Gravitational Clustering: A Simple, Robust and Adaptive Approach for Distributed Networks

Patricia Binder, Michael Muma and Abdelhak M. Zoubir

Abstract—Distributed signal processing for wireless sensor networks enables that different devices cooperate to solve different signal processing tasks. A crucial first step is to answer the question: who observes what? Recently, several distributed algorithms have been proposed, which frame the signal/object labelling problem in terms of cluster analysis after extracting source-specific features, however, the number of clusters is assumed to be known. We propose a new method called Gravitational Clustering (GC) to adaptively estimate the time-varying number of clusters based on a set of feature vectors. The key idea is to exploit the physical principle of gravitational force between mass units: streaming-in feature vectors are considered as mass units of fixed position in the feature space, around which mobile mass units are injected at each time instant. The cluster enumeration exploits the fact that the highest attraction on the mobile mass units is exerted by regions with a high density of feature vectors, i.e., gravitational clusters. By sharing estimates among neighboring nodes via a diffusion-adaptation scheme, cooperative and distributed cluster enumeration is achieved. Numerical experiments concerning robustness against outliers, convergence and computational complexity are conducted. The application in a distributed cooperative multi-view camera network illustrates the applicability to real-world problems.

Index Terms—adaptive distributed clustering, cluster enumeration, robust, outlier, multi device multi task (MDMT), wireless sensor networks, labelling.

I. INTRODUCTION

A recent and emerging research direction in distributed signal processing for wireless sensor networks (WSNs) is that of enabling cooperation among multiple heterogeneous devices dedicated to solve different signal processing tasks [1]–[5]. A crucial first step towards this so-called multiple devices multiple tasks (MDMT) paradigm is to answer the question: who observes what? [6]–[9]. For example, to arrive at a node-specific speech signal enhancement [10]–[12], all relevant speech sources must be uniquely labelled throughout the wireless acoustic sensor network. Similarly, distributed node-specific image/video enhancement requires the common labelling of all objects within a camera network [6].

To illustrate the challenging requirements for such labelling methods, consider for example a video enhancement setup, where multiple users film a nonstationary scene from different angles using their camera equipped portable devices. Each user has its own dedicated signal processing task, e.g. enhancing a specific object of interest. No prior information, such as positions of devices, registration of views or number of objects in the scene, is available, streaming-in data must be processed sequentially and little is known about the distribution of the data. Further, a central computing unit (fusion center) is not available and communication (range, bandwidth) and computation capabilities (memory, computing power), as well as battery power may be limited.

Recently, several distributed algorithms have been proposed, which frame the labelling problem in terms of cluster analysis after extracting source-specific features [6]–[8], [13], [14]. Various methods have been proposed for distributed data clustering, e.g., [14]–[27]. However, a significant drawback of common clustering algorithms is that the number of clusters has to be known a priori. In real scenarios, this information is not always available [28] or the number of clusters might be chosen improperly. Also, in a sensor network the number of clusters may change over time in a non-stationary scenario.

To the best of our knowledge, distributed cluster enumeration has only been addressed in [29], which serves here as a benchmark algorithm. For the single-node case, the question of inferring the number of clusters from the observations has been intensively studied [30]–[53]. However, most of these approaches are of high computational complexity, need to make prior assumptions on the data distribution or do not allow for an adaptive processing without the need to re-run the entire algorithm. Therefore, these methods are not suitable for the above-described object labelling task in MDMT networks.

The aim of this research is to adaptively estimate the time-varying number of clusters based on a set of streaming-in feature vectors. The proposed method is designed to be

1) adaptive - to a changing number of objects/sources,
2) robust - against outliers in the feature vectors, or in general against unknown non-spherical and possibly heavy tailed distributions of the estimated features,
3) distributed - so as to operate in a decentralized WSN, e.g., based on the diffusion-principle [14], [54],
4) sequential - so that the estimate of the number of clusters is continuously updated for streaming-in data without the need to re-run the entire algorithm,
5) computationally simple - in order to be applicable in a real WSN.

Original Contributions: A robust gravitational clustering algorithm is proposed which works for single-node and cooperative in-network clustering. The key idea is to exploit the physical principle of gravitational force between mass units. In this work, streaming-in feature vectors are considered as mass units of a fixed position in the feature space, around which mobile mass units are injected at each time instant. The cluster enumeration exploits the fact that the highest attraction on the mobile mass units is exerted by regions with a high density of feature vectors, i.e., gravitational clusters. The masses of mobile units are combined when they are in a close vicinity of each other and a threshold on the combined mass serves as detector for a cluster. In this way, the time-
varying number of clusters can be determined. By sharing
estimates among neighboring nodes via a diffusion-adaptation
scheme, cooperative and distributed cluster enumeration is
achieved. An extensive simulation-based performance anal-
ysis is provided that investigates the clustering performance
for single-node and multi-node cluster enumeration. Herein,
aspects such as robustness against outliers, computational cost
and convergence are investigated. The applicability of the
gravitational clustering algorithm is illustrated for a use-case
of labelling moving objects in a synthetic 3-D multi camera
network.

Related Work: The idea to cluster data based on the law of
gravity was first proposed by Wright in [55] where clustering
is performed by moving and merging the data points based
on gravitational force until one final cluster remains. This
approach has been extended in some works, e.g., [56], [57],
which consider multiple clusters. An overview of existing
methods which exploit the gravitational principle is provided
in [57]. In these methods, a decay term prevents that all
samples conflate into one big cluster, or a threshold is set
that determines up to which distances clusters should stay
separated and which clusters may merge. This requires prior
knowledge about the data, e.g., the minimum distance between
the clusters or the distance of the data points from their
corresponding cluster centroid, in order to assure adequate
performance. This kind of information is not always available.
Another drawback is that one cannot draw inferences from
the resulting clusters about the actual positions of the cluster
centroids since the data points (and therefore the cluster
centroids) change their positions because of their mutual
attraction. Further, such a procedure makes it difficult to adapt
to changes in the scenario without the need to re-run the
algorithm.

