J. Sens. Sens. Syst., 5, 63–72, 2016
www.j-sens-sens-syst.net/5/63/2016/
doi:10.5194/jsss-5-63-2016

JSSS Journal of Sensors and Sensor Systems Open Access

Combination of clustering algorithms to maximize the lifespan of distributed wireless sensors

Derssie D. Mebratu and Charles Kim

Electrical and Computer Engineering, Howard University, Washington DC, 20059, USA

Correspondence to: Derssie D. Mebratu (mebratu@scs.howard.edu)

Received: 9 November 2015 – Revised: 11 February 2016 – Accepted: 11 February 2016 – Published: 2 March 2016

Abstract. Increasing the lifespan of a group of distributed wireless sensors is one of the major challenges in research. This is especially important for distributed wireless sensor nodes used in harsh environments since it is not feasible to replace or recharge their batteries. Thus, the popular low-energy adaptive clustering hierarchy (LEACH) algorithm uses the "computation and communication energy model" to increase the lifespan of distributed wireless sensor nodes. As an improved method, we present here that a combination of three clustering algorithms performs better than the LEACH algorithm. The clustering algorithms included in the combination are the k-means^{++}, k-means, and gap statistics algorithms. These three algorithms are used selectively in the following manner: the k-means^{++} algorithm initializes the center for the k-means algorithm, the k-means algorithm computes the optimal center of the clusters, and the gap statistics algorithm selects the optimal number of clusters in a distributed wireless sensor network. Our simulation shows that the approach of using a combination of clustering algorithms increases the lifespan of the wireless sensor nodes by 15 % compared with the LEACH algorithm. This paper reports the details of the clustering algorithms selected for use in the combination approach and, based on the simulation results, compares the performance of the combination approach with that of the LEACH algorithm.

1 Introduction

Wireless sensor networks are being used for many different applications, such as monitoring chemical spills, detecting and assessing the extent of environmental contamination, and monitoring the movement of soldiers and weapons on the battlefield. However, their limited lifespan is a great concern when they are used in remote locations or in harsh environments.

Many different techniques have been introduced in an effort to maximize their lifespan, but these techniques have focused on having the nodes in a cluster send their data to a selected cluster head node that, in turn, reports the data to the base station. Therefore, the choice of the number of clusters and the way the cluster head node is selected are the main focuses of these techniques. Clustering and the use of cluster heads in wireless sensor networks have the potential to enhance the lifespans of a group of sensor nodes and to minimize the generation of noise in the signals exchanged between the sensor nodes and the base station (sink) (Heinzelman et al., 2000). In this approach, the cluster head organizes a reservation scheme to improve communication with the sensor nodes in the cluster, and the cluster head uses this scheme to aggregate, compress, and transmit the cluster's sensing data to the base station. Several technologies have been designed to improve the lifespan of the sensors. For example, algorithms were developed for this purpose by the energy efficient heterogeneous clustered scheme (EEHC) (Kumar et al., 2009) by the design of a distributed energy efficient clustering (DEEC)(Qing et al., 2006), and by the low-energy adaptive clustering hierarchy (LEACH) (Heinzelman et al., 2000). These goals of these algorithms were to determine the optimal number of clusters in a given number of sensor nodes and to selecting a head in a cluster of sensors. The low energy consumption clustering routing protocol (Kumar et al., 2009) improved the LEACH algorithm by utilizing the k-means algorithm that divides the sensor nodes into k clusters in the setup and steady-state phases. A major problem of the k-means algorithm was that it could not ac-

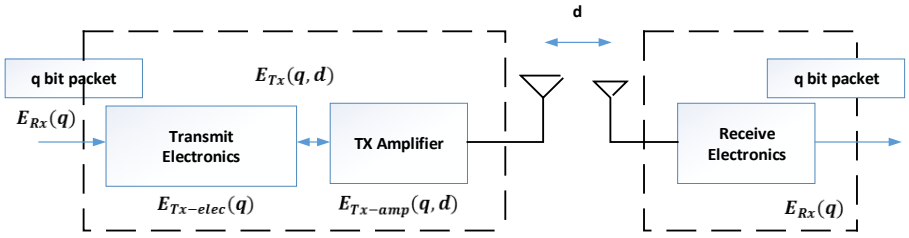

Figure 1. Radio energy model.

commodate the inevitable situation that the number of clusters gradually changed as the energy levels of the nodes decreased. Also, the method did not solve the sticky issue of initialization of the k-means process (Zhong et al., 2012). However, k-harmonic means (KHM) clustering solved the initialization problem by providing "soft membership", which assumes that a data element belongs to more than one cluster; also, if a data point is not close to any center or cluster, a "dynamic weighting function" provides a higher weight to the data element in the next iteration so that it becomes a candidate for all of the clusters. However, the KHM algorithm cannot provide an optimal number of clusters.

In this paper, we have provided detailed discussions of clustering algorithms; the combination of k-means^{++}, k-means, and gap statistics algorithms; the selective ways in which each is used and combined; and how, using the combination, the optimal number of clusters is generated, which leads to the maximum lifespan of a group of distributed wireless sensors. Before discussing the clustering algorithms and their combination, in the next section, we discuss a popular clustering algorithm, known as the low-energy adaptive clustering hierarchy algorithm (LEACH), for extending the lifespan of wireless sensors. Section 3 describes the selected clustering algorithms and their combination for determining the optimal number of clusters. Last, Sect. 4 provides the simulation results and compares the results provided by a combined clustering algorithm and the LEACH algorithm. Section 5 presents the conclusion.

