Distributed Pre-Processing of Data on Networks of Berkeley Motes Using Non-Parametric EM

Ian Davidson * S. S. Ravi †

Abstract

Inexpensive sensor networks, such as those constructed using Berkeley motes, record many missing or absurd values due to low battery levels, transmission errors, sensor device errors and node failures. However, many mining algorithms do not easily handle data with missing values and the mining literature illustrates that removing records with missing values yields worse results than if missing values are intelligently filled in. A powerful method of filling in missing values is the expectation maximization (EM) algorithm which maximizes the complete (both observed and missing) likelihood. However, typical implementations of EM require a parametric model which is prohibitive for sensor networks as the M-step typically requires collecting and transmitting the expected values for missing data to a central base station. Furthermore, distributing either the E-step or the M-step onto the network is infeasible for parametric models as Berkeley motes have neither a hardware floating point unit (FPU) nor sufficient memory to implement them. In this paper, we develop a non-parametric version of EM specifically for sensor networks. We formally show that the E-step can be solved in polynomial time. Though the problem associated with the M-step is **NP**-complete, a straightforward heuristic is possible. Therefore, our approach provides an example of *generalized* EM. Our preliminary empirical results indicate that the algorithm can restore many of the missing values. Future work will change the likelihood function that EM maximizes to include appropriate mining tasks.

Keywords: Sensor Networks, Missing Values, Mining, Non-parametric, Expectation Maximization.

1 Introduction and Motivation

A sensor network consists of nodes, each of which is equipped with a set of sensors for collecting data from the environment and additional hardware that allows the nodes to communicate with each other in a wireless fashion [1, 16]. In our work, each node is a Berkeley mote that consists of a Mica2 radio board and a MTS300 sensor board which can be used to sense temperature, light intensity and the sound level. The transmission range of each node is not more than 100 feet in ideal situations (open area, no walls or other forms of interference). We treat the sensed values as binary with 0 (false) representing typical indoor readings and 1 (true) representing above typical values. Typical values for the sensed parameters are as follows: light intensity < 500 lumens, temperature < 25.0 degrees Celsius and sound level < 20 decibels. Collecting binary values is quite common for simple sensor networks that are designed to be deployed over areas that are in excess of a square mile [3].

In practice, a sensor network may report many missing or absurd values for some of the quantities measured by the sensors. This may be due to tranmission failures, faulty sensor readings, obstructions of the sensor devices and low battery levels. For example, with our Berkeley mote network in an indoor environment, we found that even when the motes are only about ten feet from the base station, over the course of three hours, approximately three percent of packets were lost. When there are walls or other structures between a mote and the base station, the packet loss is even greater. Furthermore, when polling light and temperature readings every five seconds over the course of three hours, five percent of temperature readings and four percent of light readings were absurd (i.e., represented values that are physically impossible). When the same experiments were conducted with the nodes in a room different from the base-station, the packet loss increases to 23%, and 26% of temperature readings and 21% of light readings were absurd or missing. Therefore, even though Berkeley motes offer great potential for wide-scale deployment, techniques to deal with missing and/or absurd values generated by such networks need to be developed before the sensor network data can be mined.

We could ignore records that contain missing values. However, many mining algorithms do not easily handle missing values. Throwing away records containing

*Department of Computer Science, University at Albany - State University of New York, Albany, NY 12222. Email: davidson@cs.albany.edu.

†Department of Computer Science, University at Albany - State University of New York, Albany, NY 12222. Email: ravi@cs.albany.edu.

missing values removes legitimate values as well. For example, in our experiments mentioned above, with the motes and the base-station being in different rooms, we would have thrown away on average 23% of all records per five second snap-shot of the network. Filling in missing values using even basic schemes leads to improved results [6]. An approach that is used in many commercial data mining tools is to fill in each missing value with the most likely value [7]. This is equivalent to calculating the likelihood probability distribution over the i^{th} column's values (θ_i) from only the *observed data*. Then, the most probable value is chosen to fill in the missing value. Formally, if $y_{l,i}$ denotes the missing value of the l^{th} record's i^{th} column, then $y_{l,i} = argmax_j P(y_{l,i} = j|\theta_i)$.

Better estimates of what the missing values should be are attainable if: 1) More complex models beyond marginal probabilities are used and 2) The missing data values influence the model selection. Formally, this involves calculating the *complete* likelihood (i.e., the likelihood of both the observed and missing data). However, when the missing data is part of the likelihood calculation, no tractable solution for maximum likelihood estimation exists. Instead, a common approach is to use the expectation maximization (EM) algorithm to converge to a local maxima of the complete data likelihood.

Unfortunately, the parametric form of the EM algorithm is not amenable to sensor networks as the second step (the M-step) involves transmitting the expectation of the missing values to a central location (i.e. the base station) for aggregation. Since the EM algorithm may take many iterations to converge, this approach may require many rounds of data transmission, leading to quick depletion of the battery power at the nodes. Furthermore, since motes lack FPU hardware, neither the E-step nor the M-step can be distributed onto the network with parametric models.

In this paper we propose a non-parametric version of EM specifically for sensor networks. Our approach is designed to minimize power consumption and the necessary computations can be distributed over the network. To our knowledge, only two other papers have outlined non-parametric version of EM [2, 17]. The paper by Caruna [2] presents an EM *style* algorithm which only involves one step while the paper by Schumitzky [17] allows only a very restricted model space. Furthermore, both approaches are not readily applicable to sensor networks. We shall focus on how to fill in the missing values so that the resulting sensor network data can be mined using a variety of techniques.

The remainder of the paper is organized as follows. We begin by overviewing parametric and non-parametric EM specifically for sensor networks and then discuss our particular non-parametric model. We formally show that in this model, the computational problem associated with the E-step can be solved in polynomial time and that the corresponding problem for the M-step is **NP**-complete. However, a simple heuristic for the M-step exists, thus our algorithm is an example of *generalized* EM. Other examples of generalized EM include the Baum Welch algorithm commonly used to learn hidden Markov models [4]. We then illustrate approximations that allow distribution of the E and M steps onto the sensor network. Our empirical results presented next, illustrate the usefulness of our algorithm. Finally, we discuss and conclude our work.