Notation: The following notation is used: vectors are de-
noted by bold small letters $\mathbf{a}$ and matrices by bold capital
letters $\mathbf{A}$. All vectors are denoted as column vectors. Sets
are denoted by calligraphic letters $\mathcal{A}$ with $| \cdot |$, indicating the
cardinality of a set, the notation $\mathcal{A} \setminus i$ describes the result-
ning set after excluding element $i$ from $\mathcal{A}$ while $\| \cdot \|$ denotes the
Euclidean norm of a vector. The superscript $^\top$ denotes the
transpose operator and $\mathbf{I}_M$ stands for an $M \times M$ identity
matrix.

Organization: Section II provides the problem formulation
and data model. Section III is dedicated to the proposal
of our gravitational clustering algorithm, while Section IV
provides an extensive Monte-Carlo simulation study. Section V
concludes the paper and provides future research directions.

II. PROBLEM FORMULATION, SIGNAL MODEL AND AIMS

We consider a network of $J$ nodes whose topology is
described by a graph with nodes indexed by $j = 1, \ldots, J$. The
neighborhood of node $j$, denoted as $\mathcal{B}_j$, is the set of nodes,
including $j$, that node $j$ exchanges information with, and $|\mathcal{B}_j|$ denotes its associated cardinality. Each observation is assumed
to belong to a certain cluster $c_k$ with $k = 1, \ldots, K$ denoting
the label of the given cluster. The total number of clusters
$K$ is assumed to be unknown and might change over time.

Each cluster is described by a set of application-dependent
descriptive statistics (features).

The feature estimation process is an application-specific
research area of its own (see, e.g., [6], [7]) and is not the
focus of this article, where we seek for a generic adaptive
and robust cluster enumeration method. It is assumed that the
features have already been extracted and the uncertainty within
each cluster $k$ can be modeled by a probability distribution,
e.g., the Gaussian. Further, we account for gross estimation
errors in the feature extraction process that we consider as
outliers, thus arriving at the following observation model for
feature vectors of node $j$ at time instant $t = 1, \ldots, N$:

$$d_{kj}(t) = w_k(t) + n_{kj}(t). \tag{1}$$

Here, $w_k(t)$ denotes the class centroid, $n_{kj}(t)$ represents a
stochastic, clusters-specific uncertainty term of unspecified
distribution with associated covariance matrix $\Sigma_{kj}$, and
d$k(t), w_k(t), n_{kj}(t) \in \mathbb{R}^{d \times 1}$. For reasons of visual clarity,
we drop the index $k$ in the feature vectors and refer to them as
d$_j(t)$.

The aim of this research is to estimate the time-varying
number of clusters $K(t)$ and class centroids $w_k(t)$ based
on a set of streaming-in feature vectors $d_j(t)$. The proposed
method should be adaptive, robust, distributed, sequential and
computationally simple, as defined in Section III.

III. DESCRIPTION OF THE PROPOSED GRAVITATIONAL
CLUSTERING (GC) ALGORITHM

Gravitational Clustering (GC) is based on Isaac Newton’s
law of universal gravitation which relates the force $f$ between
two mass units with masses $m_1$ and $m_2$ and distance $||r_{12}||$
by

$$f_{12} = - f_{21} = g \cdot m_1 \cdot m_2 \cdot \frac{e_{12}}{||r_{12}||^2},$$

where $g = 6.67408 \times 10^{-11} \text{m}^3 \text{kg}^{-1} \text{s}^{-2}$ is the gravitational
constant and $e_{12}$ denotes the unit vector which points from
body 1 to body 2. GC exploits Newton’s law by modeling
the feature space as a physical space, where $g$ becomes a
tuning parameter. In contrast to the physical model and also
in contrast to [55]–[57], two different types of mass units are
introduced in this work: on the one hand, the feature vectors
are modeled as mass units of fixed position (fixed mass units).
On the other hand, artificially generated mass units (mobile
mass units) are injected into feature space. Mobile mass units
are attracted by fixed mass units, but do not interact with each
other.

GC exploits the fact that the highest attraction on the mobile
mass units is exerted by regions with a high density of feature
vectors, i.e., gravitational clusters, to which mobile mass units
gravitate and where they finally gather, governed by

$$\mathbf{x}(t) = \int_{t-1}^{t} \nu(\tau) d\tau + \mathbf{x}(t-1). \tag{2}$$

Here, $\mathbf{x}(t)$ is the new position of the mobile mass unit at time
instance $t$ to which it traveled in time $\Delta t = 1$ with velocity
The formula for the velocity is given by

$$v(t) = \int_{t-1}^{t} a(\tau) d\tau + v(t-1),$$

(3)

with \(a(t)\) denoting the acceleration and \(v(t-1)\) the current velocity. Using the relation

$$f(t) = m \cdot a(t),$$

(4)

solving for \(a\), and inserting the result in Eq. (3) yields

$$v(t) = \frac{1}{m} \int_{t-1}^{t} f(\tau) d\tau + v(t-1).$$

(5)

Equations (2)-(5) provide the basis for the algorithm proposed in this article. In the following subsections, an approach is presented which introduces the clustering procedure for the single-node case as well as the extension to distributed processing for the use in WSNs. Single-node GC is a standalone method, i.e., it can be applied to any single device clustering task and does not require a WSN.

