Asynchronous Distributed Clustering Algorithm for Wireless Sensor Networks

Cheng Qiao
Insight Centre for Data Analytics
Department of Computer Science
University College Cork, Ireland
qiao.cheng@insight-centre.org

Kenneth N. Brown
Insight Centre for Data Analytics
Department of Computer Science
University College Cork, Ireland
k.brown@cs.ucc.ie

ABSTRACT

In distributed clustering problems, nodes in a wireless sensor network must learn clusters from the data sensed across the network, without centralising the raw data. This paper presents an asynchronous distributed clustering algorithm for sensors to learn the global clusters, while respecting data privacy, and balancing communication cost and clustering quality. Different clustering algorithms including k-means and Gaussian Mixture Models, and different methods of summarising clusters to exchange between nodes are considered. In experiments on randomly generated network topologies, we demonstrate that methods which do more extensive clustering in each cycle, and which exchange descriptions of cluster shape and density instead of just centroids and data counts, achieve more consistent clustering, in significantly shorter elapsed time.

CCS Concepts

•**Networks → Wireless local area networks;**

Keywords

Distributed algorithm, clustering, Wireless sensor network

1. INTRODUCTION

Wireless Sensor Networks (WSNs) are collections of typically low-powered sensors distributed in space, communicating data to a central computer over a multi-hop wireless network, with a wide range of applications [1, 2, 3]. As the Internet of Things and Cyber Physical Systems develop, these network nodes must also include actuators, which respond to sensed data or communicated instructions to effect changes to the environment. In some applications, the actions taken by a node will depend on wider patterns observed in the network - e.g. temperature readings in indoor heating systems, or vehicle movements in traffic control - and so communication must flow back and forwards across the network. Data gathering, inference methods and decision algorithms should be designed to take account of the underlying network as well as of the application context.

Transmitting all data to a gateway for processing on a central server, or in the cloud, and then back out to the individual nodes may cause congestion in the network, may incur significant latency in the decision making, and will consume battery power on the nodes. It also creates risks for the privacy and security of the data. Further, in some cases the raw data collected by individual sensors may be owned by different entities, and sharing that data with a central authority is disallowed. To address these concerns, decision making and inference capabilities are being pushed out from the centre into the network. At the extreme, each node acts as its own decision maker, but must communicate with other nodes to learn wider network patterns [4]. The methods trade off potentially lower decision quality for reduced communication, reduced energy use and improved privacy. For example, there are some sensors deployed in a area in figure 1, and each sensor has a different reading v_i $(1 \leq i \leq 9)$. Sensors could learn the average read $\bar{v}$ by in-networking learning and message exchanging without gathering all data to central server.

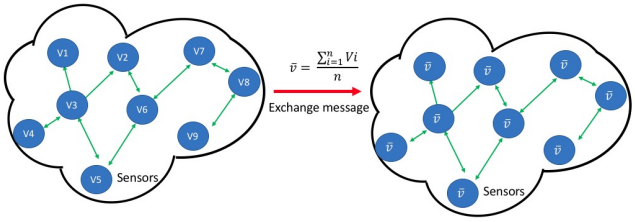

Figure 1: Problem Definition

In this paper, we consider in-network learning, and concentrate on the distributed clustering problem [5, 6, 7]. We attempt to learn the same global pattern across all nodes in a wireless mesh network, while minimising inference time and communication cost, and respecting the privacy of the raw data. We consider two different scenarios for the initial state of the network: (i) each node knows how many other nodes are in the network, and (ii) each agent is only aware of its direct communication neighbours. We develop an asynchronous mechanism for sharing information, to avoid synchronisation delays. We develop different approaches to in-network clustering, examining the trade-offs between sharing local clusters and conducting incremental clustering as information flows through the network. We design and implement different methods for describing the shapes of locally identified clusters, to improve the global clustering performance. We evaluate the methods empirically, using an asynchronous message delay simulator,

ICMLT 2019, June 21–23, 2019, Jiangxi, Nanchang, China

ACM. ISBN 978-1-4503-6323-51906. . . $15.00

DOI: https://doi.org/10.1145/3340997.3341007

and we show that the new approaches can improve clustering accuracy compared to existing methods (relative to a centralised solution) by up to 10%, while decreasing message costs by 15% and dropping convergence time by up to 85%, all without transmitting any raw data.

In the remainder of the paper, we set the context and survey related work in next section, we then define our methods, followed by the experimental set-up, and finally we present and discuss the results of the experiments.

