Self-Organizing Energy Aware Clustering of Nodes in Sensor Networks Using Relevant Attributes

Marwan Hassani [•] Emmanuel Müller [•] Pascal Spaus [•] Adriola Faqolli
Themis Palpanas [○] Thomas Seidl [•]

[•]Data Management and Data Exploration Group
RWTH Aachen University, Germany
{hassani, mueller, seidl}@cs.rwth-aachen.de

[○]Department of Computer Science
University of Trento, Italy
themis@disi.unitn.eu

ABSTRACT

Physical clustering of nodes in sensor networks aims at grouping together sensor nodes according to some similarity criteria like neighborhood. Out of each group, one selected node will be the group representative for forwarding the data collected by its group. This considerably reduces the total energy consumption, as only representatives need to communicate with distant data sink. In data mining, one is interested in constructing these physical clusters according to similar measurements of sensor nodes. Previous data mining approaches for physical clustering concentrated on the similarity over all dimensions of measurements.

We propose ECLUN, an energy aware method for physical clustering of sensor nodes based on both spatial and measurements similarities. Our approach uses a novel method for constructing physical clusters according to similarities over some dimensions of the measured data. In an unsupervised way, our method maintains physical clusters and detects outliers. Through extensive experiments on synthetic and real world data sets, we show that our approach outperforms a competing state-of-the-art technique in both the amount of consumed energy and the effectiveness of detecting changes in the sensor network. Thus, we achieve an overall significantly better life times of sensor networks, while still following changes of observed phenomena.

Categories and Subject Descriptors

H.2.8 [**Database management**]: Database applications—*Data mining*

General Terms

Algorithms, Management

Keywords

Physical Clustering, Relevant Attributes, Subspace Clustering, Sensor networks, Energy Efficiency, Change detection

SensorKDD'10, July 25, 2010, Washington, DC, USA.

1. INTRODUCTION

The communication process is the dominating energy consumer in sensor networks, particularly when this is happening over long distances. Sensor nodes need to use their full sending power to forward their sensed data to distant sink, while they can use less power when communicating locally with each other. Considering the energy limited resources in sensor networks, this motivated a lot of research on the physical clustering of sensor nodes. The idea is to divide sensor nodes into groups according to some criteria, and then selecting one node from each of these group to serve a group representative. The main task of the group representative is forwarding the readings of sensor from its group to this distant sink. As nodes need to communicate within the group (the cluster) using less energy, this considerably reduces the total consumed energy in the whole network.

Data mining approaches contributed to this problem mainly in two parts: the criteria used for clustering and the process of selecting representatives. The similarity of sensed measurements and spatial characteristics were used as a grouping measure. Thus, inside each cluster, the node with the most similar readings to the measurements of all nodes inside that cluster is selected as a cluster representative.

In both cases, the selection methodology is based on the similarity between all attributes of clustered nodes. Todays sensor nodes are collecting increasingly many number of dimensions for each sensor node. The similarity measures should cope with the increasing dimensionality of sensed data. In such data, distances grow more and more alike. The full data space is thus sparse and each nodes will be alone in its physical cluster as no global similarity between the measurements of different nodes can be observed. We are tackling this point in this work, by introducing a novel method for performing physical clustering based on the similarity over some of the sensed attributes using subspace clustering. We show that this method produces improvements in energy consumption even for low dimensional data.

In addition to the importance of saving energy, we designed our method to cope with the change detection. Detecting novelty in input stream is an important feature that has to be considered when designing any data knowledge technique in sensor networks. For example, it is an essential point in evaluating learning algorithms in drifting environments [9].

1.1 Our Contribution

The following aspects are our main contributions we included in this work:

- **Reducing the communication burden**
  In our approach, nodes do not continuously communicate with the representative. Communication is established only when a state change is detected in the monitored phenomena. By the careful construction of clusters, this communication is further reduced by using the similarity to representative readings.
- **Subspace physical clustering**
  Our novel method for building clusters according to the relevant attributes results in more consistent clusters, and helps for maintaining the clusters with less effort.
- **Outlier-aware change detection**
  We present a simple but effective method for detecting outliers in the input stream performed by each node, and another one performed by the representative to detect deviating nodes in its cluster. We show that our approach by applying this method, is still capable of detecting changes in input stream.
- **Uniform utilization of energy resources in sensor network**
  We suggest further optimization methods to our approach to uniformly distribute the usage of energy between the nodes. We cope with the cases of single-node clusters, and changing representatives according to residual energy. This results in a longer lifetime of the whole sensor network as nodes die close to each other.

The remainder of this paper is organized as follows. Section 2 mainly reviews the literature related to the physical clustering problem. Section 3 introduces some formulations and definitions used in our approach. Section 4 describes in detail our algorithm. Section 5 presents the experimental results. We conclude the paper and suggest future work in Section 6.

2. RELATED WORK

In this section we list briefly the related work to our physical clustering problem.

Traditional offline clustering algorithms e.g. [8], [4],[25] can not cope with the streaming and distributed nature of sensor nodes.

Although some **distributed versions of clustering algorithms** were established like SDBDC [11], DFEKM [12], they are still dealing with offline data and can not simply adapted to perform online distributed clustering.

Many algorithms were developed to deal with the **online distributed** clustering of data. EDISKCO [10] is an energy efficient approach for online approximative clustering of sensor data. The Distributed Grid Clustering algorithm [22] is an example of an online 2-layer distributed clustering of sensor data. ELink [16] and the Distributed Single-Pass Incremental algorithm DSIC [24] are two examples on time series clustering of sensor nodes. None of these algorithms considered the possibility of having clusters hidden in subsets of the attributes.