2 Parametric and Non-Parametric EM for Sensor Networks

When the sensor network reports a combination of observed (X) and missing (Y) values, our aim is to calculate the complete (missing and observed) maximum likelihood estimate $P(X, Y|\theta)$. This involves finding both the most probable model and actual values for the missing data. No tractable solution for this problem exists; instead, we attempt to calculate the *expectation* of the complete data likelihood $E_{P(Y|X,\theta)}[Y, X|\theta]$. To do this, we use the two step EM algorithm. The first step (the E-step) calculates the expectation of the missing nodes values ($P(Y|X, \theta)$). Given the expectations for the missing values, the second step (the M-step) calculates the value for θ that maximizes the expected complete data likelihood.

Instead of using a parametric model, we use a non-parametric model, namely the Ising spin-glass model, that is commonly used in image processing [5]. The model consists of a set $V = \{v_1, \ldots, v_n\}$ of nodes, one for each sensor node. To be consistent with the Ising spin-glass model, we assume that the nodes whose values are known take on one of the discrete values from $\{-1, +1\}$, instead of 0 and 1. Nodes whose values are missing are assumed to have the value '?'. Extensions that allow the nodes to take on many discrete values are possible and will be left for future work, though as mentioned earlier, it is typical to report binary values from Berkeley motes. Each node v_i has a set $N(v_i)$ of neighboring nodes. Therefore, the non-parametric model of the sensor network can be considered as the graph $G(V, E)$ which represents the nodes and their neighbors. Some nodes will have legitimate filled in values while others will have missing values or absurd values that can be treated as missing. In contrast to our non-parametric model, a parametric model could identify a subset of key nodes, with the sensor values at other nodes being modeled according to their distance to these key nodes and by a parametric probability

distribution (e.g. a Gaussian with parameters μ_x, σ_x, μ_y and σ_y). We now derive the probability density (mass) function for a sensor network using our non-parametric model. The the probability mass function ($P(G)$) of a particular configuration of sensor values is a function of the Hamiltonian ($H(G)$) of the underlying graph. Equations defining $H(G)$ and $P(G)$ are given below.

DEFINITION 2.1. *Let $G(V, E)$ be an undirected graph. Suppose each node $v_i \in V$ is assigned a label $\ell_i \in \{+1, -1\}$.*

$$H(G) = -\sum_{\{v_i, v_j\} \in E} \ell_i \, \ell_j. \tag{2.1}$$

$$P(G) = 1 - \frac{2^{H(G)}}{2^{|E|}} \tag{2.2}$$

As can be seen, our probability density function is a function of the edges (E) in the graph and the node labels ($\ell_1, \ldots, \ell_n$). As in the Ising spin-glass model, we sometimes refer to $H(G)$ as the **energy** associated with the given configuration. Note that the smaller the value of energy, the larger is the probability of the corresponding configuration and that the smaller the number of conflicts between a node's value and its neighbors, the lower the energy. Our model, namely the graph structure, contains no continuous parameters. Hence we use the term non-parametric to describe the model.

3 Our Non-Parametric EM Algorithm

We present our non-parametric EM algorithm along with a sketch of how the E and M steps were derived. Later sections provide the formal details of these derivations.

A key problem with the EM algorithm is initialization. However, in our situation, we can choose an initial model (graph) that is obtained by connecting each node to every other node that is within its transmission distance (100 feet for Berkeley motes). Our EM algorithm will then iteratively fill in the expected missing values and change the model by removing nodes so as to increase the probability.

3.1 The E-Step - A Sketch Given a graph $G(V, E)$, where V is the set of nodes in the sensor network, and a subset P of nodes that have fixed values, the goal of the E-step is to fill in the values for the nodes in $V - P$.

Calculating the expected values for the missing value nodes is equivalent to minimizing the Hamiltonian $H(G)$ shown in Equation (2.1). Minimizing $H(G)$ is achieved by filling in the missing values so as to minimize the number of conflicts (differences) between neighbors. Since missing value nodes can be neighbors of other missing value nodes, filling in the missing values is not straightforward. Fortunately, this computation is tractable using the following approach. (For simplicity, we leave out several technical details in the following discussion. These details are provided in Section 4.3.)

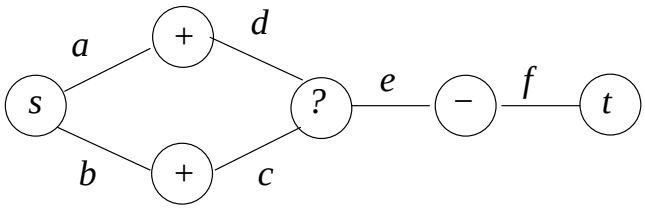

Figure 1: A Simple Illustration Why the MinCut of G_1 Minimizes the Hamiltonian

To the graph G, we add two nodes (s and t), where s has the value $+1$ and t the value -1. All of the nodes in P whose value are $+1$ are connected to s and those whose value are -1 are connected to t. This new graph is called G_1. All edges have unit weights, except that the edges involving either s or t have their weight as ∞. Then, a minimum weight edge cutset of G_1 is the minimum number of edges whose removal will create two node-disjoint subgraphs: a positive subgraph where each node has the value $+1$ or '?' and a negative subgraph where each node has the value -1 or '?'. Determining the minimum weight edge cutset of a graph can be done in polynomial time (see for example, [18]). The subgraph that a missing value node is part of determines the value to be assigned to the node. Figure 1 shows the intuition behind why a minimum cut of the graph is needed. Clearly, in that figure, the missing value should be filled in as '+'. Removing edges marked c and d creates two appropriate subgraphs but it produces the wrong missing value; removing just the edge marked e corresponds to a minimum cut and hence produces the correct result. The E-step can be approximated and easily distributed onto the network by allowing a node with a missing value to set its value to the most commonly occurring value among its neighbors. This approximation will return exactly the same solution as the one based on minimum cut, unless two missing value nodes are neighbors of each other. When there are edges between two nodes with missing values, the two approaches *may* return different results.