A. Single Node Approach for Gravitational Clustering

We begin by describing the single node approach. For every fixed mass unit with position \(d_j(t)\), which is available at time \(t\), a mobile mass unit \(u_i\) is emitted in a certain distance \(r_x\) around the fixed mass unit in space. The mobile mass unit is characterized by its position \(x_i(0)\) ∈ \(\mathbb{R}^3\) and its mass \(m_i\):

$$u_i(t) = (x_i(t)^T, m_i(t)^T), \quad i \in \mathcal{U}(t).$$

Here, \(\mathcal{U}(t)\) denotes the set of indices of all mobile mass units in a feature space at time \(t\) with its cardinality \(|\mathcal{U}(t)|\). All fixed mass units are assumed to have equal mass, i.e., \(m_d = 1\), and the mobile mass units \(u_i(0)\) have the initial mass \(m_i(0) = 1\). The distance \(r_x\) is a design parameter which must fulfill \(r_x > 0\) so that the mobile entity is not directly absorbed by the feature vector while being small enough such that the feature vector has an impact on it.

The force acting on a mobile mass unit is the superposition of all single forces emanating from each of the fixed mass units \(d_j(t)\). The formula for the total vectorial force acting on a single mobile mass unit \(u_i(t)\) is then given by

$$f_{grav,i}(t) = \sum_{n=1}^{t} g \cdot m_i(t) \cdot m_d \cdot \frac{d_j(n) - x_i(t)}{||d_j(n) - x_i(t)||^p},$$

(6)

with \(p = 3\) according to the physical model. Since we are not restricted to the physical model, \(p\) is treated as a further design parameter.

To allow a discrete-time representation, we adapt Eqs. (2) and (5) such that the new position of each attracted entity \(u_i(t)\) is calculated based on the following equations:

$$x_i(t) = v_i(t) \cdot \Delta t + x_i(t-1),$$

(7)

given the velocity

$$v_i(t) = \frac{f_{grav,i}(t)}{m_i(t)} \cdot \Delta t + v_i(t-1)$$

(8)

with \(v_i(0) := 0\) for initialization.

With decreasing distance to the feature vectors, the force acting on the mass units grows and strives to infinity as the distance goes to zero. As a consequence, and with \(\Delta t\) not being chosen infinitesimal small, the mobile mass units are accelerated so strongly that they would “shoot” past the data clusters if \(f_{grav,i}(t)\) is not limited. For this purpose a viscous damping force \(f_{damp,i}(t)\) is introduced which is antagonistic to \(f_{grav,i}(t)\), i.e.,

$$f_{damp,i}(t) = -k_{damp} \cdot v_i(t-1).$$

(9)

Here, \(k_{damp}\) is a damping parameter which is chosen such that \(0 < k_{damp} < 1\) to ensure the resulting total velocity \(v_i(t)\) of the moving mass units is positive.

Combining Eqs. (8) and (9) yields

$$v_i(t) = \frac{f_{grav,i}(t) + f_{damp,i}(t)}{m_i(t)} \cdot \Delta t + v_i(t-1).$$

(10)

In order to reduce computational cost, for every time step, the algorithm combines any two mobile mass units to a single unit if their distance is equal or smaller than a small constant \(\epsilon_r\), which should be chosen significantly smaller than the minimum expected distances between the clusters.

For this purpose, we calculate a \(|\mathcal{U}(t)| \times |\mathcal{U}(t)|\) distance matrix \(R(t)\) with elements \(r_{m,n}(t), m, n = 1, \ldots, |\mathcal{U}(t)|\) representing the distance between the mobile mass units:

$$r_{m,n}(t) = ||x_m(t) - x_n(t)||, \quad \forall m, n \in \mathcal{U}(t).$$

(11)

If two mobile mass units \(u_i(t)\) and \(u_{i'}(t)\) lie within a distance \(\epsilon_r\), i.e. \(r_{i,i'}(t) \leq \epsilon_r\), the masses of the combined mobile mass units are summed up and the position of the mobile mass unit, which has the smallest mean distance to all other mobile mass units

$$r_{\text{mean},i}(t) = \frac{1}{|\mathcal{U}(t)|} \sum_{j \in \mathcal{U}(t)} r_{ij}(t)$$

(12)

is retained, since it is more likely to be combined with other mass units in the next iteration. The other mobile mass unit is removed from the feature space. This procedure is repeated for all mobile mass units that lie within an \(\epsilon_r\)-distance. After the combination process is finished, all remaining mass units obtain a new indexing such that \(\mathcal{U}(t+1) \in \{1, 2, \ldots, |\mathcal{U}(t+1)|\}\). Details are given in Algorithm 1.

The mobile mass units continue to move towards the fixed mass units which have the highest attraction and finally remain at the positions where a balance of forces is attained, and hence \(v_i(t) \approx 0\). Note that \(v_i(t) \rightarrow 0\) for \(\Delta t \rightarrow 0\) for the state of balance of forces.
In order to further reduce computational cost, \( f_{\text{grav}, i}(t) \) can be set to 0 if \( \|d_j(t) - x_i(t)\| > d_{\text{max}} \) with \( d_{\text{max}} \) being an arbitrarily large constant.

**Determination of the Number of Clusters:**
In non-stationary scenarios, the number of clusters \( K \) depends on the time instant \( t \). For every time instant (or for every predefined time interval), the algorithm determines if a mobile mass unit indicates a cluster by checking whether its mass exceeds a threshold \( m_{\text{min}} \). This is done for all mass entities such that

\[
K(t) = \{ i \mid m_i(t) \geq m_{\text{min}} \}, \quad \forall i \in \mathcal{U}(t)
\]  

and an estimate of \( K(t) \) is obtained by

\[
\hat{K}(t) = |K(t)|.
\]

Choosing \( m_{\text{min}} > 1 \) prevents that single, “stuck” units are misinterpreted as a cluster. As a consequence, single outliers have no influence on the clustering performance. The position of the combined mass units, which indicate a cluster according to Eq. (14), provide an estimate of the cluster centroids as defined in Eq. (1):

\[
\hat{\omega}_k(t) = x_k(t), \quad \forall k \in K(t).
\]