Subspace clustering has been proposed, for today's applications with incising number of given dimensions. Subspace clustering detects clusters in arbitrary projections by automatically determining a set of relevant dimensions for each cluster [20, 15]. Thus, one is able to detect objects as part of various clusters in different subspaces. Recent research has seen a number of approaches using different definitions of what constitutes a subspace cluster [3, 13]. As summarized in a recent evaluation study [19], their common problem is that the output generated is typically huge. In recent subspace clustering algorithms we have focused on tackling redundancy [5, 6, 18]. In contrast, projected clustering assigns each object to a single projection [2, 17]. This strict partitioning of the data into projected clusters can be regarded as extreme redundancy elimination. Projected clustering results in a manageable number of clusters, but is not able to detect overlapping clusters. Both subspace clustering and projected clustering have its focus on offline data outside sensor networks. In contrast, we aim at combining clustering in subspace projections [5, 2] with physical clustering for sensor networks [10].

SERENE [7] is a framework for SElecting REpresentatives in a sensor NEtwork. It uses clustering techniques to select the subset of nodes that can best represent the rest of sensors in the network. In order to reduce communication, rather than directly querying all network nodes, only the representative sensors are queried. In this way the overall energy consumption in sensor network is reduced and consequently sensor network lifetime is extended.
To select an appropriate set of representative sensors, SERENE performs the analysis of historical readings of sensor nodes, in order to find out the correlations both in space and time dimensions among sensors and sensor readings. Sensors may be physically correlated. Sensor readings may be correlated in time. Physically correlated sensors with correlated readings are assigned to the same cluster. Then each cluster performs further analysis in order to select the sensors with the highest representation quality. The last two steps of this process are the same with the steps of clustering process in our algorithm.

Similar to our algorithm, this technique uses density-based clustering algorithm, DBSCAN [8]. Nevertheless, different from our algorithm, in SERENE approach the first stage of clustering process is analysis of historical data for detecting correlations among nodes and sensor readings. Due to restrictions of energy, computational and memory capacity in sensor nodes, this analysis can not be performed by the nodes themselves.

Continuous storing of historical data for all nodes that are spatially correlated, in order to analyze correlation of their readings, requires more memory capacity than a sensor node

possesses. Processing of all the analyses over measurements of sensors to find out correlations, needs high computation resources as well. Moreover, this process requires exchange of attribute measurements between all the nodes that are spatially correlated. This is followed by a high energy consumption in nodes, due to frequent communication and data exchange with more than one node in their clusters. Due to all these restrictions, in this approach sensor nodes can not be self organized into clusters. As a result, this technique is suitable only for those scenarios where nodes operate in a supervised way.

Another difficult part of this technique is related with maintenance of SERENE platform. With passing of time, the readings of sensor nodes change, consequently the same set of sensors may not be anymore correlated with each other, or a new correlation may appear among some other nodes. This change requires a reorganization of nodes in clusters. Reclustering process is followed by additional communication among nodes for updating historical data. This will increase the communication burden and the size of transmitted data will be significantly high. More analyses should be performed over data, meaning more resources will be consumed for computation purposes.
All the above mentioned reasons make this approach expensive in terms of energy and not easy to maintain in cases of continuous clustering applications.

In [23] a Data-Driven Processing Technique in Sensor Networks was suggested. The goal of this technique is to provide continuous data without continuous reporting, but with checks against the actual data. To achieve this goal, this approach introduces temporal and spatio-temporal suppression schemes, which use the in-network monitoring to reduce the communication rate to the central server. Based on these schemes, data is routed over a chain architecture. At the end of this chain, the nodes that are most near to central server send the aggregate change of the data to it.

Snapshot Queries [14] is another approach that introduces a platform for energy efficient data collection in sensor networks. By selecting a small set of representative nodes, this approach provides responses to user queries and reduces the energy consumption in the network. In order to select its representative, each sensor node in this approach builds a data model for capturing the distribution of measurement values of its neighbors for each attribute.

After a node decides which of its neighbors it can represent, it broadcasts its list of candidate cluster members to all its neighbors. Each node selects as its representative that neighbor that can represent it and that additionally has the longest list of candidate cluster members. This is again expensive as all messages are broadcasted and not directed to specific nodes, which might result in repeated broadcasting in case of message loss. Maintaining this model is very expensive in terms of energy, as all nodes needs to exchange all historical readings among each other. In our algorithm, each sensor node maintains a small cache of past measurements of itself for each attribute. And the control messages exchanged among nodes during the initialization phase are directed to specified nodes. As the closest state-of-the-art to our approach, we evaluate our algorithm by comparing it to Snapshot Queries [14].

3. PROBLEM FORMULATION

In this section we formally define the related problems to our algorithm.

3.1 The Representatives Selection Problem

Given a set SN of n sensor nodes $SN = \{sn_1,\ sn_2, ..., sn_n\}$ each measuring a set of attributes $\{a_1, a_2, ..., a_k\}$, an Euclidean distance function $d(sn_a, sn_b) \geq 0$; $\{a, b\} \subset \{1, 2, ..., n\}$ and a real number $\varepsilon > 0$. The problem of selecting representative nodes in SN is to find a subset $R = \{r_1, r_2, ..., r_m\} \subseteq SN$; $m \leq n$ each $r_i \in R$ is representing a set of nodes $D_i = \{sn_{i1}, sn_{i2}, ..., sn_{il}\}$, $D_i \subseteq SN$ and $\forall sn \in D_i$: $d(r_i, sn) \leq \varepsilon$ such that the measurements sensed by all members of D_i are best represented by the measurements of r_i.

Definition 1. (Physical cluster of nodes)
A physical cluster C of sensor nodes is a set D_i with a maximum number of $MaxNds > 0$ nodes represented by the representative r_i such that $\forall sn \in D_i$:

$$\sqrt{(x_{r_i} - x_{sn})^2 + (y_{r_i} - y_{sn})^2 + (z_{r_i} - z_{sn})^2} \leq \varepsilon$$

where ε is the radius of C.

