IEEE UEMCON 2017

Efficient Infrastructure Damage Detection and Localization Using Wireless Sensor Networks, with Cluster Generation for Monitoring Damage Progression

William Contreras
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
Newark, NJ 07102
wc42@njit.edu

Sotirios Ziavras
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
Newark, NJ 07102
ziavras@njit.edu

***Abstract*—Structural health monitoring (SHM) involves the development of strategies to assess the condition of instrumented engineering structures. One of the most critical applications of SHM systems is civil infrastructure. For this application, it is particularly important that SHM systems be inexpensive and easy to deploy, since the maintenance of infrastructure is often inadequately funded. Wireless sensor networks (WSN) can be very useful toward this end. We present an efficient WSN-based SHM algorithm for detecting, localizing, and monitoring the progression of damage in infrastructure applications. The algorithm utilizes a novel vibration-based pattern matching technique that is very well suited for low-power WSN nodes. During a training phase, a body of reference patterns is formed from vibrations observed at sensor nodes distributed throughout the structure. During the operational phase, observed patterns are compared to the reference patterns to determine if a match exists. Through the use of an innovative distributed algorithm, a time complexity of O(logN) is achieved for the matching process. If a match does not exist, potential damage is indicated and the reference pattern closest to the observed pattern is determined using Euclidean distance. The difference between the two patterns indicates the sensor nodes at which potential damage exists. Clusters are then formed around these sensor nodes in order to monitor the progression of local damage. Simulations are performed in MATLAB for a typical bridge deployment in order to determine the degree of overlapping that occurs as clusters are generated in response to potential damage. The simulations indicate that overlapping increases gracefully as the number of nodes experiencing damage increases.**

Keywords— Distributed algorithm, Vibration measurement, Wireless sensor network, Clustering algorithm

I. INTRODUCTION

The field of structural health monitoring (SHM) concerns the development of engineering strategies to assess the condition of instrumented engineering structures. Structural health monitoring systems can be used on many different types of structures, from civil to mechanical. One of the primary applications of SHM systems is civil infrastructure. This is a particularly important application as the risk to life and property is high in the event of structural failure. Structural health monitoring systems facilitate preventative maintenance of infrastructure, whereby the maximum lifetime can be extracted out of structures, and the risk to life and property can be minimized. Given that infrastructure is often inadequately funded, however, SHM systems should be inexpensive and easy to install. Wireless sensor networks (WSNs) are a good choice for SHM systems since the cost of deploying and operating them tends to be low. The system that we are now proposing is based upon an efficient pattern matching technique that is ideally suited for low-power, low-cost WSN sensor nodes, or motes. The algorithm uses measured vibrations from motes distributed throughout the structure to form patterns that are checked against a body of reference patterns representative of a healthy condition, to determine whether or not potential damage is occurring. If a match is not found, potential damage is indicated. The system uses *only* measured vibrations to evaluate structural condition. As such, it is output-only. It does not require detailed engineering knowledge about the structure, or information about environmental or operational parameters. This greatly simplifies the deployment and operation of the system, and thus contributes to its efficient and inexpensive nature.

Many output-only techniques have been proposed in the literature. One such technique is factor analysis [1]. An approach that has been investigated extensively is principal component analysis [2-4]. The authors of [5] propose an output-only method involving regression analysis. In [6], a method involving a spatio-temporal infinite impulse response filter is presented. In [7], the authors employ hidden Markov models to perform anomaly detection on gas turbines. The authors of [8] use transmissibility as a damage sensitive feature. In [9], the authors use Welch's t-test to look for changes over time in the distribution of statistical features obtained from the raw structural response data. The usage of autoregressive models has also been investigated extensively

[10-13]. These approaches hold promise with regard to detecting damage using only structural response data, however they tend to be more computationally intensive than the technique that we are currently proposing.

To the best of our knowledge, the least computationally intensive approach that has been proposed thus far can be found in [14]. In [14], the authors present a WSN-based system wherein a pattern matching technique is applied to vibration data from motes placed around the structure to efficiently detect damage. The methods of [15] and [16] build upon this technique by incorporating a means to localize damage, and a means to minimize the risk of false positives. They do this, however, at the expense of some computational simplicity. With the method that we are currently proposing, we build upon [14] by introducing a means to localize damage without increasing energy consumption during the normal operation of the system. In addition, we greatly improve the time complexity of the algorithm of [14], from $O(N^2)$ to O(logN). We also propose a means of generating clusters to monitor the progression of damage in local areas of the structure. We simulate the process of cluster generation in MATLAB and show that the overlapping of clusters increases gracefully as the number of motes experiencing damage increases.