2 Clustering algorithms

2.1 Low-energy adaptive clustering algorithm (LEACH)

The LEACH algorithm was developed to minimize the power consumption of wireless sensor nodes by determining the optimal number of clusters, k, in a group of distributed homogeneous wireless sensors based on the "computation and communication energy model" (Heinzelman et al., 2000). In order to determine the optimal number of clusters, k, first, the algorithm considers how much energy the head of a cluster consumes using the radio energy model depicted in Fig. 1. In the radio energy model, for a single bit transmission over a unit distance, E_{Tx} is the transmission energy dissipated, which is composed of two components, i.e., $E_{\text{Tx-elec}}(q)$, the electrical energy consumed for digital coding, modulation, and filtering a signal and $E_{\text{Tx-amp}}(q,d)$, the energy required for amplification.

Then, the total energy used to transmit a q bit message over a distance d is expressed by

$$E_{\mathrm{Tx}}(q,d) = E_{\text{Tx-elec}}(q) + E_{\text{Tx-amp}}(q,d). \quad (1)$$

The energy for k bit amplification is expressed by $E_{\text{Tx-amp}}(q,d) = q\varepsilon_{\text{fs}}d^2$, in a free space path ($\varepsilon_{\text{fs}}$) with distance squared (d^2). When a multi-path is considered, the amplification energy is defined as $E_{\text{Tx-amp}}(q,d) = q\varepsilon_{\text{mp}}d^4$, for q bits with distance to the fourth power (d^4). The LEACH algorithm proposed that the free space (fs) model be used when the distance between the transmitter and the receiver is less than the threshold distance d_o (base station distance); otherwise, the multipath (mp) model is used, as summarized below:

$$E_{\mathrm{Tx}}(q,d) = E_{\text{Tx-elec}} + q\varepsilon_{\text{fs}}d^2, \quad d < d_o, \quad (2)$$

$$E_{\mathrm{Tx}}(q,d) = E_{\text{Tx-elec}} + q\varepsilon_{\text{mp}}d^4, \quad d \geq d_o. \quad (3)$$

The receiver's energy for a q bit receipt is calculated by

$$E_{\mathrm{Rx}}(k) = qE_{\text{elec}}. \quad (4)$$

Let us now consider energy consumption by the sensor nodes in a cluster of a multi-cluster sensor network. Assuming that there are N wireless sensor nodes uniformly distributed in a square region of $M \times M$ geographical units that have k clusters, there are N/k nodes per cluster, and, in each cluster, there is one cluster head node and $(N/k)-1$ non-cluster-head nodes (or "cluster member nodes"). In a cluster, during the steady-state phase, data transfer from the nodes to a cluster head as well as from the cluster head to the sink, which is located a long distance away, so the energy of the cluster head's battery is being depleted faster that of any of the member nodes, because the cluster head receives data from the member nodes, aggregates and compresses them, and transmits the compressed data to the sink. The energy consumption of a cluster head is calculated by

$$E_{\mathrm{CH}} = qE_{\text{elec}}\left(\frac{N}{k} - 1\right) + qE_{\text{DA}}\left(\frac{N}{k}\right) + q\left(E_{\text{Tx-elec}} + \varepsilon_{\text{mp}}d^4_{\text{to BS}}\right), \quad (5)$$

where $d_{\text{to BS}}$ is the distance between the cluster head and the base station, and (E_{DA}) is the energy dissipation per bit for data aggregation and compression.

The energy consumption by a member node for transmitting a q bit message to the cluster head is defined as

$$E_{\text{Non-CH}} = q\left(E_{\text{Tx-elec}} + q\in_{\text{fs}} d_{\text{CH}}^2\right), \tag{6}$$

where d_{CH}^2 is the distance between the member nodes and the cluster head.

Now, let us calculate the energy consumption in a cluster in the aforementioned sensor network, i.e., N sensors distributed uniformly in an $M \times M$ geographical unit square area that is divided into k clusters. First, we can say that each cluster in the area takes up approximately (M^2/k) of the geographical region. Second, the location of a sensor node can be described by a Cartesian coordinate $\rho(x, y)$ (Heinzelman et al., 2000). If the area is a circle, the sensor's location can be described by a polar coordinate $\rho(r, \theta)$, where r is the radius and θ is an angle, with the radius defined by $r = M/\sqrt{\pi k}$. Third, the expected square distance in a circular area between the cluster head and the member sensor nodes is calculated by

$$E\left[d_{\text{to CH}}^2\right] = \rho \int_{\theta=0}^{2\pi} \int_{r=0}^{M} /\sqrt{\pi k} r^3 dr d\theta = \frac{\rho M^4}{2\pi k^2}, \tag{7}$$

where due to the uniform region of a node,

$$\rho = \frac{1}{(M^2/k)}, \quad E\left[d_{\text{to CH}}^2\right] = \frac{M^2}{2\pi k}, \tag{8}$$

$$E_{\text{Non-CH}} = q E_{\text{Tx-elec}} + \frac{q\varepsilon_{\text{fs}} M^2}{2\pi p}. \tag{9}$$

Fourth, the total energy consumption for a cluster is the sum of that for the cluster head and for the non-cluster head member nodes:

$$E_{\text{total}} = E_{\text{CH}} + E_{\text{Non-CH}}, \tag{10}$$

$$E_{\text{total}} = q\left(E_{\text{elec}}(N/k - 1) + E_{\text{DA}}(N/k) + 2E_{\text{Tx-elec}} + \varepsilon_{\text{mp}} d_{\text{to BS}}^4 + q\varepsilon_{\text{fs}} M^2/2\pi k\right). \tag{11}$$

