the same as they are reported in Section 3.2. After training, the selected model is tested on all four different budget distributions.

Hierarchical model training. The population size for each layer is set to 100, and for each dataset, each of the layers are trained for 200 generations. Models do not have access to the remaining budget level as an extra feature in training. The rest of model training is as described in Section 3.2. During testing, the dynamic tolerance parameter τ' is used.

Table 8. Error percentages of the methods on actual data considering different budget distributions.

	$\mathbf{B}_H$		$\mathbf{B}_{H,\sin}$		$\mathbf{B}_L$		$\mathbf{B}_{L,\sin}$	
datasets	non-H	H	non-H	H	non-H	H	non-H	H
DS_1	**22.03**	27.95	**21.97**	27.94	**23.29**	29.17	**23.30**	29.41
DS_2	32.14	**31.13**	32.19	**31.08**	32.51	**31.21**	32.54	**31.16**
DS_3	**99.61**	99.67	**99.61**	99.68	**99.61**	99.66	**99.61**	99.67
DS_4	**39.63**	43.18	**39.71**	43.19	43.79	**43.03**	43.72	**43.01**
DS_5	**40.59**	49.50	**40.64**	49.48	**42.26**	49.51	**42.32**	49.49
DS_6	**99.81**	101.2	**99.91**	101.2	**99.94**	101.2	**99.91**	101.2
DS_7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
DS_8	**36.55**	42.50	**36.55**	42.45	**36.89**	42.49	**36.93**	42.50
DS_9	11.60	**11.57**	11.60	**11.57**	11.60	**11.57**	11.60	**11.57**
DS_{10}	37.83	**37.78**	37.82	**37.77**	**37.83**	37.91	**37.83**	37.84

Table 9. p-values for actual data comparing the error of the hierarchical and non-hierarchical method considering different budget distributions.

datasets	$\mathbf{B}_H$	$\mathbf{B}_{H,\sin}$	$\mathbf{B}_L$	$\mathbf{B}_{L,\sin}$
DS_1	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
DS_2	0.005	0.003	0.012	0.008
DS_3	0.310	0.314	0.312	0.313
DS_4	0.002	0.002	0.486	0.511
DS_5	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
DS_6	0.003	0.003	0.004	0.007
DS_7	1.00	1.00	1.00	1.00
DS_8	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
DS_9	0.680	0.689	0.709	0.695
DS_{10}	0.950	0.937	0.899	0.990

Table 10. Cost percentages of the methods on actual data considering different budget distributions.

	$\mathbf{B}_H$		$\mathbf{B}_{H,\sin}$		$\mathbf{B}_L$		$\mathbf{B}_{L,\sin}$	
datasets	non-H	H	non-H	H	non-H	H	non-H	H
DS_1	15.23	**8.84**	15.23	**8.80**	15.23	**8.30**	15.23	**8.27**
DS_2	9.08	**1.84**	9.09	**1.85**	9.08	**1.74**	9.09	**1.76**
DS_3	0.031	**0.018**	0.030	**0.017**	0.031	**0.017**	0.031	**0.017**
DS_4	30.21	**0.631**	30.23	**0.631**	30.06	**0.587**	30.04	**0.597**
DS_5	46.93	**3.17**	46.93	**3.18**	46.93	**3.06**	46.93	**3.04**
DS_6	0.412	**0.001**	0.412	**0.001**	0.412	**0.001**	0.412	**0.001**
DS_7	**1.02**	2.73	**1.02**	2.73	**1.02**	2.65	**1.02**	2.67
DS_8	0.176	**0.00**	0.176	**0.00**	0.176	**0.00**	0.176	**0.00**
DS_9	0.076	**0.00**	0.076	**0.00**	0.076	**0.00**	0.076	**0.00**
DS_{10}	8.00	**1.39**	8.00	**2.21**	8.00	**2.06**	7.99	**2.06**

Results on Actual Data Table 8 reports the average prediction errors for all of the actual datasets, for both the hierarchical and non-hierarchical methods. The statistical significance of the difference in errors between these two methods is reported in Table 9. The average cost of models trained on the actual datasets for the hierarchical and non-hierarchical methods can be seen in Table 10. Table 11 reports the statistical signficance of the cost differences between the two methods.

