Hierarchical Aggregate Classification with Limited Supervision for Data Reduction in Wireless Sensor Networks*

Lu Su, Yong Yang, Bolin Ding, Jing Gao, Tarek F. Abdelzaher, and Jiawei Han
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL, 61801, USA
{lusu2, yang25, bding3, jinggao3, zaher, hanj}@illinois.edu

Abstract

The main challenge of designing classification algorithms for sensor networks is the lack of labeled sensory data, due to the high cost of manual labeling in the harsh locales where a sensor network is normally deployed. Moreover, existing classification techniques dealing with limited label information are designed for centralized databases and thus cannot be directly applied to sensor networks, since delivering all the sensory data to the sink would cost enormous energy. To address these challenges, we propose a hierarchical aggregate classification (HAC) protocol which can reduce the amount of data sent by each node while achieving accurate classification in the face of insufficient label information. In this protocol, each sensor node locally makes cluster analysis and forwards only its decision to the parent node. The decisions are aggregated along the tree, and eventually the global agreement is achieved at the sink node. In addition, to control the tradeoff between the communication energy and the classification accuracy, we design an extended version of HAC, called the constrained hierarchical aggregate classification (cHAC) protocol. cHAC can achieve more accurate classification results compared with HAC, at the cost of more energy consumption. The advantages of our schemes are demonstrated through the experiments on not only synthetic data but also a real testbed.

*Research reported in this paper was sponsored by NSF under grant CNS 1040380 and by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

General Terms

Algorithms, Measurement, Experimentation

Keywords

Sensor Networks, Classification, Data Reduction

1 Introduction

Wireless sensor networks have been prototyped for many military and civilian applications. Thus far, a wide spectrum of sensing devices, ranging from simple thermistors to micropower impulse radars, have been integrated into existing sensor network platforms. Given the enormous amount of data collected by all kinds of sensing devices, techniques which can intelligently analyze and process sensory data have drawn significant attention.

Classification, which is the task of assigning objects (data) to one of several predefined categories (classes), is a pervasive topic in data analysis and processing [1, 2]. Its basic idea is to use a function (also called classifier) "learned" from a set of training data in which each object has feature values and a class label to determine the class labels of newly arrived data. For example, consider the task of classifying bird species based on their vocalizations. Suppose we are interested in two bird species: Song sparrow and American crow, and consider two vocalization features: call intensity and call duration[1]. After learning from the training set, the following target function can be derived: 1) low call intensity & short call duration → Song sparrow, and 2) high call intensity & long call duration → American crow. Suppose there is a bird with unknown species, we can judge which class it belongs to by mapping its feature values to the corresponding class based on the learnt function.

In the context of sensor networks, species recognition like the above example is a typical category of classification applications [3, 4, 5]. Classification also plays a key role in many other applications of sensor networks. In target

[1] We use these two features simply as an illustration, in practice bird species classification requires much more complicated features as in the experiment presented in Section 6.

surveillance and tracking, sensor networks should be able to classify different types of targets, such as cars, tanks, humans and animals [6, 7, 8]. In habitat monitoring, sensor networks may distinguish different behaviors of monitored species [9, 10, 11]. In environmental monitoring, it may be desired that the environmental (e.g., weather) conditions be classified based on their impact on humans, animals, or crops [12]. In health care or assisted living, an intelligent sensor network may automatically evaluate the health status of residents, and react when they are in danger [13, 14, 15].

However, existing classification techniques designed for sensor networks fail to take into account the fact that in many applications of sensor networks, the amount of labeled training data is usually small. The lack of labeled sensory data can be attributed to the remote, harsh, and sometimes even hostile locales where a sensor network is normally deployed as well as the continuous variation of the surveilled environment. In such scenarios, manually creating a large training set becomes extremely difficult. Without sufficiently large training set, the learned classifier may not be able to describe the characteristics of each class, and tends to make bad predictions on new data. In the area of data mining and machine learning, some techniques called semi-supervised classification have been proposed in order to deal with insufficient labels [1, 2]. However, these schemes assume centralized storage and computation, and cannot be directly applied in the context of sensor networks where data are distributed over a large number of sensors.

One possible way to apply centralized classification techniques is to transport all the sensory data to the sink for offline analysis. However, it has been revealed that wireless transmission of a bit can require over 1000 times more energy than a single 32-bit computation [16]. Thus, in designing an energy-scarce sensor network, it is desired that each node locally process and reduce the raw data it collects as opposed to forwarding them directly to the sink [17, 18], especially for some data-intensive applications such as audio or video based pattern recognition. This design philosophy has great challenges in traditional low-end sensing platforms such as Mica2 mote [19]. In these platforms, sensor nodes have limited processing power, memory, and energy and hence cannot support computation intensive algorithms. Recently, some powerful sensing systems [20, 21, 22, 23] are presented, making it feasible to place the classification task locally on individual nodes so that communication energy and interference can be significantly reduced.

Moreover, in sensor networks, it often happens that multiple sensor nodes detect the same events. Different sensor nodes, due to their inaccuracy (e.g., noise in the data) and heterogeneity (e.g., different types of sensors), usually observe the events from different but complementary views. Therefore, aggregating the outputs of individual nodes can often cancel out errors and reach a much more accurate result. However, aggregation of classification results is not an easy task in the absence of sufficient labeled data due to the lack of correspondence among the outputs of different nodes.

To address the above challenges, we propose a *hierarchical aggregate classification* (HAC for short) protocol for data reduction in sensor networks, which is built on a hierarchical tree topology where all nodes detect the same events. In order to overcome the obstacles presented by insufficient labels, we suggest that sensor nodes conduct cluster analysis, which groups data points only based on the similarity of their feature values without any training. The clustering results can provide useful constraints for the task of classification when the labeled data is insufficient, since the data that have similar feature values are usually more likely to share the same class label. To reduce the amount of data delivered to the sink, we let each node locally cluster events and report only its clustering result (also called *decision* in this paper) to the parent node instead of sending the raw data which are normally multi-dimensional numbers or audio/video files. The decisions are then aggregated along the tree through an efficient algorithm called *Decision-Aggregation* which can integrate the limited label information with the clustering results of multiple sensor nodes. Finally, the global consensus is reached at the sink node.

Additionally, to control the tradeoff between the communication energy and the classification accuracy, we design an extended version of HAC, called the *constrained hierarchical aggregate classification* (cHAC) protocol. cHAC can achieve more accurate classification results compared with HAC, at the cost of more energy consumption.

To demonstrate the advantages of our schemes, we conduct intensive experiments on not only synthetic data but also a real testbed. This testbed is a good example of high-end sensing system on which various data and computation intensive classification applications can be deployed. In our evaluation, we design two experiments on the testbed. The first one is to classify different bird species based on their vocalizations, and the second one is to predict the intensity of bird vocalizations as a function of different environmental parameters measured by the sensor nodes. Both the experimental and simulation results show that the proposed protocols can achieve accurate classification in the face of insufficient label information, and provide a flexible option to tune the tradeoff between energy and accuracy.

The rest of the paper is organized as follows. Section 2 introduces the general aggregation model. In Section 3, the Decision-Aggregation algorithm, which is the aggregation function used by the HAC and cHAC protocols at each nonleaf node, is presented. We explain how the HAC protocol utilizes this algorithm to aggregate the decisions along the tree in Section 4. In Section 5, the cHAC protocol and the procedures it invokes are presented. The proposed schemes are evaluated in Section 6. Then, we summarize the related work in Section 7, and conclude the paper in Section 8, respectively.

2 Aggregation Model

We consider an aggregation tree [24, 25] T rooted at the sink node (or base station), and denote the set of sensor nodes on this tree by $\mathcal{S}_T = \{s_i,\ i = 1,2,\ldots,n_T\}$. When an event takes place, all the nodes collect sensory readings about it[2]. Let $\mathcal{E} = \{e_i,\ i = 1,2,...,t\}$ denote the sequence of events

[2]We assume the aggregation tree is constructed on a set of nodes which are deployed in proximity. Thus, they can detect the same events and have similar readings.

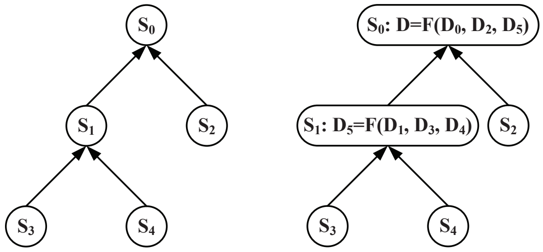


Figure 1. An example of decision aggregation

(sorted in chronological order) detected by the sensor nodes. Suppose only a small portion of the events are labeled, and our goal is to find out the labels of the rest events. The general idea of our solution is as follows. Based on its sensory readings, each node, say s_i, divides the events into different clusters through its local clustering algorithm. After that, s_i forwards the clustering result (referred to as s_i's *decision*, denoted by D_i) to its parent node. At each nonleaf node (including the sink node), the decisions of its children nodes, together with its own decision if it has one, are further aggregated. Figure 1 gives an illustrative example of decision aggregation. As can be seen, the nonleaf node s_1 aggregates the decisions of s_3 and s_4, together with its own decision. In this paper, we use function $\mathbf{F}(\cdot)$ to represent the operation of decision aggregation. Then, s_1 forwards the aggregated decision D_5 to the sink node s_0. At s_0, D_5 is further combined with s_0 and s_2's decisions. Finally, the global decision D is obtained. In the next section, we elaborate on the decision aggregation operation $\mathbf{F}(\cdot)$. Afterwards, we will disclose how the HAC and cHAC protocols invoke $\mathbf{F}(\cdot)$ to aggregate the decisions along the aggregation tree in Section 4 and Section 5, respectively.

3 Decision Aggregation

The decision aggregation function $\mathbf{F}(\cdot)$ takes the clustering decisions of multiple sensors as well as the label information as the input, and outputs a class label for each event, indicating which class the event belongs to. Although the clustering decisions do not give the concrete label assignment, they provide useful information for the classification task. $\mathbf{F}(\cdot)$ utilizes the labels from the few labeled events to guide the aggregation of clustering decisions such that a consolidated classification solution can be finally outputted. In this section, we first propose to model the decision aggregation problem as an optimization program over a bipartite graph. Then we present an effective solution and give performance analysis.

3.1 Belief Graph

Given a nonleaf node, suppose it receives n decisions from its children nodes. In each decision, the events in $\mathcal{E}$ are partitioned into m clusters[3]. Thus, we have totally $l = mn$ different clusters, denoted by $c_j,\ j = 1, 2, ..., l$. In this paper, we call these clusters the *input clusters* (iCluster for short) of a decision aggregation. On the other hand, the decision aggregation operation will output an aggregated decision also composed of m clusters, named as *output clusters* (oCluster for short).