Our algorithm shown in Figure 2 formalizes the above discussion. (Some details in the figure rely on definitions presented in Section 4.3.)

3.2 The M-Step - A Sketch Here, we are given a graph $G(V, E)$ with all node values instantiated. (Recall

Input: An undirected graph $G(V, E)$; a subset P of nodes such that each node in P has been assigned a label from $\{+1, -1\}$. (These labels cannot be changed.)

Output: A label from $\{+1, -1\}$ for each node of $V - P$ such that $H(G)$ is minimized.

Algorithm:

1. Construct the auxiliary graph $G_1(V_1, E_1)$ from G.
2. Find an s-t edge cutset C^* of minimum weight in G_1.
3. Construct graph $G_2(V_2, E_2)$, where $V_2 = V_1$ and $E_2 = E_1 - C^*$, from G_1 by deleting the edges in C^*.
4. **for** each node $v \in V - P$ **do**

   if there is a path from s to v in G_2
   then Assign the label $+1$ to v
   else Assign the label -1 to v.

Figure 2: Algorithm for the E-step

Input: An undirected graph $G(V, E)$ where each node in V has been assigned a label from $\{+1, -1\}$; a subset $P \subseteq V$ of nodes such that there is a path in G from from each node in $V - P$ to some node in P.

Output: Graph G' obtained from G by deleting a subset P' of P. (In G' also, there is a path from from each node in $V - P$ to some node in $P - P'$.) The procedure tries to obtain a graph G' whose energy is as low as possible.

Note: The g function used in Step 2(a) below is defined in Section 5.

Procedure:

1. Choose a positive integer k. (The running time increases with k.)
2. **loop**
   (a) By trying all subsets of P up to size k, find a subset P' such that $g(P') > 0$ and deleting P' does not violate the path property mentioned above. If there is no such subset, go to Step 3.
   (b) Delete the nodes in P' from P. (The graph also needs to be modified accordingly.)
3. Output the resulting graph G'.

Figure 3: Heuristic Procedure for the M-step

that all missing values were filled in by the E step.) We must now compute the most probable model (G^*) for this data. Since our model space is the set of all possible graphs with *upto* $|V| = n$ nodes, we can change the graph G by deleting edges or nodes. In this paper, we have chosen to remove nodes as this approach is more amenable to our aim of reducing power consumption. The nodes removed from G to obtain G^* are put to sleep to conserve power. These nodes may be subsequently awakened for other applications.

In Section 5, we show that the problem of determining which nodes to remove so as to minimize the Hamiltonian is **NP**-complete. Therefore, obtaining an exact solution to this problem (i.e. finding G^* that maximizes Equation (2.2)) is computationally intractable. However, we can easily find a graph structure G' that is more probable than G using the algorithm shown in Figure 3. (The definition of the g function used in Step 2(a) of the algorithm is given in Section 5.) If the variable k used in that figure is chosen to be 1, then the computation can be distributed onto the network, since each node needs to check only its immediate neighbors to determine whether it should go to sleep. Our M-step is an example of *generalized* EM [15], since the calculation yields a *more* probable model, but not necessarily the *most* probable model. Nodes which are put to sleep can be considered as outliers since removing them from the network increases the probability of the entire network/graph.

4 Derivation of the E-Step

4.1 Viewing E-Step as Energy Minimization In the E-step, we need to calculate the expected values of the nodes with missing values. Using a non-parametric model, this is simply determined by the neighbors of the node. Formally:

$$\mathrm{P}(v_i = j) = \sum_{x \in N(v_i)} \frac{1 - \delta(j, x)}{|N(v_i)|} \tag{4.3}$$

where $N(v_i)$ denotes the neighbors of node v_i and δ is the Kronecker delta function ($\delta(p, q) = 1$ if $p = q$ and 0 otherwise).

A common simplification [14] of the E step is to use a "winner take all" approach. This leads to the following E step.

$$\begin{aligned} v_i &= 1 \text{ if } argmax_j P(v_i = j) = 1 \\ &= 0 \text{ otherwise} \end{aligned}$$

Filling in the missing values with their most probable values is equivalent to calculating the *ground state* of the spin-glass model. The ground state involves setting the missing values in such a way that the probability of the configuration (given by Equation (2.2)) is maximized. This, in turn, is equivalent to minimizing the energy value given by Equation (2.1). We now explain how this minimization problem can be solved in polynomial time.

4.2 Graph Theoretic Preliminaries The model space considered in this paper consists of undirected graphs that are simple in the sense that they have no multiedges or self-loops. Let $G(V, E)$ be an undirected graph. When each node $v_i \in V$ is assigned a value $\ell_i \in \{+1, -1\}$, the Hamiltonian function (or the **energy function**) $H(G)$ (Equation (2.2)) can be rewritten using the following definitions.

(a) Each edge $\{v_i, v_j\}$ such that $\ell_i \neq \ell_j$ is called a **conflict edge**.

(b) Each edge $\{v_i, v_j\}$ such that $\ell_i = \ell_j$ is called an **agreement edge**.

LEMMA 4.1. *Let N_c and N_a denote respectively the number of conflict and agreement edges in G. Then, $H(G) = N_c - N_a$. Alternatively, $H(G) = 2\,N_c - |E|$.*

Proof: From the expression for $H(G)$ (Equation (2.1)), it is easy to see that each conflict edge contributes $+1$ to $H(G)$ and that each agreement edge contributes -1 to $H(G)$. Therefore, $H(G) = N_c - N_a$. Since each edge is either a conflict edge or an agreement edge, we have $|E| = N_c + N_a$. Therefore, $H(G)$ is also equal to $2\,N_c - |E|$. ■

The following is an easy observation which will be used later.

OBSERVATION 4.1. *Suppose $G(V, E)$ is an undirected graph where each node is labeled $+1$ or -1. If there is a path in G between a node labeled $+1$ and a node labeled -1, then the path has at least one conflict edge.* ■

4.3 An Efficient Algorithm for the E-Step The E-step of the EM algorithm for filling in missing sensor node values solves the following combinatorial problem.