Definition 2. (Spatial and non-spatial attributes)
Each node $sn \in SN$ is defined in each time stamp t by a set of attributes $\{a_{1t}, a_{2t}, ..., a_{kt}, x_{sn}, y_{sn}, z_{sn}\}$ where $\{a_{1t}, a_{2t}, ..., a_{kt}\}$ is a set of non-spatial attributes which represent the measurements of sn at time stamp t and $\{x_{sn}, y_{sn}, z_{sn}\}$ are the spatial attributes of sn.

Definition 3. (Relevant attributes)
Let $\{\mu_1(t), \mu_2(t), \cdots \mu_k(t)\}$ and $\{\sigma_1(t), \sigma_2(t), \cdots \sigma_k(t)\}$ be respectively the mean values and the standard deviations of the non-spatial attributes of l readings of sensor node sn_a at time stamp t, a non-spatial attribute a_m, where $1 \leq m \leq k$, is called a relevant attribute between two nodes sn_a and sn_b at the time t if $X_{mt}(sn_b) \in [\mu_m(t) - 2\sigma_m(t), \mu_m(t) + 2\sigma_m(t)]$ where $X_{mt}(sn_b)$ is the sensor sn_b reading of attribute m at time stamp t.

3.2 The Problem of the ECLUN Algorithm

Given a set SN of n sensor nodes with a set of attributes deployed in an environment for monitoring physical phenomena and a base station to collect these measurements, the general problem of ECLUN algorithm is to decrease the total amount of consumed energy in SN by grouping the nodes of SN into physical clusters C_i where $1 \leq i \leq n$ each represented by a representative r_i with some relevant attributes a_j where $1 \leq j \leq k$ and then sending to the base station either the readings of the relevant attributes of r_i to represent the readings of all members of C_i or the summary of the readings of all members of C_i. The target is to continuously update the base station by all important changes in the sensed phenomena.

4. ECLUN ALGORITHM

In this section we describe in details our approach. We differentiate between two phases of the algorithm. The initialization phase where physical clusters are constructed in an unsupervised way, and the running phase when these clusters are maintained and updates are sent to the representative and the server.

4.1 The Initialization Phase

Algorithm 1 gives an overview of this phase. We will next describe each of these steps in details.

Algorithm 1 Initialization Phase of ECLUN

1. *Caching of initial data*
2. *Detection of geographical neighbors*
3. *Setting relevant attributes*
4. *Estimation of representation quality for each node*
5. *Selection of local representatives*
6. *Load balancing among representatives*

4.1.1 Caching of Initial Data

Each node senses the first l measurements for each attribute and stores them in the cache of data history. These l measurements will be exchanged between nodes, and thus will decide the initial physical clustering of nodes. Therefore, outlier readings must completely be excluded in this phase. We assume that attribute measurements for each sensor are normally distributed. Therefor, nodes during this phase continuously calculate the mean and standard deviation values of their measurements for each attribute. If any new reading falls out the corresponding confidence interval $[\mu - 2\sigma, \mu + 2\sigma]$, it is suspected to be an outlier, and stored in the suspected list with a maximum length of s. If the suspected list was filled within the previous s time stamps, then its readings are considered in the main list, otherwise it is excluded completely from both lists.

4.1.2 Detection of Geographical Neighbors

Every node detects its *geographical neighbors* GN by running spatial queries with radius ε. This is done by broadcasting its ID and spatial attributes and mean values of non-spatial attributes $< ID_n, \mu_1, \mu_2, \cdots, \mu_k, x_{sn}, y_{sn}, z_{sn} >$ within a radius ε, where $\mu_1, \mu_2, \cdots, \mu_k$ are the mean values of the initial l readings of sn for each non-spatial attribute. Thus, every node becomes aware of the geographical coordinates as well as the initial readings of its neighbors, these data are stored in a list GN in each node.

4.1.3 Setting Relevant Attributes

In this step, each node decides the relevant attributes between it and each node in its GN list. According to Definition 3, the node uses the non-spatial readings received in the previous step and the statistics of its own previous l measurements to decide the relevant attributes. The threshold $Min\_Rel\_Attr \leq k$; k is the number of non-spatial attributes, decides the minimum number of relevant attributes between two nodes when one wants to represent the other. Each node excludes from the list of GN, all neighboring nodes with less than Min_Rel_Attr relevant attributes to it. The rest nodes are stored in the *candidate cluster member* CCM list.

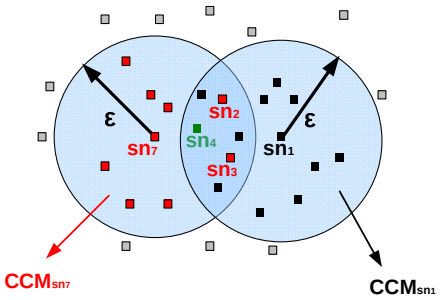

Figure 1: Candidate Cluster Members

In Figure 1, although nodes sn_2 and sn_3 are part of GN of node sn_1, they do not belong to its *candidate cluster members* CCM. Apparently, there are less than Min_Rel_Attr relative attributes between node sn_1 and each of sn_2 and sn_3, so they are both not in the CCM of sn_1.

4.1.4 Estimation of Representation Quality for Each Node

In this step of algorithm, each node analyzes how effective it is in representing its CCM nodes in the network.

Definition 4. (Representation quality)
The representation quality $RepQ$ of node sn when representing its CCM nodes is defined as:

$$RepQ(sn) = (1-\alpha)\frac{\sum_{sn_i \in CCM(sn)}(\varepsilon - d(sn, sn_i))}{\varepsilon \times \mid CCM(sn) \mid} + \alpha\frac{RE_{sn}}{IE_{sn}}$$

Where ε is the maximum radius of the possible cluster that might be represented by sn, $d(sn, sn_i)$ is the distance between sn and any of its CCM, α is a coefficient for weighting the energy, IE_{sn} and $RE_{sn}$are the initial and the residual energy of sn respectively.