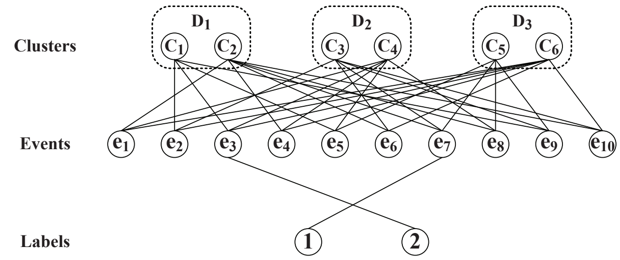


Figure 2. Belief graph of decision aggregation

In this paper, we represent the relationship between the events and the iClusters as a bipartite graph, which is referred to as *belief graph*. In belief graph, each iCluster links to the events it contains. Figure 2 demonstrates the belief graph of a decision aggregation. In this case, we suppose there are $t = 10$ events, which belong to $m = 2$ different classes. Each of the sensor nodes partitions these 10 events into $m = 2$ clusters based on its local clustering algorithm, and there are $n = 3$ decisions. Thus, we have totally $l = mn = 6$ different iClusters. Moreover, to integrate label information into the belief graph, we add one more set of vertices, which represent the labels of the events. In belief graph, the labeled events are connected to the corresponding label vertices. As shown in Fig. 2, event e_3 and e_7 are labeled, and thus link with label vertex 2 and 1, respectively.

3.2 Terminology

The belief graph can be summarized by a couple of affinity matrices:

- Clustering matrix $A = (a_{ij})_{t \times l}$, which links events and iClusters as follows:

$$a_{ij} = \begin{cases} 1 & \text{If } e_i \text{ is assigned to cluster } c_j. \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

- Groundtruth matrix $Z_{t\times m} = (\vec{z}_{1\cdot}, \vec{z}_{2\cdot}, \ldots, \vec{z}_{t\cdot})^T$, which relates events to the label information. For a labeled event e_i, its groundtruth vector is defined as:

$$z_{ik} = \begin{cases} 1 & \text{If } e_i\text{'s observed label is } k. \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

For each of the events without labels, we assign a zero vector $\vec{z}_{i\cdot} = \vec{0}$ to it.

Then, we define two sets of probability vectors that will work as the variables in the optimization problem formulated later in the next subsection:

- For an event e_i, Let L_i^e $(i = 1, 2, ..., t)$ denote the ID of the oCluster to which e_i is assigned, namely, the label of e_i. In our optimization problem, we aim at estimating the probability of e_i belonging to the k-th oCluster

[3]Most of the clustering models like K-means can control the number of clusters. We let the number of clusters equal the number of classes of the events, which is m.

($k = 1,2,...,m$), i.e., $\hat{P}(L_i^e = k|e_i)$. Thus, each event is associated with a m-dimensional probability vector:

$$\vec{x}_{i\cdot} = \{(x_{ik})|x_{ik} = \hat{P}(L_i^e = k|e_i),\ k = 1,2,...,m\} \quad (3)$$

Putting all the vectors together, we get a probability matrix $X_{t\times m} = (\vec{x}_{1\cdot}, \vec{x}_{2\cdot}, \ldots, \vec{x}_{t\cdot})^T$. After X is computed, we classify the i-th event into the k-th class if x_{ik} attains the maximum in $\vec{x}_{i\cdot}$.

- For an iCluster c_j, we also define a m-dimensional probability vector :

$$\vec{y}_{j\cdot} = \{(y_{jk})|y_{jk} = \hat{P}(L_j^c = k|c_j),\ k = 1,2,...,m\} \quad (4)$$

where L_j^c is the ID of an oCluster. In practice, $\hat{P}(L_j^c = k|c_j)$ implies the probability that the majority of the events contained in c_j are assigned to the k-th oCluster. In theory, it will serve as an auxiliary variable in the optimization problem. Correspondingly, the probability matrix for all the iClusters is $Y_{l\times m} = (\vec{y}_{1\cdot}, \vec{y}_{2\cdot}, \ldots, \vec{y}_{l\cdot})^T$.

In our design, there is a weight parameter associated with each decision. Initially, each decision is manually assigned a weight based on the properties (e.g., sensing capability, residual energy, location, etc) of the sensor node who makes this decision. The weight of each node represents the importance of the this node, and the nodes which can provide more accurate readings are assigned higher weights. The aggregated decision has a weight equal to the summation of the weights of input decisions. All the clusters within a decision have the same weight as the decision. In the rest of this paper, we use w_j to denote the weight of cluster c_j[4]. Finally, let $b_i = \sum_{k=1}^m z_{ik}$ be a flag variable indicating whether e_i is labeled or not.

3.3 Problem Formulation

With the notations defined previously, we now formulate the decision aggregation problem as the following optimization program:

$$\mathbf{P}: \min_{X,Y} \quad \sum_{i=1}^{t}\sum_{j=1}^{l} a_{ij}w_j||\vec{x}_{i\cdot} - \vec{y}_{j\cdot}||^2 + \alpha\sum_{i=1}^{t} b_i||\vec{x}_{i\cdot} - \vec{z}_{i\cdot}||^2 \quad (5)$$

$$\text{s.t.} \quad \vec{x}_{i\cdot} \geq \vec{0}, \quad |\vec{x}_{i\cdot}| = 1 \quad \text{for } i = 1,2,...,t \quad (6)$$

$$\vec{y}_{j\cdot} \geq \vec{0}, \quad |\vec{y}_{j\cdot}| = 1 \quad \text{for } j = 1,2,...,l \quad (7)$$

where $||.||$ and $|.|$ denote a vector's L2 and L1 norm respectively, and α is a predefined parameter. To achieve consensus among multiple clustering decisions, we aim at finding the optimal probability vectors of the event nodes ($\vec{x}_{i\cdot}$) and the cluster nodes ($\vec{y}_{j\cdot}$) that can minimize the disagreement over the belief graph, and in the meanwhile, comply with the label information. Specifically, the first term in the objective function (Eqn. (5)) ensures that an event has similar probability vector as the input cluster to which it belongs, namely, $\vec{x}_{i\cdot}$ should be close to $\vec{y}_{j\cdot}$ if event e_i is connected to cluster c_j in the belief graph (e.g., event e_3 and cluster c_1, c_4, c_6 in Fig. 2). The second term puts the constraint that a labeled

[4] Sometimes, we use w_i to denote the weight of decision D_i or node s_i, and hope this causes no confusion.

Algorithm 1 Decision Aggregation

Input: Clustering matrix A, Groundtruth matrix Z, parameters α, set of weights $\mathcal{W}$, and ε;
Output: The class label for each event L_i^e;

1: Initialize $Y^{(0)}$, $Y^{(1)}$ randomly.
2: $\tau \leftarrow 1$
3: **while** $\sqrt{\sum_{j=1}^{l} ||\vec{y}_{j\cdot}^{(\tau)} - \vec{y}_{j\cdot}^{(\tau-1)}||^2} > \varepsilon$ **do**
4:     **for** $i \leftarrow 1$ to t **do**
5:         $\vec{x}_{i\cdot}^{(\tau+1)} = \frac{\sum_{j=1}^{l} a_{ij}w_j\vec{y}_{j\cdot}^{(\tau)} + \alpha b_i\vec{z}_{i\cdot}}{\sum_{j=1}^{l} a_{ij}w_j + \alpha b_i}$
6:     **for** $j \leftarrow 1$ to l **do**
7:         $\vec{y}_{j\cdot}^{(\tau+1)} = \frac{\sum_{i=1}^{t} a_{ij}\vec{x}_{i\cdot}^{(\tau+1)}}{\sum_{i=1}^{t} a_{ij}}$
8:     $\tau \leftarrow \tau + 1$
9: **for** $i \leftarrow 1$ to t **do**
10:     **return** $L_i^e \leftarrow \arg\max_k x_{ik}^{(\tau)}$

event's probability vector $\vec{x}_{i\cdot}$ should not deviate much from the corresponding groundtruth vector $\vec{z}_{i\cdot}$ (e.g., event e_3 and $\vec{z}_{3\cdot}$). α can be considered as the shadow price payment for violating this constraint. Additionally, since $\vec{x}_{i\cdot}$ and $\vec{y}_{j\cdot}$ are probability vectors, each of their components must be greater than or equal to 0 and the sum should equal 1.

3.4 Solution

By checking the quadratic coefficient matrix of the objective function, we can show that **P** is a convex program, which makes it possible to find a global optimal solution. We propose to solve **P** using the block coordinate descent method [26]. The basic idea of our solution is: At the τ-th iteration, we fix the values of $\vec{x}_{i\cdot}$ or $\vec{y}_{j\cdot}$, then the objective function of **P** becomes a convex function with respect to $\vec{y}_{j\cdot}$ or $\vec{x}_{i\cdot}$. Therefore, its minimum with respect to $\vec{x}_{i\cdot}$ and $\vec{y}_{j\cdot}$ can be obtained by setting the corresponding partial derivatives (i.e., $\frac{\partial f(X,Y)}{\partial x_{ik}}$ and $\frac{\partial f(X,Y)}{\partial y_{jk}}$, $k = 1,2,\ldots,m$) to 0:

$$\vec{x}_{i\cdot}^{(\tau+1)} = \frac{\sum_{j=1}^{l} a_{ij}w_j\vec{y}_{j\cdot}^{(\tau)} + \alpha b_i\vec{z}_{i\cdot}}{\sum_{j=1}^{l} a_{ij}w_j + \alpha b_i},\ \vec{y}_{j\cdot}^{(\tau+1)} = \frac{\sum_{i=1}^{t} a_{ij}\vec{x}_{i\cdot}^{(\tau+1)}}{\sum_{i=1}^{t} a_{ij}} \quad (8)$$

The detailed steps are shown in Algorithm 1. The algorithm starts by initializing the probability matrix of input clusters randomly. The iterative process begins in line 3. First, the events receive the information (i.e., $\vec{y}_{j\cdot}$) from neighboring clusters and update $\vec{x}_{i\cdot}$ (line 5). Then, the events propagate the information (i.e., $\vec{x}_{i\cdot}$) to its neighboring clusters to update $\vec{y}_{j\cdot}$ (line 7). Finally, an event, say e_i, is assigned to the k-th oCluster if x_{ik} is the largest probability in $\vec{x}_{i\cdot}$ (line 10). According to [26] (Proposition 2.7.1), by showing the continuous differentiability of the objective function and the uniqueness of the minimum when fixing $\vec{x}_{i\cdot}$ or $\vec{y}_{j\cdot}$, we can prove that $(X^{(\tau)}, Y^{(\tau)})$ converges to the optimal point. When solving **P**, we don't take into account the constraints (Eqn. (6) and Eqn. (7)). By inductively checking the L1 norm of $\vec{x}_{i\cdot}^{(\tau)}$ and $\vec{y}_{j\cdot}^{(\tau)}$ from $\tau = 1$, it can be found out that the solution obtained by Algorithm 1 automatically satisfies the constraints.

