DEMS: A Data Mining Based Technique to Handle Missing Data in Mobile Sensor Network Applications

Le Gruenwald, Md. Shiblee Sadik, Rahul Shukla, Hanqing Yang
School of Computer Science
University of Oklahoma
Norman, Oklahoma, USA
{ggruenwald, shiblee.sadik, rahul.shukla-1, hqyang3}@ou.edu

ABSTRACT

In Mobile Sensor Network (MSN) applications, sensors move to increase the area of coverage and/or to compensate for the failure of other sensors. In such applications, loss or corruption of sensor data, known as the missing sensor data phenomenon, occurs due to various reasons, such as power outage, network interference, and sensor mobility. A desirable way to address this issue is to develop a technique that can effectively and efficiently estimate the values of the missing sensor data in order to provide timely response to queries that need to access the missing data. There exists work that aims at achieving such a goal for applications in static sensor networks (SSNs), but little research has been done for those in MSNs, which are more complex than SSNs due to the mobility of mobile sensors. In this paper, we propose a novel data mining based technique, called Data Estimation for Mobile Sensors (DEMS), to handle missing data in MSN applications. DEMS mines the spatial and temporal relationships among mobile sensors with the help of virtual static sensors. DEMS converts mobile sensor readings into virtual static sensor readings and applies the discovered relationships on virtual static sensor readings to estimate the values of the missing sensor data. We also present the experimental results using both real life and synthetic datasets to demonstrate the efficacy of DEMS in terms of data estimation accuracy.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining; G.3 [**Probability and Statistics**]: Nonparametric statistics, Statistical computing

General Terms

Algorithms, Performance, Experimentation

Keywords

Sensors, Missing Data, Mobile Sensor Networks

1. INTRODUCTION

A wireless sensor network (WSN) can be defined as a set of independent sensors which can solve cooperatively some monitoring based applications [8]. Typical applications of WSN include environmental monitoring [12], scientific investigation [15], civil structure flaw detection, battle surveillance and medical applications [18]. However, successful monitoring of any physical phenomenon is directly dependent on the appropriate deployment of the sensors [2, 13]. In a static sensor network (SSN), the sensors' positions remain stationary after the initial deployment. In addition, the areas covered by the sensors are dependent on the initial network configuration and remain unchanged over time [10]. An inappropriate deployment of sensors in a SSN may partition the monitoring area into regions either covered by at least one sensor and/or devoid of any sensors [10]. Therefore, while a covered region may be monitored by unnecessary multiple sensors, the regions uncovered by sensors may not be monitored at all leading to inaccurate results. Also, certain restrictions, such as hostile environments and disaster areas [22], make initial, manual deployment of sensors impossible. Finally, certain applications like monitoring atmosphere or ocean environment require constant mobility that can be achieved only if the sensors themselves are mobile [10]. Consequently, in recent years, much interest has been shown towards un-stationary sensors (e.g., Robomote [19]), that can re-deploy themselves according to the needs of the application. These sensors are termed as mobile sensors and their networks as mobile sensor networks (MSNs).

WSN data, in form of online data streams, arrive at the base station as real-time updated data [9]. These online data streams are infinite, unbounded and have high continuous arrival rates which do not permit complete scanning of the entire data [7]. Various factors, such as limited power and transmission capabilities of sensors, hardware failures, power outages, and network issues like disruption, package collision and external noise, cause the transmitted data to fail to reach the base station and/or be corrupted. The sensors that 'generate' these missing data are called missing sensors. A major concern with any WSN is the issue of missing sensor data. Several approaches, such as ignoring missing data, using backup sensors, re-querying the network, and utilizing data estimating techniques to estimate the values of the missing data, have been proposed to address the issue of missing sensor data [21]. Ignoring missing data is not viable for sensitive applications; using backup sensors may lead to data duplication and is expensive; and re-querying the network is unrealistic in terms of time and resource efficiency. The approach that uses data estimation has shown to be the most promising solution; however, currently it is limited to SSNs only [21, 1, 16, 6]. To the best of our knowledge, no work has been proposed to estimate the values of the missing sensor data in MSN applications.