Minimum Energy Label Assignment (MELA)

Instance: An undirected graph $G(V, E)$ and a subset $P \subseteq V$ of "preassigned" nodes; that is, each node in P has been assigned a label from $\{+1, -1\}$.

Requirement: Assign a label from $\{+1, -1\}$ to each node in $V - P$ such that $H(G)$ is minimized.

It is assumed that in G, there is a path from each node in $V - P$ (i.e., each node with a missing value) to a node in P. This assumption enables the E-step to assign values to missing nodes in an unambiguous fashion.

Our algorithm for the MELA problem relies on a transformation to the problem of finding a **minimum weight $s - t$ edge cut** in an undirected graph. The definition of such an edge cut is given below.

DEFINITION 4.1. *Let $G(V, E)$ be an undirected graph with a nonnegative weight $w(e)$ for each edge $e \in E$. Let s and t be two distinct vertices in V. An **s-t edge cutset** for G is a subset $E' \subseteq E$ such that in the graph $G'(V, E - E')$, there is no path between s and t. A **minimum weight s-t edge cutset** for G is an edge cutset whose total weight is minimum.*

The following well known result shows that minimum weight edge cutsets can be found efficiently (see for example [18]).

THEOREM 4.1. *Given an undirected graph $G(V, E)$ with a nonnegative weight $w(e)$ for each edge $e \in E$ and two distinct vertices s and t in V, a minimum weight s-t edge cutset for G can be computed in $O(|E|+|V| \log |V|)$ time.* ■

Recall that in the MELA problem, the nodes in the set $P \subseteq V$ have preassigned labels which cannot be changed. Throughout this section, we will use E^P denote the set of edges where each edge has both of its endpoints in P. Let N_c^P and N_a^P denote the number of conflict and agreement edges in E^P. (Thus, $|E^P| = N_c^P + N_a^P$.) The contribution H^P of the edges in E^P to the Hamiltonian of the graph G is therefore given by $H^P = N_c^P - N_a^P$. Note that no matter how labels are assigned to the nodes in $V - P$, the edges in E^P will always contribute H^P to the value of $H(G)$.

We now discuss how the MELA problem can be solved efficiently. Let $G(V, E)$ and $P \subseteq V$ denote the given instance of the MELA problem. Consider the auxiliary edge weighted graph $G_1(V_1, E_1)$ constructed from G as follows.

(a) $V_1 = V \cup \{s, t\}$, where s and t are two new nodes (i.e., $s \notin V$ and $t \notin V$).

(b) $E_1 = (E - E^P) \cup E_s \cup E_t$, where $E_s = \{\{s, v_i\} : v_i \in P \text{ and } \ell_i = +1\}$, and $E_t = \{\{t, v_i\} : v_i \in P \text{ and } \ell_i = -1\}$.

(c) For each edge $e \in E_1$, the weight of e, denoted by $w(e)$ is chosen as follows: if $e \in E$, then $w(e) = 1$; otherwise, $w(e) = \infty$.

We note that the auxiliary graph G_1 has a trivial s-t edge cutset of weight $|E - E^P|$. Thus, no minimum weight s-t edge cutset of G_1 can use any of the edges incident on the nodes s and t. In other words, any minimum weight s-t edge cutset of G_1 is a subset of $E - E^P$. The following lemma shows the role played by auxiliary graph in solving the MELA problem.

LEMMA 4.2. *Let $G(V, E)$ and $P \subseteq V$ constitute a given instance of the* MELA *problem. Let $H^*(G)$ denote the minimum value of the Hamiltonian function over all assignments of labels to the nodes in $V - P$. Let $G_1(V_1, E_1)$ denote the auxiliary graph of G constructed as discussed above and let W_1^* denote the minimum weight of an s-t edge cutset in G_1. Then, $H^*(G) = H^P + 2\,W_1^* - |E - E^P|$, where H^P is the contribution due to the edges in E^P.*

Proof: We prove that result in two parts.

Part 1: Here, we prove that $H^*(G) \geq H^P + 2\,W_1^* - |E - E^P|$. Consider an assignment of labels from $\{+1, -1\}$ to the nodes in $V - P$ such that the value of $H(G)$ is equal to $H^*(G)$. Let C denote the set of all the conflict edges from $E - E^P$ in the resulting assignment. As mentioned earlier, the edges in E^P contribute H^P to $H(G)$, regardless of the label assignment to the nodes in $V - P$. From Lemma 4.1, the contribution to the Hamiltonian due to the edges in $E - E^P$ is $2|C| - |E - E^P|$. Therefore, $H^*(G) = H^P + 2|C| - |E - E^P|$. Now, we have the following claim.

Claim: C is an s-t edge cutset for G_1.

Proof of Claim: Suppose C is not an s-t edge cutset for G_1. Then, there is a path from s to t in the graph $G_2(V_1, E_1 - C)$. In this path, let x be the node that is adjacent to s and let y be the node that is adjacent to t. Thus, the label of x is $+1$ and that of y is -1. Hence, by Observation 4.1, there is a conflict edge in this path. By our construction of graph G_1, this conflict edge is from the edge set $E - E^P$. This contradicts the assumption that C contains all the conflict edges from $E - E^P$, and the claim follows.

In view of the claim, G_1 has an s-t edge cutset of weight at most $|C|$. Since W_1^* is the minimum weight of an s-t edge cutset of G_1, we have $|C| \geq W_1^*$. Therefore, $H^*(G) = H^P + 2\,|C| - |E - E^P| \geq H^P + 2\,W_1^* - |E - E^P|$. This completes the proof of Part 1.

Part 2: Here, we prove that $H^*(G) \leq H^P + 2\,W_1^* - |E - E^P|$. This is done by finding an assignment of labels to the nodes in $V - P$ such that the value of the Hamiltonian for the resulting assignment is at most $H^P + 2\,W_1^* - |E - E^P|$.