According to Definition 4, $RepQ$ is greater when the members of CCM are forming a compact cluster around sn. Closer nodes means less consumed energy and much more similar measurements. Additionally, the residual energy is an important factor, as the possibility will be less later that sn gets soon out of energy. The bigger the value of α, the more the importance of the residual energy factor when selecting representatives.

4.1.5 Selection of Local Representatives

Each node decides whether it will be itself a local representative or it will be represented by any other similar node in its neighborhood. To take this decision, nodes refer to the representation quality parameter.

Each node sn_i broadcasts its $RepQ$ value to every node sn_j that belongs to its CCM. Every node $sn \in SN$ stores the list of *candidate local representatives* CLR, together with the $RepQ$ values received by them, and includes also itself

in this list. The list is ranked in a decreasing order according to *RepQ*. One of the following will happen:

1. If the current node has its own *RepQ* value in the top of this list, it announces itself as a representative.
2. If two nodes have the same *RepQ* value, then the closer node is selected as a representative for the current node
3. Otherwise, node sn is represented by the node which is having the *RepQ* in its CLR

After this step, every node either has chosen only one node as its representative, or is a representative itself. Since representatives announce themselves, each node collects the IDs of representative nodes in its neighborhood, it stores them in an internal list called *neighbor local representatives* NLR.

As we saw when building CCM, we had $Min\_Rel\_Attr \leq k$; k is the number of non-spatial attributes, we adopt this idea from the subspace clustering area [5, 2]. For many given attributes, one can hardly find two sensor nodes that can have similar measurements in all attributes, this will result in a huge number of single-node clusters. To avoid this we relax the representation criteria in such a way that the representative needs only to have some relevant attributes with its represented nodes. Algorithm 2 gives a description of the process of selecting the representative according to the relevant attributes and updating the server with relevant and non-relevant attributes by each node.

Algorithm 2 Selecting representatives per attributes

```
1.  if this_attributeis a relevant_attribute then
2.     Let it be represented by the local representative
       rep_a which is sharing the highest number of
       relevant attributes
3.  else if (other representative rep_b can represent
          this_attribute) then
4.     Some attributes are represented by rep_a others
       by rep_b
5.  else
6.     Let this_attribute be forwarded to server by rep_a
7.  end if
```

4.1.6 Load Balancing Among Representatives

To provide a uniform utilization of energy resources in sensor network, we set a threshold $MaxNds$ for the *maximum number of nodes* that can be represented by one representative. According to that, representatives decide to exclude from its cluster the most distant cluster members. The excluded node then tries to join the nearest representative in its NLR list.

At the end of the initialization phase, physical clusters C are established.

4.2 The Running Phase

The algorithm initiates the communication process only when a state change is detected. Nodes communicate with their representatives only when they detect a state change in the attribute measurements of the event they are monitoring. Similarly, representatives send data to the server only if they detect a state change in the statistics of the measurements collected from all the nodes of their clusters. We have then two possible communication paths: node-representative and representative-server.

4.2.1 Node-Local Representative Communication

Each node sn compares the current measurement values $X_{jt_i}(sn)$ on non-spatial attribute $j = (1 \cdots m)$ sensed at t_i with the mean value $\mu_j(t_{i-1})$ of the l previous measurements values of the corresponding attribute. If $|X_{jt_i}(sn) - \mu_j(t_{i-1})| \leq \delta_j$ where δ_j; $j = (1 \cdots m)$ are the measurements thresholds for attribute j, then a change in the measurements is detected and an update of X_{jt_i} should be sent to the corresponding representative. Otherwise no data is sent to the representative and old measurements sent previously to the representative by sn are used.

4.2.2 Local Representative-Server Communication

During each time stamp in the running phase, the representative executes Algorithm 3. After this, and at the same time stamp, the representative maintains C. It checks whether $X_{jt_i}(sn)$ that it has received from sn falls inside the confidence interval $[\mu_{jC}(t_{i-1}) - 2\sigma_{jC}(t_{i-1}), \mu_{jC}(t_{i-1}) + 2\sigma_{jC}(t_{i-1})]$ for each relevant attribute in C or not. If this was not the case, then sn is temporarily excluded from the t_i statistics. Its readings are saved in a list with a maximum length s. If s was filled within the previous s time stamps with readings of sn, then the representative requests sn to join another physical cluster, and forwards its s readings together with the ID of sn to the server. And sn in turn, searches for a neighboring representative in its NLR list and continues from step 5 in Algorithm 1. The only exception here will be that Min_Rel_Attr threshold does not apply, as nodes try to minimize the number of nodes representing it.

Algorithm 3 Representative running phase

```
1.  while updates are received from nodes at time t_i do
2.     if any attribute is missed then
3.        use t_{i-1} values
4.     for each relevant attribute j do
5.        μ_jC(t_i) = (1/|C|) Σ_{sn∈C}
6.        if |mu_jC(t_i) − μ_jC(t_{i-1})| ≤ ψ_j then
7.           update the server with μ_jC(t_i) and σ_jC(t_i)
8.     end for
9.  end while
```

4.3 Energy Aware Optimizations

We suggest further optimizations for the sake of energy efficiency in our algorithm.

4.3.1 Delegation of Representative Authority

The energy of local representatives decreases rapidly much more than the energy of other nodes in the network.

If a representative runs out of energy, all of its cluster nodes should recluster.This is considerably energy consuming. Furthermore, losing the representative node will cause a big lack of information about the monitored phenomena delivered by the complete cluster. We suggest a uniform utilization of

energy sources in the network, by applying a technique of delegation representative authority.