II. System Operation

A. Overview

Our system will consist of a WSN with motes distributed throughout the structure of interest. Our SHM algorithm will consist of two primary phases: 1) a training phase and 2) an operational phase. During the training phase, the motes of the system will be grouped into an initial set of clusters. During the operational phase, in order to monitor the spread of damage, new clusters will be added according to where potential damage is detected. Each cluster will have a single cluster head, and will run a separate instance of the operational phase of the algorithm. The motes in the cluster that communicate with the cluster head will be referred to as member motes.

B. Operational Phase

During the operational phase, in each cluster, the motes will turn on in a synchronized fashion at regular intervals, and collect acceleration data from the structure (for a duration on the order of a second). When done collecting data, each mote will use its acceleration data to calculate the value of the designated damage sensitive feature. A damage sensitive feature is a quantity calculated from the raw system response data that is sensitive to the presence of damage. A time series average is an example of a damage sensitive feature. Each mote will then quantize its damage sensitive feature value using (1). The resulting value will be referred to as a subrange value. In (1), x is the value of the damage sensitive feature and Y is a constant that determines the size of the quantization increment. Once each mote of the cluster has in its possession a subrange value, the motes will share their subrange values amongst each other to form a pattern, or tuple, of subrange values. The tuple will be checked against a body of reference tuples obtained from the healthy structure during the training phase to determine if potential damage is occurring. The total number of tuples in the body of reference tuples will be denoted by P.

$$v = \left\lfloor \frac{x}{Y} \right\rfloor \tag{1}$$

The tuple will be checked in pieces, in a distributed fashion on the motes of the cluster. The number of motes in a given cluster will be denoted by N. Graphically, the way in which the tuple will be checked will resemble a binary tree. The leaves of the tree represent each mote in the cluster. For our explanation, we will assume that the number of motes in the cluster, N, is a power of two, and thus that the binary tree is full. Additionally, each mote in the cluster will contain the entire body of reference tuples. The process is illustrated in Fig. 1 for N=4. In the figure, the s variables represent the subrange values for each mote. In the highest level of the tree (the level which contains the leaves) each mote checks its subrange value against the subrange values corresponding to itself in the body of reference tuples. After this, every other mote (i.e. the first mote, the third mote, the fifth mote, etc) will transmit its subrange value to the mote adjacent to it in the highest level of the tree. The nodes in the second to highest level of the tree will represent the receiving motes. In this level, each receiving mote will combine the received subrange value with its own subrange value to form a 2-tuple, and will check the 2-tuple against the 2-tuples in the body of reference tuples corresponding to the two motes from which the subrange values came. Once this is done, every other receiving mote will transmit its 2-tuple to the receiving mote adjacent to it in the second to highest level. The nodes in the third to highest level of the tree will represent the new receiving motes. Each new receiving mote combines the received 2-tuple with its own 2-tuple to form a 4-tuple, and checks the 4-tuple against the 4-tuples in the body of reference tuples corresponding to the motes from which the subrange values came. This process will continue until the full tuple is formed at the level of the root node and checked against the full body of reference tuples. The root node will represent the cluster head, which will check

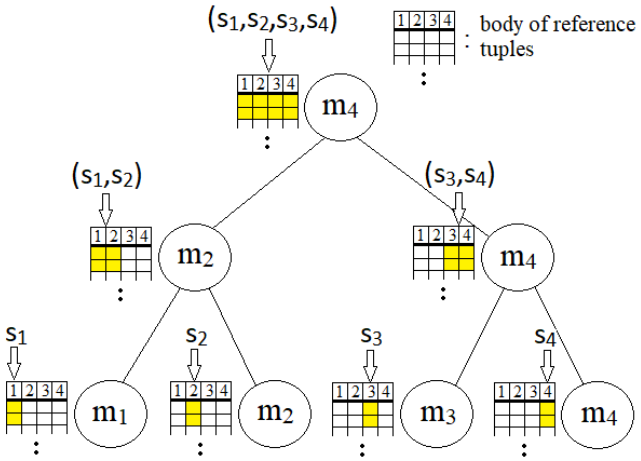

Fig. 1. Illustration of the tuple matching process for a cluster where N=4. On the leaf level of the tree, single subrange values are checked; on the second level, 2-tuples are checked; at the root node (the cluster head) the entire tuple is checked.

the full tuple. The tuple will be checked this way to make the system robust against individual mote failure, and to allow results to be generated more quickly in the event potential damage is discovered.