Table 1 shows the first two iterations of the Decision-Aggregation algorithm (with $\alpha = 20$ and $w_j = 1$ for all j) for the belief graph shown in Fig. 2. We start with uniform

Table 1. Iterations of Decision Aggregation

$Y^{(1)}$	$X^{(1)}$	$Y^{(2)}$	$X^{(2)}$
(0.5,0.5)	(0.5,0.5)	(0.3913,0.6087)	(0.4710,0.5290)
(0.5,0.5)	(0.5,0.5) (0.0652,0.9348)	(0.5725,0.4275)	(0.4686,0.5314) (0.0536,0.9464)
(0.5,0.5)	(0.5,0.5) (0.5,0.5)	(0.5870,0.4130)	(0.4710,0.5290) (0.4710,0.5290)
(0.5,0.5)	(0.5,0.5) (0.9348,0.0652)	(0.4130,0.5870)	(0.5290,0.4710) (0.9464,0.0536)
(0.5,0.5)	(0.5,0.5) (0.5,0.5)	(0.6087,0.3913)	(0.5314,0.4686) (0.5290,0.4710)
(0.5,0.5)	(0.5,0.5)	(0.4275,0.5725)	(0.5290,0.4710)

probability vectors for the six clusters ($Y^{(1)}$). Then the probabilities of the events without labels are calculated by averaging the probabilities of the clusters they link to. At this step, they all have (0.5, 0.5) as their probability vectors. On the other hand, if the event is labeled (e.g., e_3 and e_7 in this example), the labeled information is incorporated into the probability vector computation where we average the probability vectors of the clusters each event links to and that of the groundtruth label (note that the vote from the true label has a weight α). For example, e_3 is adjacent to c_1, c_4, c_6 and label 2, so we have $\vec{x}_{3\cdot}^{(1)} = \frac{(0.5,0.5)+(0.5,0.5)+(0.5,0.5)+\alpha\cdot(0,1)}{3+\alpha} = (0.0652, 0.9348)$. During the second iteration, $\vec{y}_{j\cdot}^{(2)}$ is obtained by averaging the probabilities of the events it contains. For example, $\vec{y}_{1\cdot}^{(2)}$ is the average of $\vec{x}_{2\cdot}^{(2)}$, $\vec{x}_{3\cdot}^{(2)}$, $\vec{x}_{5\cdot}^{(2)}$ and $\vec{y}_{9\cdot}^{(2)}$, which leads to $(0.3913, 0.6087)$. The propagation continues until convergence.

3.5 Performance Analysis

It can be seen that at each iteration, the algorithm takes $O(tlm) = O(tnm^2)$ time to compute the probability vectors of clusters and events. Also, the convergence rate of coordinate descent method is usually linear [26] (in practice, we fix the iteration number as a constant). Thus, the computational complexity of Algorithm 1 is actually linear with respect to the number of events (i.e., t), considering that the number of nodes involved in each aggregation (i.e., n) and the number of classes (i.e., m) are usually small. Thus, the proposed algorithm is not more expensive than the classification/clustering schemes, and thus can be applied to any platform running classification tasks. Furthermore, since wireless communication is the dominating factor of the energy consumption in sensor networks, our algorithm actually saves much more energy than it consumes.

4 Hierarchical Aggregate Classification

Here we introduce the *Hierarchical Aggregate Classification* (HAC) protocol. HAC applies the Decision-Aggregation algorithm on each of the nonleaf nodes to aggregate all the decisions it collects. The output of the algorithm, i.e., the aggregated decision is forwarded upwards by the nonleaf node, and serves as one of the input decisions in the aggregation at its parent node. The message carrying the decision consists of t entries, corresponding to t events. In each entry, the index of an event and the ID of the oCluster to which this event belongs are stored. As previously defined, the ID of each oCluster is the label of this oCluster. However, these labels may not be useful in later aggregations, because the oClusters will probably be combined with other clusters whose labels are unknown. For instance, at the sink s_0 (shown in Fig. 1(b)), the labeled clusters in decision D_5 are combined with unlabeled clusters in D_0 and D_2. Finally, the global decision is obtained at the sink node, and each event is assigned a predefined label.

5 Constrained Hierarchical Aggregate Classification

In this section, we introduce an extended version of the HAC protocol, called the *Constrained Hierarchical Aggregate Classification* (cHAC) protocol. cHAC also uses the Decision-Aggregation algorithm as the aggregation function. However, different from HAC which requires that each nonleaf node aggregates all the decisions it collects, cHAC intelligently plans the decision aggregations throughout the tree such that more accurate classification results can be obtained at the cost of more energy consumption. In the rest of this section, we first use a simple example to illustrate how energy and accuracy are correlated during the process of decision aggregation. Then we present the cHAC protocol, together with the procedures it invokes. Finally, the performance of cHAC is analyzed.

5.1 Tradeoff between Energy and Accuracy

Hierarchical aggregate classification, as discussed in the previous sections, can improve the classification accuracy as well as the consumption of communication energy through combining decisions coming from different sensor nodes. However, decision information is inevitably lost during the process of aggregation, and this may hurt the accuracy of the aggregated decision.

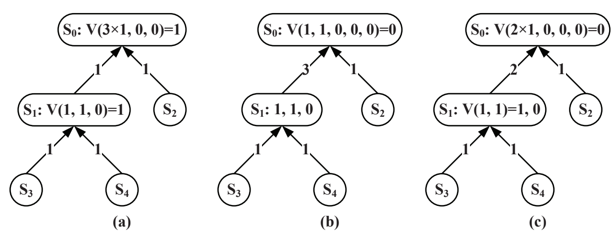


Figure 3. An example of energy-accuracy tradeoff

Let's continue to study the example shown in Fig. 1. Suppose all of the five sensors detect an event and try to predict the label of this event. To simplify the explanation, in this case we assume the sensor nodes are doing classification (not clustering). There are two classes with label 0 and 1 respectively. The local decisions of the nodes are: $D_0 = D_1 = D_2 = 0$ and $D_3 = D_4 = 1$ (recall that D_i is s_i's decision). Here we use a simple method, i.e., majority voting, as the aggregation function $\mathbf{F}(\cdot)$. Note that only in this example we use majority voting as the aggregation function, in every other part of this paper, the aggregation function means the Decision-Aggregation algorithm. Intuitively, given that the weight of each node is 1, the aggregated result of all the decisions should be 0, since there are more 0s than 1s among the 5 atomic decisions.

Figure. 3(a) illustrates the process of hierarchical aggregate classification along the aggregation tree, where the numbers on the links imply the number of decisions transmitted along this link. At node s_1, D_1, D_3 and D_4 are combined. The resultant decision, which is D_5, is evaluated to be 1, since the majority of input decisions (D_3 and D_4) choose label 1. Then, D_5 is sent to the sink s_0, where the final aggregation happens. Since D_5 is the combination of three atomic decisions, its weight is the aggregate weight of three nodes, i.e., $w_5 = 3$. Therefore, the final aggregation is calculated as follows: $\mathbf{F}(w_5 D_5, w_0 D_0, w_2 D_2) = \mathbf{F}(3 \times 1, 0, 0) = 1$. Clearly, this result is not accurate, since more than half of the nodes predict the label of the event to be 0. The problem lies in the aggregation at node s_1, where some decision information, i.e., $D_1 = 0$ is lost[5].

5.2 Problem Formulation

To address this problem, we propose to trade energy for accuracy. Specifically, we add a constraint to the hierarchical aggregate classification, namely, in each decision aggregation along the tree (including the aggregation at the sink), the weight of each input decision cannot be larger than a predefined percentage of the total weight of all this aggregation's input decisions. Formally, suppose $\mathcal{D}$ is the set of the input decisions involved in an aggregation (note that $\mathcal{D}$ is *NOT* the set of all nodes' decisions), then it must be satisfied that $\frac{w_i}{\sum_{D_k \in \mathcal{D}} w_k} \leq \delta$ for $\forall\, D_i \in \mathcal{D}$, where w_i (w_k) denotes the weight of decision D_i (D_k), and δ is a predefined percentage. In this paper, we call δ the *weight balance threshold*, and the constraint defined above the *weight balance constraint*. The weight balance threshold δ is a control parameter which can tune the tradeoff between energy and accuracy.

Intuitively, the smaller δ is, the larger number of decisions are combined in each aggregation, and thus the aggregated decision is closer to the global consensus. For example, if all the sensor nodes have the same weight and $\delta = \frac{1}{n}$ (n is the total number of sensor nodes), the weight balance constraint requires that each combination takes at least n decisions, which indicates that all the decisions need to be sent to the sink node without any aggregation on the half way. Clearly, the result of this aggregation perfectly represents the global consensus. Moreover, when δ is small, to guarantee that a large number of decisions are combined in each aggregation, some decisions have to be transmitted for more than one hop along the aggregation tree, resulting in more transmissions.

For the simplicity of presentation, we assume that in each transmission, only one decision is forwarded. We are interested in the problem of *Constrained Hierarchical Aggregate Classification*: Under the weight balance constraint, among all the legal ways (solutions) which can aggregate the atomic decisions along the aggregation tree to reach a consensus at the sink, we want to pick the one with the minimum total number of transmissions. In fact, the hierarchical aggregate classification problem discussed in previous sections is equivalent to the case when $\delta = 1$.

Let's get back to the example shown in Fig. 3. Suppose in this case, δ is set to be 0.5. Apparently, the aforementioned solution (Fig. 3(a)) does not satisfy this constraint, since the weight percentage of D_5 in the final aggregation is more than half. Thus, although the absolute minimum transmission number (which is 4) is achieved in this case, it is not a valid solution. In contrast, a feasible solution is shown in Fig. 3(b). In this scenario, node s_1 does not make any aggregation, but directly forwards the decisions (D_1=0, D_3=1 and D_4=1) to the sink. This will surely satisfy the balance constraint. In addition, this solution actually achieves the highest accuracy, since no information is lost before arriving at the sink node. However, it results in unnecessary energy consumption (6 transmissions in total). Finally, Fig. 3(c) shows the optimal solution. Specifically, node s_1 combines two out of the three decisions ($D_5 = \mathbf{F}(D_3, D_4) = \mathbf{F}(1,1) = 1$), and delivers the resultant decisions (D_1 and D_5) to the sink through 2 transmissions. This solution spends 5 transmissions, the minimum energy consumption that can be achieved under the weight balance constraint. More importantly, the global decision achieved by this solution is 0, which correctly represents the majority of the nodes.