Finally, the optimal number of clusters, k, can be determined by setting the derivative of E_{total} with respect k to zero, resulting in

$$k = \frac{\sqrt{N}\sqrt{\epsilon_{\text{fs}}} M}{\sqrt{2\pi}\sqrt{\epsilon_{\text{mp}}} d_{\text{to BS}}^2} \tag{12}$$

$$\epsilon_{\text{fs}} = 10\,\text{pJ bit}^{-1}\,\text{m}^{-2}, \quad \epsilon_{\text{mp}} = 0.0013\,\text{pJ bit}^{-1}\,\text{m}^{-4}.$$

Based on Eq. (12), let us assume that the number of sensor nodes (N) and the network region (M) are constant, but the base station distance (d) increases; subsequently, the optimal number of clusters (k) decreases. Ultimately, some clusters have many sensor nodes when the number of clusters decreases due to k is the inverse squared distance. As Haibo et al. (2010) described, a cluster head with many sensor nodes consumes more energy than a cluster head with a few sensor nodes, because it aggregates, receives, and compresses more sensing information than a cluster head with few sensor nodes. In addition, if there is a large distance between a cluster head and the base station, the cluster head node consumes more energy than it would if the distance were shorter. If the current cluster head runs out of energy, the entire wireless sensor network is no longer operational. The main challenge is to minimize the power consumption of the cluster head, especially when many sensor nodes are allocated to a single cluster.

2.2 k-means^{++} algorithm

The k-means^{++} algorithm is used to assign the initial center of the k-means algorithm. Since the k-means algorithm randomly chooses the initial centroid, it is not guaranteed that clustering by the k-means algorithm is optimal. For example, if the initial random centroid is far away from the cluster's true center, the number of iterations required to optimize the centroid takes longer, and an incorrect clustering result may be obtained (Arthur et al., 2007; Avros et al., 2012). To remedy these problems, the k-means^{++} algorithm randomly selects the initial center from the sensor nodes' locations, but their location depends on their squared distances from the closest center that already has been selected.

For example, the first single initial center (c_1) is selected randomly; however, the remaining centers, such as those in the range from (c_2) to (c_l), are calculated based on the steps described below.

First, let us assume that the sensor nodes are represented by $X = (x_1, \ldots, x_n)$ and that l centers are represented as C, where $C = (c_1, \ldots, c_l)$. The distance between each sensor node and (c_1) is calculated by

$$D_1 = \|x_1 - c_1\|^2, \quad D_2 = \|x_2 - c_1\|^2 \tag{13}$$

and $D_n = \|x_n - c_1\|^2$.

The distance of each sensor nodes and over the average distance is calculated by

$$p(x_1) = \frac{D_1^2}{D_1^2}, \quad p(x_2) = \frac{D_2^2}{D_1^2 + D_2^2}, \tag{14}$$

$$p(x_n) = \frac{D_n^2}{D_1^2 + D_2^2 + \ldots + D_n^2}.$$

Second, the algorithm generates a random number. Then, one of the values of $p(x_1), p(x_2), \ldots, p(x_n)$ close to a random number (i.e., x_i) becomes the second center. For example, for the random number of $R \approx p(x_4)$, the sensor node x_4 becomes (c_2); otherwise, the algorithm generates another

value. The third step is to choose the third center (c_3). The distance is calculated as

$$D_1^2 = \min(\|x_1 - c_1\|^2 \|x_1 - c_2\|^2),$$
$$D_2^2 = \min(\|x_2 - c_1\|^2 \|x_2 - c_2\|^2), \quad (15)$$
$$D_3^2 = \min(\|x_3 - c_1\|^2 \|x_3 - c_2\|^2).$$

The distance of each sensor node and over the average distance of sensor nodes is also calculated as

$$p(x_1) = \frac{D_1^2}{D_1^2}, \quad p(x_2) = \frac{D_2^2}{D_1^2 + D_2^2}, \quad (16)$$
$$p(x_n) = \frac{D_n^2}{D_1^2 + D_2^2 + \ldots + D_n^2}.$$

Again, the algorithm generates a random number to choose one of the values of $p(x_1), p(x_2), \ldots, p(x_n)$. The process of selecting the initial centers using the above steps continues until l centers are selected.

Moreover, Arthur et al. (2007) chose the initialization center of a data set one by one in a controlled fashion using the k-means^{++} algorithm. For example, the first initial center was selected randomly in a sensing region, but the subsequent centers depended on the value of the previous center. For example, c_2 depends on c_1, and c_3 depends on c_2 and c_1. If we expand the illustrative example, the k-means^{++} algorithm can be conveniently generalized for any number of nodes and clusters.

The first step is to choose the first single initial center (c_1) randomly. The second step is to compute the distance between all sensor nodes and (c_1) and choose c_2 by the following:

$$D_i = \|x_i - c_1\|^2, \quad (17)$$

$$p(x_1) = \frac{D_1^2}{D_1^2}, \quad p(x_2) = \frac{D_2^2}{D_1^2 + D_2^2}, \quad (18)$$
$$p(x_n) = \frac{D_n^2}{D_1^2 + D_2^2 + \ldots + D_n^2}.$$

The algorithm generates a random number. Then, one of the values of $p(x_1), p(x_2), \ldots, p(x_n)$ close to the random number, x_i, becomes a second center, c_2. Third, recompute the distance vector to choose the third center as

$$D_i^2 = \min(\|x_i - c_1\|^2 \|x_i - c_2\|^2, \ldots, \|x_i - c_l\|^2). \quad (19)$$