2. RELATED WORK

Data aggregation is an essential operation in WSNs to prolong the lifetime of sensors [9, 10]. The topology of the network has a significant influence on the performance of different data aggregation techniques, and so different algorithms have been developed for different topologies, including tree-based [11], cluster-based [12], chain-based [13] and structure-free [14]. One common technique is to identify a subset of nodes as cluster heads, which gather data from neighbouring nodes [15, 16], aggregate the data, and then transmit summaries as needed to the central base station. Deciding which nodes should be cluster heads is a challenging task [17], and static methods in which the cluster heads do not change introduce single points of failure into the network.

For distributed data mining, in which more general patterns must be inferred, Distributed K-Means [6] was proposed for use in large-scale peer-to-peer wired networks. The k-means algorithm is distributed by inserting an exchange of messages between neighbouring nodes after each internal iteration of the basic K-means loop (the selection of centroids, and the assignment of individual data points to each centroid) on the node's local data. After assigning the data points, each node transmits a description of its centroids and the number of data points associated with them to its neighbours. Once a node has received an update from all of its neighbours, it proceeds to the next round. It assumes centroids are listed in a specified order, and computes a weighted average of the centroids, which it then uses as the initial centroids for the next iteration. The algorithm terminates after successive updates do not change the centroids beyond a specified threshold. In this scheme, no raw data is exchanged, thus satisfying the basic privacy requirements, and the centroid summaries reduces the amount of data being transmitted. However, the algorithm requires synchronisation, which in wireless networks may introduce significant delay, and may fail to converge if one node has initial data whose distribution is significantly different from the average. The exchange of just centroids and counts also discards information about the shape of the cluster. When clusters have significantly different shapes, this may again cause problems for convergence.

To address the cluster shape issue, for tree-based network topologies, Bendechache et al. [18] represent each cluster by its boundary points, and then exchange the boundary and the number of internal points. Messages flow up the tree to the root. When a node receives multiple descriptions, it computes a new description by merging any overlapping clusters into a single cluster, and computing the new boundary. The algorithm is initialised with a much higher k value than is expected, to allow this cluster merging to reduce the number of clusters. This approach solves the issue of arbitrary shapes, but introduces failure points at each aggregation, and requires a secondary communication from root to leaves to disseminate the final clusters. It also exposes some of the raw data to neighbouring nodes, in order to specify the cluster boundary.

Gossip-based aggregation methods for distributed K-means are proposed by Fatta et al. [7] and Bénézit et al [8]. Each node selects a small subset of its neighbours to communicate with at each round, and includes a damping factor in the weighted average to avoid oscillations. Although global synchronisation is avoided, local synchronisation is still required. In practice, the nature of the gossip algorithm means that many rounds are required before convergence. Since there is no global synchronisation, nodes are unaware of when other nodes have converged, and so another algorithm running in parallel is required to detect termination, or the algorithm is forced to terminate after a fixed number of rounds.

3. METHOD

First, we make some assumptions about the underlying network. The network

1. is connected – there is a multi-hop path between any pair of nodes;
2. uses point-to-point communication – i.e. each message transmission is sent to a single neighbouring node;
3. ensures reliable delivery of the messages, but this may require retransmissions on failure, and so there is a message-specific delay for any transmission;
4. begins clustering at a pre-scheduled time – all nodes start clustering at approximately the same time, with the same expected number of clusters;
5. supports a parallel process to detect termination of the algorithm.

Our Asynchronous Distributed Clustering (ADC) algorithm framework, which runs on each node in parallel, is shown in Algorithm 3. First, the node clusters its local dataset. It then summarises the result of the clustering, and transmits those summaries to neighbours. After receipt of cluster summaries from neighbours, the node processes those descriptions and repeats by running the local clustering algorithm again. Once a node detects that successive iterations do not change the clusters significantly, and that there are no incoming messages, it terminates. Note that this is an algorithm framework, and that there are many different choices to be made for a particular instantiation, on (e.g.) the choice of centroids, the local clustering, the selection of neighbours, the summary of the clusters, and the termination criterion, as discussed below.

Algorithm 1: Algorithm for scenario 1

```
1  Agents run [local clustering] to completion ;
2  Agents transmit [[summaries]] with agent ID to neighbours;
3  while not stabilised or messages received do
4      if Receive terminate message then
5          Update local knowledge and transmit terminate
           message to all neighbours;
6          Terminated;
7      else if Receive final summary then
8          (Compute global view);
9          Send converge command to neighbours;
10     else
11         Transmit the new summary received;
```

Initial centroids We consider the same pre-scheduled initial centroids for all nodes, random selection of the initial centroids, and

Algorithm 2: Algorithm for scenario 2

```
Agents run local clustering to completion ;
Transmit [representation] with model name to neighbors;
while not stabilised or messages received do
    if Received new message then
        M = (combined model);
        if M ≠ Null then
            Sending M to neighbours ;
    else
        turn to sleep but ready to work;
```

Algorithm 3: ADC

```
Choose initial centroids ;
while not stabilised or messages received do
    Run local clustering algorithm;
    Summarise the clustering result;
    Exchange cluster descriptions with neighbours;
    Incorporate received neighbours' cluster information;
```

selection of the initial centroids by each node independently based on the current state of their data.