MSNs consist of sensors placed on mobile platforms like Robomote [19]. In addition to the issues common to any data stream application, MSN applications have certain additional constraints. MSN applications are broadly divided into relocation and continuous coverage based applications [10, 22]. The spatial relation between two sensors is distorted by the mobility of mobile sensors; hence the spatial relationship between two mobile sensors is difficult to obtain in MSNs. Moreover, the history data of a mobile sensor that are generated at different locations may not necessarily possess the spatial or temporal relationships with the data in the current round of sensor readings. Finally, mobile sensors have the capability of moving themselves which costs lots of energy; so power outage occurs more often on mobile sensors than on static sensors; hence, instances of missing data are more pronounced in MSNs. In this paper we propose a data mining based solution for estimating the values of the missing sensor data in MSN applications, called DEMS (Data Estimation for Mobile Sensors). DEMS is a novel concept that addresses the issues associated with mobile sensors by utilizing virtual static sensors. DEMS establishes these virtual static sensors by dividing the entire monitoring area into hexagons and associating each hexagon's center with a virtual static sensor. It converts each mobile sensor reading into an equivalent virtual sensor reading. When a mobile sensor reading is missing, DEMS uses the spatial and temporal association rules among the virtual sensor readings that it discovers based on the history virtual sensor readings to compute the estimated value of the missing mobile sensor reading.

The rest of the paper is organized as follows: Section 2 discusses the related work; Section 3 describes DEMS; Section 4 presents the performance evaluation comparing DEMS with the three existing techniques: Average, Spirit [16], and TinyDB [11]; and Section 5 provides the conclusions and future work.

2. RELATED WORK AND ISSUES

Approaches for estimating the values of the missing sensor data (or approaches for estimating missing data for short), as of now, have been limited to SSNs only. TinyDB [11] is a prominent information extracting system for sensor networks. TinyDB does data estimation for a missing sensor by averaging the readings of other sensors for a particular round. However, it does not work well if a non-linear relationship exists among sensors and the sensors do not report similar readings. SPIRIT [16] uses auto-regression for finding correlations using hidden variables inside the history data of a sensor. It estimates missing data by predicting changes in data patterns using hidden variables as a summary of data correlation among all the history data. However, it does not consider the sensor readings from other sensors for the current round; therefore it is unable to find the current relationships among the data which may affect its accuracy. The Kalman filter [21] uses the dynamic linear model to predict missing data based on the history data. However, the dynamic nature of data distribution may introduce instances when the same sensor reports a completely different value in the current round compared to the previous rounds. This may cause erroneous results.

FARM [5] uses association rules among sensor readings to estimate missing data. It uses a novel data freshness framework to address the temporal nature of data. Further, it implements a data compaction scheme to store history data. Its estimated data are fairly accurate compared to those of statistical methods. However, its limitation is that it establishes association rules among similar sensor readings only; thus, only equivalent relationships are mined.

Mining Autonomously Spatio-Temporal Environmental Rules (MASTER) [1] is a comprehensive spatio-temporal association rules mining framework which provides both a competitive data estimation method and an exploratory tool to investigate the evolution of patterns of the sensor data in static sensor networks. MASTER is well equipped to discover spatial and temporal association rules among the sensors. This framework includes a novel data structure called MASTER-tree which stores the history data synopsis (the moments) for each sensor and represents the association rules among the sensors. An example of an association rule in MASTER is $S_1[10, 20], S_2[40, 90] \to S_3[30, 40]$ where S_1, S_2 and S_3 are three sensors, S_1 and S_2 are called the antecedent sensors and S_3 is called the consequent sensor of the rule. This rule implies when the sensor reading of S_1 is between 10 and 20 and the sensor reading of S_2 is between 40 and 90, the sensor reading of S_3 would be between 30 and 40. Each node in the MASTER-tree represents a sensor except the root node which represents an empty node; and each path/sub-path starting from the root node represents an association rule. Hence a MASTER-tree is capable of representing any kind of relationships among the sensors which participate in the MASTER-tree.