Calculate $p(x_1), \ldots, p(x_n)$ as Eq. (16) and generate a random number close to $p(x_1), \ldots, p(x_n)$ to choose the third center (c_3). The difference between Eqs. (17) and (19) is that Eq. (17) is used to calculate the distance between the initial center (c_1) and the sensor nodes, whereas Eq. (19) is used to calculate the distance based on (c_1) and (c_2). In general, all remaining centers, such as c_l, are calculated as

$$D_n^2 = \min(\|x_i - c_1\|^2, \ldots, \|x_i - c_l\|^2), \quad (20)$$

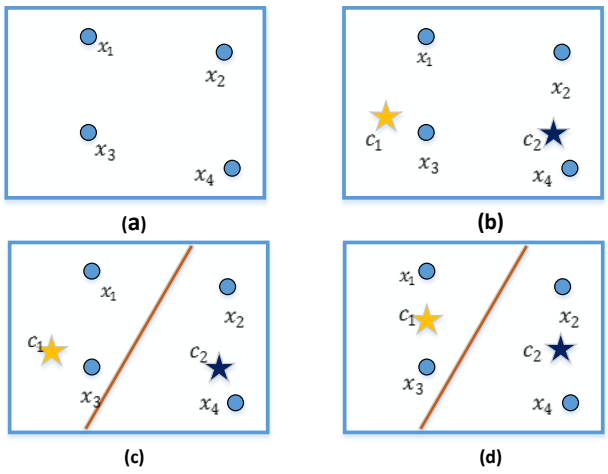

Figure 2. k-means algorithm diagram: **(a)** location of the sensor nodes; **(b)** initial centers; **(c)** new center after multiple iterations; **(d)** optimal centers.

$$p(x_1) = \frac{D_1^2}{D_1^2}, \quad p(x_2) = \frac{D_2^2}{D_1^2 D_2^2}, \ldots, \quad (21)$$
$$p(x_n) = \frac{D_n^2}{D_1^2 + D_2^2 + \ldots + D_n^2}.$$

2.3 k-means algorithm

The k-means algorithm is a method of grouping or classifying sensor nodes into k numbers of groups/clusters (Zhong et al., 2012). This technique selects an optimal center location of a cluster from which the sum of the squared distances to the locations of the sensor nodes is minimized.

Figure 2 illustrates how the k-mean algorithm is used to select an optimal center. First, sensor nodes are represented as (x_1, x_2, x_3, x_4) in Fig. 2a, and let us randomly choose two centers, called c_1 and c_2 (Fig. 2b). Next, calculate the distance between each sensor node to the two centers, $\|x_1 - c_1\|^2, \ldots, \|x_4 - c_1\|^2$ and $\|x_1 - c_2\|^2, \ldots, \|x_4 - c_2\|^2$. Third, group sensor nodes are based on sensor nodes' minimum distance to the centers. For example, if x_1 and x_2 are closest to c_1, then x_1 and x_2 will be in the same group. Similarly, if x_3 and x_4 are closest to c_2, then x_3 and x_4 will be in the same group. In addition, Fig. 2c shows that sensor nodes are grouped based on the closest distance to the centers. Four, calculate a new center for sensor nodes, which are in the same group. For example, $c_{1_{\text{new}}} = \frac{1}{2}\{(x_1 - c_1)^2 + (x_2 - c_1)^2\}$ and $c_{2_{\text{new}}} = \frac{1}{2}\{(x_1 - c_2)^2 + (x_2 - c_2)^2\}$. Last, we continue to calculate the center based on the previous equation until the new center is the same as the previous center location. When the previous and the new center location are the same, the centers are optimal, shown in Fig. 2d.

If we expand the illustrative example, the k-means algorithm can be generalized conveniently for any number of nodes and clusters. In general, the locations of n sensor nodes are represented by X, where $X = (x_1, \ldots, x_n)$, and l centers are represented by C, where $C = (c_1, \ldots, c_l)$. The k-means objective function, which minimizes the distance between sensor node (x_i) and the cluster center (c_j), is defined as

$$\mathrm{KM}(X, C) = \sum_{i=1}^{n} \left\| x_i - c_j \right\|^2 \tag{22}$$

$i = 1, \ldots, n$ and $j = 1, \ldots, l$,

where

$$c_j = \frac{1}{u_j} \sum_{x_i \epsilon u_j} x_i. \tag{23}$$

The cluster center c_j represents the current estimation of the location of the center of cluster j, and u_j is the number of sensor nodes in cluster j.

2.4 Gap statistics

"Gap statistics" is a standard technique for determining the optimal number of clusters for a data set (or a group of sensor nodes) by comparing the observed weight curve to the expectation of a referenced weight curve (Tibshirani et al., 2001).

The observed weight is the sum of the distance between all observed sensor nodes (actual data) and the center of the cluster; the referenced weight is the sum of the distance between all referenced sensor nodes (ideal) and the center of the cluster (Yan, 2005; Zhang, 2001). The observed weight and the expectation of the referenced weight can be derived mathematically as shown below.

First, let us assume that the sensor nodes are represented by $X = (x_1, \ldots, x_n)$. Also, if there are sensor nodes in a cluster, the distance between each of them is defined by

$$\begin{aligned} D_k &= \sum_{ii}^{'} d'_{ii'} \qquad i = (1, \ldots, n) \\ &= \sum_{i=1}^{n} \sum_{i'=1}^{n} \|x_i - x_{i'}\|^2, \\ &= (x_1 - x_1)^2 + (x_1 - x_2)^2 + (x_1 - x_3)^2 + (x_2 - x_1)^2 \\ &\quad + (x_2 - x_2)^2 + \ldots + (x_n - x_n)^2 \end{aligned} \tag{24}$$

$x_1 - x_1 = 0$, $x_2 - x_2 = 0, \ldots, x_n - x_n = 0$.