Local clustering We consider three local clustering methods. First, we use a single internal iteration of k-means (centroid selection and association of points to centroids). Second, we use k-means run to completion on each round. Finally, we fit a Gaussian Mixture Model to the data. In some cases, we allow a node to omit local clustering until it has received messages containing summaries from all other nodes.

Cluster summaries We consider four different summaries: (i) centroid location and count of points in each cluster; (ii) centroid location, count and shape and density description, discussed below; (iii) independent Gaussians fitted to each cluster in turn; and (iv) the GMM if generated during clustering. To generate the shape and density description for (ii), we first apply Principal Component Analysis to each cluster, to generate the axes of the cluster shape, centred on the centroid. For each positive and negative axis, we generate the maximal data point, the 80th percentile point, and the 40th percentile point. We then adjust the 80th and 40th percentile points to midway between that point and next closest greater point. Taken together, this produces three bounding boxes (or hyperrectangles), containing 100%, 80% and 40% of the points respectively, oriented along the PCA axes, and thus approximates the shape and density of the points in the cluster. Figure 2 shows an example generation of the bounding boxes for a 2-dimensional data set.

Data exchange While a node is doing local clustering, more messages may arrive from neighbours. We considered two approaches: transmit the result of local clustering, and then read any stored messages and repeat; read stored messages and repeat clustering, until the inbox is empty, and then transmit the clustering results. In practice, transmitting before reading new messages produced better performance, so we only report those results. As well as transmitting their own summaries, we allow each node to send on summaries received from other nodes without modification if needed. Further, to each summary we attach a list of all nodes whose data was used to generate the clusters. This supports the delayed clustering above, where a node waits until it has received information from all known nodes. In some cases, we also allow a node to issue a request for reduced data summaries (summaries built from a specific subset of nodes). Finally, we consider case where messages are sent to all direct neighbours, and where messages are sent to a randomly selected subset of neighbours.

Incorporating neighbours' descriptions For incorporating the simplest cluster summaries, we compute the weighted average centroids. For cases where we receive more descriptive summaries, we generate new data points by sampling from those descriptions, and we sample in proportion to the cluster counts. The sampled data is then combined with the local data, and provides the input to next round of local clustering. For the PCA/bounding box description, we generate points uniformly at random in the 40% box, and then generate points uniformly in each sector to extend to the 80% box, and then repeat to fill the remainder of the 100% box (Fig 3, for the same case as Fig 2). For the Gaussian summaries, we simply sample from the Gaussian distribution.

We can use our framework to recreate the method of [6] (use 'no further improvement' as terminating condition), by all nodes starting with the same pre-scheduled initial centroids, running a single internal iteration of k-means, counting the data points assigned to each cluster, transmitting a list of pairs of centroid location and centroid count to all neighbours, waiting until all neighbours have transmitted, then computing a weighted average location for each centroid. Similarly, we can replicate the method of [7], by all nodes starting with the same pre-scheduled initial centroids, running a full iteration of k-means, counting the data points assigned to each cluster, transmitting a list of pairs of centroid location and centroid count to a single randomly selected neighbours, then computing a weighted average location for each centroid.

3.1 Agent knowledge: known number of nodes

We consider two different scenarios for the initial knowledge nodes have of the rest of the network. In this first scenario, we assume that each node in the network knows the total number of other nodes. In any uncontrolled process, we run the risk of creating a feedback problem, where the neighbours of a node send it summaries which already incorporate that node's own cluster descriptions. For example, A sends its summaries to B, which incorporates the data, generates new summaries, and sends them on to C; C updates its global view, and passes it on to A; if A then incorporates that model into it local data, it will effectively give double the weight to its own data. To avoid this problem, each agent can simply transmit its own summaries labelled with its own ID, and then simply relay other received previously unseen summaries without re-clustering. Once it has received the correct number of summaries, it can combine them all to get a global picture. At that point, it can transmit the global summary, with a flag to indicate that no new information will be received. Each agent receiving this termination message stops its own process, adopts the global summary, and retransmits the message. We call this algorithm variant Model Merge after Filtering 1 (MMF1), shown in Algorithm 4. We note that there is some privacy loss here, since at least one agent will see the individual summaries from each other node.