MASTER limits the number of sensors in one MASTER-tree by clustering the sensors into small groups and producing an individual MASTER-tree for each cluster. The advantage of the clustering step is twofold: (1) the clustering step arranges spatially co-related sensors into a cluster, and (2) it limits the number of sensors in a MASTER-tree which restricts the exponentially large number of association rules into a more manageable number. As each data round arrives, MASTER finds the appropriate MASTER-tree for each sensor and updates the MASTER-tree based on the arrived sensor readings. At any particular time, if a sensor reading is missing, MASTER finds the appropriate MASTER-tree for the missing sensor and evaluates the support and confidence of the association rules where the missing sensor appears as consequent. MASTER finds the best association rule comparing the obtained support and confidence with the user-defined minimum support and minimum confidence. Finally, it uses the best association rule and the current sensor readings of the antecedent sensors in the best association rule to estimate the consequent sensor's reading. Interested readers are referred to [1] for further details.

MASTER was designed for SSNs. It has the following deficiencies. The cluster formation step is solely based on the spatial attributes of a sensor. In a MSN, the spatial data of a sensor are changing; therefore the prior knowledge about sensor locations is not enough for MSNs even though spatial clustering works very well in SSNs. One possible solution for this problem is re-clustering whenever a sensor changes its location, but re-clustering is very computation-intensive and may cause loss of the history data, and thus loss of history data synopsis (the moments) stored in the MASTER-tree. Hence location-based clustering for mobile sensors does not produce any meaningful result. Moreover, in a MSN, a reading of a sensor is accompanied by the location of the sensor. So, if a sensor is missing, it is very likely that the reading and the location from that sensor will be missing together. Hence the estimation technique must estimate both dimensions for the missing sensors, which means that location prediction has to be an inherent part of the technique.

In a SSN, association rule mining can be used to discover the relations among sensors. According to Tobler's first law of geography [23], geographically close sensors are more correlated than the distant one. In a MSN, the distance between the mobile sensors changes over time; therefore the correlation changes over time too. The association rules among the sensors represent the correlation among them. If the mobile sensors change their locations, the correlations among them change; hence the association rules previously obtained based on the sensor data will no longer be valid for the new locations. This has two-fold implications on MASTER: (1) any previously explored rules may not be valid anymore; and (2) previously formed clusters may not be valid at all. In the extreme case, the history data from the same sensor may no longer be valid to estimate the missing data of the same sensor in the current round of data. This is because the old data are based on the previous locations of the sensor, whereas the new data are based on the new locations. So the methods (e.g., Kalman Filter [21]) which use history data to estimate new data will also become invalid in such a situation.

Motivated by the drawbacks of MASTER, in this paper we propose a new technique, called DEMS, for MSN applications. DEMS makes use of virtual static sensors that tackles the problems of location-aware clustering of real mobile sensors. It also tackles the problem of having no related history information for the current round of data from real mobile sensors. Moreover, DEMS addresses the issue of missing location of a real mobile sensor and is capable of predicting the next location for a missing real mobile sensor. The details of DEMS are presented in the next section.

3. THE PROPOSED DEMS

This section describes our technique, DEMS. It starts with a brief overview of DEMS followed by a detailed description of our novel concept of virtual static sensor and its significance. Finally it presents the MASTER-tree used for data mining and the estimation module for DEMS.

3.1 The Overview of DEMS

In DEMS, we exploit the spatial and temporal relations between sensor readings to estimate the missing sensor data. First we divide the entire monitoring area into hexagons based on a user-defined radius. Each hexagon corresponds to a virtual static sensor (VSS) placed at the center of the hexagon and covering the entire hexagon. A VSS is an artificial sensor, i.e. it does not exist physically in real life applications, but it exists in our technique as a synthetic sensor which mirrors a real static sensor. Each VSS has a unique identifier. DEMS converts the real mobile sensor readings into VSS readings based on the mobile sensors' current locations. Figure 1 shows A as the monitoring area covered by a MSN that is divided into 14 hexagons with 14 VSSs,