The complete gravitational clustering procedure is exemplarily presented in Fig. 1 for different time instants with \( p = 2, \epsilon_r = 1, r_x = 2, m_{\text{min}} = 7 \) and 5% outliers, where, at first, data samples from 5 clusters of different shapes are streaming in one at a time. Starting at \( t = t_0 \), for each feature vector (represented by the blue stars) available at that time, an associated mobile mass unit (denoted by the red circles) drawn at random from a Gaussian distribution with \( \mathcal{N}(d_n, r_x \mathbf{I}_n) \) (see subfigure (a)) is emitted. With increasing time, the feature vectors begin to form clusters and the mobile mass units move towards the feature vectors. When they gather in the cluster centers, they eventually fuse with other mobile mass units in their direct environment if their distance is less or equal to \( \epsilon_r \). In Fig. 1 this is indicated by the thickness of the red circles which is proportional to the mass of the units. Once a cluster is found according to Eq. (13), the respective mobile mass unit is represented by a big red colored cross (see subfigure (b)). At \( t = t_2 \), all 5 clusters have formed and have been detected by the GC algorithm, i.e. \( \hat{K}(t_2) = 5 \) (see subfigure (c)),. At time instant \( t \) with \( t_2 < t < t_3 \), a new cluster with centroid \( \hat{\omega}_5 = (-4, -11)^\top \) is generated additionally to the already existing ones so that from now on data samples from 6 clusters are streaming in. After a few further time steps, GC adapts to the changing scenario and provides the correct estimate of the number of clusters in the scene (see subfigure (d)). This ability makes it useful for dynamic scenarios where new objects enter the scene, such as in multi-view camera networks. This example also illustrates that the algorithm is robust to a certain amount of outliers which is further evaluated in Subsections III-A1 and IV-B.

A summary of the GC algorithm is provided in Algorithm 2.

![Fig. 1: Exemplary procedure of the GC at different time instants with \( p = 2 \) and 5% outliers. The red circles denote the mobile mass units and the blue stars represent the fixed mass units. The thickness of the red circles represents the mass of the units. Once a cluster is found, the respective mobile mass unit is indicated by a large red colored cross.](image-url)
The task of the WSN is to solve a cluster enumeration problem jointly by communicating the results within the sensor neighborhood, $B_j$, $j = 1, \ldots, J$. The proposed procedure for a cooperative network clustering procedure extends GC by the distributed adaptive communication scheme provided in Fig. 3 (time indices are left out for reasons of visual clarity). Since the D-GC is based on the diffusion adaptation scheme [54], it does not require a fusion center. Instead information diffuses via local neighborhood communication throughout the network. As a result, the adaptation and learning performance is increased compared to non-cooperative sensor networks [59].

Having observed $t$ feature vectors $d_j$, at time instant $t$, each node $j$ collects them in a $q \times t$ dimensional matrix $D_j(t)$, where each column consists of an observed feature vector. Optionally, $d_j(t)$ is exchanged within $B_j$ before adapting, i.e., the number of clusters is determined based on $D_j(t)$ which additionally contains all available $d_i, l \in B_j$. In this case, the gravitational force for each node $j$ and each mobile mass unit $x_i(t)$ becomes

$$f_{grav,i,j}(t) = \sum_{l \in B_j} \sum_{n=1}^{t} q \cdot m_i(t) \cdot m_d \cdot \frac{d_i(n) - x_i(t)}{\|d_i(n) - x_i(t)\|^p}.$$ 

In order to save communication cost, it is also possible to exchange aggregated information, e.g., the current estimates of the cluster centroids $\hat{w}_k(t)$, instead of the feature vectors. If the exchange of $d_j(t), j \in B_j$ is left out, $D_j(t)$ contains only the feature vectors observed by node $j$ itself, i.e., $D_j(t) = d_j(t)$, and the method proceeds analogously.

The intermediate cluster number estimate $\hat{K}_j(t)$ is improved upon by including received neighboring estimates $\hat{K}_j(l), l \in B_j$ which are combined to form the final decision. To robustify against false decisions at the node level within the network, the “combine” step at node $j$ is chosen as

$$\hat{K}_j(t) = \text{median}\{\hat{K}_j(l)\}, \quad l \in B_j. \quad (16)$$

An overview of other possible combination rules for the “combine” step of the diffusion algorithm is given, e.g., in [60].

As data streams in, the steps shown in Fig. 3 are repeated to provide an online in-network estimate $\hat{K}_j(t)$.

The D-GC algorithm is summarized in Algorithm 3.
covariance matrices as given in the Appendix.

The simulations are performed with the following parameters for GC: one mass unit with position \( x_t \) is placed around each new feature vector whereas its position is drawn at random from a Gaussian distribution with \( \mathcal{N}(d_n, r_x I_q) \) and \( r_x = 1 \). The parameters of GC are set as follows: \( g \) is set to 1, \( \kappa_{\text{damp}} = 0.8 \), \( \epsilon_r = \sqrt{q} \) with \( q \) being the data dimension, and \( m_{\text{min}} = 7 \). The parameter

\[
p \equiv p(x_1, x_2) = (\log_{10} (\|x_1 - x_2\| + 1) + 2
\]

is chosen throughout the simulations which is close to \( p = 2 \) for close distances, but weakens the influence of far away clusters. The minimum and maximum number of clusters which is considered for model order selection of X-Means and PG-Means are set to \( K_{\min} = 1 \) and \( K_{\max} = 9 \), respectively.