3.2 Agent knowledge: direct neighbourhood only

In this scenario, we assume that each node knows only its immediate neighbours, and does not know the identity of other nodes, or even the size of the network. The same feedback problem as in the previous scenario still applies. In this case, we cannot wait until all summaries are received, since no node knows how many summaries are expected. Instead, we exploit the IDs that we can attach to each model. When a node receives a summary constructed only

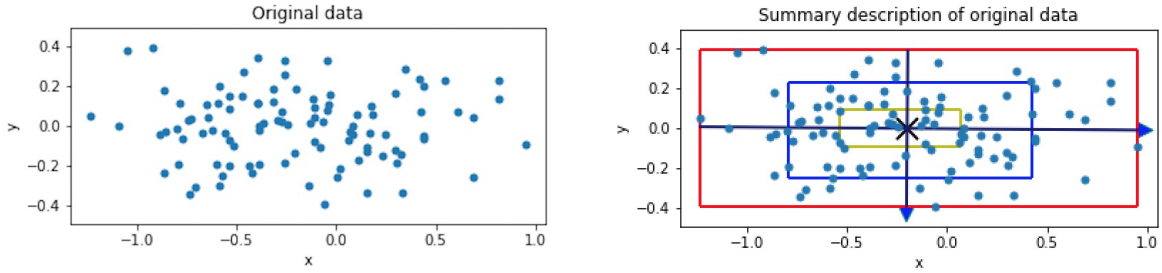

Figure 2: Original data and its summary description

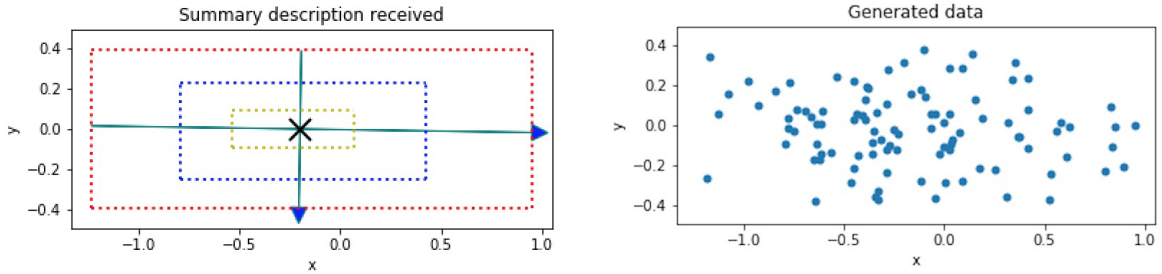

Figure 3: Bounding boxes received and regenerated data based on it

Algorithm 4: MMF1 for node N_i in n node network

Require: Local dataset X_i;
Input : Message box $\boldsymbol{B}$
Output: Final representatives of clusters
Randomly generate k initial centroids in the space;
Clustering X_i with specific centroids as input;
Create empty table $\boldsymbol{T}$;
while *not terminated* **do**
  if *Message box* $\boldsymbol{B}$ *is not empty* **then**
    if *Receive terminate message* **then**
      Send terminate message to all neighbours;
      Terminated;
    else if *Receive data message x and* $x \notin \boldsymbol{T}$ **then**
      Add x to $\boldsymbol{T}$;
      if *len(T) == n* **then**
        Compute global view;
        Send converge command to neighbours;
      else
        Share message x with neighbours;

from IDs it has not seen before, it applies the basic summary incorporation approach. If it receives a summary constructed entirely from IDs it has already seen, it ignores it. If it receives a summary built from some old and some new IDs, it then applies a model subtraction procedure, to avoid the feedback problem. It generates sample data points for the parent model, then generates temporary data points for the model to be subtracted, and for each temporary data point, it removes the closest data point generated for the parent model. If it does not have an appropriate set of smaller models to subtract, it identifies a small missing set, and sends a request to one or more neighbours which transmitted a superset. Those neighbours either reply with the summaries, or issue their own requests in turn. The algorithm, MMF2, is described in Algorithm 5.