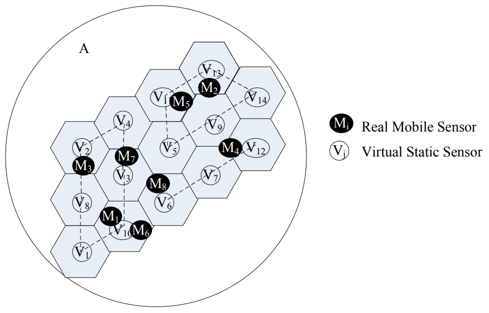

Figure 1: Monitoring area and hexagons

$V_1 \ldots V_14$, and 7 real mobile sensors, $M_1 \ldots M7$. Using agglomerative clustering [3], DEMS clusters the VSSs based on their locations into clusters and creates a MASTER-tree for each cluster. The dotted lines that connect the centers of the hexagons in Figure 1 show three clusters ($V_1, V_2, V_3, V_8, V_{10}$), ($V_6, V_7, V_{12}$) and ($V_5, V_9, V_{11}, V_{13}, V_{14}$). MASTER-tree records the data for the VSSs. For each missing mobile sensor reading, its estimated value is computed using the three major steps: (1) mapping the missing real mobile sensor to its corresponding VSS; (2) estimating the missing VSS reading using the discovered spatial and temporal association rules among the history VSS readings, and (3) converting the estimated VSS reading into the corresponding real mobile sensor reading.

In a MSN, a sensor reading reported is accompanied by the sensor location where the reading was obtained. Whenever a mobile sensor reading is missing (we call this a missing mobile sensor for short), it is likely that both the location and the reading will be missing together. To find the appropriate location of a missing mobile sensor we always keep track of mobile sensors' locations. A mobile sensor's location is mapped to a hexagon and the consecutive locations of a mobile sensor are mapped to a sequence of hexagons. We refer to a sequence of hexagons as a mobile sensor's trajectory. We mine the mobile sensor trajectories and predict the missing location based on the history trajectories. Morzy [14] proposed a pattern tree based approach for mining trajectories and predicting future locations, which we adopt for DEMS. DEMS maintains a single pattern tree of trajectories for all the mobile sensors. As small devices like sensors often use the same protocol for relocation [10, 19], it is reasonable to assume that they have similar patterns of movement; therefore DEMS maintains a single pattern tree of trajectories for all the mobile sensors and uses a single pattern tree instead of an individual pattern tree for each mobile sensor. This trajectory pattern tree is used to predict a missing mobile sensor's location. The predicted location is used to map a mobile sensor to a VSS. Since sensors repeat the mobility pattern for relocation, history based trajectory mining is more promising than random walk models.

3.2 The Virtual Static Sensor

In SSNs, every sensor monitors a fixed region and a sensor's reading reflects an event occurring within this region; but in MSNs, owing to their mobile nature, the region being monitored varies with time. However, as in SSNs, the sensor readings for MSNs still reflect events occurring within a particular region. Our concept of virtual static sensors is directly motivated by the above fact. Every VSS, like sensors in SSNs, 'monitors' a fixed region called its coverage area. An event occurring within a VSS's coverage area is reflected in its readings. However, unlike sensors in SSNs, VSSs do not have real existence and do not 'report' data to a base station. On the contrary, they are 'created' in our technique virtually to ease the spatio-temporal data mining.

A VSS reports a reading if there exists at least one real mobile sensor in the coverage area. A VSS is active if it reports in the current round and is inactive otherwise. VSS readings are readings of the real mobile sensor(s) which are present in the VSS's coverage area. In situations when multiple real mobile sensors are in a VSS's coverage area, the VSS reports the average of all the real mobile sensors' readings. There are two reasons for considering the average reading: (1) multiple sensors monitoring the same small coverage area most likely will report similar readings; and (2) any event occurring in the common coverage area will be reflected in the readings of all the sensors monitoring that area. As a hexagon is the atomic coverage region in DEMS, the radius of each hexagon is usually small enough to assure the variance of real sensors' readings from the same hexagon to be minimal, and averaging all readings from sensors from the same hexagon will be close to the real value of the corresponding region. A VSS is called a missing VSS if one real mobile sensor exists or expected to exist within the coverage area of that particular VSS and the reading from the real mobile sensor is missing.