Therefore, $D_k = 2n_k \sum_{i=1}^{n} \|x_i - \overline{x}\|^2$,

where $\overline{x} = \frac{x_{1'} + x_{2'} + \ldots + x_{n'}}{n}$, and $\overline{x}$ is the center of the cluster, n is the number of sensor nodes, and $d_{i,i'}$ is the distance between two nodes (i and i'), k is the number of clusters,

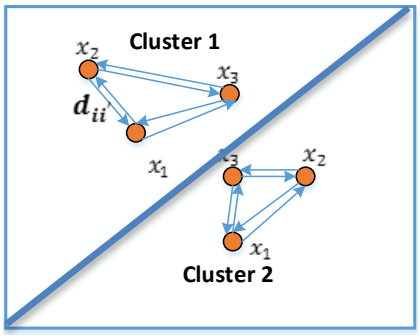

Figure 3. Sensor nodes in a cluster.

($k = 1, \ldots, g$), and g is the maximum number of clusters. The weight in the k cluster is defined by

$$W_k = \sum_{k=1}^{g} \frac{1}{2n_k} D_k. \tag{25}$$

Figure 3 also illustrates the distance between each of the sensor nodes, the number of clusters, and the number of sensor nodes in a cluster. For example, $k = 2$, $n_1 = 3$, and $n_2 = 3$, D_1 is the total distance between sensor nodes to the center at cluster 1, and D_2 is the total distance between sensor nodes to the center at cluster 2.

Second, the algorithm generates the referenced weight by adding a small noise into the original sensor nodes or the observed sensor nodes. The referenced weight is W_k, and the referenced weight dispersion is W^*_{kb}; k is the number of clusters, $k = 1, \ldots, g$, and b refers to the reference data sets, $b = (1, 2, \ldots, B)$, where B is the maximum number of data sets. For example, when $k = 3$ and $b = 5$, the algorithm generates five different locations for sensor nodes which are distributed across three clusters.

Third, the algorithm calculates the expected value of the referenced weight, $E^*_n(W_{kb})$, and n is the number of sensor nodes. In order to analyze the difference between observed weight and the expected value of referenced weight, the algorithm uses the logarithmic scale graph since it shows a visual differentiation between observed and referenced weight. Therefore, the observed weight is represented as $\log(W_k)$, and the expected referenced weight is represented as $E^*_n(\log(W_{kb}))$.

As expressed above, the main goal of the gap statistics method is to compare the curve of the observed weight $(\log(W_k))$ to the curve that represents the expectation of a referenced weight $(E^*_n\{\log(W_{kb})\})$ to determine the optimal number of clusters based on the maximum gap between the two curves. As Yan (2005) and Zhang (2001) describe, the number of optimal clusters can be found when $(\log(W_k))$ falls the farthest below the expected referenced weight dispersion curve.

However, when there is a small gap between the $\log(W_k)$ curve and the expected referenced weight curve

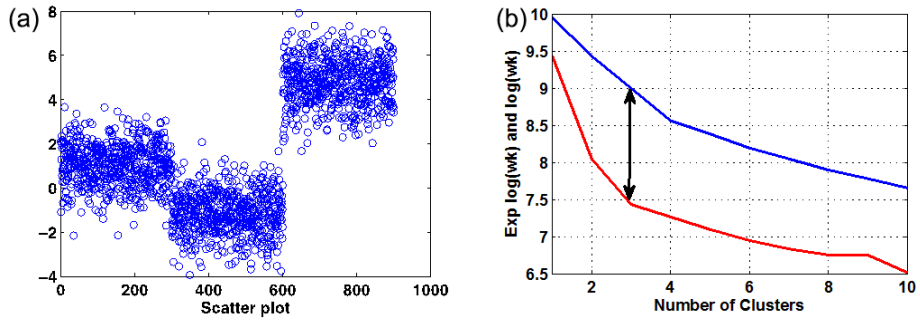

Figure 4. Results of the example with three clusters: **(a)** sensor nodes; **(b)** weight dispersion, W_k, as a function of k number of clusters.

$(E_n^*\{\log(W_{kb})\})$, the cluster is not optimal because the observed sensor nodes have noise that is the same as that of the referenced weight sensor nodes. Conversely, when there is a maximum gap between the $\log(W_k)$ curves and the expected referenced weight curve $(E_n^*\{\log(W_{kb})\})$, the cluster is optimal. In other words, the observed sensor nodes have very small noise at the maximum gap compared to that of the referenced sensor nodes, which are generated with noise. In this discussion, the term "noise" indicates that the sensor nodes are not close to each other and that they do not form the optimal number of clusters.

For example, Fig. 4a shows a scatter graph in which the sensor nodes are distributed across three clusters; one cluster is well separated from the other two clusters, which are connected. Figure 4b shows that using the gap statistics algorithm determines the optimal number of clusters in Fig. 4a. As Fig. 4b shows, the increased number of clusters results in decreased weight. The red line indicates the location of the original sensor nodes within the cluster and has observed weight $(\log(W_k))$; the graph shows a rapid decrease up to cluster number 2, and, then, it decreases slowly from cluster numbers 3–10. In addition, the blue line is the referenced weight, $(E_n^*\{\log(W_{kb})\})$. The optimal number of clusters is determined to be three, because, at that point, the gap between the two lines is at its maximum.

