Distributed k-means algorithm

Gabriele Oliva, *Member, IEEE,* and Roberto Setola, *Senior Member, IEEE*

Abstract—In this paper we provide a distributed implementation of the k-means clustering algorithm, assuming that each node in a wireless sensor network is provided with a vector representing an observation (e.g., sensorial data, position or both). The proposed algorithm, by means of one-hop communication only, partitions the nodes into groups that have small in-group and large out-group distances. Since the partitions may not have a direct relation with the topological clusters, an algorithm for the computation of the centroid of a sparse subset of the nodes is provided. Depending on the nature the observation of each node, it is possible to obtain a position clustering, a measure clustering or a hybrid clustering.

Index Terms—k-means, clustering, distributed algorithms, consensus

1 Introduction

Wireless sensor networks are becoming more and more present in everyday life, as a powerful tool to measure and put together spatially dispersed information, to be used for several applications, such as monitoring, robot coordination, indoor localization, rescue, etc.

Dealing with huge sets of multidimensional data is, in many cases, a non-trivial issue, and there is the need to provide agile and efficient methodologies to grasp the fundamental aspects of the ongoing situation, as well as to let the network of nodes organize in groups that are homogeneous according to distance or other criteria, e.g., for rescue or coordination purposes.

The above goals can be reached by means of clustering algorithms such as the k-means algorithm [14], that partitions a set of observations into groups with small in-group and large out-group distances.

In this paper we provide a fully distributed implementation of the k-means algorithm for a network of nodes, each holding a (possibly high dimensional) measure of piece of information.

1.1 Related Work

In the literature parallel and distributed k-means implementations have been motivated by the need to categorize huge data sets of high-dimensional observations. In [1], [2], [3] the focus is mainly on multiprocessor architectures; in [4] a parallel implementation of fuzzy clustering techniques is provided; in [5] a parallel version of k-means is provided with a specific focus on privacy preserving aspects.

Another typical distributed application involving clustering is the so called *consensus clustering*, where a set of m agents execute each a clustering algorithm over the same data set and the nodes have to reach a consensus on the optimal partition [6], [7], [8].

In the context of wireless sensor networks there are several attempts to implement distributed clustering schemes, especially with respect to probabilistic approaches: in [27] an incremental scheme is given; in [9] the distributed approach is based on a gossip algorithm; in [10] distributed consensus algorithms are used to reach consensus on a Mixture of Gaussians distribution; a scheme that requires the graph to be a tree is provided in [11], and a scheme based on the method of multipliers is given in [12]; in [28] a decentralized clustering algorithm is provided for a hierarchical sensor network, composed of a static backbone of cluster heads and a set of low-end sensors; in [29] a distributed clustering algorithm for a sensor network is given assuming multiple power levels in sensor nodes.

With the exception of [12], [11], those works are limited to a parameter estimation or to specific network topologies or configurations.

In [13] distributed clustering schemes are provided by transforming the original clustering problem in several smaller clustering problems and then using consensus constraints and single-hop communication to reach a global solution.

1.2 Contribution

With respect to previous approaches, we provide a formalization of the problem where each node in a wireless sensor network is provided with a single observation, i.e., a vector containing sensorial information and/or its position in the space.

Each node therefore, instead of solving a smaller clustering problem, has to solve an assignment problem with respect to a set of centroids (i.e., finding the nearest one) as well as contribute to the refinement of the centroids interacting with the other nodes. The procedure is executed by means of one-hop communication only and the result is proven to be exactly the same of the centralized k-means algorithm. This is done by

• *G. Oliva and R. Setola are with the Complex Systems & Security Laboratory, University Campus Bio-Medico, via A. del Portillo 21, 00128, Rome, Italy.*
E-mail: g.oliva@unicampus.it r.setola@unicampus.it

resorting to distributed consensus algorithms such as average-consensus and max-consensus [16], [17], [18]. Specifically, in the proposed approach it is possible to solve three different classes of problems:

- **Position clustering**: the observation of each node coincides with its position and the algorithm partitions the nodes according to their positions;
- **Measure clustering**: the nodes execute the clustering algorithm with respect to sensorial observations, without taking into account their position;
- **Hybrid clustering**: the observation of each node includes both sensorial information and position, eventually suitably scaled in order to reach the desired combination: the clustering will therefore be on both position and measures.