The total monitoring region for any MSN or SSN is fixed either due to application specifications or hardware constraints. However, we further sub-divide the MSN's monitoring region into fixed size hexagons with a VSS 'covering' each particular hexagon. We choose hexagonal coverage area as they do not suffer from overlapping or uncovered regions as in the case of circular coverage area. Thus, in our monitoring area, we do not encounter regions where a real mobile sensor can map to multiple VSS (for overlapping regions) or cannot map to any VSS (for uncovered regions). Two virtual static sensors are neighbors if their covered hexagons share at least one edge. Due to the static nature of VSSs, they have a static spatial relation among themselves and can be co-related too. Finally consecutive readings from a VSS are originated from the same location and can show temporal relationships among them. Hence VSS readings are directly stored in our MASTER-tree. So, in DEMS, the MASTER-tree represents the relationships among the VSSs. We assume that at any instance, all the mobile sensors report their readings to the base station, which is then mapped to the corresponding VSSs. Figure 2 shows the mapping algorithm in details. For each real mobile sensor, DEMS finds the appropriate VSS (lines 3 & 4) using a geometric mapping between location and hexagon. If the location of the real mobile sensor is missing, DEMS predicts the expected location for the real mobile sensor and maps it to the appropriate VSS for that predicted location. If the mobile sensor reading is missing, DEMS marks the corresponding VSS as missing. Finally, in the loop from lines 11 to 18, each VSS is marked appropriately as active, inactive or missing. At any particular time, only the active virtual static sensors are stored in their appropriate MASTER-trees.

```
Procedure mapReal2Virtual(RealSensorData listRSData,
VirtualSensorData listVSData)
  1   for each real sensor rs
  2     if(rs is not missing)
  3       location ← listRSData(rs).Location
  4       vs ← findVirtualSensor(location)
  5      listVSData(vs).addReading(listRSData(rs).Reading)
  6     else
  7       location ← predictLocation(rs)
  8       vs ← findVirtualSensor(location)
  9       listVSData(vs).status←missing
 10   end loop
 11   for each virtual static sensor vs
 12     if(listVSData(vs) has data)
 13       listVSData(vs).status←active
 14   listVSData(vs).reading←average(listVSData(vs).Readin
      gs)
 15     else
 16       if(listVSData(vs).status is not missing)
 17         listVSData(vs).status ←inactive
 18   end loop
end procedure
```

Figure 2: Mapping mobile sensor readings to virtual static sensor readings

3.3 The MASTER-tree Projection Module

A MASTER-tree is like a pattern tree, which is used to represent arbitrary relationships among all Boolean itemsets [4]. A pattern tree is equivalent to a spanning tree of a binary hypercube which represents all possible Boolean items relationships; but the computational complexity of a pattern tree is exponential. However, grouping items into a set of clusters and pruning the pattern tree or its equivalent hypercube lowers the computational complexity. A pattern tree unduly favors only the right most leaf node and extracts the relationships of this node with all other nodes. A MASTER-tree does not suffer from those issues of a pattern tree. It combines the various pattern trees regarding each node and prunes the common paths in the resulting tree and forms a new tree called a MASTER-tree [1].

In a MASTER-tree, each tree node represents a VSS. The data distribution of a particular VSS node over a particular vector space is stored in each node. The complete vector space, in which the VSS readings occur, is discretized into a finite number of cells. Technically, for each cell, an arbitrarily accurate data distribution function or probability distribution function can be represented by an infinite number of moments in statistical theory. However, computationally, only a finite number of moments plus element counters are stored in the MASTER-tree nodes (typically the first four moments). An element counter is the number of VSS readings belonging to the cell associated with the corresponding MASTER-tree node. For each cell, a few moments are stored, and the cells across nodes are linked following the MASTER-tree paths. These cells and links form a grid structure (GS). As GS depends on a finite number of cells and a fixed number of nodes in a particular cluster, it does not grow exponentially with the increase in the number of rounds of sensor readings. Thus, the MASTER-tree projection module is to establish a MASTER-tree for each cluster and then to incrementally update the GS as a new round of sensor readings arrives. This maintains the up-to-date association rules among the VSSs in a cluster to serve data analysis purposes. Interested readers are refer to [1] for details about this module.