The generation of outliers considers a certain percentage of samples that are affected by a process which is modeled by an additive contaminative distribution [58] based on Eq. (1) with

\[
n_{k_j}(t) = e_{k_j}(t) + \zeta(t) o_{k_j}(t),
\]

where \( e_{k_j}(t) \) represents an uncertainty term which models uncertainties following a specific probability distribution \( E \), e.g. the Gaussian, while \( o_{k_j}(t) \) denotes the contaminating outlier process that is independent of \( e_{k_j}(t) \) and \( \zeta(t) \) is a stationary random process for which

\[
\zeta(t) = \begin{cases} 1 & \text{with probability } p_c \\ 0 & \text{with probability } (1 - p_c) \end{cases}.
\]

The displayed results represent the averages that are based on 100 Monte-Carlo runs.

**B. Robustness Against Outliers**

In order to evaluate the performance of GC in the presence of outliers or when the feature vectors do not follow a Gaussian distribution, the behavior of the algorithms for different underlying probability distributions is tested for the two data sets as a function of the number of available feature vectors per cluster. Since the benchmark algorithms are not adaptive, we let the algorithms under test process data batches in steps of 10 at once for a better comparability, i.e., they are performed based on 10, 20, 30, ..., \( K \cdot 50 \) available feature vectors per cluster. Starting with one initial cluster, after every 50 feature vectors per cluster, the number of clusters is increased by one and feature vectors belonging to the new clusters stream in. This ground truth for \( K(t) \) is visualized by the solid black line, e.g. in Figs. 4 and 5.

1) \textbf{Data-1:} We first assume that no outliers are present and that the feature vectors are Gaussian distributed with \( d_n \sim \mathcal{N}(w_k, \Sigma_k) \), where \( w_k \) and \( \Sigma_k \) are given in the Appendix. The result is depicted in Figure 4 (a), where the error bars visualize the estimation error and have a distance of each one standard deviation above and below the curve. Except for some difficulties of the X-Means for a single cluster, the algorithms show similar and satisfactory results.
Laplace-distributed feature vectors while retaining the same distributed data does not always hold [58]. The results for Fig. 4: Evaluation of Data-1 for different underlying probability distributions for the feature vectors and outliers. However, in real-life scenarios, the assumption of Gaussian-distributed data does not always hold [38]. The results for Laplace-distributed feature vectors while retaining the same parameters as before is presented in Fig. 4 (b). The figure shows the high sensitivity of the PG-Means to deviations from Gaussianity.

Since real measurements may also suffer from outlying data, the following simulation evaluates the performance of the algorithms under test for the presence of \( p_e = 5\% \) outliers in the feature vectors. The outliers are generated by drawing a sample at random from a probability distribution and adding it to the original sample (additive outliers) as specified in Eq. (17). In real-world scenarios, outliers usually do not follow any specific distribution. We choose \( \Omega_{kj}^{t}(t) \) to follow a non-symmetric distribution, i.e., a skewed heavy-tailed distribution like the chi-square distribution with different degrees of freedom \( v \) for each cluster. This is done in order to create a non-symmetric outlier distribution instead of a constant shift of the mean of the outlier distribution for all clusters. In this manner, a randomly drawn vector of dimension \( q \) with \( v_1 = 3 \) is added to 5 % of data vectors for the first cluster and a vector with \( v_2 = 5 \) is subtracted from the corresponding feature vectors of cluster 2. For cluster 3, a different random number is drawn for each direction in feature space, i.e. the outliers are generated with \( v_{3,1} = 4 \) and \( v_{3,2} = 1 \) for \( x \) and \( y \) direction, respectively, whereby \( v_{3,2} = 1 \) is subtracted from the \( y \)-component. For the fourth cluster, we have \( v_{4,1} = 2 \), \( v_{4,2} = -3 \) and the outliers for the last cluster are generated by drawing samples at random from a Gaussian distribution such that \( \Omega_{kj}^{t} \sim \mathcal{N}(\omega_o, \Sigma_o) \) with \( \omega_o = (0, 0)^\top \) and \( \Sigma_o = 3I_2 \). The result is depicted in Fig. 4 (c). A small amount of outliers in the data causes significantly worse estimation results for both the X-Means and PG-Means. The increase in the estimation error is less pronounced for the proposed GC algorithm.

2) Data-2: For the second, more complex data set with \( \omega_k \) and \( \Sigma_k \) as given in the Appendix, we evaluate the performance of GC, X-Means and PG-Means analogously as for Data-1 for the outlier-free case and for the presence of \( p_e = 5\% \) Gaussian-distributed additive outliers. The result for outlier-free feature vectors is depicted in Fig. 5 (a) and Fig. 5 (b) presents the estimation outcome for 5% Gaussian-distributed outliers with \( \Omega_{kj}^{t} \sim \mathcal{N}(\omega_o, \Sigma_o) \) and \( \omega_o = (0, 0, 0)^\top \) and \( \Sigma_o = 3I_3 \).

The corresponding RMSE and probability of correct detection results are given in Tables II and III.

| Simulation | RMSE |
|------------|------|
| GC         | 0.152 0.220 0.267 |
| X-Means    | 0.150 0.425 1.092 |
| PG-Means   | 1.065 2.077 3.841 |

| Simulation | RMSE |
|------------|------|
| GC         | 0.168 0.071 0.376 |
| X-Means    | 1.317 3.986 4.711 |

TABLE I: RMSE of the estimated number of cluster centroids compared to the ground truth for the algorithms under test for different simulation scenarios.

![Graphs showing average estimated number of clusters as a function of the number of feature vectors for different scenarios.](image)

Fig. 4: Evaluation of Data-1 for different underlying probability distributions for the feature vectors and outliers.

![Graphs showing average estimated number of clusters as a function of the number of feature vectors for different scenarios.](image)

(a) Average estimated number of clusters as a function of the number of feature vectors.

(b) Average estimated number of clusters as a function of the number of feature vectors, where the feature vectors follow a Laplace distribution.

(c) Average estimated number of clusters as a function of the number of feature vectors, where the feature vectors are contaminated with 5% chi-square distributed additive outliers.