Consider algorithm in Figure 2. Using the assumption that there is a path in G from each node in $V - P$ to a node in P, it is easy to see that Step 4 of the algorithm assigns a label from $\{+1, -1\}$ to each node of $V - P$. Further, the assignment ensures that the only conflict edges from $E - E^P$ in the resulting assignment are those in C^*. Therefore, by Lemma 4.1, the value of the Hamiltonian function $H(G)$ for this assignment of labels to the nodes in $V - P$ is given by $H(G) = H^P + 2\,|C^*| - |E - E^P|$. Since C^* is a minimum weight s-t edge cutset, we have $|C^*| = W_1^*$. Therefore, $H(G) = H^P + 2\,W_1^* - |E - E^P|$. Since there is an assignment of labels to the nodes in $V - P$ such that the Hamiltonian function of G has a value of $H^P + 2\,W_1^* - |E - E^P|$, it follows that $H^*(G) \leq 2\,W_1^* - |E - E^P|$. This completes the proof of Part 2 as well as that of the lemma. ■

A direct consequence of the above lemma is that the algorithm in Figure 2 computes an optimal solution to the MELA problem. The running time of the algorithm is dominated by Step 2, where a minimum weight s-t edge cutset of G_1 is constructed. As mentioned in Theorem 4.1, this step can be carried out in $O(|E| + |V| \log |V|)$ time. The following theorem summarizes the above discussion.

THEOREM 4.2. *The E-step of the EM algorithm for filling in the missing sensor values can be solved in $O(|E| + |V| \log |V|)$ time, where $|V|$ is the number of nodes and $|E|$ is the number of edges in the given sensor network.* ■

5 Derivation of the M-Step

The M-step requires finding the most probable model given the observed and filled in missing values (from the E-step). We model the M-step by a graph problem where the goal is to delete a set of nodes so that the energy of the resulting graph is as small as possible. Since the goal of the EM algorithm is to fill in the missing values, only nodes that had preassigned labels in the E-step are candidates for deletion. (In other words, nodes whose values are to be filled in cannot be deleted.) When a node is deleted from a graph, all the edges incident on that node are also deleted. Recall that in the graph used in the E-step, there must be a path from each node with a missing value to a node with a value $+1$ or -1. Therefore, the deletion process used in the M-step must ensure that this path property continues to hold in the resulting graph. A precise formulation of the problem is as follows.

Minimum Energy Node Deletion (MEND)

Instance: An undirected graph $G(V, E)$ where each node in V has been assigned a label from $\{+1, -1\}$; a subset $P \subseteq V$ of nodes with preassigned values. (Note: The E-step assigned values to the nodes in $V - P$.)

Requirement: Find a subset $P' \subseteq P$ so that in the graph G' obtained by deleting from G the nodes in P', each node in $V - P$ has a path to some node in $P - P'$ and $H(G')$ is a minimum over all such subsets.

As stated above, MEND is an optimization problem. A decision version of the problem can be obtained in an obvious manner by introducing a parameter B and changing the requirement to the following question: Is a set of nodes P' whose deletion produces a graph G' such that G' has the path property mentioned above and the energy of G' is at most B? For convenience, we will use MEND to denote both the decision and optimization versions of the problem. (The usage will be clear from the context.)

We now show that the MEND problem is computationally intractable. Our proof uses a reduction from the following problem which is known to be **NP**-complete [12].

Exact Cover by 3-Sets (X3C)

Instance: A set $X = \{x_1, x_2, \ldots, x_n\}$, where $n = 3q$ for some positive integer q; a collection $Y = \{Y_1, Y_2, \ldots, Y_m\}$ of subsets of X, where $|Y_j| = 3$, $1 \leq j \leq m$.

Question: Is there a subcollection $Y' = \{Y_{j_1}, Y_{j_2}, \ldots, Y_{j_q}\}$ consisting of q sets such that the union of the sets in Y' is equal to X?

THEOREM 5.1. *The* MEND *problem is* ***NP****-complete.*

Proof: It is easy to see that MEND is in **NP**. To prove **NP**-hardness, we use a reduction from the X3C problem defined above. Given an instance I of the X3C problem, we create an instance I' of the MEND problem as follows. For each element $x_i \in X$, we create a node v_i, $1 \leq i \leq n$. Similarly, for each subset $Y_j \in Y$, we create a node w_j, $1 \leq j \leq m$. Let $V_1 = \{v_1, v_2, \ldots, v_n\}$ and $V_2 = \{w_1, w_2, \ldots, w_n\}$. The node set V for the graph $G(V, E)$ is given by $V = V_1 \cup V_2$. The edge set E is constructed as follows. If $Y_j = \{x_{j_1}, x_{j_2}, x_{j_3}\}$, then we add the edges $\{w_j, v_{j_1}\}$, $\{w_j, v_{j_2}\}$ and $\{w_j, v_{j_3}\}$ to the graph. This completes the construction of the graph G. Each node w_j is assigned the label $+1$ ($1 \leq j \leq m$) and each node v_i is assigned the label -1 ($1 \leq i \leq n$). The set P from which nodes can be deleted is V_2. Note that each edge in the graph is a conflict edge. Further, the graph has exactly $3m$ edges. Therefore, the initial energy value of G is $3m$. The energy bound B for the graph after node deletion is set to n. This completes the construction of the MEND problem instance I'. It is clear that the construction of I' can be carried out in polynomial time. We now argue that there is a solution to the MEND instance I' if and only if there is a solution to the X3C instance I.

Suppose there is a solution to the X3C instance I given by $Y' = \{Y_{j_1}, Y_{j_2}, \ldots, Y_{j_q}\}$. A solution to the MEND instance is obtained by deleting from G all the nodes from V_2 *except* w_{j_1}, w_{j_2}, ..., w_{j_q}. Let G' denote the resulting graph. Note that each set in Y' has exactly three elements and Y' is a solution to the X3C instance I. Therefore, in G', each node from V_2 is of degree three and each node from V_1 is of degree one. Hence, G' has the path property mentioned above. Further, the number of edges in G' is exactly n, and each edge is a conflict edge. Therefore $H(G') = n$ as required. Thus, we have a solution to the MEND instance I'.

