A Genetic Programming Approach to Cost-Sensitive Control in Resource Constrained Sensor Systems

Afsoon Yousefi Zowj
Dept. of Computer Science
University of Vermont
ayousef1@uvm.edu

Josh C Bongard
Dept. of Computer Science
University of Vermont
jbongard@uvm.edu

Christian Skalka
Dept. of Computer Science
University of Vermont
skalka@cs.uvm.edu

ABSTRACT

Resource constrained sensor systems are an increasingly attractive option in a variety of environmental monitoring domains, due to continued improvements in sensor technology. However, sensors for the same measurement application can differ in terms of cost and accuracy, while fluctuations in environmental conditions can impact both application requirements and available energy. This raises the problem of automatically controlling heterogeneous sensor suites in resource constrained sensor system applications, in a manner that balances cost and accuracy of available sensors. We present a method that employs a hierarchy of model ensembles trained by genetic programming (GP): if model ensembles that poll low-cost sensors exhibit too much prediction uncertainty, they automatically transfer the burden of prediction to other GP-trained model ensembles that poll more expensive and accurate sensors. We show that, for increasingly challenging datasets, this hierarchical approach makes predictions with equivalent accuracy yet lower cost than a similar yet non-hierarchical method in which a single GP-generated model determines which sensors to poll at any given time. Our results thus show that a hierarchy of GP-trained ensembles can serve as a control algorithm for heterogeneous sensor suites in resource constrained sensor system applications that balances cost and accuracy.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence

Keywords

Genetic Programming, Resource Constrained Sensor Systems, Cost-Sensitive Control, Sensor Fusion

1. INTRODUCTION

Resource constrained sensor systems (RCSS) such as Wireless Sensor Networks have revolutionized environmental monitoring by combining low cost with flexibility in sensor capabilities [29]. They have been used in diverse environmental monitoring applications and continue to be adapted in new fields. Because RCSS are often, even typically, deployed in remote locations, and thus rely on combinations of battery power and energy harvesting, a major challenge in RCSS 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 [28]. In this paper, we propose new sensor control algorithms for RCSS with heterogeneous sensor suites that balance cost and accuracy, obtained using genetic programming (GP) techniques.

By "heterogeneous sensor suite", we mean RCSS 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 [28]. 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 [28], 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) [22]. We envision extending Snowcloud to incorporate ground based light detection and ranging (LIDAR) scanning [4] 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 3.4 and described graphically in Figure 2. Note that while the Snowcloud system is an intended application of this scheme, it

GECCO '15, July 11 - 15, 2015, Madrid, Spain

DOI: http://dx.doi.org/10.1145/2739480.2754751

can be generalized to any RCSS 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 [21]. To obtain predictive models themselves, in this work we use genetic programming (GP). 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 nonlinearities [24], while the spatial distribution of SWE is nonlinear because it is influenced simultaneously by various forcing effects [25]. Nonlinear predictors are therefore desirable. Furthermore, recent results [6] show that GP has advantages over other approaches (such as C4.5) due to associated techniques for preventing overfitting, e.g. treating model size minimization as an objective [11]. 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 [6]. 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 [5]. We therefore hypothesize that a hierarchical approach will outperform a non-hierarchical approach in settings where multiple sensors with differing predictive abilities, and we explore this comparison in our experiments.

2. RELATED WORK

Previous work on adaptive sampling [28] has aimed to reduce sampling rates in RCSS applications to balance sensor cost and accuracy. In particular, Alippi *et al.* [3] 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 [15]. 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 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.* [13] have used a probabilistic method to predict the communication cost for a given deployment topology. Papadimitriou *et al.* [17] 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 [26, 9]. This method has been widely used, for example in visual monitoring [16, 18] and target tracking [19, 23]. 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 [12]. However, we are not concerned with fault tolerance, but strictly between selecting sensors from heterogeneous suites. Willett *et al.* [28] 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 [14]. 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.* [27]. 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 sensor control in RCSS during deployments.

Outside of the adaptive sampling and sensor fusion fields, multi-objective optimization has been used for cost-sensitive modeling. For example Kim [11] set error as one objective and tree size as another, as we do here. Zhao [30] 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.

3. 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].

3.1 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.

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.

3.2 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. Here, inputs are sensor values and 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 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^{train} and d^{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^{train}, i.e.,

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

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, the current readings of the sensors 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 data set:

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

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 was incorporated into the fitness objectives during the optimization process [7].

Age: We employed the Age-Fitness Pareto Optimization (AFPO) method [20], 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 an 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.