![Graphs showing average estimated number of clusters as a function of the number of feature vectors for different scenarios.](image)

C. Distributed Cluster Enumeration

In the following scenario, the performance of the proposed algorithm is tested for a distributed WSN where \( J = 10 \) nodes are randomly distributed in space. Each node is connected to the \( |B_j| = 4 \) neighboring nodes which have the smallest Euclidean distance. For the clustering procedure, again Data-2 is used. Each node is subject to a contamination with a certain
receive feature vectors including 5\% of outliers.

Three nodes receive outlier-free feature vectors, three nodes receive outlier-contaminated feature vectors, subject to contamination.

Estimates of the number of clusters, $\hat{K}$, are communicated within each neighborhood $B_j$. The number of estimates $K^*$ that are exchanged and Figure 4 (c) considers the case of a non-cooperative network, i.e., the nodes do not communicate at all. It can be seen that the propagation of outlier-contaminated data can result in worse clustering compared to the non-cooperative network scenario. In contrast to its competitors, GC is able to correctly estimate the number of clusters for all variations of data exchange in the evaluated scenarios.

As GC provides not only an estimate of the number of clusters but also of the location of the cluster centroids, the following simulation evaluates the accuracy of the location estimates. As a benchmark, this paper considers the Distributed K-Means (DKM) algorithm by Forero et al. [17]. The basic idea of the DKM is to cluster the features into a preset number of groups, such that the sum of squared-errors is minimized. For details, see [17]. Table III provides the root-mean-square error (RMSE) of the estimated cluster centroids for both GC and DKM compared to the ground truth for this scenario. The results for the DKM are based on 20 and 50 iterations with parameters $p = \nu = 2$, where $p = 2$ enables soft clustering and $\nu = 2$ is the tuning parameter which yields the best results in the performance tests in [17]. As a result, GC is able to provide precise estimates of the cluster centroid locations. In the given scenario, it outperforms the benchmark method with a RMSE being constantly low, independently of the number of available feature vectors per cluster.

D. Convergence

In order to demonstrate the convergence of GC after a limited number of iterations, the temporal behavior of the mass units, which are placed in different initial distances to the feature vectors, is evaluated. For this purpose, a single cluster with 50 feature vectors, $d_n \sim \mathcal{N}(w_k, \Sigma_k)$, is generated with $w = (3, 3)^\top$ and covariance matrix $\Sigma = (0.3, 0.3)^\top I_2$. One

![Figure 6(a)](image)

Fig. 6 (a): Average estimated number of clusters as a function of the number of feature vectors.

![Figure 6(b)](image)

Fig. 6 (b): Average estimated number of clusters as a function of the number of feature vectors per node, where the feature vectors are contaminated with 5\% Gaussian-distributed additive outliers.

![Figure 6(c)](image)

Fig. 6 (c): 0.647 0.207 0.140

Simulation $\bar{p}_{corr}$

| RMSE | GC | X-Means | PG-Means |
|------|----|---------|----------|
| Fig. 6(a) | 0.035 | 2.269 | 4.736 |
| Fig. 6(b) | 0.147 | 1.839 | 3.649 |
| Fig. 6(c) | 0.201 | 1.682 | 4.335 |

Table III: RMSE of the estimated cluster centroids compared to the ground truth for different amounts of feature vectors per cluster.

Table IV: RMSE of the estimated number of cluster centroids compared to the ground truth for the algorithms under test for the distributed clustering scenario.
Fig. 6: Evaluation of Data-2 for a distributed network compared for different variations of data exchange: the exchange of both feature vectors and estimates of the number of clusters (a), exchanging the estimated number of clusters only (b) and a non-cooperative scenario with no data exchange at all (c).

Table V: Probability of correct decision $p_{corr}$ for the algorithms under test for the distributed clustering scenario.

| Simulation | GC | X-Means | PG-Means |
|------------|----|---------|----------|
| Fig. 6 (a) | 0.999 | 0.832 | 0.002 |
| Fig. 6 (b) | 0.978 | 0.439 | 0.283 |
| Fig. 6 (c) | 0.960 | 0.772 | 0.303 |

Fig. 7: Average distance of mass units to cluster center as a function of the number of time steps $t_{it}$.

Mobile mass unit per feature vector is placed in feature space with an associated position that is drawn at random from a Gaussian distribution with $\mathcal{N}(\mu_n, \sigma^2)$. GC is executed and the average Euclidean distance between the mobile mass units and the cluster centroid is calculated for each time step. As the mobile mass units approach the cluster, their distance to the centroid decreases continuously until a minimum distance $\epsilon_{\min}$ is reached, after which the algorithm is said to have converged.

Fig. 7 provides the average number of time steps $t_{it}$ that the GC algorithm requires to achieve convergence for the cases $\sigma = 0.5$, $\sigma = 3$ and $\sigma = 7$. It is shown that the mobile mass units converge to the true cluster centroid during the clustering process and remain in their position even for large initial distances from the cluster centroid. Small values of $\sigma$ result in lower computation times.

E. Computational Cost

Since the energy consumption in wireless devices should be kept as low as possible, the computational cost of the algorithms in use is crucial. In order to compare the algorithms in this aspect, the computation time as a function of the number of available feature vectors is provided by Fig. 8 and given in seconds (using an Intel Core i5 Quad Core 760 PC1156). Because of its performance results and disproportionately high running times, the PG-Means is not considered in this simulation.

V. Multi-View Camera Network Application

The performance of the D-GC algorithm in a cooperative WSN application is evaluated based on the following multi-
Fig. 8: Comparison of computation times for GC and X-Means for different values of $K_{\text{max}}$.

Fig. 9: A wireless cooperative camera network continuously observing a scene. The top image depicts the setup with $J = 3$ cameras observing the scene from different viewpoints. The bottom left and right images show the frames captured at the same time instant by camera 1 and 3, respectively.