3 Combination of the clustering algorithms

As summarized above, the LEACH (Heinzelman et al., 2000) algorithm uses a computation and communication energy model to increase the lifespan of the sensor nodes. But the method is still far from being a complete and optimal solution to the problem. For example, the LEACH algorithm selects a fixed number of clusters, but it ignores the fact that some of the sensor nodes in a cluster can be reallocated to another cluster. It also ignores the fact that the cluster head's energy will be depleted quickly when too many sensor nodes remain in a single cluster, because more energy is required for aggregating, compressing, and transmitting more information. With this background of partial solutions to the problem, our intention was to attain a complete solution by using other

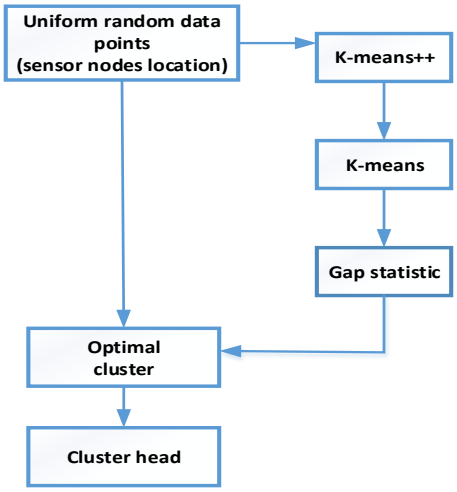

Figure 5. Combination of the three clustering algorithms.

clustering algorithms that were developed for other purposes. This section provides details concerning how they were used. The operation of wireless sensor nodes is divided into three phases, i.e., setup, advertisement, and steady state. In this research, we focused only on the setup phase. During the setup phase, first, the sensor nodes identify their locations and positions and then transmit the information to a base station. At the base station, where this combined algorithm is located and runs, the k-means^{++} algorithm generates the initial center for the sensor nodes' location. Second, the k-means algorithm chooses the optimal centers of the clusters. Finally, the gap statistics algorithm is used to select the optimal number of clusters for the nodes.

Figure 5 shows the steps that are used to choose the optimal number of clusters based on the three clustering algorithms (k-means^{++}, k-means, and gap statistics).

In the first step, we represent the location of the sensor node. In the second step, we initialize the cluster's center based on the k-means^{++} algorithm. In the third step, we choose the optimal center for the cluster based on the k-means algorithm. In the fourth step, we used the gap statistic algorithm to calculate the optimal number of clusters.

The first step starts with a number of sensor nodes represented by $X = (x_1, \ldots, x_n)$ In the second step, we calculate the initial centers for the sensor nodes based on Eqs. (17)–(19).

Third, we calculate the optimal centers of the distributed sensor network based on the k-means algorithm using Eqs. (22) and (23). The k number of clusters is defined as $(k = 1, \ldots, g)$. Using Eqs. (24)–(25), the sum of the clusters' weight (W_k) is calculated, and the mean of a reference weight (W^*_{kb}) is generated. b refers to the reference data sets, $b = (1, 2, \ldots, B)$, where B is the maximum number of data sets. In order to analyze the difference between the observed weight and the expected value of the referenced weight, the algorithm uses a logarithmic scale graph since it shows a visual differentiation between the observed weight and the referenced weight. Therefore, the observed weight is represented as log (W_k), and the expected referenced weight is represented as $E^*_n(\log(W_{kb}))$. The gap statistics is defined by

$$\mathrm{Gap}_n(k) = E^*_n\{\log(W_{kb})\} - \log(W_k). \quad (26)$$

As expressed above, the main goal of the gap statistics method is to compare the curve of the observed weight $(\log(W_k))$ to the curve that represents the expected reference weight $(E^*_n\{\log(W_{kb})\})$ to determine the optimal number of clusters based on the maximum gap between the two curves.

$$\max(\mathrm{Gap}_n(k)) \approx \hat{k}_{\mathrm{opt}} \quad (27)$$

4 Simulation and discussion

4.1 Test sensor network and scope of simulation

The test sensor network is of the sensor nodes randomly distributed between $u(0, 0)$ and $u(100\,\mathrm{m}, 100\,\mathrm{m})$ as illustrated in Fig. 6, with their location expressed as $X = [(x_{ij})]$, where $(i = 1, 2, \ldots, n)$ and $(j = 1, 2, \ldots, k)$. In addition, the base station (sink) is assumed to be at (50 m, 175 m).

For the simulation of the test sensor network, we used the LEACH algorithm's simulation parameters, as indicated in Table 1. For example, the initial energy for each of the sensor nodes was set to 0.5 J. Each of the data messages were 525 bytes long, and the broadcast packet size header was 25 bytes long.

The radio electronics energy was $50\,\mathrm{nJ\,bit^{-1}}$, and the radio transmitter energy was set to $10\,\mathrm{pJ\,bit^{-1}\,m^{-2}}$ or $0.0013\,\mathrm{pJ\,bit^{-1}\,m^{4}}$. The cluster head collects data from the sensor nodes and aggregates those data prior to sending them to the base station. The energy used to aggregate the data (E_{DA}) was $5\,\mathrm{nJ\,bit^{-1}\,signal^{-1}}$.

4.2 Code structures for the clustering algorithms

The simulation steps of the three combined algorithms are described in Table 2. First, the k-means^{++} algorithm simulates choosing the initial center of the sensor nodes; second, the k-means algorithm simulates the calculation of the optimal center. Third, the gap statistics algorithm simulates the calculation of the optimal number of clusters.

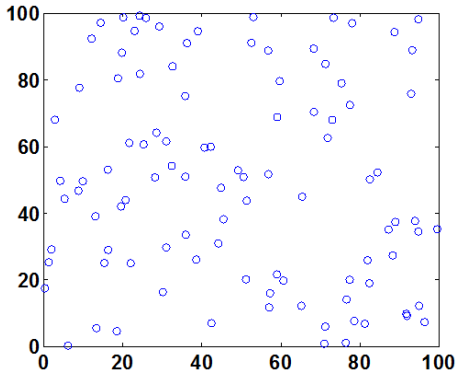

Figure 6. 100 wireless sensor nodes in the area of the sensing network.

Table 1. Simulation parameters.