Table 11. P-values for actual data comparing the cost of the hierarchical and non-hierarchical method considering different budget distributions

datasets	$\mathbf{B}_H$	$\mathbf{B}_{H,\sin}$	$\mathbf{B}_L$	$\mathbf{B}_{L,\sin}$
DS_1	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
DS_2	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	0.00086
DS_3	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
DS_4	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
DS_5	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
DS_6	0.024	0.024	0.025	0.024
DS_7	0.513	0.515	0.618	0.585
DS_8	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
DS_9	0.164	0.164	0.164	0.164
DS_{10}	0.019	0.019	0.016	0.016

5 Discussion

In this Section we reflect on the reason for and meaning of our quantitative results described in Sections 3 (basic results) and 4.3 (results with a dynamic budget). Overall, our experimental results show that in any case, sampling costs of models generated by the hierarchical method are significantly lower than models generated by the non-hierarchical method. Non-hierarchical modes use more expensive sensors with higher frequency. Results also show that hierarchical models achieve similar error rates as those incurred by non-hierarchical models as datasets grow larger, though non-hierarchical models do achieve lower error for small datasets especially when a dynamic budget is considered. Also notable is that results in Section 4.3 suggest that the hierarchical method obtains models that are more effectively sensitive to noise than models generated by the non-hierarchical method, when the budget level shrinks from high to low.

5.1 Basic Results with a Static Budget

The results presented in Section 3 suggest that the hierarchical method is better at balancing cost and accuracy than the non-hierarchical approach. We believe this is because meaningful sensor control conditions for managing cost are complex and require considerable computational effort to be discovered. Using hand-tuned prediction uncertainty to drive sensor control is more effective. As mentioned in Section 2.2, in these experiments a basic genetic programming approach was deployed. We anticipate that if we were to use a more powerful underlying GP approach, the error of both hierarchical and non-hierarchical models would be reduced.

Synthesized Data The hierarchical method achieved significantly better accuracy and significantly lower cost than the non-hierarchical using synthesized data.

Average error. As can be seen in Figure 3, the hierarchical method achieved significantly better accuracy than the non-hierarchical method for $\tau < 0.4$. In general, results show that higher tolerance allows the algorithm to accept more

uncertainty in the prediction and rely on less expensive sensors which are less accurate. This avoids the use of more expensive sensors, but causes average error to rise. A tolerance of $\tau < 0.4$ is apparently the threshold where average error in the hierarchical method exceeds that of the non-hierarchical method, when analysing results for different values of tolerance included in this study.

Average cost. Results reported in Figure 3 show that the hierarchical method significantly outperforms the non-hierarchical method with regard to cost on this dataset, even when tolerance is low. This suggests that the use of variance in ensemble predictions to serve as a proxy for prediction uncertainty is not easy to learn, and serves as a good mechanism for control. Results suggest that $\tau = 0.1$ is a "sweet spot" for balancing cost and accuracy, though the value could be increased or decreased if greater frugality or accuracy were needed, respectively.

Actual Data For testing on actual data, we fixed $\tau = 0.1$ due to results on synthetic data demonstrating a good balance between cost and accuracy with this tolerance level.

Average error. Table 6 shows that the average error of the hierarchical and the non-hierarchical methods were not significantly different, except for datasets DS_1, DS_2 and DS_4 where the latter method achieves better prediction accuracy. This is probably due to the characteristics of these datasets, where the difference between the least prediction variances $v_{\mathbf{r}_i}^{(\mathbf{g})}$s and the greatest ones is large. The majority of sensors in these datasets are not informative but have low costs and the remaining sensors are more informative but come with higher costs. Thus, lower levels of the hierarchy "struggle" compared to upper ones in terms of accuracy. Nevertheless, accuracy rate with the hierarchical method is still competitive even in these cases, and cost reduction is significant. Also, it can be seen that as the size of the datasets grows, the difference between the error rate of the non-hierarchical and the hierarchical methods decreases, and in the three largest datasets the hierarchical method also achieves lower error rates.

Average cost. The hierarchical method achieved significantly lower cost than the non-hierarchical method on all of the real world datasets, as shown in Table 6, except for DS_3 and DS_6. As represented in Table 5, in these two datasets, just a small subset of sensors are relatively informative. Since the tolerance parameter for the hierarchical method is low, the hierarchical method employs more informative sensors. Taken together, results shown in Table 6 clearly indicates an advantage of the hierarchical method for balancing cost and accuracy.

5.2 Results with a Dynamic Budget

Now we consider the results provided in Section 4.3 on synthesized data and ten actual datasets when a possibly dynamic budget is taken in to account. The results obtained from the experiments in Section 4.3 suggest that the hierarchical method is more successful in balancing cost and prediction accuracy compared to the non-hierarchical method as the number of observations in the dataset grows. The hierarchical method produces much less costly models, which results

in less noise accumulating on the sensors. This conservation can be crucial to reduce system down time if longer time periods are required to replenish the budget.