Each local representative is aware of its residual energy. At the time it notices that its energy capacity is decreased under a certain threshold (for instance: 50% of its initial energy as it started to represent this cluster), local representative requests the residual energy values of nodes in C. The authority of representing the cluster is delegated to the node with the highest residual energy including current representative. If none of the cluster nodes has more residual energy than current representative, then it continues being the representative of its cluster and performs later the same check again.

4.3.2 Optimization in Case of Single Node Cluster

In such a scenario, sensor node has to communicate with distant server for updating only its measurements. To avoid this, each node that is alone in its cluster sends lazily *'join requests'* to its neighbor representatives. Each neighbor representative then checks whether the attribute measurements are relevant to its cluster. Accordingly it might join that cluster or keep representing itself. In case of more acknowledgments, it selects the nearest neighbor representatives. Receiving no-acknowledgment means that the node is selecting different data than its neighbors and will keep representing itself. This might mean that either this node is corrupted or measuring some local event.

5. EXPERIMENTAL EVALUATION

To evaluate the performance of ECLUN, we performed a set of experiments to test the effectivity of each feature of ECLUN, and to compare the performance of ECLUN with the state-of-the-art competing algorithm, Snapshot Queries [14]. We start by describing our real and synthetic datasets in 5.1, then our evaluation methodology for each set of experiments in 5.2, in 5.3 we describe the settings of our experiments and then we conclude this section by discussing the experimental results in 5.4.

5.1 Datasets

We have used three real datasets in addition to one synthetic dataset for evaluating ECLUN. We give a description of each with some of the parameter settings applied with them on both ECLUN and Snapshot Queries. Unless otherwise stated, these parameter settings applies to all experiments.

5.1.1 Real Datasets

Intel Berkeley Research Lab: Intel Lab 1
Out of the 54 nodes readings collected in [1]. Three nodes had a huge number of missing readings, therefore we used the clean readings of 51 nodes each contains 4-parameters readings taken every 31 seconds. The clean processed dataset contained 15730 readings. We have mapped these time stamps into 5 days, 15 hours, 27 min and 10 sec period of time. The network topology was selected to be as close as possible to original nodes topology. When applying this dataset on any algorithm we set the initial energy IE of each node to 295 Joules. We call this real dataset *Intel Lab 1* in our next experiments.

Table 1: Generated events in the synthetic dataset

	Event 1	**Event 2**
Values per dimension	D1{Low} D2{High} D3{Low}	D1{High} D2{Low} D3{High}
Time stamps [From, to]	[0,200], [1000,1100], [10000,11000]	[300,350], [1000,1100], [2000,2500]
Most affected node	Node 2, coordinates: (2,0)	Node 47, coordinates: (4,6)

Intel Berkeley Research Lab: Intel Lab 2
To get more readings, we have excluded 5 nodes from the original Intel lab dataset. This resulted in 23077 healthy readings. For small missed values in between, we have always inserted the last received value instead of later missed readings. Again we set the topology of the network in both evaluated algorithms to be as close as possible to the original topology. When applying this dataset on any algorithm we set the initial energy IE of each node to 10000 Joules. We call this dataset *Intel Lab 2*.

I9 Sensor Dataset
Explanation about this one-dimensional dataset with 40589 readings can be found in [10]. The 16 nodes were randomly inserted to the algorithms without mapping the coordinates of network topology. When applying this dataset on any algorithm we set the initial energy IE of each node to 1100 Joules.

5.1.2 Synthetic Dataset

The synthetic dataset was generated mainly for evaluating the response of each of the competing algorithms to some inserted changes in the monitored phenomena.We generated readings for 49 sensor nodes distributed in one 7x7 grid with 12000 3-dimensional readings for each node. The normally distributed random readings were mainly simulating the humidity, light and temperature attributes sensed by TelosB nodes [21]. The total range of each attribute was divided into three subranges: Low, normal and High. Inserted events are any combination of three ranges, each taken from an attribute. We have generated 2 different in different parts of the network, details about these events are depicted in Table 1.

5.2 Evaluation Methodology

We evaluated ECLUN from three different perspectives:

1. **Evaluating each Feature of ECLUN:** We have tested the effect of each feature of ECLUN by evaluating for every feature two versions of ECLUN, one containing this feature and the other not. The measure was the total number of dead nodes in the whole network with the progress of time.

2. **Energy Consumption:** Two measures were performed to evaluate the energy consumption of ECLUN with that of Snapshot Queries, the total number of dead

nodes in the network, and the total amount of consumed energy in Joules.

3. **Detection of Changes in Input Stream:** We wanted to see the values of readings delivered to the server by the representatives in each algorithm on time stamps where we synthetically inserted events as in Table 1.

5.3 Setup of the Experiments

For evaluating the energy consumption, in all experiments we used the energy model described in [10]. For all experiments of ECLUN, we had the following settings on all datasets: the radius of covered nodes by the range of each node: $\varepsilon = 2$, the maximum number of nodes in one physical cluster: $MaxNds = 4$ and the delegation authority threshold: 50% of initial energy. For **Intel Lab 1** and **Intel Lab 2** datasets, we have selected the number of initial readings $l = 10$, the threshold of relevant attributes for representing $Min\_Rel\_Attr = 2$, the node-representative update thresholds: δ_j; $j = (1 \cdots 4)$ as $(0.2, 0.2, 0.2, 0.2)$ and the representative-server update thresholds: ψ_j; $j = (1 \cdots 4)$ as $(0.2, 0.2, 0.2, 0.2)$. For **I9** dataset: $l = 10$, $Min\_Rel\_Attr = 1$, $\delta_1 = 0.2$ and $\psi_1 = 0.2$. To have fair results, the parameter settings of Snapshot Queries were always identical to that of ECLUN whenever they apply. We set the error threshold T_j; $j = (1 \cdots 4)$ to $(5, 5, 5, 5)$. According to [14], these values deliver the best results in terms of number of participating nodes in each cluster on the one hand, and an accepted representation error on the other hand.