3.4 The Data Estimation Module

The data estimation module computes the estimated value for the missing mobile sensor. Initially, the location of the missing mobile sensor is predicted based on the user-defined minimum support and minimum confidence using Morzy's approach [14]. If the algorithm fails to predict the next location, DEMS uses the last reported location of the missing mobile sensor as its current location. Location prediction is preceded by mapping the missing mobile sensor to the corresponding VSS, which is called missing VSS. The estimated missing mobile sensor reading is the estimated missing VSS reading computed from the MASTER-tree.

The data estimation module accomplishes the task in an iterative way. First it obtains the prior distribution of the missing VSS ($mVSS$) from the MASTER-tree, i.e., the rule $\Phi \rightarrow mVSS$ (here Φ means empty). If the rule satisfies the user-defined error margin and the minimum support and minimum confidence thresholds, the rule holds and the estimated value is produced by taking the average of the prior distribution of mVSS. However, if the error margin requirement is not satisfied, the related information from the other tree nodes (VSSs) is considered for re-estimation. Here, the data estimation module chooses one more new antecedent node to infer the mVSS's reading. As every node represents a VSS, a node can be an antecedent node if the corresponding VSS is active. The initial relevant subspace for the antecedent node is simply the cell picked up based on its current reading. When the actual support does not satisfy the minimum support threshold, the relevant subspace is augmented iteratively until the actual support is no less than the minimum support. However, if the support requirement cannot be satisfied even if the relevant space reaches its upper limit, i.e., the complete subspace, the module removes this node and considers a new prior node. This process of adding a new antecedent node is repeated until the estimation procedure meets one of the following conditions: (1) a rule that satisfies the minimum support, minimum confidence and maximum error margin is found, or (2) no more nodes within the cluster is to be added to the antecedent nodes set. The procedure then returns the estimated value using the last expected value (the average) over the obtained consequent subspace. The estimated mVSS's reading is directly used as the estimated reading for the missing mobile sensor.

4. EXPERIMENTAL DESIGN AND RESULT ANALYSIS

In this section, we compare DEMS with two existing algorithms: SPIRIT [16] and TinyDB [11]. Although both TinyDB and SPIRIT are designed for static sensors, it can be argued that they can still be used for data estimation for mobile sensors disregarding sensor's mobility. We also compare it with the Average which is a statistical baseline method where the missing reading is estimated by averaging all other known sensor readings of the current round.

4.1 Experimental Datasets

4.1.1 DAPPLE Project Dataset

The real life dataset is obtained from the DAPPLE project [17]. The data are about carbon monoxide (CO) readings collected over a period of two weeks around Marylebone Road in London. The mobile sensors monitoring the atmospheric CO level are attached to PDAs which store these readings. The data sampling rate of the sensors is once every second. The software on the PDAs generates log files containing the CO levels with the locations and the timestamps associated with the readings. Each reading was carried out with a single sensor kit every second for a duration of about 45 minutes over a two weeks period. Simultaneous use of multiple sensors (usually three) was limited to some days only. For our experimental purposes, we considered the instances when three sensors were simultaneously recording CO pollution levels for a considerable period of time. We chose Thursday, , when three sensors were simultaneously recording for about 32 minutes, resulting in 600 rounds (after disregarding the missing rounds) of CO readings.