As a matter of fact, the constrained hierarchical aggregate classification (cHAC) problem is an NP-complete problem. We prove this proposition by the following theorem:

Theorem 1. The constrained hierarchical aggregate classification problem is NP-complete.

PROOF. First of all, we restate the cHAC problem as a decision problem. That is, we wish to determine whether the decision aggregation along a given tree can be done at the cost of exactly k transmissions. In this proof, we will show that the equal-size partition problem (ePartition for short), a known NP-complete problem, can be reduced to cHAC, i.e., ePartition $\leq_P$ cHAC. The equal-size partition problem is to decide whether a given set of integers can be partitioned into two "halves" that have both the same size (number of integers) and the same sum (summation of the integers).

The reduction algorithm begins with an instance of ePartition. Let $\mathcal{A} = \{a_1, a_2, \cdots, a_n\}$ ($n \geq 8$) be a set of integers. We shall construct an instance of cHAC (denoted by Φ) such that $\mathcal{A}$ can be equally partitioned if and only if the answer to Φ is yes.

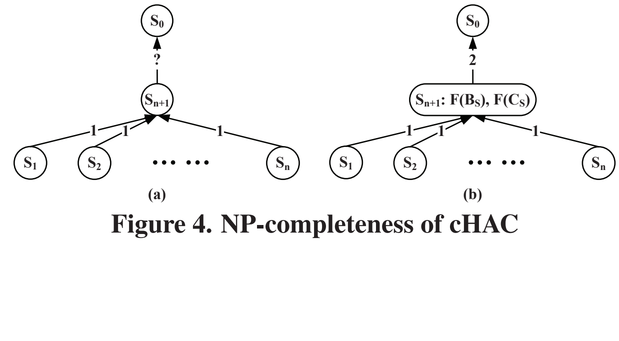


Figure 4. NP-completeness of cHAC

[5]Note that in this simple example, s_1 can send the numbers of 1s and 0s picked by its children (together with itself) to achieve better consensus compared with sending the majority voting result only. However, this solution cannot work for the decision aggregation problem where only clustering results are available. Since the same cluster ID may represent different classes in different decisions, we cannot simply count the number of labels assigned by the decisions. Furthermore, when the number of classes is large, this solution will consume excessive energy.

Φ is constructed as follows. An aggregation tree is shown in Fig. 4(a). The root node s_0 has a single child, which is s_{n+1}. There are n nodes $(s_1, s_2, \cdots, s_n)$ connected to s_{n+1}. Suppose the weight of node s_i, $i = 1,2,\cdots,n$ is $a_i + M$, where M is a very large positive integer. Moreover, the weights of s_0 and s_{n+1} are $\frac{\sum_{i=1}^{n} a_i + nM}{2}$ and 0, respectively. In this case, the weight balance threshold is set to be $\delta = \frac{1}{3}$. Then, we introduce the formal description of Φ: Is there a way to solve the cHAC problem on the tree shown in Fig. 4(a) such that the total number of transmissions is exactly $n+2$.

Suppose $\mathcal{A}$ can be partitioned into two equal-size subsets with equal summation. We denote these two subsets by $\mathcal{B}$ and $\mathcal{C}$. Without loss of generality, we suppose $\mathcal{B} = \{a_1, a_2, \cdots, a_{\frac{n}{2}}\}$ and $\mathcal{C} = \{a_{\frac{n}{2}+1}, a_{\frac{n}{2}+2}, \cdots, a_n\}$, and thus we have $\sum_{i=1}^{\frac{n}{2}} a_i = \sum_{j=\frac{n}{2}+1}^{n} a_j$. Correspondingly, we put nodes $s_1, s_2, \cdots, s_{\frac{n}{2}}$ in $\mathcal{B}_S$ and $s_{\frac{n}{2}+1}, s_{\frac{n}{2}+2}, \cdots, s_n$ in $\mathcal{C}_S$ (as shown in Fig. 4(b)). Since $w_i = a_i + M$, it can be derived that $\sum_{i=1}^{\frac{n}{2}} w_i = \sum_{j=\frac{n}{2}+1}^{n} w_j$, namely, $\mathcal{B}_S$ and $\mathcal{C}_S$ have the same total weight. In addition, no elements in $\mathcal{B}_S$ and $\mathcal{C}_S$ violate the weight balance constraint given that M is a large integer. Then, we combine decisions in $\mathcal{B}_S$ and $\mathcal{C}_S$ at node s_{n+1}, and use 2 transmissions to send the combined ones to s_{n+1}. Furthermore, since the weight of each combined decision is $\frac{\sum_{i=1}^{n} a_i + nM}{2}$, which equals s_0's weight, the three decision (two combined decisions and s_0's decision) can be aggregated at node s_0 without violating the weight balance constraint (recall that $\delta = \frac{1}{3}$). Furthermore, since n transmissions are needed to move the clustering decision of each leaf node to node s_{n+1}, altogether $n+2$ transmissions are used during this process, and thus the answer to Φ is yes.

Conversely, suppose the answer to Φ is yes. Since n transmissions (from s_i to s_{n+1}) are inevitable, we have to use 2 transmissions to send decisions from s_{n+1} to s_0. It is easy to see that the only way to achieve this is to combine the decisions at s_{n+1} into $\mathcal{B}_S$ and $\mathcal{C}_S$ with the same weight $\frac{\sum_{i=1}^{n} a_i + nM}{2}$, and then send them to s_0. For M is large, $\mathcal{B}_S$ and $\mathcal{C}_S$ must have the same size, and thus the corresponding halves $\mathcal{B}$ and $\mathcal{C}$ in $\mathcal{A}$ also have the same sum $\frac{\sum_{i=1}^{n} a_i}{2}$ (and of course, the same size). So Φ is yes implies that the ePartition instance is yes.

In summary, ePartition $\leq_P$ cHAC is proved, and thus cHAC is NP-hard. Furthermore, it is straightforward to show that cHAC$\in$ NP, and thus a conclusion can be drawn that cHAC is NP-complete. $\square$

The NP-completeness of the constrained decision aggregation problem makes it hopeless to find the optimal solution in polynomial time. In the rest of this section, we'll introduce an approximate solution, which is proved to have a constant approximation ratio and a polynomial complexity.

5.3 Constrained Decision Aggregation

In this subsection, we introduce the key function of our solution, which is called *constrained decision aggregation*. The constrained decision aggregation procedure works at each nonleaf tree node, except the sink. It partitions the decisions gathered at this node into different sets and invokes Decision-Aggregation to combine the decision sets which respect the weight balance constraint.

Intuitively, to guarantee that the final decision aggregation at the sink node is legal, each decision arriving at the sink should have a weight smaller than or equal to $W = \lfloor \delta W_T \rfloor$ (where W_T denotes the total weight of all the sensor nodes on the aggregation tree). Therefore, at any nonleaf node except the sink, the summation of the weights of all the input decisions involved in any aggregation must not exceed W. This is an additional constraint called the *weight summation constraint* for each nonleaf node. Also, W is referred to as the *weight summation threshold*. From the analysis above, it is easy to see that any solution to the constrained hierarchical aggregate classification problem satisfies this constraint.

Consequently, at each nonleaf node, before aggregate the decisions, we need to solve the following problem first: Consider a nonleaf node s_0 with n children nodes s_i, $i = 1,2,...,n$. The goal is to divide the decision set $\mathcal{D} = \{D_1, D_2, ..., D_n\}$ into the minimum number of subsets such that each multi-decision subset respects both the weight summation constraint and the weight balance constraint. Since node s_0 spends one transmission to forward each aggregated subset or single-decision subset to its parent, minimizing the subset number implies the minimization of transmission number. To solve this problem, we introduce Decision-Selection, an algorithm which can pick a valid subset of decisions as long as there exists one. Afterwards, we give a formal definition of the Constrained-Decision-Aggregation procedure which iteratively invokes Decision-Selection and Decision-Aggregation to select and combine the decisions.

Decision-Selection is a dynamic programming based approach. First of all, we define a couple of notations. (a) $V[i,w]$: $V[i,w] = 1$ if out of the first i decisions in $\mathcal{D}$, it is possible to find a subset in which the aggregate weight of all the decisions is exactly w, and $V[i,w] = 0$ otherwise. (b) $keep[i,w]$: $keep[i,w] = 1$ if decision D_i is picked in the subset whose total weight is w, and $keep[i,w] = 0$ otherwise. The initial settings of $V[i,w]$ are described as below:

$$V[0,w] = 0 \qquad \text{for } 0 \leq w \leq W \tag{9}$$

$$V[i,w] = -\infty \qquad \text{for } w < 0 \tag{10}$$

$$V[i,w_i] = 1 \qquad \text{for } 1 \leq i \leq n \tag{11}$$

In Decision-Selection, $V[i,w]$ is recursively calculated based on Eqn. (12) for $1 \leq i \leq n$ and $0 \leq w \leq W$.

$$V[i,w] = \max(V[i-1,w], V[i-1,w-w_i]) \tag{12}$$

Algorithm 2 describes the detailed steps of Decision-Selection. Given a particular decision D_i and a weight sum w, what we are concerned about is under which condition D_i could be selected to the output decision set (i.e., set $keep[i,w]$ to be 1), which can not be directly seen from Eqn. (12). There are two possible cases. Case 1 happens in line 8. In this case, among the first $i-1$ decisions, we can find a subset with total weight $w - w_i$ (i.e., $V[i-1, w-w_i] = 1$), but cannot find a subset with total weight w (i.e., $V[i-1,w] = 0$). Obviously, D_i should be selected; Case 2 is in line 12. In this case,

Algorithm 2 Decision Selection

Input: Weight summation threshold W, set of weights $\mathcal{W}$, set of input decisions $\mathcal{D}$, weight balance threshold δ;
Output: Set of decisions $\mathcal{A}$ satisfying both weight summation constraint and weight balance constraint;

1: $\mathcal{A} \leftarrow \Phi$
2: **for** $w \leftarrow 0$ to W **do**
3:     $V[0,w] \leftarrow 0$;
4: **for** $i \leftarrow 1$ to n **do**
5:     **for** $w \leftarrow 0$ to W **do**
6:         **if** $w_i < w$ and $V[i-1, w-w_i] > V[i-1,w]$ **then**
7:             $V[i,w] \leftarrow V[i-1,w-w_i]$;
8:             $keep[i,w] \leftarrow 1$;
9:         **else if** $w_i = w$ **then**
10:             $V[i,w] \leftarrow 1$;
11:             **if** $V[i-1,w] = 0$ **then**
12:                 $keep[i,w] \leftarrow 1$;
13:             **else**
14:                 $keep[i,w] \leftarrow 0$;
15:         **else**
16:             $V[i,w] \leftarrow V[i-1,w]$;
17:             $keep[i,w] \leftarrow 0$;
18: **for** $w \leftarrow W$ downto 1 **do**
19:     $m \leftarrow w$;
20:     **for** $i \leftarrow n$ downto 1 **do**
21:         **if** $keep[i,m] = 1$ and $w_i \leq \lfloor \delta w \rfloor$ **then**
22:             $\mathcal{A} \leftarrow \mathcal{A} \bigcup \{D_i\}$;
23:             $m \leftarrow m - w_i$;
24:     **if** $\mathcal{A} \neq \Phi$ **then**
25:         break;
26: **return** $\mathcal{A}$

though $w = w_i$, we put D_i into the selected set only when no subset among the first $i-1$ decisions has a total weight of w (i.e., $V[i-1,w] = 0$), since the algorithm only picks one set for a particular weight sum. Decision-Selection has an important prerequisite: *$\mathcal{D}$ must be sorted in the ascending order of weight*. The motivation of this prerequisite can be better understood via Theorem 2.