Algorithm 5: MMF2 for node N_i

Require: Local dataset X_i;
Input : Message box $\boldsymbol{B}$
Output: Final representatives of clusters
Randomly generate k initial centroids in the space;
Clustering X_i with specific centroids as input;
Sending representatives and model name M_i to neighbours;
while *not terminated* **do**
  if *Message box* $\boldsymbol{B}$ *is not empty* **then**
    if *Received message* **then**
      M = combined model (Algorithm 6);
      if $M \neq Null$ **then**
        Sending M to neighbours ;
  else
    turn to sleep but ready to work;

Algorithm 6 shows the procedure for model subtraction. First, the final model $\boldsymbol{f_{model}}$, which contains all unique message it saw, is computed and the largest subset $\boldsymbol{l_{sub}}$ of $\boldsymbol{f_{model}}$ is computed as well. For instance, sensor i received two messages: abc and $bcef$, then the final model $\boldsymbol{f_{model}}$ is $abcef$ and $\boldsymbol{l_{sub}}$ is $bcef$.

After that, $\boldsymbol{d_{set}}$ is defined as the difference between $\boldsymbol{l_{sub}}$ and $\boldsymbol{f_{model}}$. Then, the elements of $\boldsymbol{d_{set}}$ are searched in History table $\boldsymbol{T}$. If any item is still missing, send out a request to specific neighbours who have transmitted this message before. Finally, models for $\boldsymbol{f_{model}}$ are combined by substracting and regenerating and the regenerated dataset is clustered to get the new model.

4. EMPIRICAL EVALUATION

Algorithm 6: Model Subtraction

Input : History table $\boldsymbol{T}$, Message list $\boldsymbol{m_i}$ $(0 \leq i \leq N)$
Output: Final Model $\boldsymbol{f_{model}}$

H ← set of node IDs in $\boldsymbol{T}$;
$\boldsymbol{f_{model}}$ ← union of H & node IDs in $\boldsymbol{m}$;
Add $\boldsymbol{m}$ to $\boldsymbol{T}$;
if $S == H$ **then**
    return Null
else
    $\boldsymbol{l_{sub}}$ ← biggest subset from $\boldsymbol{T}$;
    $\boldsymbol{d_{set}}$ ← $\boldsymbol{f_{model}} \setminus \boldsymbol{l_{sub}}$;
    Find elements of $\boldsymbol{d_{set}}$ missing from $\boldsymbol{T}$;
    Send request to neighbours to get those models;
Combine models for $\boldsymbol{f_{model}}$ by substracting and regenerating;
Clustering regenerated dataset;
Summarise the clusters;
return new model of clusters;

We evaluate a number of different instantiations of the ADC framework. MMFi-P uses full k-means, exchanging PCA and Bounding Box summaries, regenerating data by sampling, for the two MMF variants. Similarly, MMFi-SG uses full k-means, but then exchanges separate Gaussian models with counts fitted to each cluster. MMFi-GMM fits a Gaussian Mixture Model for clustering, and exchanges the GMSS with counts. We also evaluate [6] partial k-means, exchanging centroids and counts with all neighbours, and [7, 8] (full k-means, exchanging weighted centroids and counts with a single neighbour). To simulate the operation of the algorithms on a wireless mesh network, we implement an Asynchronous Message Delay Simulator, based on [19]. Random network topologies are generated based on [20], where the probability p_N is larger than $\frac{(1+\varepsilon)ln(N)}{N}$, where ε is a positive constant, to ensure that a random graph generated is a connected graph (we set $\varepsilon = 1$). To generate dense graph topologies, p_N is set to 0.8. The initial raw data is generated in k clusters, sampled from a two-dimensional normal distribution, scaled to the range [0,1] before being assigned randomly to network nodes. Each node is given 200 data points and the number of clusters (k) in all simulations is set to 5. Each message transmission delay is generated uniformly from the range [0.5, 1.0], and nodes begin their work at time randomly selected in [0, 0.1s].

Before we distribute the data points over the nodes, we run a single centralised k-means, to get a benchmark clustering result. We use the precentage of membership mismatch (PMM) as our formal measure of clustering error.

$$PMM^{(i)} = 100\frac{|\vec{x} \in X^{(i)} : L_c(\vec{x}) \neq L_p^i(\vec{x})|}{|X^{(i)}|} \quad (1)$$

where $L_c(\vec{x})$ denotes the label of the cluster to which $\vec{x}$ is assigned at the end of centralized K-means and $L_p^i(\vec{x})$ denotes the label of the cluster to which $\vec{x}$ is assigned once the node reaches the termination state.

Since GMM is a soft clustering method, which assigns to a single data point a probability of membership of each cluster, PMM may be unfair to GMM, since it can never achieve 100% clustering accuracy. In order to compare with the hard assignment obtained by k-means, we assume that the final assignment of a data point to a cluster is determined by the highest probability. In addition to clustering accuracy, we measure the convergence time, which records the interval between the time when first agent initialised and last agent terminated. For communication costs, we record the total number of received messages. For each measure, we compute Mean, Min, Max, Median and Standard Deviation over 50 random problem instances at each setting. The notation is summarised in Table 1.