Table 1: Average MAEs for DAPPLE project dataset

Approach	Average MAE
DEMS	0.0
Average	1.2717
TinyDB	0.6331
SPIRIT	0.9437

4.1.2 Factory Floor Temperature Dataset

Besides the above real life application dataset, we also synthesized a factory floor temperature dataset [20] which exhibits dynamically changing phenomena. In this experiment, machines are placed on a grid floor. Initially, all machines are off and the starting temperatures for all grid points are set to zero. The boundary grid point temperature is held at zero throughout the experiment. Then some machines will be turned on for a number of rounds; the temperatures on these machines will reach a high constant temperature and heat will disperse on the floor. At each time step, a grid point is updated using the heat transfer formula used in [20]. In this simulation, 100 mobile sensors were roaming around in random directions to monitor the factory floor and reported the temperature readings from different locations at different points in time. In our simulations, we sampled the mobile sensor readings once per hour. In total we gathered 5000 rounds of readings from 100 sensors.

4.2 Performance Comparison Studies

In this section we compare the performances of DEMS, Average, SPIRIT [16], and TinyDB [11] in terms of mean absolute error (MAE). MAE is calculated using the following formula: $MAE = \frac{\sum_{i=1}^{n} |e_i - v_i|}{n}$ where e_i is the estimated value, v_i is the original value for the i-th data point, and n is the total number of data points. MAE is thus the magnitude, not the percentage, of the error. We specifically study the impacts of the number of rounds of sensor readings on the estimation accuracy.

4.2.1 Result for DAPPLE Project Dataset

Figure 3 shows the change of MAE with the change of number of rounds of sensor readings. The MAE value of 0 for DEMS implies that DEMS estimates the missing data with no error. A possible reason is that the DAPPLE project

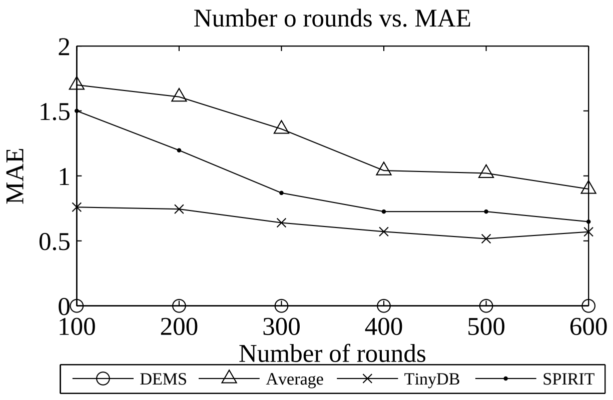

Figure 3: Number of rounds vs. MAE for DAPPLE project dataset

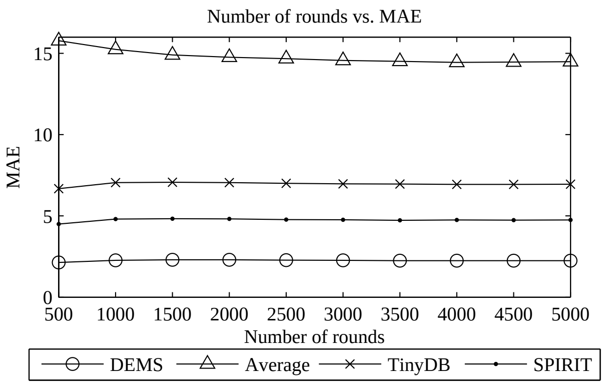

Figure 4: Number of rounds vs. MAE for Factory floor temperature dataset

dataset has very few variations (the CO levels are within the range 0 6) and the sensors have very high spatial correlations. In most cases the readings in the same hexagon are the same. Hence, DEMS produces a zero error in terms of MAE. The MAEs for other approaches are comparatively high at the beginning and become stable at the end as the number of rounds increases.

Table 1 shows the average MAE for all the approaches. DEMS almost perfectly estimates the missing values while Average gives the highest error compared to SPIRIT and TinyDB.

4.2.2 Result for Factory Floor Temperature Dataset

We performed a similar study for the factory floor temperature dataset. This dataset have more variations (temperatures are in the range 0 100C) compared to the DAPPLE project dataset. Figure 4 shows the change of MAE with respect to the change of number of rounds. The MAE for each approach remains almost constant when the number of rounds changes. As this dataset has more variations than the DAPPLE project dataset, even though DEMS still performs better than the other techniques, its performance is not as good as its performance with the DAPPLE project dataset.