3.3 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. Successful solutions may evolve which only visit nodes containing references to expensive sensors—which are then polled—if less expensive sensors report certain combinations of values that signal these sensors are unlikely to predict well given the current circumstances. $e_{t_j}^{\text{train}}$ and $c_{t_j}^{\text{train}} * \text{age} * \text{size}$ are employed as the two main objectives in the optimization process.

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 is going to be used. It is assumed here that s_1 tends to make 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 sen-

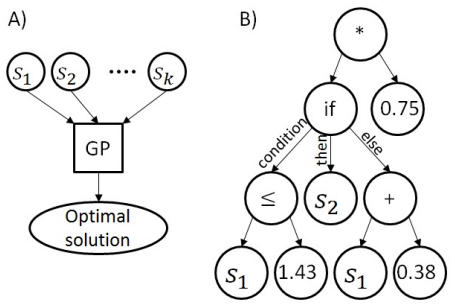

Figure 1:A)Non-hierarchical framework.B)A non-hierarchical sample solution.

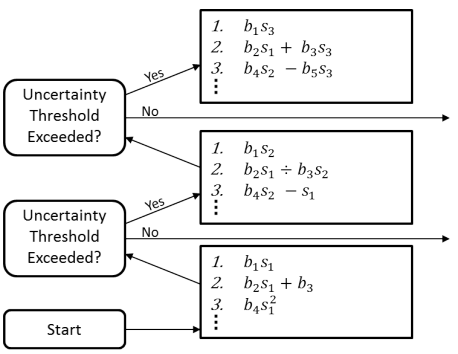

Figure 2: Hierarchical framework.

sors. 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.

3.4 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. First, 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 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 [21]: 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 somewhat 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

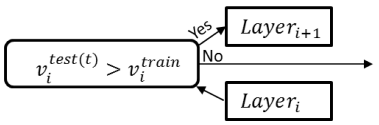

Figure 3: Using the difference between training data prediction variance and test data prediction variance as the condition for switching between model layers.

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 this paper, the variance is considered as a proxy for entropy, but any other entropy related metric could be used instead. Figure 3 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 (1) demonstrates how this parameter is used to determine which level should be activated for prediction.

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

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 ensures that the conditional is only true when the prediction variance on the testing data is greater than the prediction variance on the training data which has a high probability of occurring. Thus, the method tends to extract the predictions from the solutions on the uppermost layer. The latter ensures that the conditional is only true whenever the variance on the testing data is finite, which is always true. In this case, the first layer always provides the prediction. 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.

Table 1: Available sensors and their features.

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

4. 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.

4.1 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., $b_{i,j}$s, were randomly selected in the range $[0, 1]$. Then, random numbers were generated for $g^{(t)}$, 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 were repeated 40 times.

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 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.

Hierarchical setup. The population size for each layer is 100 and each layer was trained for 100 generations, for the sake of fairness in comparisons. 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 $\tau = 0.0, 0.1, 0.2, 0.4, 0.6, 0.8$.

4.1.1 Results on Synthesized Data

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 4, 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

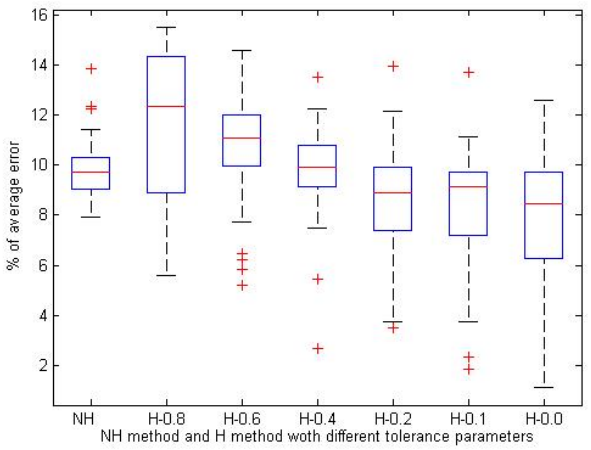

Figure 4:Average error on the test data for the non-hierarchical and the hierarchical methods with different tolerance parameters. Statistical significance of these results are represented in Table 2.A)

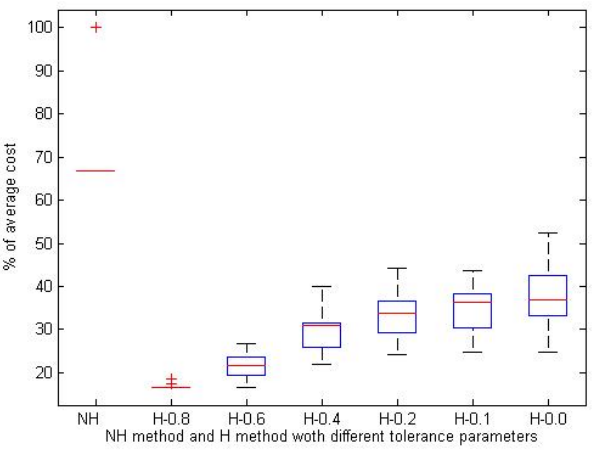

Figure 5:The average cost on the test data for the non-hierarchical and the hierarchical methods with different tolerance parameters. Statistical significance of these results are represented in Table 2.B)