MMF1	MMF scheme under Scenario 1
MMF2	MMF scheme under Scenario 2
P	Describe clusters by bounding boxes
SG	Fit clusters by separate gaussians
GMM	Gaussian Mixture Model
R-msg	Received data message
R-poll	Received poll message
t_0	First agent learned global view
t_1	Last agent learned global view
t_2	Convergence time for algorithm

Table 1: Notation

None of our methods use a central controller, and we assume an additional process which detects termination if required. For termination time, we record the last time at which any node or message was active. For MMF2, we also measured the time that the first node reached stability (t_0), and the time that the last node reached stability (t_1). In those MMF2 experiments, t_2 is the final termination.

The number of messages received is not necessarily sufficient to measure communication cost, since large messages require more packets and thus longer transmission times. For IEEE 802.11ac, the maximum size of a single packet (MPDU) is 11,454 bytes. For our experiments, we limit the number of clusters to 5. Five centroid locations plus counts, or five Gaussians plus counts, can be described in just over 1000 bytes. The principal components and bounding boxes for five clusters can be described in just 9,272 bytes, and thus, for our experiments, each model description can be transmitted in a single packet, and so the total number of messages is a good proxy for communication energy costs.

First we evaluate performance under scenario 1, in which nodes know the size of the network. The results for densely connected networks of 10 nodes are shown in Table 2. MMF1-P and -SG show the highest accuracy (relative to centralised k-means); Both of these methods use full k-means at each node to generate the initial clustering and the final clustering, but differ in the summaries that are exchanged. MMF1-P requires the fewest messages, but takes significantly longer than MMF1-SG to terminate – this appears to be because of the extra time required to analyse the clusters using PCA and to generate the bounding boxes. MMF1-GMM requires the least time to stabilise, and achieves reasonably high accuracy. The existings methods from [6], [7] and [8] are outperformed on all measures. In Table 3, we show the result for sparsely connected networks of 10 nodes. MMF-1 achieves the highest accuracy and requires the least amount of messages to converge. Again, MMF1-GMM converges faster than other methods and achieves high accuracy. Again, [6], [7] and [8] are outperformed on all measures. To summarise, selecting between MMF1-P, -SG and -GMM will depend on the relative costs of inaccuracy, message transmission, and elapsed time.

In table 4 we show the result of for densely connected networks of 10 nodes for scenario 2, in which nodes only know of the existence of their immediate neighbours[1]. We report three different time measures. MMF2-P again requires fewest messages. MMF-

[1]we omit [6], [7] and [8], which are identical to the first scenario.

Name	Measures	Mean	S.D
MMF1-P	Time	8.6	3.09
	Accuracy	98.42 [72.5 100.0 99.7]	4.02
	R-msg	20.89 [11.5,25.68, 23.43]	9.12
MMF1-SG	Time	2.24	0.1
	Accuracy	98.46 [72.6 99.9 99.8]	5.33
	R-msg	33.85 [25.5, 40.53, 33.34]	8.69
MMF1-GMM	Time	1.68	0.08
	Accuracy	94.29 [70.7 99.7 97.15]	8.15
	R-msg	36.51 [27.80, 43.89, 36.0]	7.26
[6]	Time	12.23	6.71
	Accuracy	87.1 [66.1 99.6 90.45]	10.53
	R-msg	42.22 [27.5, 52.65, 42.29]	26.54
	R-poll	45.91 [31.73, 57.61, 44.73]	26.32
[7]	Time	61.61	26.52
	Accuracy	89.21 [20.7, 100.0, 98.7]	15.83
	R-msg	104.27 [96.84, 110.63, 104.73]	3.67
[8]	Time	20.56	19.68
	Accuracy	78.9 [33.1, 99.9, 81.05]	16.35
	R-msg	139.67 [127.72, 148.56, 139.81]	5.42

Table 2: Comparison of MMF1 method in 10-agent dense network. Results show mean results for each measure over 50 runs, with min, max and median in brackets.