A hashing approach will be used by each mote to search the body of reference tuples. The hash key for a given tuple will be the sum of the subrange values that constitute the tuple. For a hash function, a simple modulus can be used, as it is anticipated that the tuples, and thus the hash keys, should have a fairly uniform distribution. If a great deal of clustering occurs (many keys to a few hash values) due to a lack of uniformity in the distribution of the hash keys, a hash function tailored to the distribution of the hash keys can be developed. The hash tables can be resized if the body of reference tuples is updated.

If, at any point in the tuple matching process the searching mote does not find a match for the observed tuple, an alert is generated indicating that damage could potentially be present. The alert is sent to the cluster head, and the cluster head forwards the alert to the base station. In addition to this, the cluster head initiates a process to localize the damage. The cluster head will immediately gather the subrange values from all of the member motes to form the N-tuple. The cluster head then determines which reference tuple the offending observed tuple is closest to, in terms of Euclidean distance. Using (2), the cluster head calculates the Euclidean distance between the observed tuple and each reference tuple. In (2), s_o denotes an observed subrange value and s_r denotes a reference subrange value. The numbers in the subscripts denote the mote with which the subrange value is associated. The cluster head then determines which Euclidean distance is smallest in order to determine which reference tuple the observed tuple is closet to. Once the closest tuple is determined, it is determined which subrange values of the observed tuple deviate from those of the reference tuple. This can be done by subtracting, in an element-wise fashion, the subrange values of the observed tuple from those of the closest reference tuple. The motes for which the difference is non-zero are the motes for which damage is assumed to be occurring.

$$d = \sqrt{\left(s_{o,1} - s_{r,1}\right)^2 + \left(s_{o,2} - s_{r,2}\right)^2 + \cdots + \left(s_{o,N} - s_{r,N}\right)^2} \quad (2)$$

Once it is determined which motes are experiencing potential damage, the cluster head sends this information to the base station. The base station then adds clusters to monitor the progression of damage. A cluster is formed around each mote where potential damage is detected. Each mote where potential damage is detected is made into a cluster head and the motes within transmission range of these motes are included in the cluster. This allows the progression of damage to be monitored in the vicinity of the mote where potential damage is initially detected. In our method we allow clusters to overlap. As a result of this, as potential damage is detected, motes will have memberships to more than one cluster. The rank of a mote is defined to be the total number of clusters that the mote belongs to.

C. Pseudocode

Algorithm 1. Operational Phase Algorithm for Each Cluster

```
1:  while (1)
2:    for i = 1:N
3:      data = collect_vibration_data ( );
4:      x_i = calculate_feature_value ( data );
5:      v_i = ⌊x_i / Y⌋;
6:    end
7:    for i = 1:logN
8:      transmit_subrange_values ( );
9:      result = tuple_matching ( );
10:     if result == 0
11:       transmit_alert ( );
12:     end
13:   end
14:   if result == 0
15:     gather_subrange_values ( );
16:     for i = 1:P
17:       d_i = sqrt( (s_o,1 - s_r,1,i)^2 + (s_o,2 - s_r,2,i)^2 + ... + (s_o,N - s_r,N,i)^2 )
18:     end
19:     tuple_index = minimum ( d );
20:     v_closest_tuple = body_of_reference_tuples (tuple_index);
21:     damage_nodes = non_zero (v_observed - v_closest_tuple);
22:     transmit_data_to_base_station ( damage_nodes );
23:   end
24: end
```