We consider a wireless camera network as shown in Fig. 9 where a set of spatially distributed cameras (nodes) records the scene from different angles. By communicating with each other, the aim is to adaptively determine a network-wide estimate of the number of recorded cars. Due to the different viewpoints, the number of observed objects per camera at the same time instant can differ significantly. For this reason, we selected cameras 1-3 to cooperate such that $J = |B_j| = 3$: camera 1 and 2 have a similar view on the scene whereas camera 3 does not see all of the cars recorded by camera 1 and 2. Plus, the cars enter and leave the scene with a delay compared to the other cameras. A video of 95 frames is recorded by the cameras where three cars enter the scene at different time instants. A Gaussian Mixture Model foreground detector is used to separate moving objects from the background in order to extract feature vectors for the clustering process. The feature vectors are composed of SURF [61] and color features. For the color histogram, the detected object is subdivided into three concentric rings. For each ring, a 10-bin histogram per color channel is computed. Concatenating the three histograms for each of the three color channels results in a 90-dimensional color feature. Combining it with the SURF features yields a 211-dimensional feature vector for each object. It is a challenging scenario since the video has a low resolution, the cameras observe the cars from different angles and there is a low amount of available feature vectors. Because of the previous results, we will compare only DX-Means and D-GC in the following. For this scenario, the feature vectors lie close in space which is taken into account by setting the parameters to $K_{\text{max}} = 5$, $r_x = 0.01$, $r_{\text{min}} = 2$, $d_{\text{max}} = 3$ and $\epsilon_t = 0.12$. The cameras exchange their estimates of the number of object (cars) in the scene and decide on a common estimate according to Eq. (16).

The average results based on 100 Monte-Carlo runs are provided in Fig. 10. The variance is a result of the random initialization of the DX-Means and the random initial positioning of mobile mass units for GC. While the DX-Means needs time to converge to the true value, the D-GC provides a close representation of the ground truth as seen by camera 2.

VI. CONCLUSION

A novel clustering algorithm was proposed, which can be adapted to various single-node and in-network scenarios. It provides both an estimate for the number of clusters as well as for the positions of their centroids. A performance comparison to the X-Means and PG-Means has been provided where our proposed method shows promising results. Unlike the benchmarks, the GC is potentially real-time capable and, up to a certain amount, robust against outliers in the data. Using a standard set of default parameters provides good results for the diverse scenarios examined in this work. By including knowledge about the data set in defining the parameters, GC can be further adapted to very specific scenarios, resulting in an increased performance.

Future work will include the application of this algorithm to real-world speech source labelling for distributed signal en-
hancement multi-device-multi-task (MDMT) wireless acoustic sensor networks [10], object labelling in distributed multi-view camera networks as well as labelling of semantic information based on occupancy grid maps for autonomous mapping and navigation with multiple rescue robots [62].

ACKNOWLEDGMENTS

The work of P. Binder was supported by the LOEWE initiative (Hessen, Germany) within the NICER project. The work of M. Muma was supported by the project HANDiCAMS which acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 323944.

APPENDIX

The cluster centroids and diagonal entries of the corresponding covariance matrices for the data sets Data-1 and Data-2 as introduced in Subsection IV.A are given as follows:

**Data-1:**

\[
\begin{align*}
\mathbf{w}_1 &= (-1,0)^\top, \\
\mathbf{w}_2 &= (4,0)^\top, \\
\mathbf{w}_3 &= (0.5)^\top, \\
\mathbf{w}_4 &= (9,4)^\top, \\
\mathbf{w}_5 &= (3,0)^\top, \\
\Sigma_1 &= (0.2,0.4)^\top \Sigma_2, \\
\Sigma_2 &= (0.6,0.6)^\top \Sigma_3, \\
\Sigma_3 &= (0.4,0.2)^\top \Sigma_4, \\
\Sigma_4 &= (0.2,0.2)^\top \Sigma_5, \\
\Sigma_5 &= (0.3,0.5)^\top \Sigma_\alpha = (0.0,0.4,0.4,0.4)^\top \Sigma_3
\end{align*}
\]

\(\alpha = 0.15\).

REFERENCES

[1] J. Plata-Chaves, A. Bertrand, M. Moonen, S. Theodoridis, and A. M. Zoubir, “Heterogeneous and multitask wireless sensor networks—applications, and challenges,” IEEE J. Sel. Topics Signal Process., vol. 11, no. 3, pp. 450–465, 2017.

[2] A. Hassani, A. Bertrand, and M. Moonen, “Cooperative integrated noise reduction and node-specific direction-of-arrival estimation in a fully connected wireless acoustic sensor network,” Signal Process., vol. 107, pp. 68 – 81, 2015, special Issue on ad hoc microphone arrays and wireless acoustic sensor networks Special Issue on Fractional Signal Processing and Applications.

[3] Y. Tong, L. Liu, M. Zhao, J. Chen, and H. Li, “Adaptive fusion algorithm of heterogeneous sensor networks under different illumination conditions,” Signal Process., vol. 126, pp. 149 – 158, 2016, signal Processing for Heterogeneous Sensor Networks.

[4] J. Chen and J. D. Schizas, “Distributed information-based clustering of heterogeneous sensor data,” Signal Process., vol. 126, pp. 35 – 51, 2016, signal Processing for Heterogeneous Sensor Networks.

[5] J. Szurley, A. Bertrand, and M. Moonen, “Distributed adaptive node-specific signal estimation in heterogeneous and mixed-topology wireless sensor networks,” Signal Process., vol. 117, pp. 44 – 60, 2015.

[6] F. K. Teklehaymanot, M. Muma, B. Béjar-Haro, P. Binder, A. M. Zoubir, and M. Vetterli, “Robust diffusion-based unsupervised object labelling in distributed camera networks,” in Proc. 12th IEEE AFRICON, April 2015.