Table 2 shows the average MAE for all the approaches. The average errors produced by Average, SPIRIT and TinyDB are about seven times, three times, and two times more than that produced by DEMS, respectively. DEMS is thus very effective in estimating missing sensor data.

Table 2: Average MAEs for Factory floor temperature dataset

Approach	Average MAE
DEMS	2.2538
Average	14.7787
TinyDB	6.9621
SPIRIT	4.7472

5. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a new technique (DEMS) to estimate missing data in MSN applications. Experimental results show that the estimated values computed by DEMS are more accurate than those produced by the existing techniques: Average, SPIRIT [16], and TinyDB [11]. For future work, we will consider the case when multiple mobile sensors report data at different times. We envision scenarios where considerable delays may exist between each sensor's readings. Finally as DEMS currently is designed for single hop MSNs only, we plan to expand the scope of DEMS to include multi-hop MSNs, mobile base station, and clustered MSNs.

6. ACKNOWLEDGEMENT

This work has been supported in part by the NASA under the grant No. NNG05GA30G.

7. REFERENCES

[1] H. Chok and L. Gruenwald. An online spatio-temporal association rule mining framework for analyzing and estimating sensor data. In *IDEAS*, 2009.

[2] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. Saluja. Sensor deployment strategy for target detection. In *1st ACM international workshop on Wireless sensor networks and applications*, 2002.

[3] W. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods.

[4] C. Giannella, J. Han, J. Pei, X. Yan, and P. Yu. Mining frequent patterns in data streams at multiple time granularities. *Next Generation Data Mining, Eds. H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha*, 2003.

[5] L. Gruenwald, H. Chook, and M. Aboukhamis. Using data mining to estimate missing sensor data. In *ICDMW*, 2007.

[6] L. Gruenwald, H. Yang, S. Sadik, and R. Shukla. Using data mining to handle missing data in multi-hop sensor network applications. In *MobiDE*, 2010.

[7] S. Guha, N. Koudas, and K. Shim. Data streams and histrograms. In *ACM Symposium on Theory of Computing*, 2001.

[8] T. Haenselmann. *An FDL'ed Textbook on Sensor Networks*. GNU Free Documentation License, 2005.

[9] N. Jiang and L. Gruenwald. Research issues in data stream association rule mining. *ACM Sigmod Record*, 2006.

[10] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley. Mobility improves coverage of sensor networks. In *MobiHoc*, 2005.

[11] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tinydb: An acquisitional query processing system for sensor networks. *Transactions on Database Systems*, 2005.

[12] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *1st ACM international workshop on Wireless sensor networks and applications*, 2002.

[13] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M.B.Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *InfoCom*, 2001.

[14] M. Morzy. Mining frequent trajectories of moving objects for location prediction.

[15] N. Oceanic and A. Administration. *Advisory Comittee on Commercial Remote Sensing*. http;//metar.noaa.gov, Last access 2010.

[16] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, 2005.

[17] D. project. *UCL Carbon Monoxide Data Collection at DAPPLE Site*. http://www.cs.ucl.ac.uk /research /vr /Projects /envesci /DAPPLE2004 /UCLDAPPLE.html, Last access 2010.

[18] L. Schwiebert, S.Gupta, and J.Weinmann. Research challenges in wireless networks of biomedical sensor. In *MobiCom*, 2001.

[19] G. Sibley, M. Rahimi, and G. Sukhatme. Robomote - a tiny mobile robot platform for large-scale sensor networks. In *ICRA*, 2002.

[20] A. Silberstein, K. Munagala, and J. Yang. Energy-efficient monitoring of extreme values in sensor networks. *ACM Sigmod Record*, 2006.

[21] N. Vijayakumar and B. Plale. Missing event prediction in sensor data streams using kalman filters. *Knowledge Discovery from Sensor Data, Eds. Taylor and Francis*, 2009.

[22] G. Wang, G. Cao, T. parta, , and W. Zhang. Sensor relocation in mobile sensor networks. In *InfoCom*, 2005.

[23] W.Tobler. *A Computer Movie Simulating Urban Growth in the Detroit Region*. Economic Geography, 1970.