The operational phase algorithm performed in each cluster is delineated in Algorithm 1. In the first *for* loop, which begins in Line 2, each mote gathers vibration data, calculates the value of the statistical feature, and calculates the subrange value. Note that, since the motes do this simultaneously as described in Section II-B, the iterations of the loop will execute in parallel, not sequentially. In the second *for* loop, which begins in Line 7 of Algorithm 1, the motes perform the distributed tuple checking process. At each iteration, the subrange values are transmitted to the appropriate motes. The receiving mote forms a tuple out of the received subrange values and checks the tuple against the body of reference tuples to determine if a match exists. If a match exists, the *tuple_matching* function returns a value of one; if a match does not exist, a value of zero is returned and an alert is transmitted to the cluster head. If a match is not found, the process of localizing the damage is executed by the cluster head. The process is indicated by the *if* statement beginning at Line 14 of Algorithm 1. The cluster head first gathers the subrange values from the motes and forms the observed N tuple. It then determines the Euclidean distance between the N tuple and each of the reference tuples,

which is indicated by the *for* loop beginning at Line 16 (recall that *P* is the total number of reference tuples). The minimum Euclidean distance is determined and the associated *N* tuple is selected from the body of reference tuples. The selected reference tuple is subtracted in an element-wise fashion from the observed tuple and the difference, whose non-zero elements indicate the motes at which damage is potentially occurring, is transmitted to the base station. The base station forms new clusters based upon the indicated motes.

D. Time Complexity

There are two possible outcomes from the operational phase algorithm performed in each cluster: either potential damage will be discovered, or it will not be discovered. Of these two possibilities, the latter will occur much more frequently and will constitute the normal operation of the system. Most often, then, only the data collection and tuple checking processes (the first two *for* loops of Algorithm 1) will run. The time complexity associated with the data collection process (the first *for* loop) is O(1). This is because it takes a fixed amount of time to collect data, calculate the value of the damage sensitive feature, and calculate the subrange value. Regarding the tuple checking process (the second *for* loop), the main factor that effects execution time is *N*. The time complexity for this process is O(logN), since the number of iterations in the search process is equal to *logN*, and the complexity of each iteration is (owing to the hashing approach) O(1).

III. Simulation Results

To test our algorithm, we developed a MATLAB simulation. The primary goal of our simulation was to make a determination as to how much overlapping of clusters occurs when potential damage is detected. In the simulation, we did a case study on an average-size bridge, 10 meters wide by 100 meters long. We ran the simulation for two separate cases. In the first case, the motes were distributed in a regular fashion over the length of the bridge. In particular, five motes were distributed evenly over the width of the bridge, every five meters. One hundred motes were simulated in total. An illustration of this distribution of motes can be seen in Fig. 2. In practice, such a distribution might be implemented in order to monitor the superstructure of a bridge, where the motes are placed on longitudinally-oriented I-beams supporting the bridge deck. Such an application is illustrated in Fig. 3. In the second case, 100 motes were distributed in a random fashion over the length of the bridge. In each case, we assumed the transmission range of the motes to be 20m, which is the indoor range of the popular TelosB mote (the underside of the bridge will be roughly similar to an indoor environment).

For each case, in the simulation, the coordinates of the motes were first established. Subtractive clustering was used to establish the initial clusters. Subtractive clustering is a fast algorithm for determining the number of clusters and the cluster centers for a particular data set [17]. In particular, MATLAB's 'subclust' function was used to generate the initial cluster heads. Note that clusters generated using subtractive clustering do not overlap. Each mote was assigned to the nearest cluster head. Damage was then simulated at several different numbers of motes. For each number of motes experiencing damage, the average number of motes for each rank was determined (recall that the rank of a mote is the total number of clusters that the mote belongs to). As an example, it was determined how many motes, on average, belong to three clusters when five motes are experiencing damage. To do this, each number of motes experiencing damage was simulated 10000 times (we found that this value allowed for sufficient convergence about the average). As an example, damage was simulated at two motes 10000 times. Each time, the motes experiencing damage were selected randomly from the 100 total motes in the WSN, and the number of motes for each rank was determined. In the end, for each number of motes experiencing damage, the average number of motes for each rank was calculated. An example of the network when two motes are experiencing damage can be seen in Fig. 4.

The results of our simulations can be seen in Table I and Table II. Table I shows the results for the regular distribution of motes and Table II shows the results for the random distribution of motes. Note that the highest rank possible for a given number of motes experiencing damage is equal to the number of motes experiencing damage plus one. Each mote will belong to at least an initial training phase cluster (from subtractive clustering), and can belong to as many additional clusters as there are motes experiencing damage. As such, in the tables, for each number of motes experiencing damage, all of the average values for ranks exceeding the number of motes experiencing damage plus one, are equal to zero. A rank equal

Table I. Data for the regular distribution. The average number of motes is shown for each rank, for several different numbers of motes experiencing damage.