Name	Measures	Mean	S.D
MMF1-P	Time	8.72	1.85
	Accuracy	98.73 [72.9 99.9 99.7]	3.82
	R-msg	21.54[21.52, 17.78, 26.57]	3.47
MMF1-SG	Time	3.01	0.13
	Accuracy	96.92 [72.7 99.9 99.8]	8.08
	R-msg	29.74 [17.07, 46.15, 28.96]	2.52
MMF1-GMM	Time	2.57	0.16
	Accuracy	96.23 [73.0 99.7 98.65]	6.20
	R-msg	30.94[18.02, 48.82, 30.25]	2.56
[6]	Time	16.62	6.38
	Accuracy	88.56 [60.5 99.9 92.05]	10.58
	R-msg	29.59[16.03, 50.92, 28.37]	13.13
	R-poll	31.33[17.42,51.84,30.50]	12.59
[7]	Time	42.38	15.59
	Accuracy	81.21 [19.40 100.0 81.4]	15.48
	R-msg	87.56 [77.32,96.56,88.13]	6.41
[8]	Time	71.68	45.36
	Accuracy	78.43 [41.6 100.0 79.95]	14.01
	R-msg	204.16 [180.54, 224.47, 203.82]	12.47

Table 3: Comparison of different schemes under MMF1 method in a 10-agent sparse network

SG again has the highest accuracy. The relative performance of MMF2-GMM has improved, with close to the highest accuracy, and clearly the fastest termination time. [6] is still outperformed by the other methods, although the number of messages (including both data messages and polling messages) are no longer significantly higher. [7] and [8] are still outperformed on all measures. The result for sparse networks are shown in table 5. MMF2-SG shows the highest accuracy and MMF2-P requires the fewest messages but takes substantially longer time than MMF2-SG to terminate. MMF2-GMM converges much faster than other methods and still achieves high accuracy. The methods in [6], [7] and [8] are outperformed on all measures.

In summary, the methods that use full clustering at each node on each cycle, and which exchange more informative descriptions, outperform [6]. Taking MMF2-GMM as representative, it achieves 10 percentage points higher accuracy relative to centralised k-means (reduces the misclassification rate from 12% to 1.7%), reduces the message count by 20%, and reduces elapsed time by 75%.

In all cases, we respect the privacy of the original data, and do not exchange any individually identifiable data points. However, we do lose some privacy compared to [6], in that we can now identify data distributions for individual nodes.

Name	Measures	Mean	S.D
MMF2-P	$[t_0, t_1, t_2]$	[5.83, 10.14, 11.17]	[0.52, 2.25, 2.22]
	Accuracy	96.88 [72.6 99.5 98.35]	5.07
	R-msg	35.95 [22.12,47.84,35.04]	2.0
MMF2-SG	$[t_0, t_1, t_2]$	[1.77, 2.47, 3.66]	[0.06, 0.09, 0.22]
	Accuracy	98.29 [72.6 100.0 99.8]	6.09
	R-msg	79.40[51.32, 103.14, 77.15]	1.80
MMF2- GMM	$[t_0, t_1, t_2]$	[1.20, 1.87, 3.07]	[0.05, 0.11, 0.18]
	Accuracy	97.82 [72.9 100.0 99.7]	6.31
	R-msg	69.40[44.07, 91.12, 67.37]	3.12

Table 4: MMF2 method in 10-agent dense networks

Name	Measures	Mean	S.D
MMF2-P	$[t_0, t_1, t_2]$	[6.04, 9.18, 13.79]	[1.15, 1.00, 2.89]
	Accuracy	93.04 [74.5 98.1 94.7]	5.15
	R-message	32.71[16.94, 56.76, 32.71]	3.0
MMF2-SG	$[t_0, t_1, t_2]$	[2.33, 3.44, 5.97]	[0.16, 0.18, 0.71]
	Accuracy	96.57 [72.5 100.0 99.8]	8.84
	R-message	63.31[32.39,111.36,63.31]	3.75
MMF2- GMM	$[t_0, t_1, t_2]$	[1.77, 2.82, 5.94]	[0.13, 0.17, 0.95]
	Accuracy	96.15 [73.0 100.0 99.5]	8.51
	R-message	57.61[29.73, 99.85, 57.60]	3.78

Table 5: Comparison of different schemes under MMF2 method in a 10-agent sparse network

5. CONCLUSION AND FUTURE DIRECTIONS

We proposed an asynchronous distributed clustering algorithm framework for wireless mesh networks, which respects data privacy, while balancing communication cost and clustering quality. Methods that use full k-means clustering at each node each cycle, and which exchange cluster shape and density descriptions, require fewer messages and give higher accuracy relative to centralised k-means, compared to previous methods. Distributed Gaussian Mixture Model clustering is almost as high in accuracy, and requires less elapsed time. The methods do, however, leak some privacy about the data sensed by each individual node. The results show that more informative cluster descriptions improve distributed clustering.

For a more comprehensive conclusion, we will extend the evaluation, to consider larger networks and different data distributions. We will extend our algorithms and experiments to include networks in which nodes fail (e.g. because of limited batteries). We will address application scenarios in which subgroups of nodes sense different data distributions from the rest of the network, and problems where the distributions change over time. Finally, we will extend our methods to handle different inference problems, identifying which problems can or should be handled in the network, and which require transmission to a central server for more extensive analysis.