[7] S. Chouvardas, M. Muma, K. Hamaidi, S. Theodoridis, and A. M. Zoubir, “Distributed robust labeling of audio sources in heterogeneous wireless sensor networks,” in Proc. 40th IEEE Int. Conf. Acoustics, Speech and Sign. Process. (ICASSP), December 2015, pp. 5783–5787.

[8] M. H. Bahari, J. Plata-Chaves, A. Bertrand, and M. Moonen, “Distributed labelling of audio sources in wireless acoustic sensor networks using consensus and matching,” in 2016 24th Eurp. Sign. Process. Conf. (EUSIPCO). IEEE, 2016, pp. 2345–2349.

[9] A. Bertrand and M. Moonen, “Distributed signal estimation in sensor networks where nodes have different interests,” Signal Process., vol. 92, no. 7, pp. 1679 – 1690, 2012.

[10] ——, “Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology,” IEEE Trans. Signal Process., vol. 59, no. 5, pp. 2196–2210, May 2011.

[11] A. Hassan, J. Plata-Chaves, A. Bertrand, and M. Moonen, “Multi-task wireless acoustic sensor network for node-specific speech enhancement and doa estimation,” in 2016 IEEE Sensor Array and MultiChannel Signal Process. Workshop (SAM). IEEE, 2016, pp. 1–5.

[12] A. Hassan, J. Plata-Chaves, M. H. Bahari, M. Moonen, and A. Bertrand, “Multi-task wireless sensor network for joint distributed node-specific signal enhancement, LCMV beamforming and DOA estimation,” IEEE J. Sel. Topics Signal Process., vol. 11, no. 3, pp. 518–533, 2017.

[13] P. Binder, M. Muma, and A. M. Zoubir, “Robust and computationally efficient diffusion-based classification in distributed networks,” in Proc. 40th IEEE Int. Conf. Acoustics, Speech and Signal Process. (ICASSP), Brisbane, Australia, December 2015, pp. 3432–3436.

[14] P. Binder, M. Muma, and A. Zoubir, “Robust and adaptive diffusion-based classification in distributed networks,” EURASIP J. Adv. Signal Process., vol. 2016, no. 1, pp. 1–13, 2016.

[15] M. Hai, S. Zhang, L. Zhu, and Y. Wang, “A survey of distributed clustering algorithms,” in Proc. IEEE Int. Conf. Industr. Controle Electr. Engin. (ICICCE), August 2012, pp. 1142–1145.

[16] R. D. Nowak, “Distributed EM algorithms for density estimation and clustering in sensor networks,” IEEE Trans. Signal Process., vol. 51, no. 8, pp. 2245 – 2253, August 2003.

[17] P. Forero, A. Cano, G. B. Giannakis et al., “Distributed clustering using wireless sensor networks,” IEEE J. Sel. Topics Signal Process., vol. 5, no. 4, pp. 707–724, 2011.

[18] X. Zhao and A. H. Sayed, “Distributed clustering and learning over networks,” IEEE Trans. Signal Process., vol. 63, no. 13, pp. 3285–3300, July 2015.

[19] ——, “Clustering via diffusion adaptation over networks,” in Proc. International Workshop on Cognitive Information Processing (CIP), May 2012, pp. 1–6.

[20] S. Khawatmi, A. M. Zoubir, and A. H. Sayed, “Decentralized clustering over adaptive networks,” in Proc. 23rd European Signal Process. Conf. (EUSIPCO). Nice, France, Nice, France, 2015, pp. 2745–2749.

[21] P. Sasi Kumar and S. Khara, “K-means clustering in wireless sensor networks,” in 2012 Fourth Int. Conf. on Computational Intell. and Communication Networks, Nov 2012, pp. 140–144.

[22] P. Shen and C. Li, “Distributed information theoretic clustering,” IEEE Trans. Signal Processing, vol. 62, no. 13, pp. 3442–3453, 2014.

[23] M. Nasim, S. Qaisar, and S. Lee, “An energy efficient cooperative hierarchical mimo clustering scheme for wireless sensor networks,” Sensors, vol. 12, no. 1, pp. 92–114, 2012.

[24] J. Qin, W. Fu, H. Gao, and W. X. Zheng, “Distributed k-means algorithm and fuzzy c-means algorithm for sensor networks based on multilayer consensus theory,” IEEE Trans. Cybern., vol. 47, no. 3, pp. 772–783, March 2017.

[25] Q. Liu, W. Fu, J. Qin, W. X. Zheng, and H. Gao, “Distributed k-means algorithm for sensor networks based on multi-agent consensus theory,” in 2016 IEEE Int. Conf. on Industrial Technology (ICIT), March 2016, pp. 2114–2119.

[26] D. Gu, “Distributed EM algorithm for gaussian mixtures in sensor networks,” IEEE Trans. on Neural Networks, vol. 19, no. 7, pp. 1154–1166, 2008.

[27] D. Pastor, E. Dupraz, and F.-X. Sochenlau, “Decentralized clustering based on robust estimation and hypothesis testing.”

[28] B. Issac and N. Israr, Case Studies in Secure Computing: Achievements and Trends. Taylor & Francis, 2014.

[29] F. Teklehaymanot, M. Muma, J. Lin, and A. M. Zoubir, “In-network adaptive diffusion-based clustering for distributed classification/labeling,” in European Signal Process. Conf. (EUSIPCO) 2016, April 2016.

[30] C. Fraley and A. E. Raftery, “How many clusters? which clustering method? answers via model-based cluster analysis,” The Comput. J., vol. 41, no. 8, p. 578, 1998.

[31] S. Salvador and P. Chan, “Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms,” in 16th IEEE Int. Conf. on Tools with Artificial Intell., Nov 2004, pp. 576–584.

[32] M. Yan and K. Ye, “Determining the number of clusters using the weighted gap statistic,” Biometrics, vol. 63, no. 4, pp. 1031–1037, 2007.