	Number of motes experiencing damage				
Rank	1	3	5	7	9
1	67.1948	30.1316	13.9339	6.5943	3.1834
2	32.8052	44.7821	33.3566	20.9148	12.3974
3	0	21.6509	32.2513	30.6722	23.468
4	0	3.4354	16.0782	25.0527	26.9556
5	0	0	`3.9847	12.4537	19.9017
6	0	0	0.3953	3.671	10.0152
7	0	0	0	0.5951	3.3196
8	0	0	0	0.0462	0.6776
9	0	0	0	0	0.0781
10	0	0	0	0	0.0034
11	0	0	0	0	0

Table II. Data for the random distribution. The average number of motes is shown for each rank, for several different numbers of motes experiencing damage.

	Number of motes experiencing damage				
Rank	1	3	5	7	9
1	63.1521	25.9154	11.1563	4.8683	2.3109
2	36.8479	42.9456	28.7344	16.7529	9.5263
3	0	25.7148	32.4731	27.2286	18.787
4	0	5.4242	20.2801	26.6824	24.8512

5	0	0	6.5016	16.5499	22.3397
6	0	0	0.8545	6.3648	13.9986
7	0	0	0	1.4362	6.1507
8	0	0	0	0.1169	1.7208

9	0	0	0	0	0.2952
10	0	0	0	0	0.0196
11	0	0	0	0	0

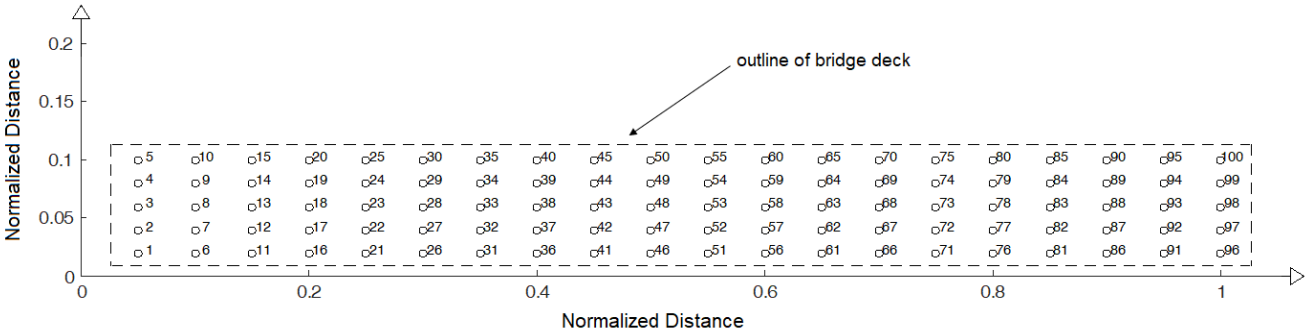

Fig. 2. Plot from MATLAB showing the motes regularly distributed over the simulated region.

to one indicates membership to only a training phase cluster.

In Table I and Table II, it can be seen that for a given rank, as the number of motes experiencing damage increases, in general, the average number of motes initially increases and then begins to decrease. The eventual decrease in the number of motes per rank makes sense given that, as the number of motes experiencing damage increases, there are more ranks that a given mote can belong to. At first, it tends to increase, however. Looking at the average rank for each number of motes experiencing damage can help explain this.

The average rank for each number of motes experiencing damage can be seen in Fig. 5 for three different cases: 1) the regular distribution of motes, 2) the random distribution of motes, and 3) the hypothetical case where motes are evenly spread between the different possible ranks for a given number of motes experiencing damage. If motes were evenly spread among the ranks, the average number of motes per rank would monotonically decrease as the number of motes experiencing damage increases. In this case, the average rank for a given number of motes experiencing damage would simply be the arithmetic mean of all the possible ranks for that number of motes. From Fig. 5 it can be seen that the average rank for the regular and random distributions increases at a slower rate than it does for the case where motes are evenly spread among the ranks. This indicates that, unlike the case where the motes are