Now, suppose there is a solution consisting of the node set P' to the MEND instance I'. Let G' denote the graph after the nodes in P' are deleted from G. We have the following claim.

Claim: G' contains exactly $q = n/3$ nodes from V_2.

Proof of Claim: Let V_2' denote the set of nodes from V_2 in G'. We have two cases to consider.
Case 1: Suppose $|V_2'| < q$. Note that each node in V_2' is of degree three. Thus, the total number of edges from the nodes in V_2' is at most $3(q-1) = 3q - 3 < n$. These $3q - 3$ edges are incident on the n nodes in V_1. Thus, at least one node in V_1 has degree zero. Such a node does not have a path to a node in $P - P'$. This is a contradiction since the solution to I' must satisfy the path property.
Case 2: Suppose $|V_2'| > q$. Again, each node in V_2' is of degree three. Thus, the total number of edges from the nodes in V_2' is at least $3(q+1) = 3q + 3 > n$. Each of these is a conflict edge and there are no agreement edges in G'. Thus, the energy of G' is greater than n. This is a contradiction since the solution to I' has an energy value of at most n. The claim follows.

From the above claim, it can be seen that each node from V_2 in G' has a degree of three and that each node in V_1 has a degree of one. It follows that the sets corresponding to the nodes from V_2 in G' form a solution to the X3C instance I. This completes the proof of Theorem 5.1. ■

In view of Theorem 5.1, it is unlikely that the M-step of the EM algorithm can obtain a global minimum energy configuration in polynomial time. So, it is reasonable to look for heuristic methods that reduce the energy iteratively and reach a local minimum. We discuss one such method below.

Suppose we are given an undirected graph $G(V, E)$ with a $+1$ or -1 label for each node and a subset $P \subseteq V$ of candidate nodes that can be deleted. Consider any nonempty subset P^1 of P. Of the edges incident on the nodes in P^1, let N_c^1 and N_a^1 denote respectively the number of conflict edges and agreement edges. Define the **gain** of P^1, denoted by $g(P^1)$, to be the value

$N_c^1 - N_a^1$. Let us call P^1 a **candidate deletion set** if $g(P^1)$ is *greater than* zero and the graph G' obtained by deleting P^1 from G continues to have the path property mentioned above. The reason why P^1 is useful is that the graph G' obtained by the deleting P^1 has a smaller energy value than G. (In fact, $H(G') = H(G) - g(P^1)$.) It will be prohibitively expensive in terms of running time to find a candidate deletion set by trying all possible subsets of P. To make the process efficient, we *fix* an integer k and try only subsets of size at most k. Thus, in each iteration, this method will examine $O(|P|^k)$ subsets. This is reasonable for small values of k. As mentioned earlier, for $k = 1$, the computation can be distributed over the sensor network. When a candidate deletion set is found, the deletion of the corresponding nodes is guaranteed to decrease the energy. When no candidate deletion set of size at most k is available, the procedure stops with a local minimum solution. An outline of the resulting heuristic algorithm is shown in Figure 3.

6 Empirical Results

We present preliminary results for our work. Using the TOSSIM sensor network simulator and MATLAB, we created an artifical sensor network on a uniformly spaced 50×50 grid, with a sensor node at each grid point. Each node has its eight immediate neighbors as its initial neighborhood. It is helpful to remember the our M step removes nodes from the network to increase the probability of the entire network. These removed nodes can be considered as anomalies/outliers.

For our simple *BOX* example (see Figure 4), we randomly removed 10% of all values and then applied our distributed non-parametric EM algorithm to restore them. Examples of networks with missing and then restored values are shown in Figures 5 and 6. Note that a few nodes around the boundary of the simple pattern are turned off. We repeated this experiment ten times. As expected, we found that on average only 0.4% of all missing values were restored to their **incorrect** values for this simple problem.

For our *CIRCLES* example (see Figure 7), we randomly removed 10% of all values and then applied our algorithms to restore them. Examples before and after the restoration data sets are shown in Figures 8 and 9. We found that on average only 1.0% of all missing values were **not** restored to their correct values. We also found that many of the nodes along the boundaries of adjacent circles were turned off as they can be considered anomalous.

As a test of our algorithm's ability to turn off nodes that are outliers, we constructed a *RANDOM* example (see Figure 10) where each node's value is generated randomly. We then randomly removed 10% of all values and applied our algorithms to restore them. Examples before and after restoration data sets are shown in Figures 11 and 12. We found that on average, 2.1% of all missing values were restored to their incorrect values. More importantly, our algorithm turned off the vast majority of nodes indicating that most of the node values were anomalies as would be expected for random data.

7 Conclusion and Future Work

We have presented preliminary results from a line of research for mining using the resource constrained Berkeley mote sensor network platform. We believe that this is an important area as Berkeley motes are an inexpensive and popular sensor network platform that is commercially available in kit form. The basic sensing boards can record temperature, light and noise. Often this information is converted to binary data according to a user settable threshold. The information generated from these motes regularly contains missing or absurd values due to transmission errors, low battery levels, sensor reading errors and node failures. However, many mining algorithms do not easily handle missing data.

In this paper, we examined the use of EM algorithm to fill in the missing values as is the standard practice in statistics. However, parametric EM would quickly consume the battery life of the motes as it requires repeated transmission of the expected values to a central base-station for aggregation. Instead, we propose a novel *non-parametric* EM algorithm that is motivated by the Lattice spin-glass literature. The E-step in our approach leads to a problem that can be solved efficiently. Even though the problem associated with the M-step is is computationally intractable, we proposed a heuristic that finds a more probable (but not necessarily the most probable) model. Therefore, our approach is an example of *generalized* EM [15].

A significant benefit of using non-parametric models is that the E and M steps can be distributed onto the sensor network. This would not be possible for parametric EM since the motes lack floating point hardware and the memory-space needed to execute the corresponding algorithms.