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	$\tau = 0.2$	$\ll 0.001$
$\tau = 0.1$	$\ll 0.001$	$\tau = 0.1$	$\ll 0.001$
$\tau = 0.0$	$\ll 0.001$	$\tau = 0.0$	$\ll 0.001$

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 $c^{\text{test}}_{t_j}$. The average cost of the hierarchical method is the average of $c^{\text{test}}_{L_i}$, when 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 5 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.

4.2 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 treat the individual features as individual sensors. Each experiment in this section were repeated 30 times.

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

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 $v^{(\mathbf{g})}_{\mathbf{r}_i}$ for all 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, in each dataset.

DS No.	$\min v^{(\mathbf{g})}_{\mathbf{r}_i}$	$\max v^{(\mathbf{g})}_{\mathbf{r}_i}$
DS_1	0.1766	0.6918
DS_2	0.0307	0.5441
DS_3	0.0002	0.2578
DS_4	0.0076	0.7911
DS_5	0.0112	0.2478
DS_6	0.000	0.096
DS_7	0.0157	0.1527
DS_8	0.0005	0.4542
DS_9	0.0001	0.1897
DS_{10}	0.0037	0.0263

Table 6: Average error percentages and the corresponding P-values for the hierarchical and the non-hierarchical methods.

DS No.	NH: error %	H: error %	P-value
DS_1	20.85	25.81	$\ll 0.001$
DS_2	25.90	28.93	$\ll 0.001$
DS_3	126.49	202.12	0.565
DS_4	29.19	36.70	$\ll 0.001$
DS_5	35.29	39.68	0.393
DS_6	110.63	111.08	0.223
DS_7	0.00	0.00	0.082
DS_8	37.59	28.65	0.194
DS_9	10.79	10.67	0.197
DS_{10}	32.11	29.88	0.423

Table 7: Average cost percentages and the corresponding P-values for the hierarchical and the non-hierarchical methods.

DS No.	NH: cost %	H: cost %	P-value
DS_1	38.89	12.33	0.022
DS_2	23.18	1.26	$\ll 0.001$
DS_3	6.83	15.81	$\ll 0.001$
DS_4	32.90	4.18	$\ll 0.001$
DS_5	53.63	28.63	0.004
DS_6	0.00	0.98	0.040
DS_7	11.58	7.35	0.005
DS_8	2.55	0.00	$\ll 0.001$
DS_9	0.02	0.00	$\ll 0.001$
DS_{10}	20.62	3.88	0.009

Non-hierarchical setup. The population size is 200 and for each dataset with k features, it is trained for $200 * k$ 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, each layer was trained for 200 generation. 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 conductedof for tolerance parameter $\tau = 0.1$. This value is selected based on the results in 4.1 and will be discussed in more detail in Section 5.1.

4.2.1 Results on Actual Data

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 the datasets as a percentage of error. It can be seen that for some 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 generally, 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 4.1.1, the average cost is represented as the percentage of the maximum possible cost. Table 7 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.

5. DISCUSSION

Our results in all experiments 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. Furthermore, our results show that the latter approach better supports dynamic adaptation to changes in available energy, through modulation of tolerance. The non-hierarchical approach cannot adapt to such changes without retraining from scratch, or aggressive online learning. As mentioned in Section 3.2, 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.

In the remainder of this Section we discuss results as they pertain specifically to experiments with synthesized and actual data.

5.1 Synthesized Data

Average error. As can be seen in Figure 4, 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.

Average cost. Results reported in Figure 5 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 in Figures 4 and 5 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.

5.2 Actual Data

For testing on actual data, we fixed $\tau = 0.1$ due to results on synthetic data demonstrating a nice 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 $v^{(\mathbf{g})}_{\mathbf{r}_i}$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 informative enough but come with very 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 the real world datasets, as shown in Table 7, 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 Tables

6 and 7 clearly indicate an advantage of the hierarchical method for balancing cost and accuracy.

6. CONCLUSION AND FUTURE WORK

All resource constrained sensor systems have to 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 paper, 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 bigger and more complex, competitive and sometimes lower error rates are achieved by the hierarchical method. Future work includes consideration of how to dynamically tune the balance of cost and accuracy based on available energy and budget. Other directions for future work include methods for online learning to support adaptation of control algorithms to particular deployments, and application of hierarchical control algorithms in real resource constrained sensor system deployments.