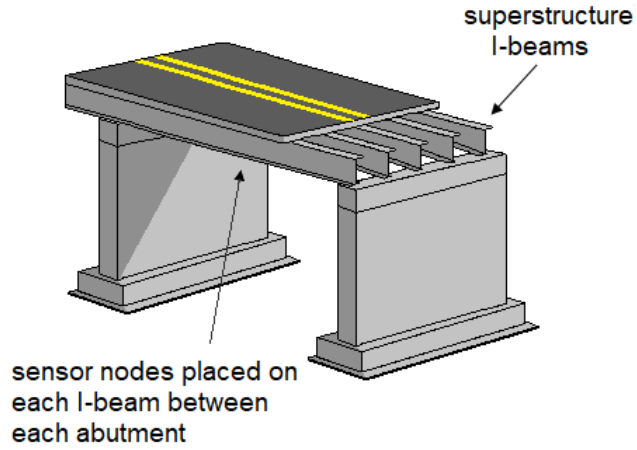

Fig. 3. Illustration of an application of the simulated system. Motes can be placed on I-beams supporting the bridge deck to monitor the bridge's superstructure.

evenly spread between ranks, there is a bias toward lower ranks as the number of motes experiencing damage increases. It can be concluded, therefore, that for the system of our case study (a typical case) it is more difficult to achieve higher ranks than it is to achieve lower ranks, unlike the average case where motes are evenly spread among the ranks. This helps to explain why, as the number of motes experiencing damage increases, the average number of motes per rank first increases and then, only when the number of motes experiencing damage is sufficiently high, begins to decrease.

It is interesting to note that average rank increases slightly faster with the random distribution than it does with the regular distribution. As such, it is somewhat easier to achieve a larger number of overlapping clusters with the random distribution than it is with the regular distribution. This might be because, with the random distribution, mote density is higher in certain regions than with the regular distribution. Note that for the regular and random distributions, average rank was computed for each number of motes through the use of a weighted arithmetic mean. The weight for each rank corresponds to the

average number of motes for that rank. Also note that, in Fig. 5, the R-squared value for each fit is equal to one, indicating that the data is perfectly linear.

IV. Conclusion

We have proposed a novel SHM algorithm that utilizes a vibration-based pattern matching technique that is well suited for low-power, low cost motes. Through the use of an efficient distributed algorithm, a time complexity of O(logN) is achieved. Localization of potential damage is performed using a nearest-neighbor approach with Euclidean distance. Clusters are generated around motes where damage is suspected in order to monitor the progression of damage. Simulation results from MATLAB indicate that cluster membership for a given mote grows gracefully as the number of motes experiencing damage increases. Future work includes testing the nearest neighbor approach to assess the accuracy with which it localizes damage.

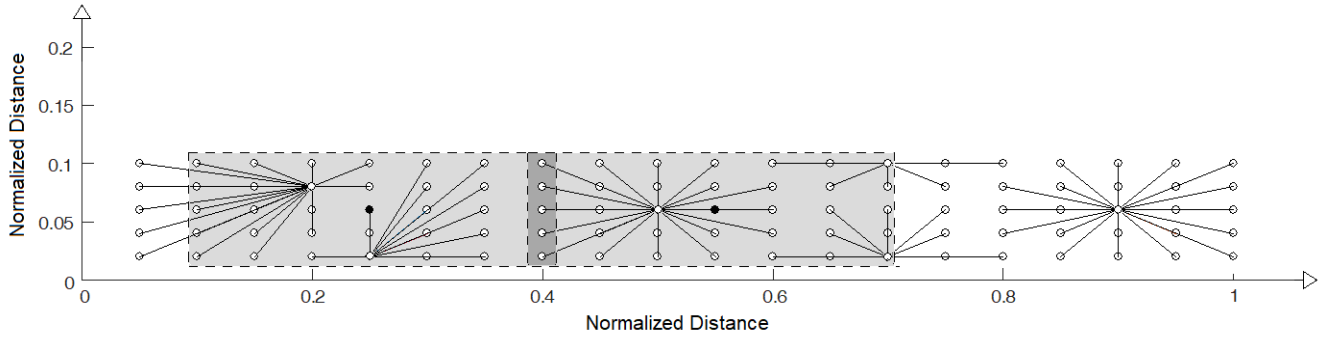

Fig. 4. Plot from MATLAB showing the clusters formed during the training phase using subtractive clustering and the clusters formed during the operational phase in response to potential damage being detected. Damage was simulated at two motes. These motes are indicated with a black fill. The clusters from the training phase are indicated by line segments connecting the motes to the cluster heads. The clusters from the damage detection process are indicated by grey regions. The dark grey illustrates an overlapping of the two clusters formed in response to potential damage being detected.

References

[1] V. Lamsa and T. Raiko, "Novelty detection by nonlinear factor analysis for structural health monitoring," IEEE International Workshop on Machine Learning for Signal Processing, 2010, pp. 468-473.