5.4 Experimental Results

5.4.1 Results of Features Evaluation

In each of the following selected two experiments, we test a feature in ECLUN, by comparing the energy consumption of two versions of ECLUN that differ only in including this feature or not.

Node - Representative and Representative - Server Communications

This feature enables the update of the representative or the server to occur only when a change is detected. Excluding it means that the nodes always communicate with the representative whenever they have a new reading, and the representative in turn always communicates with the server.

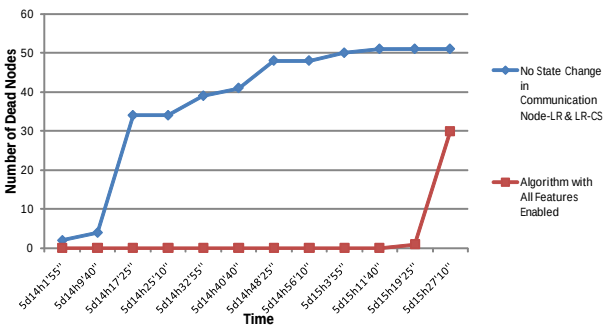

Figure 2: Testing the ECLUN feature of performing the update only when a change is detected using the Intel Lab 1 dataset

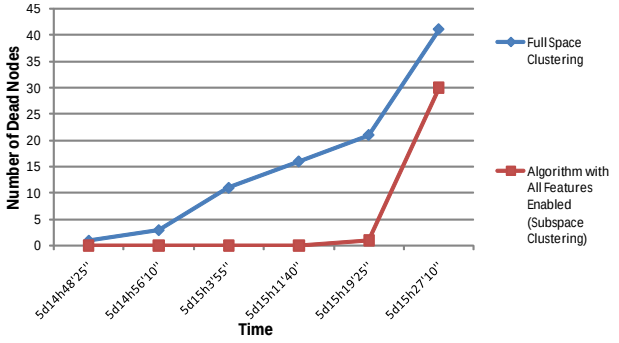

Figure 3: Testing the ECLUN feature of subspace clustering using the Intel Lab 1 dataset

As expected, the results in Figure 2 shows that this feature extends the whole network life time. Without using this features nodes start to die in the network after 5 days, 14 hours, 1 minute and 55 seconds, while by using the first node dies around 1 hour and 20 minutes after that. Additionally, by the end of dataset 21 nodes are still alive when enabling this feature, while all nodes die when disabling it. As we will see in the change detection results, this feature is not delaying important changes.

Subspace Clustering:(Clustering per Relevant Attributes)

Disabling this feature means that a node can only be represented by nodes that are relevant to it in all spaces (attributes). The possibility for nodes to find such a representative in its neighbors will be very low. Which ends with a self representation by the node. As depicted in Figure 3, using this feature delays the death of first node around 31 minutes and increases the number of the still-alive nodes by 11 with the end of the simulation. The impact of the subspace clustering is even stronger with higher dimensions.

5.4.2 Results of Energy Consumption Evaluation

We evaluate here the energy consumption of ECLUN and Snapshot Queries algorithm [14]. Figure 4 depicts the residual energy in Joules of each sensor node after the initialization phase. As shown, the initialization phase of Snapshot Queries consumes more energy than that of ECLUN. This is because of the extensive messages of big sizes that are exchanged between nodes in Snapshot Queries during this phase. Although this initialization phase happens not so often in ECLUN through reclustering, our experiments showed that it happens more likely in Snapshot Queries. This is due to the subspace nature that ECLUN uses. It can be seen also in Figure 4, that the energy consumption in ECLUN is balanced between all the nodes after this phase, in contrast to Snapshot Queries, where selected representatives consumes more energy than others even during this phase.

Figure 5 presents a comparison between two versions of each algorithm. For ECLUN, we used the all-features version and another without the previous features tested in 5.4.1 and without the delegation of authority optimization. For Snapshot Queries, we applied the two forms of changing the representative with the decrease of energy suggested in [14].

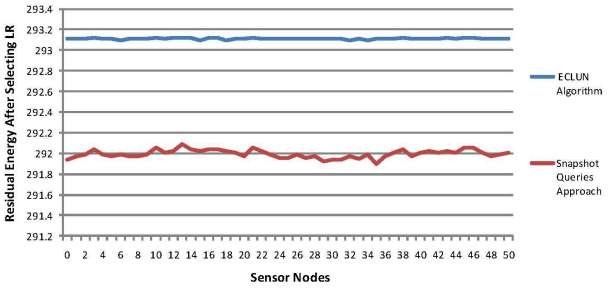

Figure 4: The residual energy in each of the 51 nodes after the process of selecting representatives in ECLUN and Snapshot Queries using Intel Lab 1

Table 2: Total energy consumption of ECLUN and Snapshot queries in Joules

Dataset	Number of Nodes	Number of Readings per Node	**ECLUN** Energy Consumption [Joules]	**Snapshot** Energy Consumption [Joules]
Intel Lab 2	49	23077	**22552.5**	25546.9
I9	16	40589	**13837.1**	14094.9

The first one is similar to ECLUN, where nodes are invited to send their residual energy and the one with the highest residual energy is selected. The other approach randomly selects the next representative, we called this (Snapshot Queries with randomized representatives). The two versions of ECLUN extend considerably the network life time much more than both of the versions of Snapshot Queries. The better version of Snapshot Queries starts to lose nodes around 7 hours and 15 minutes earlier than the normal ECLUN. When the dataset ends, ECLUN has still 21 alive nodes, while the two versions of Snapshot Queries almost lose all of their nodes. Another important feature is that the nodes in ECLUN die close to each other, which yields a better usage of the network resources and more data about observed phenomena. Figure 6 and Table 2 show the efficiency of ECLUN over Snapshot Queries for different sizes of data with different dimensionality.