6. ACKNOWLEDGMENT

This research was supported by Science Foundation Ireland under Grant Number SFI/12/RC/2289.

References

[1] Harb, Hassan, Abdallah Makhoul, Samar Tawbi, and Raphaël Couturier. "Comparison of different data aggregation techniques in distributed sensor networks." IEEE Access 5 (2017): 4250-4263.

[2] Ting, Ying-Yao, Chi-Wei Hsiao, and Huan-Sheng Wang. "A Data Fusion-Based Fire Detection System." IEICE Trans Information and Systems 101.4 (2018): 977-984.

[3] Ilyas, Mohammad, and Imad Mahgoub, eds. Handbook of sensor networks: compact wireless and wired sensing systems. CRC press, 2004.

[4] Zeng L, Li L, Duan L, Lu K, Shi Z, Wang M, Wu W, Luo P. Distributed data mining: a survey. Information Technology and Management, 2012, 13(4): 403-409.

[5] P. A. Forero, A. Cano and G. B. Giannakis, "Distributed Clustering Using Wireless Sensor Networks," in IEEE Journal of Selected Topics in Signal Processing, vol. 5,no. 4, pp. 707-724, Aug. 2011.

[6] Datta S, Giannella C, Kargupta H. Approximate distributed k-means clustering over a peer-to-peer network. IEEE Transactions on Knowledge and Data Engineering, 2009, 21(10): 1372-1388.

[7] Di Fatta, G., Blasa, F., Cafiero, S., Fortino, G. Fault tolerant decentralised k-means clustering for asynchronous large-scale networks[J]. Journal of Parallel and Distributed Computing, 2013, 73(3): 317-329.

[8] Bénézit F, V. Blondel, P. Thiran, J. Tsitsiklis and M. Vetterli. Weighted gossip: Distributed averaging using non-doubly stochastic matrices, 2010 IEEE Int Symp on Information Theory, Austin, TX, 2010, 1753-1757.

[9] X.Kui, J.Wang, S.Zhang, and J.Cao. Energy balanced clustering data collection based on dominating set in wireless sensor networks, Ad-hoc Sensor Wireless Netw. J., 24, 99-217, 2015.

[10] Y. Lu, I. S. Comsa, P. Kuonen, and B. Hirsbrunner, Dynamic data aggregation protocol based on multiple objective tree in wireless sensor networks, in Proc. IEEE 10th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Processing, 1-7 2015.

[11] C.Wang, L.Xing, V.M.Vokkarane, and Y.Sun, Reliability of wireless sensor networks with tree topology, Int. J. Performability Eng., 8, 213-216, 2012.

[12] K.Bhakare, R.Krishna, and S.Bhakare. An energy-efficient grid based clustering topology for a wireless sensor network, Int. J. Comput. Appl. 39 (14), 2012.

[13] H. A. Marhoon, M. Mahmuddin, and S. A. Nor, Chain-based routing protocols in wireless sensor networks: A survey, J. Eng. Appl. Sci., 10 (3), 1389-1398, 2015.

[14] C.-M. Chao and T.-Y. Hsiao, Design of structure-free and energy- balanced data aggregation in wireless sensor networks, J. Netw. Comput. Appl., 17, 229-239, 2014.

[15] Lu, Huang, Jie Li, and Mohsen Guizani. "Secure and efficient data transmission for cluster-based wireless sensor networks." IEEE transactions on parallel and distributed systems 25.3 (2014): 750-761.

[16] M. Shanmukhi and O. B. V. Ramanaiah, Cluster-based comb-needle model for energy-efficient data aggregation in wireless sensor networks, in Proc. Appl. Innov. Mobile Comput., 2015, 42-47.

[17] Ahmed, A. and Qazi, S. Cluster head selection algorithm for mobile wireless sensor networks. In Open Source Systems and Technologies (ICOSST), 120-125, 2013.

[18] Bendechache M, Le-Khac N A, Kechadi M T. Hierarchical aggregation approach for distributed clustering of spatial datasets. IEEE 16th International Conference on Data Mining Workshops. 2016: 1098-1103.

[19] Roie Zivan and Amnon Meisels. 2006. Message delay and DisCSP search algorithms. Annals of Mathematics and Artificial Intelligence 46, 4, 2006, 415-439.

[20] Erdös, P. and Rényi, A. A. On random graphs I. Publicationes Mathematicae, 6:290-297, 1959.