Theorem 2. Decision-Selection can return a valid decision set satisfying both the weight summation constraint and the weight balance constraint, as long as there exists such a set in $\mathcal{D}$.

PROOF. There is no doubt that given a number $1 < w < W$, Algorithm 2 can identify a subset of $\mathcal{D}$ whose weight summation is exactly equal to w, if such a subset really exists. There are at most W subsets found by the algorithm (stored in $keep[i,w]$), and all of them satisfy the weight summation constraint. Thus, we only need to show that if none of these selected subsets can satisfy the weight balance constraint, there does not exist a legal decision subset in $\mathcal{D}$. The key point is, given a weight summation w, there may exist multiple valid subsets, and among them, the subset (denoted by $\mathcal{D}^*(w)$) whose last decision has the smallest weight is the most likely to satisfy the weight balance constraint. This is because the decisions are sorted in the ascending order of the weight. Thus, given a decision set, if the last decision, which has the largest weight, satisfies the constraint, all the preceding decisions in this set satisfy the constraint as well. The Decision-Selection algorithm guarantees to pick $\mathcal{D}^*(w)$, since a decision D_i is selected (i.e., $keep[i,w] \leftarrow 1$, line 8 and 12) only when no valid subset exists among the first $i-1$ decisions (i.e., $V[i-1,w] = 0$, line 6 and 11). For example, suppose we have a sorted decision set $\mathcal{D} = \{w_1 = 2, w_2 = 3, w_3 = 4, w_4 = 7\}$, with the weight sum $w = 9$ and the balance threshold $\delta = \frac{1}{2}$. It is obvious that there are two subsets of $\mathcal{D}$ whose weight sum equals w. They are $\mathcal{D}_1 = \{2,3,4\}$ and $\mathcal{D}_2 = \{2,7\}$. Among them, only $\mathcal{D}_1$, whose last decision has a smaller weight, can satisfy the weight balance constraint. In the algorithm, $keep[3,9]$ will be set by 1, and $keep[4,9]$ is assigned to be 0 since $V[3,9] = 1$. Finally, lines 18-25 exhaustively check $\mathcal{D}^*(w)$ (line 21) with w ranging from W down to 1, and thus will not miss a valid subset if it does exist. $\square$

Algorithm 3 Constrained Decision Aggregation

Input: Set of input decisions $\mathcal{D}$
Output: Set of output decisions Ω

1: Sort $\mathcal{D}$ in the ascending order of weight;
2: **repeat**
3:     $\mathcal{A} \leftarrow$ Decision-Selection($\mathcal{D}$);
4:     $\mathcal{D} \leftarrow \mathcal{D} - \mathcal{A}$;
5:     $\mathcal{C} \leftarrow \mathcal{C} \bigcup \{$Decision-Aggregation($\mathcal{A}$)$\}$;
6: **until** $|\mathcal{A}| = 0$
7: $\mathcal{R} \leftarrow \mathcal{D}$;
8: $\Omega \leftarrow \mathcal{C} \bigcup \mathcal{R}$;
9: **return** Ω

With Decision-Selection, we are able to design the Constrained-Decision-Aggregation algorithm. As shown in Algorithm 3, after sorting $\mathcal{D}$ in the ascending order of weight, Constrained-Decision-Aggregation iteratively invokes Decision-Selection (line 3), and combines the returned decision set (i.e., $\mathcal{A}$) through the procedure of Decision-Aggregation (line 5). The resultant aggregated decisions are stored in a set $\mathcal{C}$. This process repeats until no more set is found by Decision-Selection, then the residual decisions left in $\mathcal{D}$ are moved to another set $\mathcal{R}$. In line 8, the union of $\mathcal{C}$ and $\mathcal{R}$ forms the set of output decisions Ω. Finally, $|\Omega|$ transmissions are consumed by the subtree root s_0 to forward the output decisions in Ω to its parent node.

5.4 Constrained Hierarchical Aggregate Classification

The constrained hierarchical aggregate classification (cHAC) protocol invokes the Constrained-Decision-Aggregation procedure at each nonleaf node except the sink, and aggregates the decisions along the tree. Specifically, suppose n subtrees are connected to a nonleaf node s_0. cHAC applies Constrained-Decision-Aggregation to each of the subtrees T_i, resulting in a set of aggregated decisions $\mathcal{C}_i$ and a set of residual decisions $\mathcal{R}_i$. After arriving at s_0, these sets form two union sets, which are $\mathcal{C}_0 = \bigcup_{i=1}^{n} \mathcal{C}_i$ and $\mathcal{R}_0 = \bigcup_{i=1}^{n} \mathcal{R}_i$. Then, cHAC employs Constrained-Decision-Aggregation on $\mathcal{R}_0$, and puts the newly aggregated decisions in $\mathcal{C}_0$. After the algorithm terminates, no decisions left in $\mathcal{R}_0$ can be further combined. Subsequently, s_0 spends $|\Omega_0| = |\mathcal{C}_0| + |\mathcal{R}_0|$ transmissions to forward the decisions. Altogether, $\sum_{i=0}^{n} |\Omega_i|$ transmissions are consumed during this process. In each transmission, the cHAC protocol uses the same message format as the HAC protocol to carry the decision.

At the sink node, the procedure of Decision-Aggregation (not Constrained-Decision-Aggregation) is called, since none of the arrived decisions has a weight larger than $W = \lfloor \delta W_T \rfloor$. Finally, the global consensus is achieved. It is easy to see, the cHAC protocol can be easily implemented in a distributed manner. Each node only needs to collect the decision information from its children and make local aggregations. Therefore, no global coordination is needed.

5.5 Performance Analysis

In this subsection, we show the approximation ratio and the computational complexity of not only the Constrained-Decision-Aggregation algorithm but also the whole constrained hierarchical aggregate classification process. We start with the analysis of the Constrained-Decision-Aggregation algorithm. First of all, we have the following observation.

Lemma 1. After Constrained-Decision-Aggregation terminates, there are at most one decision in $\mathcal{C}$ whose weight is no more than $\frac{W}{2}$.

The basic idea of the proof is: suppose there are two such decisions, they are resulted from two decision subsets whose weight summation is no more than $\frac{W}{2}$. However, the Decision-Selection algorithm should have combined these two subsets into one which satisfies both the weight balance constraint and the weight summation constraint. In fact, if there exists a decision in $\mathcal{C}$ with a weight less than or equal to $\frac{W}{2}$, we move it from $\mathcal{C}$ to $\mathcal{R}$.

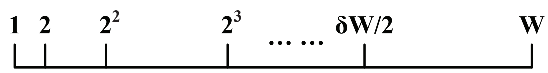


Figure 5. Intervals

According to Algorithm 3, the residual set $\mathcal{R}$ contains the decisions that cannot be aggregated. We project these decisions onto an interval $[1, W]$ based on their weights. Then, we divide the interval $[1, W]$ into subintervals by the points 2^i, $i = 1, 2, ..., \lfloor \log_2(\delta W/2) \rfloor$ (together with the point $\delta W/2$), as shown in Fig. 5. Before proving the approximation ratio of Constrained-Decision-Aggregation in Theorem 3, we first prove the following claim in Lemma 2.

Lemma 2. Within each interval, there are less than $\frac{2}{\delta}$ decisions in $\mathcal{R}$.

PROOF. By contradiction, suppose there are $\frac{2}{\delta}$ decisions within an interval delimited by the point 2^{i-1} and 2^i. The sum of their weights is less than W, for $i \leq \lfloor \log_2(\delta W/2) \rfloor$. Within $[2^{i-1}, 2^i]$, the aggregate weight of these decisions is at least $\frac{2}{\delta} \cdot 2^{i-1} = \frac{2^i}{\delta}$, while the weight of a single decision is at most 2^i. Thus, the weight percentage of any decision in this interval is at most δ, which satisfies the weight balance constraint. This contradicts with the fact that Constrained-Decision-Aggregation leaves them uncombined, so the proof is completed. □

Theorem 3. Suppose OPT is the transmission number in the optimal solution to the constrained decision aggregation problem, and SOL is the transmission number in the solution found by the Constrained-Decision-Aggregation algorithm. We have $\text{SOL} \leq \frac{2}{\delta} \cdot \left(\text{OPT} + \log_2 \frac{\delta W}{2}\right)$.

PROOF. First of all, we define some notations. Let $W_{\mathcal{D}}$ denote the total weight of the decisions in $\mathcal{D}$. In addition, $\mathcal{R}_{\leq}$ and $\mathcal{R}_{>}$ are two subsets of $\mathcal{R}$ in which the weights of decisions are smaller than or equal to $\delta W/2$ and larger than $\delta W/2$, respectively. An lower bound of OPT is $W_{\mathcal{D}}/W$, since every (aggregated) decision has a weight at most W. In our algorithm, decisions in $\mathcal{D}$ are partitioned and aggregated into two sets $\mathcal{C}$ and $\mathcal{R}$, and we have $\text{SOL} = |\mathcal{C}| + |\mathcal{R}| = |\mathcal{C}| + |\mathcal{R}_{\leq}| + |\mathcal{R}_{>}|$. For all the decisions in $\mathcal{C}$ whose weights are at least $W/2$ and all the decision in $\mathcal{R}_{>}$ whose weights are at least $\delta W/2$, we have $|\mathcal{C}| + |\mathcal{R}_{>}| \leq \frac{W_{\mathcal{D}}}{\delta W/2} = \frac{2}{\delta} \cdot \frac{W_{\mathcal{D}}}{W} \leq \frac{2}{\delta} \cdot \text{OPT}$, for $\text{OPT} \geq W_{\mathcal{D}}/W$. In addition, by Lemma 2, we have $|\mathcal{R}_{\leq}| \leq \frac{2}{\delta} \cdot \log_2 \frac{\delta W}{2}$. Thus, combining the above two inequalities, we can derive the promised bound $\text{SOL} \leq \frac{2}{\delta} \cdot \left(\text{OPT} + \log_2 \frac{\delta W}{2}\right)$. □

Then, the computational complexity of Constrained-Decision-Aggregation is given by Theorem 4.