Especially for the measure and hybrid clustering problems, there is the non-negligible risk to obtain clusters composed of a sparse subsets of nodes, hence we provide a distributed algorithm to solve the issue.

1.3 Paper Outline

The outline of the paper is as follows: After a notation subsection that concludes this introduction, Section 2 reviews the k-means clustering algorithm, while Section 3 outlines distributed consensus algorithms; Section 4 introduces the proposed distributed k-means algorithm, while section 5 details the proposed centroid computation algorithm for sparse clusters; finally some simulation results and conclusions are provided in Sections 6 and 7.

1.4 Notation

n	number of observation/nodes;
d	size of the observations;
x_i	$d \times 1$ vector of the ith observation/node;
k	number of clusters for k-means algorithm;
S_i	ith cluster;
S	set of clusters;
r_{ij}	decision variable representing the assignment of observation x_i to cluster S_j;
c_j	$d \times 1$ vector of the centroid of cluster S_j;
D	objective functional of the k-means algorithm;
M	number of iterations of the k-means algorithm;
$\mathcal{G}$	weighted graph;
$\mathcal{V}$	set of nodes of graph $\mathcal{G}$;
$\mathcal{E}$	set of edges of graph $\mathcal{G}$;
$\mathcal{W}$	set of the weights of the edges $\mathcal{E}$ of graph $\mathcal{G}$;
v_i	ith node of graph $\mathcal{G}$;
(v_i, v_j)	edge connecting two nodes v_i and v_j of graph $\mathcal{G}$;
w_{ij}	weight of edge (v_i, v_j);
$\mathcal{N}_i$	neighborhood of node i;
$z_i(t)$	state variable of an node in a consensus algorithm at step t;
τ	sampling time of consensus algorithms;
$\chi(\cdot)$	function to be computed by χ-consensus;
$u_i(t)$	input for the ith node in a consensus algorithm at step t;
$u_i(\mathcal{N}_i, t)$	neighborhood-dependent input for the ith node in a consensus algorithm at step t;
T_{max}	maximum number of iterations for consensus algorithms;
η	stop parameter for average consensus algorithm;
m	length of the diameter of the graph $\mathcal{G}$;
ρ	communication threshold;
s_{jh}	hth subcluster of cluster S_j;
$\sigma(S_j)$	set of subclusters of cluster S_j;
i^*	identifier of the node elected as leader;
$z_i^t(t)$	$kd \times 1$ vector to be used to obtain information on all centers at step t;
$z_i^t(T_{max})$	$kd \times 1$ vector containing the stack of the coordinates of the k centers at step t;
$k_i^*(t)$	id of the current nearest center for node i at step t;
$\mu_i(t)$	$d \times 1$ vector representing the choice of the nearest center for node i at step t;
$\mathcal{N}_i^c(t)$	subset of $\mathcal{N}_i$ containing nodes with the same choice of the nearest center at step t;
$\mathcal{G}^c(t)$	subgraph of $\mathcal{G}$ induced by $\mathcal{N}_i^c(t)$;
NS_i	network size counter for node i;
$\text{SCS}_{ij}(t)$	size of the subcluster of $S_j(t)$node i belongs to at step t;
$\phi_{ij}(t)$	boolean membership of node i to the cluster $S_j(t)$ at step t;
$\text{SCN}_{ij}(t)$	counter of subcluster number in $S_j(t)$ for node i at step t;
e^h	hth unit versor;
$e_i(t)$	versor selected by ith node at time step t;
$\text{ID}_{ij}(t)$	ith greatest index among the subcluster leaders in $S_j(t)$

2 K-Means Algorithm

Consider a set of n observation $x_1, \ldots, x_n$, where each observation x_i is a vector in $\mathbb{R}^d$. The *k-means algorithm* [14] algorithm aims at partitioning the n observations into k ($k \leq n$) sets or *clusters* $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the following function:

$$D = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} ||x_i - c_j||^2 \tag{1}$$

where $r_{ij} = 1$ if observation x_i is assigned to the set S_j and $r_{ij} = 0$ otherwise, and c_j is the centroid of the observations within the set S_j.