Parameter	Value
Network field	From (0,0) to (100,100)
Number of nodes	100
Base station	At (50,175)
Initial energy	0.5 J
Data packet size	525 bytes
Broadcast packet size	25 bytes
E_{elec}	$50\,\mathrm{nJ\,bit^{-1}}$
ϵ_{fs}	$10\,\mathrm{pJ\,bit^{-1}\,m^{2}}$
ϵ_{emp}	$0.0013\,\mathrm{pJ\,bit^{-1}\,m^{-4}}$
E_{DA}	$5\,\mathrm{nJ\,bit^{-1}\,signal^{-1}}$
Threshold distance (d_o)	75 m

4.3 Simulation

Step 1: determination of the optimal center of the cluster using the k-means algorithm

The k-means^{++} algorithm and the k-means algorithm were used to generate the optimal location of the center of the sensor nodes. For example, in Fig. 7, the optimal center is marked by "X", and the sensor nodes are marked by gray, blue, green, cyan, dark blue, black, and Red.

Step 2: determination of the optimal number of clusters using gap statistics

After the optimal location of the center of the sensor nodes was calculated, the gap statistics algorithm determined the optimal number of sensor nodes by comparing the observed weight curve $(\log(W_k))$ to the expected reference weight curve $E^*_n\{\log(W_{kb})\}$.

Table 2. Combination of clustering algorithms.

Algorithm 1: k-means^{++}
Require: generate a uniform random number sensor nodes' location 1: $c_1 \leftarrow$ select a single center from uniformly distributed sensor node location X 2: while $c_i < k\, d_o$ $>$ k is the number of cluster 3: sample $x \in X$ with probability $\frac{D_i^2}{\sum_{i=1} D_i^2}$ 4: $c_i \leftarrow c_i \cup \{x\}$ end while $>$ select a new center 5: end while
Algorithm 2: k-means
6: use Initial center from k-means^{++} $C \subset X$ $>$ $C = c_1 \ldots c_l$ 7: repeat 8: for all $x \in X$ find KM(X, C) (closet center $c \in C$ to x) 9: for all $i \in k$ let $c_j =$ average $\{x \in X \vert \mathrm{KM}(X, C) = c_j\}$ $>$ $j = 1, \ldots, l$ 10: until The set C is unchanged
Algorithm 3: gap statistics
Require: cluster the observed data, with the number of clusters fixed at $k = 1, 2, \ldots, g$ 11: for $k = 1 \rightarrow g\, d_o$ 12: $D_k \leftarrow \sum_{i,i'} d_{ii'}$ 13: $W_k \leftarrow \sum_{k=1}^{g} \frac{1}{2n_k} D_k$ $>$ total distance within clusters 14: end for
Require: generate reference data W_{kb}^*, $>$ $b = 1, 2 \ldots B$, $k = 1, 2, \ldots, g$ 15: for $k = 1 \leftarrow g\, d_o$ 16: for $b = 1 \rightarrow B\, d_o$ 17: $D_k \leftarrow \sum_{i,i'} d_{ii'}$ 18: $W_{kb}^* \leftarrow \sum_{k=1}^{g} \frac{1}{2n_k} D_k$ 19: end for 20: end for 21: $\mathrm{Gap}_n(k) = E_n^* \{\log(W_{kb})\} - \log(W_k)$ 22: $\max(\mathrm{Gap}_n(k)) \approx \hat{k}_{\mathrm{opt}}$

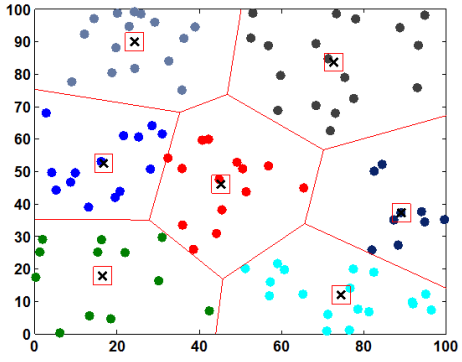

Figure 7. Sensor nodes grouped in seven clusters.

Figure 8 shows the observed and reference weight functions versus the number of clusters. In addition, the red dots on the red curve are the observed weight curve ($\log(W_k)$). The blue curve is the reference weight curve $E_n^* \{\log(W_{kb})\}$ for different numbers of clusters, which was used to calculate the gap statistics. The optimal number of clusters was estimated to be seven because the maximum gap between the reference (blue) and the observed (red) curves reached its maximum at the seven-cluster point.

Step 3: Comparison of the LEACH algorithm and the combination of clustering algorithms

We compare our approaches with the LEACH algorithm's approaches to determine which method provided a longer lifespan for the wireless sensor nodes.

As discussed in Sect. 2, the LEACH algorithm determines the optimal number of clusters, k, in a group of distributed homogeneous wireless sensors based on the "computation and communication energy model".

To assess the two methods, we used the LEACH algorithm to choose a cluster head within sensor nodes in a cluster. For example, sensor nodes randomly chosen from 0 to 1. When

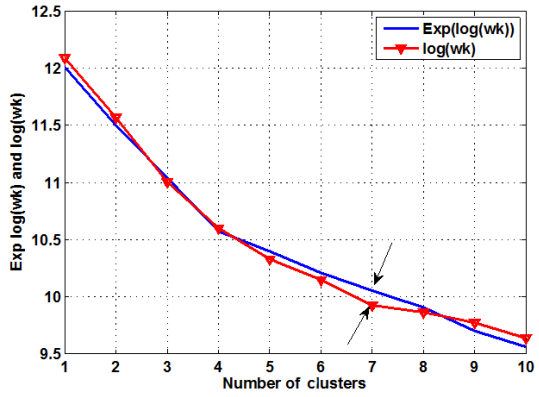

Figure 8. log(mean) dispersion of reference and log dispersion original data sets.