Our distributed non-parametric EM algorithm, functionally, fills in the values for nodes that do not report a value or report an absurd value. In addition, since our M step removes nodes from the network/graph to increase the overall probability, we are effectively removing outliers. Future work will investigate more complex likelihood functions that incorporate the mining task as well data pre-processing. For example, clustering is easily facilitated by the introduction of a latent vari-

able (Q) which is the cluster ID, for each node. The complete data likelihood to maximizes would then be $P(Q, X, Y|\theta)$.

Our preliminary experimental results show that for synthetic sensor network data, the algorithm is capable of restoring the vast majority of missing values correctly. However, additional empirical work must be done to determine and overcome the pragmatic problems in the approach.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless Sensor Networks: A Survey", *Computer Networks*, Vol. 38, 2002, pp. 393–422.

[2] R. Caruna, "A Non-Parametric EM-Style Algorithm for Imputing Missing Values", AI-Stats 2001.

[3] *Communications of the ACM*, Special Issue on Wireless Sensor Networks, June 2004.

[4] R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification* (2nd edition), Wiley Interscience, 2000.

[5] I. Davidson and G. Paul, "Locating Secret Messages in Images", 10^{th} SIG KDD Conference 2004.

[6] P. Domingos and M. Pazzani, "Beyond independence: Conditions for the optimality of the simple Bayesian Classifier", ICML, 1996.

[7] H. Edelistein, Two Crows Data Mining Report, Two Crows Corporation 1999.

[8] E. Elnahrawy and B. Nath, "Poster Abstract: Online Data Cleaning in Wireless Sensor Networks", *Proc. SenSys'03*, Los Angeles.

[9] E. Elnahrawy and B. Nath, "Cleaning and Querying Noisy Sensors", *Proc. WSNA'03*, San Diego, CA.

[10] E. Elnahrawy and B. Nath, "Statistical Approaches to Cleaning Sensor Data", book chapter in *Distributed Sensor Networks*, Edited by S. S. Iyengar and R. R. Brooks, CRC Press, 2003.

[11] E. Elnahrawy and B. Nath, "Context-Aware Sensors", *Proc. European Workshop on Wireless Sensor Networks*, 2003, pp. 77–93.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman and Co., San Francisco.

[13] *IEEE Computer*, Special Issue on Sensor Networks, Aug. 2004.

[14] M. Kearns, Y. Mansour, and A. Y. Ng, . "An information-theoretic analysis of hard and soft assignment methods for clustering", *Proc. UAI 1997*.

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

[16] C. S. Raghavendra, K. M. Sivalingam and T. Znati (Editors), *Wireless Sensor Networks*, Kluwer Academic Publishers, Norwell, MA, 2004.

[17] Schumitzky, A., NonParametric EM Algorithms for Estimating Prior Distributions, TR 90-2, Department of Mathematics, University of Southern California.

[18] M. Stoer, F. Wagner, "A Simple Min-Cut Algorithm", *J. ACM*, Vol. 44, No. 4, July 1997, pp. 585–591.

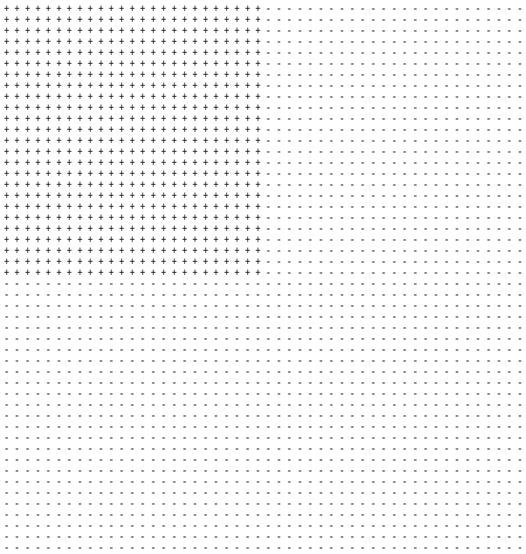

Figure 4: Original Box Data for a 50×50 uniformally spaced sensor network.

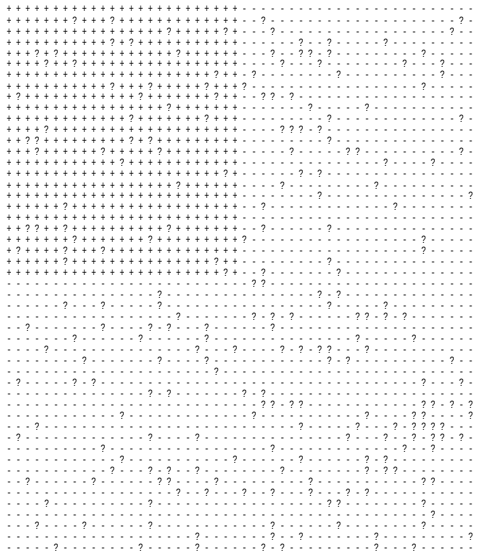

Figure 5: Box Data with Missing Values for a 50×50 uniformally spaced sensor network. The symbol '?' indicates a missing value.

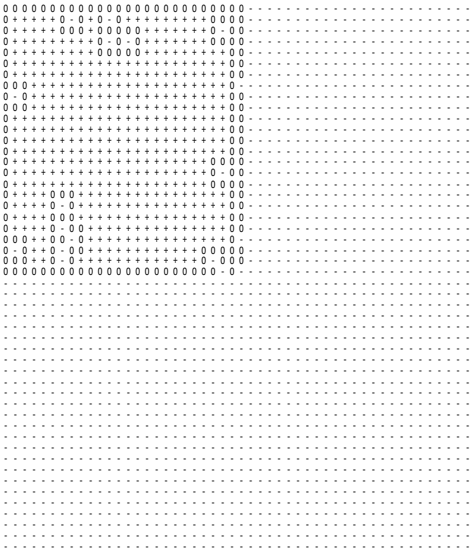

Figure 6: Recovered Box Data for a 50×50 uniformally spaced sensor network. The symbol 'O' indicates a node to be turned off.