It also can be seen in Table 10 that the hierarchical method produces models that adapt their sensor sampling strategy based on the current budget, since they reduce their costs when the budget level goes from high to low. In contrast, models produced by the non-hierarchical method do not change their cost when the models are presented with the low budget level. The reason why the non-hierarchical method does not make use of the remaining budget to change its behaviour is at the moment unclear. Models produced by the hierarchical method incur lower cost when the budget level is low than when the budget level is high. The dynamic tolerance parameter employed in the hierarchical method successfully balances the cost of the hierarchical method to the remaining budget considering the results reported in Figure 5 and Table 10.

Synthesized Data The results using synthesized data (Figure 5) demonstrate that the hierarchical method adapts to the changing budget better than the non-hierarchical method. The hierarchical models obtain about the same prediction accuracy as the non-hierarchical models, but with significantly lower cost.

Average error. As can be seen in Figure 5, the difference between the error rate of the hierarchical and non-hierarchical methods is low. The p-values reported in Table 7a suggest that this difference is insignificant when the budget level is low. When the hierarchical method must work within the confines of a low budget, it produces models that only infrequently poll high-cost sensors. In this manner, the hierarchical models keep the overall cost of prediction low, which results in a higher remaining budget for the remainder of the period during which predictions are requested. Keeping the budget high in turn results in sensor readings with higher accuracy.

The non-hierarchical models however maintain high accuracy by polling the more accurate sensors more frequently, which incurs a higher cost. This approach eventually causes prediction accuracy to suffer, since it increases the noise in sensor readings as the budget decreases. As can be seen in Table 7a, when the budget is low, the non-hierarchical models are not able to maintain their superior accuracy rates.

Average cost. Figure 5 shows that the hierarchical method generates significantly lower cost models compared to the non-hierarchical method, for all of the budget distributions considered. The hierarchical method keeps cost low in two ways. First, the model hierarchy is constrained by design in the sensors it samples, depending on the hierarchy level. Second, the certainty threshold can be tuned to become more restrictive as the budget drops.

If the budget is low, then the dynamic tolerance parameter forces the hierarchical model to tolerate more uncertainty in its predictions. In contrast, the non-hierarchical method generates models that tend to use more expensive sensors more frequently. The results shown in Table 7b support this claim. As in the basic setting with a static budget, the non-hierarchical method has difficul-

ties discovering the proper conditions for sampling various sensors, that can be manually tuned into the threshold parameter for the hierarchical method.

Actual Data With actual data, the hierarchical models are able to adapt to budget fluctuations better than the non-hierarchical models, but the non-hierarchical models achieve higher accuracy on smaller datasets.

Average error. Table 8 shows that the non-hierarchical method achieves better error rates compared to the hierarchical method on datasets for which most of the sensors are non informative, but a few are with very high cost. For example, as shown in Table 5, in dataset DS_8 with 19 sensors, the difference in accuracy and cost between the most informative and the least informative sensors is on the order of 10^3. The most informative sensor is able to explain 45.42% of the output variance while the least informative sensor explains just 0.05% of the output variance. The difference in accuracy and cost between the other sensors and the most informative sensor are almost the same, except for the four most informative sensors. In this case, the non-hierarchical method uses the informative sensors in order to achieve high accuracy, whereas the hierarchical method tries to find a model with less cost. The order of difference for the most informative and the least informative sensors in DS_3 and DS_9 is high, but this difference order reduces for the other sensors in those datasets and also the most informative sensor in these datasets are not that informative (25.89% and 18.97% respectively). For the rest of the datasets, the order of difference is not that high compared to DS_8.

Also, in the non-hierarchical method, a model and its descendants could have been refined through the entire training period, whereas in the hierarchical method the training effort is distributed among the hieararchy layers. This means that individual model lineages have much less time to be improved, compared to non-hierarchical model lineages. Even so, Tables 8 and 10 show that when the budget level drops from high to low, the cost of the hierarchical models drops further than the drop observed in the non-hierarchical models. In this way the budget is better conserved during deployment and sensor noise is ameliorated.

As can be seen in Table 8, the error rates of the non-hierarchical models grow more than the error rates of the hierarchical models when the budget level decreases. Moreover, as the sizes of the datasets grow (from DS_1 to DS_{10}), the differences between the error rates of the hierarchical and non-hierarchical models decrease. For the two largest data sets (DS_9 and DS_{10}), the average error rate of the hierarchical and the non-hierarchical models are not statistically significantly different.