Theorem 4. Constrained-Decision-Aggregation has a computational complexity of $O(n^2 \delta W)$[6].

PROOF. First of all, Constrained-Decision-Aggregation sorts the decisions in $\mathcal{D}$, which takes a running time of $O(n \log n)$. Then, in the loop between line 2 and 6, Decision-Selection is repeatedly called, and each takes $O(nW)$. Since under the weight balance constraint, the number of decisions picked by Decision-Selection in each iteration must be no less than $\frac{1}{\delta}$, the number of times that Decision-Selection is called is at most δn. Therefore, the overall computational complexity of the Constrained-Decision-Aggregation procedure is $O(n \log(n)) + \delta n O(nW) = O(n^2 \delta W)$. □

Next, we give the approximation ratio and the computational complexity of the whole constrained hierarchical aggregate classification process by Theorem 5 and Theorem 6, respectively.

Theorem 5. Suppose OPT is the transmission number in the optimal solution to the constrained hierarchical aggregate classification problem, and SOL is the transmission number in the solution found by the cHAC protocol. Then, we have $\text{SOL} \leq \frac{2}{\delta} \cdot \left(1 + \log_2 \frac{\delta W}{2}\right) \cdot \text{OPT}$.

PROOF. The proof is similar to the proof of Theorem 3 in spirit, but the bound we derived is a bit weaker. Suppose the tree has n nodes (excluding the sink node). Let $\text{OPT} = \text{OPT}_1 + \text{OPT}_2 + \ldots + \text{OPT}_n$, where OPT_i is the number of transmissions from node s_i to its parent in the optimal solution. Similarly, $\text{SOL} = \text{SOL}_1 + \text{SOL}_2 + \ldots + \text{SOL}_n$, where SOL_i is the number of transmissions from s_i to its parent in the solution obtained by the cHAC protocol.

In case that s_i is a nonleaf node, the cHAC protocol takes the aggregated decisions from its children as the input. Intuitively, the lower bound of OPT_i is the optimal solution (denoted by $\widetilde{\text{OPT}}_i$) to the problem that takes all the atomic decisions without being aggregated as the in-

[6] In this analysis, we do not consider Decision-Aggregation, since it can be decoupled from Constrained-Decision-Aggregation.

put. Thus, similar to the analysis in Theorem 3, we have $\mathrm{SOL}_i \leq \frac{2}{\delta} \cdot \left(\widetilde{\mathrm{OPT}}_i + \log_2 \frac{\delta W}{2} \right) \leq \frac{2}{\delta} \cdot \left(1 + \log_2 \frac{\delta W}{2} \right) \cdot \widetilde{\mathrm{OPT}}_i$. If s_i is a leaf node, it is apparent that $\mathrm{SOL}_i = \mathrm{OPT}_i$. Since $\widetilde{\mathrm{OPT}}_i \leq \mathrm{OPT}_i$, summing them up for $i = 1, 2, \ldots, n$, we can derive the promised bound $\mathrm{SOL} \leq \frac{2}{\delta} \cdot \left(1 + \log_2 \frac{\delta W}{2} \right) \cdot \widetilde{\mathrm{OPT}} \leq \frac{2}{\delta} \cdot \left(1 + \log_2 \frac{\delta W}{2} \right) \cdot \mathrm{OPT}$. $\square$

Theorem 6. The cHAC protocol has a computational complexity of $O(n_T^2 W)$.

Recall that here n_T denotes the total number of nodes on the aggregation tree. In the worst case, the Decision-Selection algorithm is called for $O(n_T)$ times, and each takes $O(n_T W)$ time. Therefore, the overall computational complexity of the cHAC protocol is $O(n_T^2 W)$.

6 Performance Evaluation

In this section, we evaluate the proposed schemes on i) Synthetic data, and ii) A solar-powered sensor network testbed. For the reason of comparison, we design two baseline methods. Both of the baselines adopt the strategy of majority-voting, and they are different in the ways of generating votes. The first baseline method, which we call *Clustering Voting*, suggests that each node locally groups the events into different clusters (this part is the same as our scheme), and then count the labeled events (i.e., how many events belong to a particular label) in each cluster. Within a cluster, all the events are assigned the label with the largest count. For example, suppose there are totally 100 events in a cluster, with three events labeled 1 and two events labeled 0, then all the 100 events are labeled 1 according to the clustering voting scheme. Finally, the nodes vote to decide the label of each event. The second baseline, called *Classification Voting*, lets each node apply classification algorithms (such as decision tree, SVM) on the labeled data, and predict the labels of the rest. Then, the events are labeled based on the vote. The detailed experimental results are shown and explained in the next two subsections.

6.1 Experiment on Synthetic Data

In this part, we evaluate our schemes on synthetic data. First of all, we randomly build aggregation trees with the number of tree nodes ranging from 20 to 80. The height of the trees increases with the augment of tree size. In particular, the trees of height 3, 4, 5 and 6 contain around 20, 30, 50 and 80 nodes, correspondingly. In order to achieve diversity, we apply different clustering algorithms (such as K-means, spectral clustering) to different nodes. Suppose each node has a weight between 0 and 5. For each height, we evaluate different tree topologies and record the average results.

Next, we give a brief description on how the synthetic data is generated. In this experiment, we assume there are 10 different types of sensors, corresponding to 10 features of the events (e.g., temperature, humidity, etc). Suppose the events are drawn from 5 different classes (labels), and we randomly assign the groundtruth labels (from 5 classes) to 10000 events. For each event, based on its assigned label, we generate its feature values from a Gaussian distribution in a 10-dimensional (each dimension corresponds to a feature) data space $\mathbb{R}^{10}$. Therefore, the collection of the events

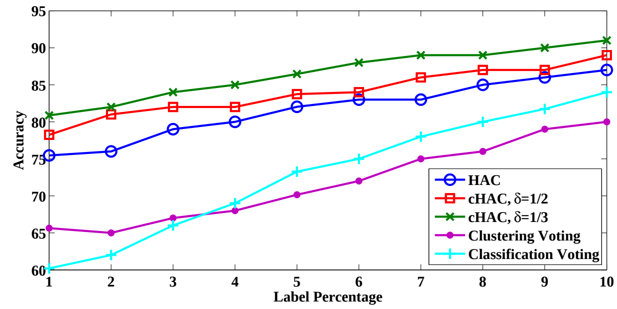
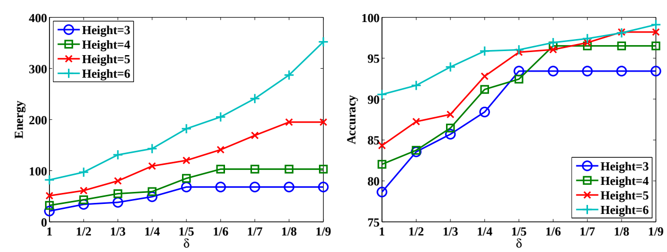


Figure 6. Comparison of accuracy (Synthetic data)

are drawn from a Gaussian mixture model with 5 components, each of which corresponds to a class. After creating the events, we generate the sensory readings of the nodes. For each node on the tree, we randomly assign a subset of the previously defined (10 types of) sensors to it. For each type of sensor assigned to this node, we generate its sensory reading of each event as follows: we first copy the corresponding feature value of the event and then add random Gaussian noise to this value. In this way, different nodes with the same types of sensors would have different sensory readings.



(a) δ on energy (b) δ on accuracy

Figure 7. Impact of δ on energy and accuracy (Synthetic data)

Figure 6 compares the classification accuracy (percentage of the correctly classified events) of the proposed hierarchical aggregate classification (HAC) and constrained hierarchical aggregate classification (cHAC) (with weight balance threshold $\delta = 1/2$, $1/3$, and $1/4$, respectively) protocols, and the two baseline schemes. As can be seen, when the percentage of labeled events is less than 10, the proposed protocols can always achieve better performance than the baselines. Moreover, with the decrease of label percentage, the accuracy of all the five methods degrade. Among them, the accuracy of the classification voting decreases the fastest. This is reasonable since compared with clustering methods, classification models are normally more sensitive to the label percentage. Another notable point is that the cHAC with smaller δ can achieve higher accuracy.

Figure 7 demonstrates the impact of weight balance threshold δ on the communication energy (in terms of the total number of messages transmitted by the tree nodes) and the classification accuracy of the cHAC protocol. In this experiment, we assume 5% of the events are labeled, and test four groups of aggregation trees, with height 3, 4, 5 and 6 respectively. As expected, when δ decreases, no matter of the tree size, more energy is consumed (Figure 7(a)), and

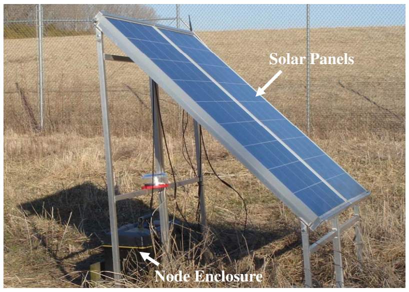


Figure 8. Outside look of a solar-powered sensor node

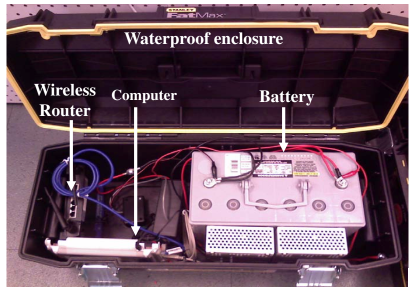


Figure 9. Inside look of a solar-powered sensor node

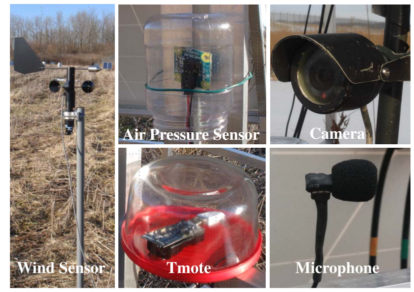


Figure 10. Different types of sensors on the nodes

higher accuracy can be achieved (Figure 7(b)). This confirms our scheme's capability of trading energy for accuracy. Furthermore, given a δ, the trees with larger height tend to have higher accuracy, however, at the cost of more energy consumption. This is because usually better diversity can be obtained when more nodes are involved in the aggregation. Finally, when δ becomes lower than a threshold (e.g., $\delta = \frac{1}{5}$ for height-3 trees), the accuracy cannot be improved any more, since all the atomic decisions (locally made by each node) have been sent to the sink.