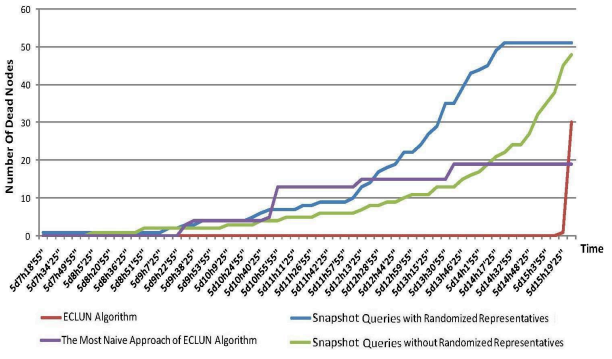

Figure 5: Number of dead nodes in different versions of ECLUN and Snapshot Queries using Intel Lab 1

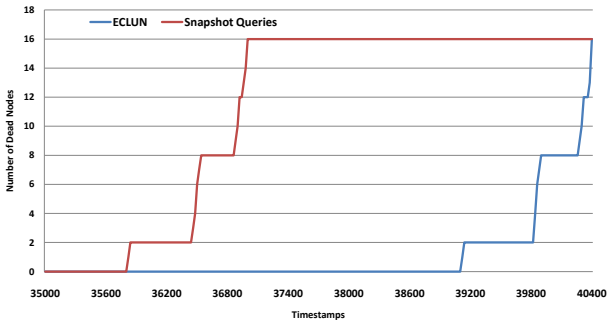

Figure 6: Number of dead nodes using the I9 dataset

5.4.3 Results of Change Detection Evaluation

For evaluating this measure, we used the synthetic dataset. Figure 7 depicts the input events affecting some parts of the sensor network, and the corresponding output sent by ECLUN and Snapshot Queries to the server at the same time stamp. Figures 7(a) and 7(b) show the input and ECLUN output Event 1 at time stamp:11. Snapshot Queries detected this event at time stamp:12 Figure 7(c). Obviously, ECLUN was not only able to detect this event exactly when it appeared, it could also deliver the involved nodes in this event with few false positives. Snapshot Queries detected the change event with a delay of one time stamp, then delivered the data of only one node out of the six involved in Event 1. Figures 7(d), 7(e) and 7(f) describe the input, ECLUN output and Snapshot Queries output at time stamp:1000. ECLUN detected the event at the same time stamp but was less accurate than at time stamp:11. This is due to the fact that at time stamp:11, ECLUN has clustered the nodes according to the first $l = 10$ readings. Event 1 was existing during that interval, and thus detecting it was much more accurate. Snapshot Queries could not detect this event at all. Figures 7(g), 7(h) and 7(i) depict the same sequence for Event 2 at time stamp: 1050.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a novel algorithm for an energy aware physical clustering of sensor nodes. Our algorithm considers both spatial and data similarities when building these physical clusters. Nodes in our suggested approach make use of established data mining techniques like subspace clustering for joining physical clusters according to relevant attributes, and outlier detection for online exclusion of outlying readings. We further suggested a powerful method for the maintenance of the constructed clusters. This enables the network to adapt with different changes of observed phenomena in an unsupervised way, while consuming less energy. We proved the efficiency and effectiveness of our approach through comprehensive experiments.

In the future, we would like to combine our sensor stream data clustering approach (EDISKCO [10]) with this node clustering approach. This can further save energy, and might improve the correctness of approximative solutions applied on stream sensor data. With the huge exchange of data in physical clustering of sensor nodes, security looks like an important issue. We aim at tackling this point by extending our outlier ranking techniques. As an additional require-