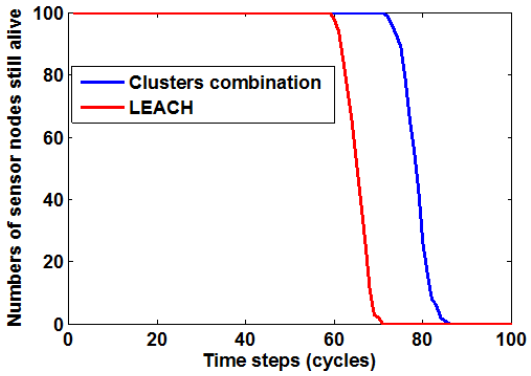

Figure 9. Lifespans of homogenous wireless sensor nodes: (red) LEACH algorithm; (blue) combination of clustering algorithms.

the randomly chosen value is less than the $T(n)$, the sensor node becomes a cluster head; otherwise, a different sensor node chooses another random number to become a cluster head.

The value of $T(n)$ is calculated based on the probability of a sensor node becoming a cluster head and the number of rounds. For example, if there are 20 sensor nodes in a cluster, the probability of becoming a cluster head for each sensor node is $p = 1/20 = 0.05$. After the first cluster head is chosen, the probability of 1 of the remaining 19 sensor nodes becoming a cluster head in the next round is $1/19$. Thus, the number of rounds required for every sensor node to become a cluster head is $r = 1/p$.

$$T(n) = \begin{cases} \dfrac{p}{1 - p \cdot (r \cdot \bmod 1/p)} & \text{if } n \epsilon G \\ 0 & \text{otherwise} \end{cases}, \tag{28}$$

where r is the number of rounds remaining, G is a group of sensor nodes that have not yet become cluster heads in the previous rounds, p is the expected probability to become a cluster head, and n is a sensor node.

The operation of the LEACH algorithm depends on the rounds. Each round has two phases, i.e., a setup phase and a steady-state phase. During the setup phase, the number of clusters and the cluster head are selected. In the steady-state phase, data are transferred from the sensor nodes to cluster head, which sends them to the base station.

4.4 Comparison of performance

Figure 9 shows the number of sensors still alive over time and shows the advantage of using the combination of the clustering algorithms (blue curve) over the LEACH algorithm (red curve). The energy of the sensor nodes begins to diminish at $t = 62$ cycles using the LEACH algorithm, while it begins to diminish at $t = 75$ cycles using the combination of clustering algorithms. In the LEACH algorithm, all of sensor nodes became inactive at $t = 73$ cycles, whereas they lasted up to 87 cycles in the combination of clustering algorithms. Overall, the combination of clustering algorithms provided 15 % greater lifespan for the sensor nodes than the LEACH algorithm.

5 Conclusions

To improve the lifespan of sensor networks, we proposed using a combination of clustering algorithms, i.e., the k-means algorithm, the k-means^{++} algorithm, and gap statistics, and we compared that approach with the use of the popular LEACH algorithm. In applying the clustering algorithms, the k-means algorithm was used to classify or group sensor nodes into k clusters based on their locations. Also, the k-means^{++} algorithm obtained more appropriate initial center locations for the k-means algorithm, which allowed the optimization of the cluster's center, and gap statistics was used to select the optimal number of clusters for a wireless sensor network.

Our simulation demonstrated the advantage of using the combination of clustering algorithms over using the LEACH algorithm in that the lifespan of the wireless sensor nodes was increased by 15 %.

Edited by: R. Morello
Reviewed by: two anonymous referees

References

Arthur, D. and Vassilvitskii, S.: k-means^{++}: the advantage of careful seeding, 18th Symposium on Discrete Algorithms, New Orleans, Louisiana, 7–9 January 2007, 1027–1035, 2007.

Avros, R., Granichin, O., Shalymov, D. Volkovich, Z., and Weber, G.: Randomized Algorithm of Finding the True Number of Clusters Based on Chebychev Polynomial Approximation, in: Data Mining: Foundation and Intelligent Paradigms, Springer, 23, doi:10.1007/978-3-642-23166-7, 2012.

Haibo, Z., Wu, Y., Hu, Y., and Xie, G.: A novel stable selection and reliable transmission protocol for clustered heterogeneous wireless sensor networks, Comput. Commun., 33, 1843–1849, 2010.

Heinzelman, W., Chandrakasan, A., and Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks, The 33rd Hawaii International Conference on System Science, Maui, Hawaii, 4–7 January 2000, p. 8020, doi:10.1109/HICSS.2000.926982, 2000.

Kumar, D., Aseri, T., and Patel, R. B.: EEHC: Energy efficient heterogeneous clustered scheme for wireless sensor networks, Comput. Commun., 32, 662–667, 2009.

Qing, L., Zhu, Q., and Wang, M.: DEEC: Design of a distributed energy-efficient clustering algorithm for Heterogeneous wireless sensor networks, Comput. Commun., 29, 2230–2237, 2006.

Tibshirani, R., Walther, G., and Haste, T.: Estimating the number of clusters in a data set via the gap statistic, J. Roy. Stat. Soc. B, 63, 411–423, 2001.

Yan, M.: Method of Determining the Number of Clusters in a Data Set and a New Clustering Criterion, PhD dissertation, Virginia Polytechnic Institute and State University, 23–73, 2005.

Zhang, B.: Generalized K-Harmonic Means – Dynamic weighting of Data in Unsupervised Learning, 1st SIAM international Conference on Data Mining (SDM'2001), Chicago, USA, 5–7 April, 1–13, 2001.

Zhong, S., Wang, G., Leng, X., Wang, X., Xue, L., and Gu, Y.: A Low Energy Consumption Clustering Routing Protocol Based on K-Means, Software Engineering and Applications, 5, 1013–1015, 2012.