6.2 Experiment on Real Testbed

In this part, we test the performance of the proposed protocols on our solar-powered sensor network testbed [20]. This outdoor testbed is located on the edge of a forest. Currently, 9 nodes have been deployed and running since August 2008. Figure 8 and Figure 9 show the outside and inside look of a node, which comprises of a low-power PC to provide computing capability as well as a wireless router to support wireless communication among nodes. The nodes are equipped with multiple types of sensors, and thus can provide a broad spectrum of sensing capabilities for different environmental monitoring applications. Figure 10 shows some of the sensors integrated with the nodes, which can collect the sensory readings of temperature, humidity, light, wind speed, wind direction, and air pressure. In addition, the nodes are also equipped with microphones and cameras which are used to record audio and video information of wildlife. Readers can refer to [20] for more details on the system software and hardware architecture, as well as some implementation issues.

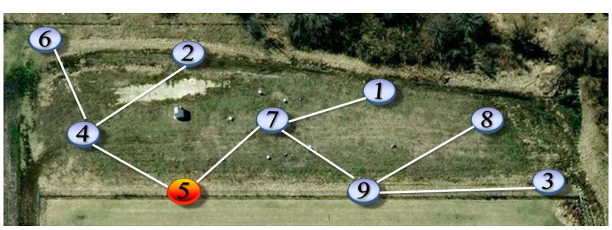


Figure 11. Tree topology of the 9 deployed nodes

We construct an aggregation tree on the 9 deployed nodes, as shown in Fig. 11. In this tree, node 5 works as the sink, and all the nodes have weight 1. Furthermore, to avoid packet collision and overhearing during the process of decision aggregation, we employ a distributed aggregation scheduling algorithm proposed by [27]. Under this scheduling strategy, at any time slot only a subset of the sensor nodes are allowed to send packets and their transmissions do not interfere with each other. The wireless interfaces of the rest nodes are shut down so as to save the energy of idle listening.

To illustrate our approach, we design two experiments on this testbed, which are explained respectively later in this section.

6.2.1 *Classification of Bird Species*

In this experiment, we target on automatically recognizing different bird species based on their vocalizations. Bird species classification is a typical pattern recognition problem and has been extensively studied in recent years [3, 28, 29, 30]. Building bird species recognition system normally involves two phases: feature extraction phase and classification phase. In the feature extraction phase, bird vocalizations are represented with a few acoustical parameters (features) of the sound. Here the philosophy is that features should be selected so that they are able to maximally distinguish sounds produced by different bird species (classes). The most widely used parametrization method is the model of Mel-frequency cepstral coefficients (MFCC), which is adopted in this experiment. After the features are extracted, each audio data point is reduced to a vector of features. Subsequently, in the classification phase, classification or clustering algorithms can be directly applied on the feature vectors like usual classification tasks.

Three bird species: *song sparrow*, *American crow*, and *red-winged blackbird* are studied in this test. They are among the most frequently observed species around the place where the testbed is deployed. We select 4000 time periods within each of which the vocalizations of one species are recorded by all the sensor nodes of the testbed. The duration of each period is 1.5 seconds. The goal of this experiment is to determine the bird species (class) by which the vocalizations are produced within each time period, given that only a small percentage of the 4000 time periods are labeled by the above three species.

Initially, each sensor node locally extracts the MFCC features of each audio clip. Then we apply the same five methods (i.e., the proposed protocols and two baseline schemes) as in the preceding experiment to the extracted feature vectors, and observe their performance. The accuracies achieved by the five methods are shown in Fig. 12. Due to the unexpected noise in both the feature values and the labels, the curves are not as smooth as those shown in the experiment

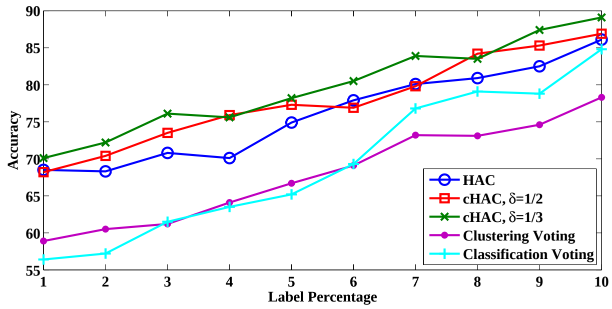
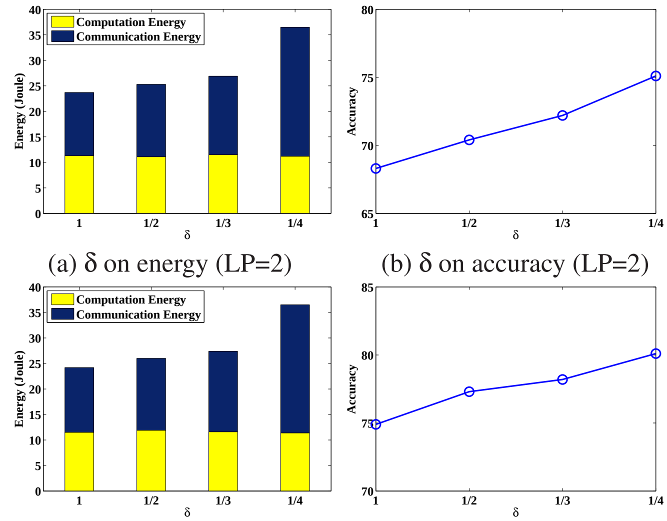


Figure 12. Comparison of accuracy (Species classification)

on synthetic data. However, we can still observe the same curve patterns as discovered in the previous experiment. In short, the proposed schemes always perform better than the baseline methods given a label percentage less than 10.

Different from the experiment on synthetic data in which energy consumption is approximated by the number of transmissions, here we measure the real energy consumption of not only communication but also computation on the testbed. In particular, the computation energy include the energy consumed by the classification (or clustering) algorithms as well as the HAC (or cHAC) protocol. We do not take into account the energy consumption of sensing since it is usually smaller than that of computation and communication, and more importantly, the sensing energy is the same no matter what kind of classification strategy is used. On the other hand, the communication energy is referred to as the energy consumption of sensor nodes by transmitting or receiving packets. As previously discussed, the extra energy expenditure caused by overhearing and idle listening is eliminated by carefully scheduling the on/off of each node's wireless interface.

Figure 13 demonstrates the tradeoff between energy and accuracy tuned by weight balance threshold δ (when the cHAC protocol is applied). In this test, we study two scenarios when the percentage of labeled data is 2 (LP=2) and 5 (LP=5), respectively. Since in this case the aggregation tree has only 9 nodes, there are only four possible choices on δ. However, the relation between energy and accuracy can still be clearly observed. Figure 13(a) and (c) describe the impact of δ on the total computation energy as well as the total communication energy under two label percentages. As one can see, in either case, the computation energy is nearly invariant regardless of δ, since the clustering algorithm locally executed by each node dominates this category of energy consumption. In contrast, the communication energy increases with the decrease of δ, resulting in the growth of total energy expenditure. On the other hand, as shown in Fig. 13(b) and (d), the classification accuracy is improved when δ goes down. Therefore, our expectation that the cHAC protocol could trade energy for accuracy is realized. As a comparison, we also measure the energy needed to transport all the extracted feature vectors to the sink. The number is 1073.41 joules, which is difficult for the energy-scarce sensing systems to afford. This is because the MFCC features are usually of high dimension and thus it costs enormous energy to deliver them.



(a) δ on energy (LP=2) (b) δ on accuracy (LP=2)

(c) δ on energy (LP=5) (d) δ on accuracy (LP=5)

Figure 13. Impact of δ on energy and accuracy (Species classification)

6.2.2 *Classification of Bird Vocalization Intensity*

We design the second experiment to test the proposed schemes on more types of sensors. We build a classifier that attempts to predict the intensity of bird vocalizations as a function of the aforementioned six environmental parameters (features): temperature, humidity, light, wind speed, wind direction and air pressure. The ground-truth intensity of bird vocalizations can be measured using microphones located on the nodes. In this experiment, we define three classes (labels), corresponding to three levels of vocalization intensity: 1) High intensity, 2) Medium intensity, and 3) Low intensity. The objective of this experiment is as follows. Given the collected environmental data, among which a small percentage has been labeled into the above three categories, we want to decide the labels of the rest data. The experiment spans a period of one month. Every 10 minutes, the sensors of each node record the sensory readings corresponding to six features of the environment. Thus, at the end of the month, each node has collected about 4000 event readings. During this month, the intensity of bird vocalizations is also measured and averaged over 10 minute intervals. This average value is taken as the ground truth.

In this experiment, we test one more baseline scheme, called *Data Aggregation*. Different from the five schemes evaluated in the preceding experiments, instead of aggregating the decisions, data aggregation directly transports and averages the raw data along the aggregation tree, and applies centralized classification techniques on the averaged data at the sink. The comparison results of accuracy are exhibited in Fig. 14. As can be seen, the accuracy of data aggregation is higher than that of the HAC protocol, but lower than the accuracy of cHAC when all the decisions are delivered to the sink (i.e., $\delta = 1/4$). This is because some information is lost during the process of data aggregation. More importantly, data aggregation consumes about 64.87 joules of energy to deliver the sensory readings to the sink, much larger than the communication energy consumed by the proposed protocols.

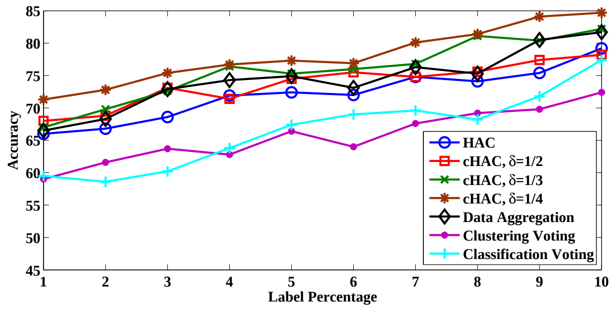
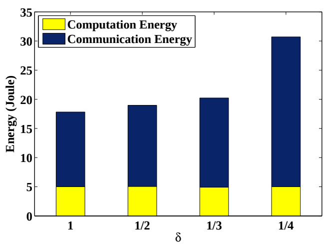
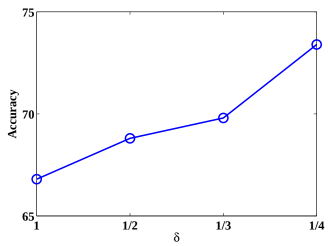


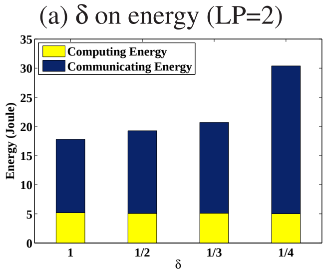
Figure 14. Comparison of accuracy (Intensity classification)



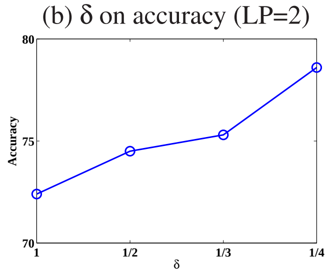
(a) δ on energy (LP=2)



(b) δ on accuracy (LP=2)



(c) δ on energy (LP=5)



(d) δ on accuracy (LP=5)

Figure 15. Impact of δ on energy and accuracy (Intensity classification)

The reason is that for each event, along each tree link our solution forwards only the index of the class to which the event belongs, while data aggregation transmits a vector of up to six numbers. Thus, data aggregation is not an economic solution from the perspective of energy.