Average cost. As can be seen in Table 10, the hierarchical models always achieve lower cost compared to the non-hierarchical models, except for DS_7. Data set DS_7 has five sensors which all explain a small amount of the outcome variance. Hierarchical models attempt to predict the outcome using these sensor readings, but the non-hierarchical models rarely employ any of those sensors since they have so little predictive value. Instead the hierarchical models use the training effort to find a constant that predicts the outcome, at no cost.

Otherwise, as seen in Table 10, hierarchical models generally reduce cost more than non-hierarchical models when initial costs go from high to low, as is the case for synthesized data and for the same reasons (or so we hypothesize).

6 Conclusion and future work

Wireless sensor networks often face a trade-off between measurement accuracy and the cost of sensor sampling. In networks supporting multiple sensor types, it is therefore desirable to develop cost-sensitive control algorithms that sample more expensive sensors only when necessary. In this Chapter, a hierarchical method is proposed where GP solutions are sorted in a hierarchy of layers based on the cost of the sensors they use. Switching to the next more expensive layer takes place only if the prediction variance indicates uncertainty at lower layers. We compare this method to a non-hierarchical GP method where cost is treated as an additional optimization objective in fitness selection. In experiments using a synthesized dataset and ten real datasets, the hierarchical method is shown to have significantly lower prediction costs than the non-hierarchical method. As the datasets grow larger and more complex, competitive and sometimes lower error rates are achieved by the hierarchical method. In a second set of experiments, we consider a more sophisticated setting where the current budget level (e.g. power levels) is available as a sensor reading, and lower budgets have a direct impact on sensor accuracy. The non-hierarchical method in this case uses the remaining budget level in order to induce a model that adapts to the budget and sensor noise. In the hierarchical method the remaining budget is used in the decision to switch between layers. The results from experiments show that when the methods are altered to dynamically tune the balance of cost and accuracy based on available energy and budget in the presence of noise, the hierarchical method achieves significantly lower cost. As datasets grow larger, the hierarchical method achieves a competitive error rate as compared to the non-hierarchical method. Future work includes a consideration of methods for online learning to support adaptation of control algorithms to particular deployments, and the application of hierarchical control algorithms in real wireless sensor network deployments.

Acknowledgements

This work was supported in part by the NSF awards PECASE-0953837 and INSPIRE-1344227.

References

1. github code public repository. `http://git.io/vfmGB`, accessed: 2015-04-18
2. UCI machine learning repository. `http://archive.ics.uci.edu/ml/datasets.html`, accessed: 2015-02-03
3. Snowcloud: A Complete System for Snow Hydrology Research. ACM (2013)

4. Alippi, C., Anastasi, G., Galperti, C., Mancini, F., Roveri, M.: Adaptive sampling for energy conservation in wireless sensor networks for snow monitoring applications. In: IEEE 4th International Conference on Mobile Adhoc and Sensor Systems, MASS 2007, 8-11 October 2007, Pisa, Italy. pp. 1–6 (2007)
5. Bair, E.H., Davis, R.E., Finnegan, D.C., LeWinter, A.L., Guttmann, E., Dozier, J.: Can we estimate precipitation rate during snowfall using a scanning terrestrial lidar? In: International Snow Science Workshop. pp. 923–929. Anchorage, AK (2012)
6. Brockhoff, Dimo, Zitzler, Eckart: Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In: Parallel Problem Solving from Nature-PPSN IX, pp. 533–542. Springer (2006)
7. Buckingham, D., Skalka, C., Bongard, J.: Inductive learning of snowpack distribution models for improved estimation of areal snow water equivalent. Journal of Hydrology (2015), accepted for Publication.
8. Donoho, D.L.: Compressed sensing. IEEE Transactions on Information Theory 52(4), 1289–1306 (2006)
9. Hall, D., Llinas, J.: Multisensor data fusion. CRC press (2001)
10. Hornby, G.: ALPS: the age-layered population structure for reducing the problem of premature convergence. In: Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006. pp. 815–822 (2006)
11. de Jong, E.D., Pollack, J.B.: Multi-objective methods for tree size control. Genetic Programming and Evolvable Machines 4(3), 211–233 (2003)
12. Kim, D.: Structural risk minimization on decision trees using an evolutionary multiobjective optimization. In: Genetic Programming, pp. 338–348. Springer (2004)
13. Koushanfar, F., Slijepcevic, S., Potkonjak, M., Sangiovanni-Vincentelli, A.: Error-tolerant multimodal sensor fusion. In: IEEE CAS Workshop on Wireless Communication and Networking. pp. 5–6 (2002)
14. Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection, vol. 1. MIT press (1992)
15. Krause, A., Guestrin, C., Gupta, A., Kleinberg, J.: Near-optimal sensor placements: Maximizing information while minimizing communication cost. In: Proceedings of the 5th international conference on Information processing in sensor networks. pp. 2–10. ACM (2006)
16. Maleki, S., Pandharipande, A., Leus, G.: Two-stage spectrum sensing for cognitive radios. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2010, 14-19 March 2010, Sheraton Dallas Hotel, Dallas, Texas, USA. pp. 2946–2949 (2010)
17. Malloy, M.L., Nowak, R.D.: Near-optimal adaptive compressed sensing. IEEE Transactions on Information Theory 60(7), 4001–4012 (2014)
18. Martinelli, A.: Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. IEEE Transactions on Robotics 28(1), 44–60 (2012)
19. Papadimitriou, C., Beck, J.L., Au, S.K.: Entropy-based optimal sensor location for structural model updating. Journal of Vibration and Control 6(5), 781–800 (2000)
20. Pohl, C., Genderen, J.V.: Review article multisensor image fusion in remote sensing: concepts, methods and applications. International Journal of Remote Sensing 19(5), 823–854 (1998)
21. Ren, H., Rank, D., Merdes, M., Stallkamp, J., Kazanzides, P.: Multisensor data fusion in an integrated tracking system for endoscopic surgery. IEEE Transactions on Information Technology in Biomedicine 16(1), 106–111 (2012)