[2] Z. Hongyang, G. Junqi, X. Xiaobo, B. Rongfang, and S. Yunchuan, "Environmental effect removal based structural health monitoring in the Internet of Things," Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2013, pp. 512-517.

[3] W.-H. Hu, E. Caetano, and Á. Cunha, "Structural health monitoring of a stress-ribbon footbridge," Engineering Structures, vol. 57, no. 0, pp. 578-593, 2013.

[4] J. Kullaa, "Structural health monitoring under nonlinear environmental or operational influences," Shock and Vibration, p. 9, 2014.

[5] N. H. M. Kamrujjaman Serker, Z. Wu, and S. Li, "A nonphysics-based approach for vibration-based structural health monitoring under changing environmental conditions," Structural Health Monitoring, vol. 9, no. 2, pp. 145-158, 2010.

[6] D. Gorinevsky and G. Gordon, "Spatio-temporal filter for structural health monitoring," American Control Conference, 2006.

[7] A. D. Kenyon, V. M. Catterson, S. D. J. McArthur, and J. Twiddle, "An agent-based implementation of hidden Markov models for gas turbine condition monitoring," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 44, no. 2, pp. 186-195, 2014.

[8] M. Bocca, J. Toivola, L. M. Eriksson, J. Hollmen, and H. Koivo, "Structural health monitoring in wireless sensor networks by the embedded Goertzel algorithm," IEEE/ACM International Conference on Cyber-Physical Systems, 2011, pp. 206-214.

[9] A. Scianna, Z. Jiang, R. Christenson, and J. DeWolf, "Implementation of a probabilistic structural health monitoring method on a highway bridge," Advances in Civil Engineering, vol. 2012, 2012.

[10] Q. W. Zhang, "Statistical damage identification for bridges using ambient vibration data," Computers & Structures, vol. 85, no. 7–8, pp. 476-485, 2007.

[11] E. Hidalgo et al., "Wireless structural health monitoring system based on autoregressive models," 38th Annual Conference on IEEE Industrial Electronics Society, 2012, pp. 6035-6040.

[12] M. M. Alves et al., "Damage prediction for wind turbines using wireless sensor and actuator networks," Journal of Network and Computer Applications, vol. 80, pp. 123-140, 2017.

[13] S. Hoell and P. Omenzetter, "Optimal selection of autoregressive model coefficients for early damage detectability with an application to wind turbine blades," Mechanical Systems and Signal Processing, vol. 70–71, pp. 557-577, 2016.

[14] W. Contreras and S. Ziavras, "Wireless sensor network-based pattern matching technique for the circumvention of environmental and stimuli-related variability in structural health monitoring," IET Wireless Sensor Systems, vol. 6, no. 1, pp. 26-33, 2016.

[15] W. Contreras and S. Ziavras, "Low-cost, efficient output-only infrastructure damage detection with wireless sensor networks," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. PP, no. 99, pp. 1-10, 2017.

[16] W. Contreras and S. Ziavras, "Wireless sensor network-based infrastructure damage detection constrained by energy consumption," 7th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference, 2016.

[17] M. N. Halgamuge, S. M. Guru, and A. Jennings, "Energy efficient cluster formation in wireless sensor networks," Tenth International Conference on Telecommunications, 2003.

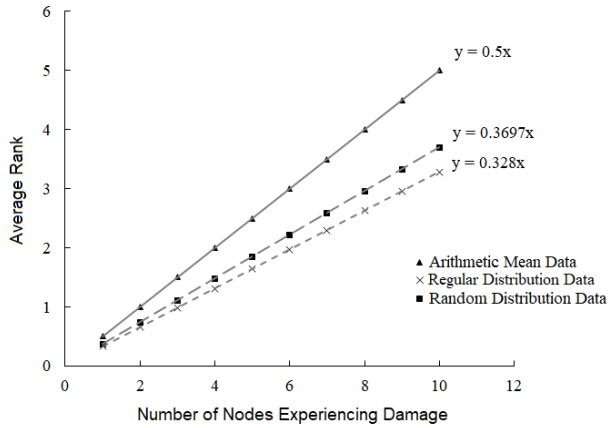

Fig. 5. Average rank for each number of nodes experiencing damage, for three cases: 1) regular distribution of motes, 2) random distribution of motes, and 3) the case where motes are evenly spread between the ranks.