Acknowledgements

This work was supported in part by the NSF award PECASE-0953837 and DARPA award MSEE-W911NF-11-1-0076.

7. 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] C. Alippi, G. Anastasi, C. Galperti, F. Mancini, and M. Roveri. 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*, pages 1–6, 2007.

[4] E. H. Bair, R. E. Davis, D. C. Finnegan, A. L. LeWinter, E. Guttmann, and J. Dozier. Can we estimate precipitation rate during snowfall using a scanning terrestrial lidar? In *International Snow Science Workshop*, pages 923–929, Anchorage, AK, 2012.

[5] Brockhoff, Dimo, Zitzler, and Eckart. Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In *Parallel Problem Solving from Nature-PPSN IX*, pages 533–542. Springer, 2006.

[6] D. Buckingham, C. Skalka, and J. Bongard. Inductive learning of snowpack distribution models for improved estimation of areal snow water equivalent. *Journal of Hydrology*, 2015. Accepted for Publication.

[7] E. D. de Jong and J. B. Pollack. Multi-objective methods for tree size control. *Genetic Programming and Evolvable Machines*, 4(3):211–233, 2003.

[8] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[9] D. Hall and J. Llinas. *Multisensor data fusion*. CRC press, 2001.

[10] G. Hornby. 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*, pages 815–822, 2006.

[11] D. Kim. Structural risk minimization on decision trees using an evolutionary multiobjective optimization. In *Genetic Programming*, pages 338–348. Springer, 2004.

[12] F. Koushanfar, S. Slijepcevic, M. Potkonjak, and A. Sangiovanni-Vincentelli. Error-tolerant multimodal sensor fusion. In *IEEE CAS Workshop on Wireless Communication and Networking*, pages 5–6, 2002.

[13] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 2–10. ACM, 2006.

[14] S. Maleki, A. Pandharipande, and G. Leus. 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*, pages 2946–2949, 2010.

[15] M. L. Malloy and R. D. Nowak. Near-optimal adaptive compressed sensing. *IEEE Transactions on Information Theory*, 60(7):4001–4012, 2014.

[16] A. Martinelli. 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.

[17] C. Papadimitriou, J. L. Beck, and S.-K. Au. Entropy-based optimal sensor location for structural model updating. *Journal of Vibration and Control*, 6(5):781–800, 2000.

[18] C. Pohl and J. V. Genderen. Review article multisensor image fusion in remote sensing: concepts, methods and applications. *International Journal of Remote Sensing*, 19(5):823–854, 1998.

[19] H. Ren, D. Rank, M. Merdes, J. Stallkamp, and P. Kazanzides. Multisensor data fusion in an integrated tracking system for endoscopic surgery. *IEEE Transactions on Information Technology in Biomedicine*, 16(1):106–111, 2012.

[20] M. Schmidt and H. Lipson. Age-fitness pareto optimization. In R. Riolo, T. McConaghy, and E. Vladislavleva, editors, *Genetic Programming Theory and Practice VIII*, volume 8 of *Genetic and Evolutionary Computation*, pages 129–146. Springer New York, 2011.

[21] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 287–294, New York, NY, USA, 1992. ACM.

[22] C. Skalka and J. Frolik. Snowcloud: A complete data gathering system for snow hydrology research. In *Real-World Wireless Sensor Networks*, pages 3–14. Springer, 2014.

[23] D. Smith and S. Singh. Approaches to multisensor data fusion in target tracking: A survey. *IEEE Trans. Knowl. Data Eng.*, 18(12):1696–1710, 2006.

[24] H. Tabari, S. Marofi, H. Z. Abyaneh, and M. R. Sharifi. 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.

[25] U. Tappeiner, G. Tappeiner, J. Aschenwald, E. Tasser, and B. Ostendorf. GIS-based modelling of spatial pattern of snow cover duration in an alpine area. *Ecol. Model.*, 138:265–275, 2001.

[26] E. L. Waltz and D. M. Buede. Data fusion and decision support for command and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(6):865–879, 1986.

[27] D. Wang, H. Ahmadi, T. F. Abdelzaher, H. Chenji, R. Stoleru, and C. C. Aggarwal. 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*, pages 1–8, 2011.

[28] R. Willett, A. Martin, and R. Nowak. 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*, pages 124–133, 2004.

[29] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Comput. Netw.*, 52(12):2292–2330, 2008.

[30] H. Zhao. A multi-objective genetic programming approach to developing pareto optimal decision trees. *Decision Support Systems*, 43(3):809–826, 2007.