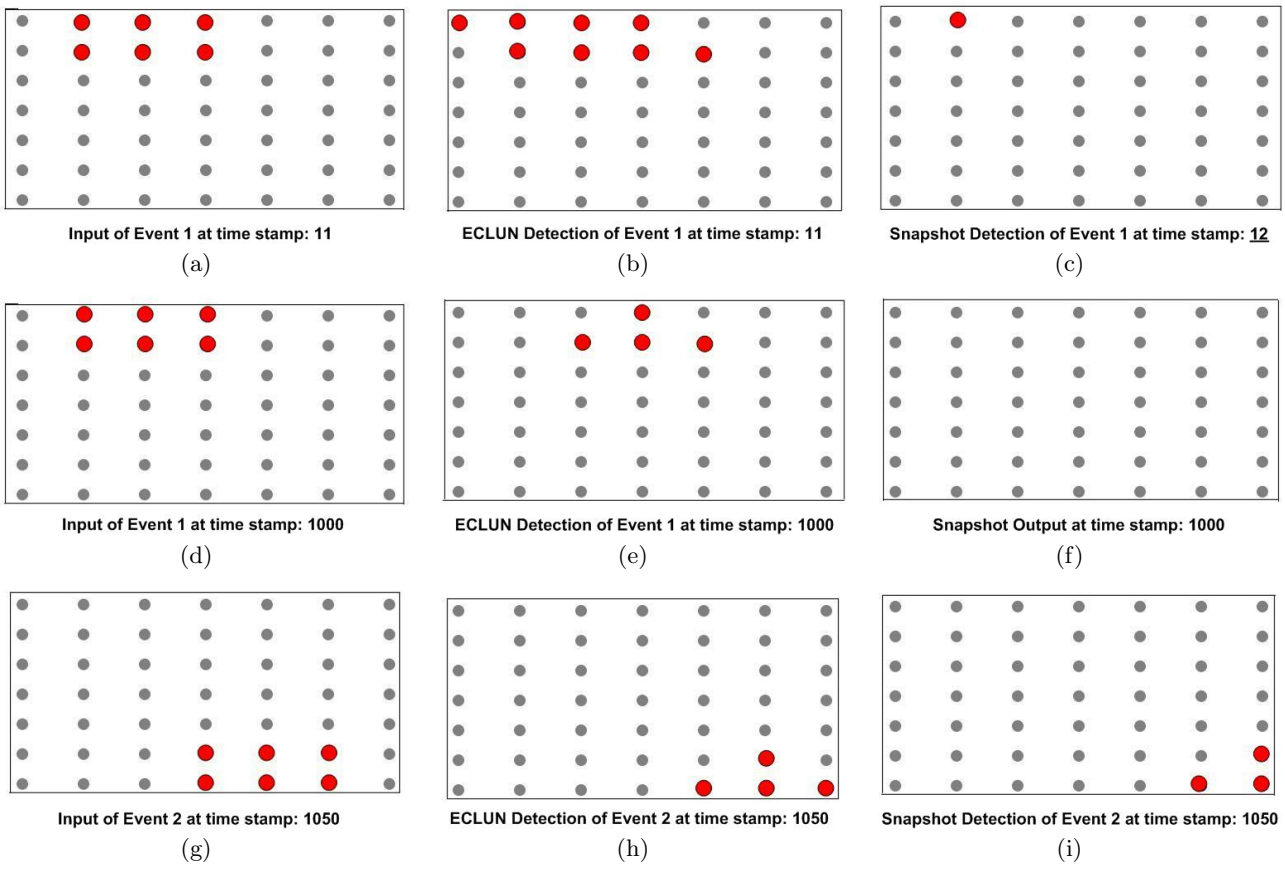

Figure 7: Change detection evaluation using the Synthetic Dataset with 2 inserted events

ment, we would like to extend our approach additionally to consider subspace physical clustering of mobile sensor nodes.

Acknowledgments

This research was funded in part by the cluster of excellence on Ultra-high speed Mobile Information and Communication (UMIC) of the DFG (German Research Foundation grant EXC 89).

7. REFERENCES

[1] Dataset of intel berkeley research lab. In *Online under http://db.csail.mit.edu/labdata/labdata.html*, 2004.

[2] C. Aggarwal, J. Wolf, P. Yu, C. Procopiuc, and J. Park. Fast algorithms for projected clustering. In *SIGMOD*, pages 61–72, 1999.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD*, pages 94–105, 1998.

[4] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings ACM SIGMOD' 99 Int Conf. on Management of Data*, 1999.

[5] I. Assent, R. Krieger, E. Müller, and T. Seidl. DUSC: Dimensionality unbiased subspace clustering. In *ICDM*, pages 409–414, 2007.

[6] I. Assent, R. Krieger, E. Müller, and T. Seidl. INSCY: Indexing subspace clusters with in-process-removal of redundancy. In *ICDM*, pages 719–724, 2008.

[7] E. Baralis and T. Cerquitelli. Selecting representatives in a sensor network. In *In Proceedings of the SEBD*, pages 351–360, 2006.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[9] A. R. Ganguly, J. Gama, O. A. Omitaomu, M. M. Gaber, and R. R. Vatsavai. *Knowledge Discovery from Sensor Data*. 2008.

[10] M. Hassani, E. Müller, and T. Seidl. EDISKCO: Energy efficient distributed in-sensor-network k-center clustering with outliers. In *In Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data SensorKDD*, pages 39–48, 2009.

[11] E. Januzaj, H.-P. Kriegel, and M. Pfeifle. Scalable density-based distributed clustering. In *Proceedings 8th European Conference on Principles and Practice of Knowledge Discovery in Databases PKDD*, 2004.

[12] R. Jin, A. Goswami, and G. Agrawal. Fast and exact out-of-core and distributed k-means clustering. *Knowledge and Information Systems*, 10(1):17–40, 2006.

[13] K. Kailing, H.-P. Kriegel, and P. Kröger. Density-connected subspace clustering for

high-dimensional data. In *SDM*, pages 246–257, 2004.

[14] Y. Kotidis. Snapshot queries: Towards data-centric sensor networks. In *Proceeding of the 21st International Conference on Data Engineering, ICDE*, 2005.

[15] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *TKDD*, 3(1), 2009.

[16] A. Meka and A. K. Singh. Distributed special clustering in sensor networks. In *EDBT 2006, LNCS 3896*, pages 980–1000, 2006.

[17] G. Moise, J. Sander, and M. Ester. P3C: A robust projected clustering algorithm. In *ICDM*, pages 414–425, 2006.

[18] E. Müller, I. Assent, S. Günnemann, R. Krieger, and T. Seidl. Relevant subspace clustering: Mining the most interesting non-redundant concepts in high dimensional data. In *ICDM*, pages 377–386, 2009.

[19] E. Müller, S. Günnemann, I. Assent, and T. Seidl. Evaluating clustering in subspace projections of high dimensional data. *PVLDB*, 2(1):1270–1281, 2009.

[20] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations*, 6(1):90–105, 2004.

[21] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, IPSN*, 2005.

[22] P. P. Rodrigues, J. Gama, and L. Lopes. Clustering distributed sensor data streams. In *ECML PKDD 2008, LNAI. Springer-Verlag*, 2008.

[23] A. Silberstein, R. Braynard, G. Filpus, G. Puggioni, A. Gelfand, K. Munagala, and J. Yang. Data-driven processing in sensor networks. In *Proceedings of the 3rd Biennal Conference on Innovative Data Systems Research (CIDR)*, 2007.

[24] J. Yin and M. M. Gaber. Clustering distributed time series in sensor networks. In *In Proceedings of the Eighth IEEE Conference on Data Mining, ICDM*, 2008.

[25] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings ACM SIGMOD' 96 Int Conf. on Management of Data*, 1996.