The tradeoff between energy and accuracy can be observed in Fig. 15. In this experiment, the ratio of communication energy over computation energy is larger than that of the species classification experiment. This is because in this case the dimension of data (which is 6) is smaller, and thus the clustering algorithms consume less energy. Clearly, by tuning weight balance threshold δ, the cHAC protocol can trade energy for accuracy.

7 Related Work

In sensor networks, data reduction strategies aim at reducing the amount of data sent by each node [17]. Traditional data reduction techniques [17, 18, 31] select a subset of sensory readings that is delivered to the sink such that the original observation data can be reconstructed within some user-defined accuracy. For example, [17] presents a data reduction strategy that exploits the Least-Mean-Square (LMS) to predict sensory readings without prior knowledge or statistical modeling of the sensory readings. The prediction is made at both the sensor nodes and the sink, and each node only needs to send the readings that deviate from the prediction. [31] explores the use of a centralized predictive filtering algorithm to reduce the amount of transmitted data. It eliminates the predictor on sensor nodes. Instead, it relies on a low level signaling system at the sink that instructs the nodes to transmit their data when required. In contrast, the proposed decision aggregation algorithms summarize the sensory readings by grouping similar events into the same clusters, and report only the clustering results to the sink. The proposed schemes focus more on the similarity among the data, and hence can be regarded as a kind of decision level data reduction.

As discussed in the introduction, classification techniques for sensor networks have been widely studied. For example, [12] also studies hierarchical data classification in sensor networks. In this paper, local classifiers built by individual sensors are iteratively enhanced along the routing path, by strategically combining generated pseudo data and new local data. However, as other classification schemes for sensor networks, this paper assumes that a large amount of labeled data are available, which is impractical in sensor networks. To solve the problem of learning from distributed data sets, people have designed methods that can learn multiple classifiers from local data sets in a distributed environment and then combine local classifiers into a global model [32, 33, 34]. However, these methods still focus on learning from a training set consisting of sufficient labeled examples.

Moreover, the proposed problems and solutions in this paper are different from the following work: 1) Data aggregation [24, 25, 35, 36], which combines the data coming from different sensors that detect common phenomena so as to save transmission energy. Data aggregation techniques often apply simple operations, such as average, max and min, directly on the raw data, and thus are different from the proposed decision aggregation schemes which combine the clustering results from multiple sensor nodes. 2) Multisensor data fusion [37, 38], which gathers and fuses data from multiple sensors in order to achieve higher accuracy. Typically, it combines each sensor's vector of confidence probabilities that the observed target belongs to predefined classes. It is similar to supervised classification, since each confidence probability corresponds to a labeled class. 3) Ensemble classification [39, 40], which combines multiple supervised models or integrates supervised models with unsupervised models for improved classification accuracy. Existing ensemble classification methods cannot be directly applied to our problem setting because they conduct centralized instead of distributed classification and they require sufficient labeled data to train the base models.

8 Conclusions

In this paper, we consider the problem of classification with limited supervision in sensor networks. We propose two protocols, HAC and cHAC, which work on tree topologies. HAC let each sensor node locally make cluster analysis and forward the decision to its parent node. The decisions are aggregated along the tree, and eventually the global consensus is achieved at the sink node. As an extension of HAC, cHAC can trade energy for accuracy, and thus is able to provide flexible service to various applications.

9 References

[1] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[2] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2011.

[3] J. Cai, D. Ee, B. Pham, P. Roe, and J. Zhang, “Sensor network for the monitoring of ecosystem: Bird species recognition,” in *ISSNIP*, 2007.

[4] W. Hu, V. N. Tran, N. Bulusu, C. T. Chou, S. Jha, and A. Taylor, “The design and evaluation of a hybrid sensor network for cane-toad monitoring,” in *IPSN*, 2005.

[5] D. Duran, D. Peng, H. Sharif, B. Chen, and D. Armstrong, “Hierarchical character oriented wildlife species recognition through heterogeneous wireless sensor networks,” in *PIMRC*, 2007.

[6] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J. A. Stankovic, T. Abdelzaher, and B. H. Krogh, “Lightweight detection and classification for wireless sensor networks in realistic environments,” in *SenSys*, 2005.

[7] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, “A line in the sand: A wireless sensor network for target detection, classification, and tracking,” *Computer Networks*, vol. 46, pp. 605–634, 2004.

[8] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, “Distributed target classification and tracking in sensor networks,” in *Proceedings of the IEEE*, 2003, pp. 1163–1171.

[9] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, “Wireless sensor networks for habitat monitoring,” in *WSNA*, 2002.

[10] Y. Guo, P. Corke, G. Poulton, T. Wark, G. Bishop-Hurley, and D. Swain, “Animal behaviour understanding using wireless sensor networks,” in *LCN*, 2006.

[11] B. Sheng, Q. Li, W. Mao, and W. Jin, “Outlier detection in sensor networks,” in *MobiHoc*, 2007.

[12] X. Cheng, J. Xu, J. Pei, and J. Liu, “Hierarchical distributed data classification in wireless sensor networks,” in *MASS*, 2009.

[13] E. M. Tapia, S. S. Intille, and K. Larson, “Activity recognition in the home using simple and ubiquitous sensors,” in *Pervasive*, 2004.

[14] K. Lorincz, B.-r. Chen, G. W. Challen, A. R. Chowdhury, S. Patel, P. Bonato, and M. Welsh, “Mercury: a wearable sensor network platform for high-fidelity motion analysis,” in *Sensys*, 2009.

[15] Z. Zeng, S. Yu, W. Shin, and J. C. Hou, “PAS: A Wireless-Enabled, Cell-Phone-Incorporated Personal Assistant System for Independent and Assisted Living,” in *ICDCS*, 2008.

[16] K. C. Barr and K. Asanovic, “Energy aware lossless data compression,” in *MobiSys*, 2003.

[17] S. Santini and K. Römer, “An adaptive strategy for quality-based data reduction in wireless sensor networks,” in *INSS*, 2006.

[18] K. Römer, “Discovery of frequent distributed event patterns in sensor networks,” in *EWSN*, 2008.

[19] J. L. Hill and D. E. Culler, “Mica: A wireless platform for deeply embedded networks,” *IEEE Micro*, vol. 22, no. 6, pp. 12–24, 2002.

[20] Y. Yang, L. Wang, D. K. Noh, H. K. Le, and T. F. Abdelzaher, “Solarstore: enhancing data reliability in solar-powered storage-centric sensor networks,” in *MobiSys*, 2009.

[21] R. Pon, M. A. Batalin, J. Gordon, A. Kansal, D. Liu, M. Rahimi, L. Shirachi, Y. Yu, M. Hansen, W. J. Kaiser, M. Srivastava, S. Gaurav, and D. Estrin, “Networked infomechanical systems: a mobile embedded networked sensor platform,” in *IPSN*, 2005.

[22] L. Girod, M. Lukac, V. Trifa, and D. Estrin, “The design and implementation of a self-calibrating distributed acoustic sensing platform,” in *SenSys*, 2006.

[23] J.-C. Chin, N. S. V. Rao, D. K. Y. Yau, M. Shankar, Y. Yang, J. C. Hou, S. Srivathsan, and S. Iyengar, “Identification of low-level point radioactive sources using a sensor network,” *ACM Trans. Sensor Networks*, vol. 7, pp. 21:1–21:35, October 2010.

[24] S. M. Michael, M. J. Franklin, J. Hellerstein, and W. Hong, “Tag: a tiny aggregation service for ad-hoc sensor networks,” in *OSDI*, 2002.

[25] B. Krishnamachari, D. Estrin, and S. B. Wicker, “The impact of data aggregation in wireless sensor networks,” in *ICDCSW*, 2002.

[26] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1995.

[27] B. Yu, J. Li, and Y. Li, “Distributed data aggregation scheduling in wireless sensor networks,” in *INFOCOM*, 2009.

[28] S. E. Anderson, A. S. Dave, and D. Margoliash, “Template-based automatic recognition of birdsong syllables from continuous recordings,” *The Journal of the Acoustical Society of America*, vol. 100, no. 2, pp. 1209–1219, 1996.

[29] P. Somervuo, A. Harma, and S. Fagerlund, “Parametric representations of bird sounds for automatic species recognition,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, pp. 2252–2263, Nov. 2006.

[30] S. Fagerlund, “Bird species recognition using support vector machines,” *EURASIP J. Appl. Signal Process.*, vol. 2007, pp. 64–64, January 2007.

[31] M. Cordina and C. J. Debono, “Maximizing the lifetime of wireless sensor networks through intelligent clustering and data reduction techniques,” in *WCNC*, 2009.

[32] A. Lazarevic and Z. Obradovic, “The distributed boosting algorithm,” in *KDD*, 2001.

[33] N. V. Chawla, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, “Learning ensembles from bites: A scalable and accurate approach,” *J. Mach. Learn. Res.*, vol. 5, pp. 421–451, 2004.

[34] P. Luo, H. Xiong, K. Lü, and Z. Shi, “Distributed classification in peer-to-peer networks,” in *KDD*, 2007.

[35] J. Gao, L. Guibas, N. Milosavljevic, and J. Hershberger, “Sparse data aggregation in sensor networks,” in *IPSN*, 2007.

[36] L. Su, Y. Gao, Y. Yang, and G. Cao, “Towards optimal rate allocation for data aggregation in wireless sensor networks,” in *MobiHoc*, 2011.

[37] D. L. Hall and J. Llinas, *Handbook of multisensor data fusion*. CRC Press, 2001.

[38] G. Xing, R. Tan, B. Liu, J. Wang, X. Jia, and C.-W. Yi, “Data fusion improves the coverage of wireless sensor networks,” in *MobiCom*, 2009.

[39] J. Gao, F. Liang, W. Fan, Y. Sun, , and J. Han, “Graph-based consensus maximization among multiple supervised and unsupervised models,” in *NIPS*, 2009.

[40] T. Dietterich, “Ensemble methods in machine learning,” in *Proc. 1st Int. Workshop on Multiple Classifier Systems, Lecture Notes in CS, 1857*. Springer, 2000.