22. Schmidt, M., Lipson, H.: Age-fitness pareto optimization. In: Riolo, R., McConaghy, T., Vladislavleva, E. (eds.) Genetic Programming Theory and Practice VIII, Genetic and Evolutionary Computation, vol. 8, pp. 129–146. Springer New York (2011), http://dx.doi.org/10.1007/978-1-4419-7747-2_8
23. Seung, H.S., Opper, M., Sompolinsky, H.: Query by committee. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. pp. 287–294. COLT '92, ACM, New York, NY, USA (1992), http://doi.acm.org/10.1145/130385.130417
24. Skalka, C., Frolik, J.: Snowcloud: A complete data gathering system for snow hydrology research. In: Real-World Wireless Sensor Networks, pp. 3–14. Springer (2014)
25. Smith, D., Singh, S.: Approaches to multisensor data fusion in target tracking: A survey. IEEE Trans. Knowl. Data Eng. 18(12), 1696–1710 (2006)
26. Tabari, H., Marofi, S., Abyaneh, H.Z., Sharifi, M.R.: Comparison of artificial neural network and combined models in estimating spatial distribution of snow depth and snow water equivalent in Samsami basin of Iran. Neural Comput. Appl. 19(4), 625–635 (2010)
27. Tappeiner, U., Tappeiner, G., Aschenwald, J., Tasser, E., Ostendorf, B.: GIS-based modelling of spatial pattern of snow cover duration in an alpine area. Ecol. Model. 138, 265–275 (2001)
28. Waltz, E.L., Buede, D.M.: Data fusion and decision support for command and control. IEEE Transactions on Systems, Man, and Cybernetics 16(6), 865–879 (1986)
29. Wang, D., Ahmadi, H., Abdelzaher, T.F., Chenji, H., Stoleru, R., Aggarwal, C.C.: Optimizing quality-of-information in cost-sensitive sensor data fusion. In: Distributed Computing in Sensor Systems, 7th IEEE International Conference and Workshops, DCOSS 2011, Barcelona, Spain, 27-29 June, 2011, Proceedings. pp. 1–8 (2011)
30. Willett, R., Martin, A., Nowak, R.: Backcasting: adaptive sampling for sensor networks. In: Proceedings of the Third International Symposium on Information Processing in Sensor Networks, IPSN 2004, Berkeley, California, USA, April 26-27, 2004. pp. 124–133 (2004)
31. Yick, J., Mukherjee, B., Ghosal, D.: Wireless sensor network survey. Comput. Netw. 52(12), 2292–2330 (2008)
32. Zhao, H.: A multi-objective genetic programming approach to developing pareto optimal decision trees. Decision Support Systems 43(3), 809–826 (2007)
33. Zowj, A.Y., Bongard, J.C., Skalka, C.: A genetic programming approach to cost-sensitive control in resource constrained sensor systems. In: Proceedings of the Genetic and Evolutionary Computation Conference,GECCO 2015, Madrid, Spain, July 11-15, 2015. pp. 1295–1302 (2015)