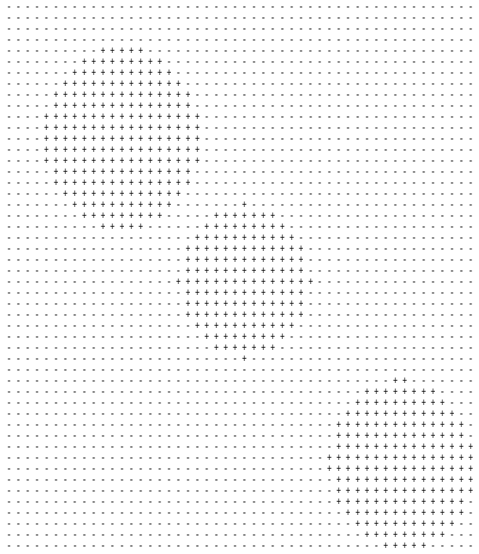

Figure 7: Original Multiple Circles Data for a 50×50 uniformally spaced sensor network.

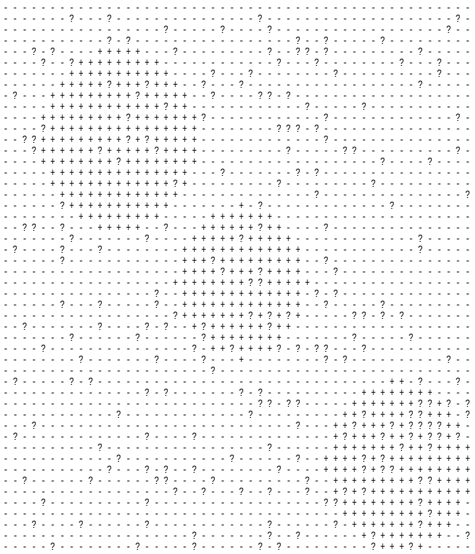

Figure 8: Multiple Circles Data with Missing Values for a 50×50 uniformally spaced sensor network. The symbol '?' indicates a missing value.

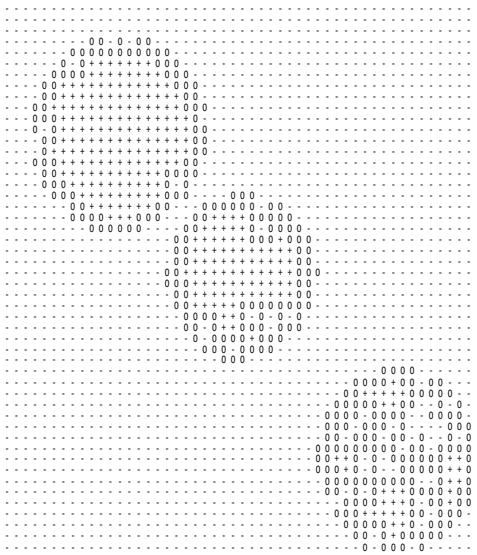

Figure 9: Recovered Multiple Circles Data for a 50×50 uniformally spaced sensor network. The symbol 'O' indicates a node to be turned off.

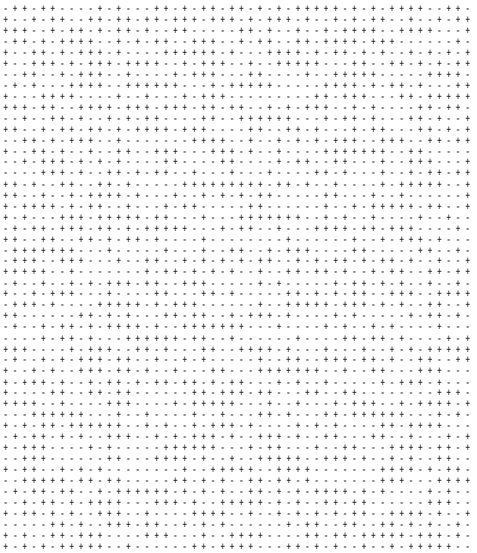

Figure 10: Original Random Data for a 50×50 uniformally spaced sensor network.

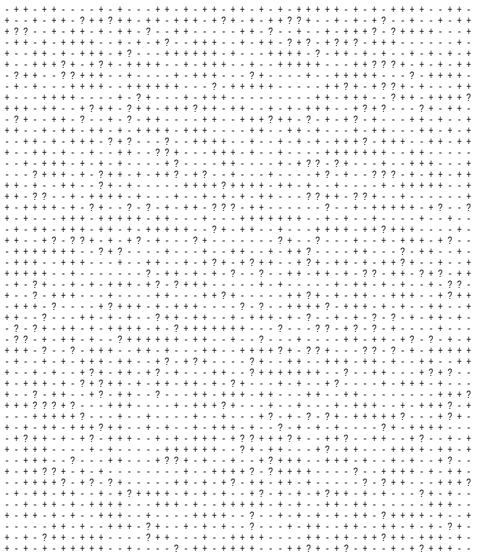

Figure 11: Random Data with Missing values for a 50×50 uniformally spaced sensor network. The symbol '?' indicates a missing value.

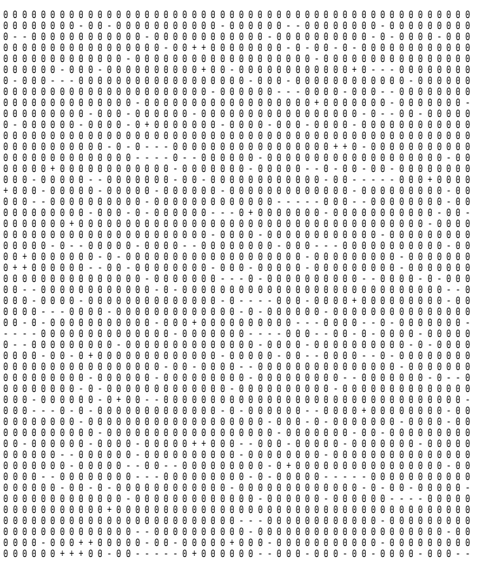

Figure 12: Recovered Random Data for a 50×50 uniformally spaced sensor network. The symbol 'O' indicates a node to be turned off.