Starting with a random set of centroids $c_1(0), \ldots c_k(0) \in \mathbb{R}^d$ the algorithm alternates an *assignment* and an *update* step.

During the assignment step, each observation x_i is assigned to the set characterized by the nearest centroid,

i.e.:

$$r_{ih}(t) = \begin{cases} 1 & \text{if } h = \ \text{argmin}_j ||x_i - c_j(t)|| \\ 0 & \text{else} \end{cases} \quad (2)$$

During the update step each c_j is updated as the centroid of the observations associated to S_j, i.e.:

$$c_j(t+1) = \frac{\sum_{i=1}^{n} r_{ij}(t) x_i}{\sum_{i=1}^{n} r_{ij}(t)} \quad (3)$$

The two steps are iterated until convergence or up to a maximum of M iterations.

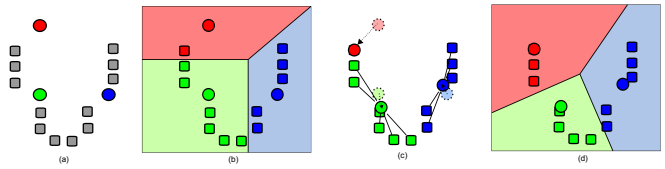

Fig. 1. Example of execution of k-means algorithm (source; Wikimedia Commons available under GNU Free Documentation License v. 1.2).

Figure 1 is an example of execution of the algorithm for a set of $n = 12$ observations in $\mathbb{R}^2$ and for $k = 3$. Specifically Figure 1.(a) shows the initial centroids, Figure 1.(b) and Figure 1.(c) are the assignment and update steps for the first iteration, while Figure 1.(d) is the assignment step for the second iteration.

The k-means algorithm is granted to converge to a local optimum value, while there is no guarantee to converge to the global optimum[14], [15]. Since there is a strong dependency on the initial choice of the centroids, a common practice is to execute the algorithm several times and select the best solution.

For the computational complexity, note that for each iteration, for each of the n observation and for each of the d components of the observations the algorithm calculates the difference with each of the k centers, hence the complexity is $O(dknM)$ [15].

Remark 1: Note that a center that is not chosen in the first step by any agent has a chance to be never chosen, e.g. when the random center is chosen very far from any observation and center. In this case a possible solution is to replace the center with a new random center. As the density of observations grows, however, the risk of this singular case becomes less and less likely.

3 Distributed Consensus Algorithms

In the following we will review the average consensus problem and the max consensus problem.

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$ be a *weighted graph*, where $\mathcal{V}$ is a set of n *nodes* $v_1, \dots v_n$ and $\mathcal{E}$ is the set of *edges* (v_i, v_j) and W is the set of *weights* w_{ij} associated to each edge (v_i, v_j). A graph is said to be *undirected* if $(v_i, v_j) \in \mathcal{E}$ whenever $(v_j, v_i) \in \mathcal{E}$, and is said to be *directed* otherwise. A graph $\mathcal{G}$ is *connected* if for any $v_i, v_j \in \mathcal{V}$ there is a path whose endpoints are in v_i and v_j, without necessarily respecting the orientation of edges. A graph $\mathcal{G}$ is balanced if for each node $v_i \in \mathcal{V}$

$$\sum_{j=1}^{n} w_{ij} = \sum_{j=1}^{n} w_{ji} \quad (4)$$

i.e., the sum of the weights of incoming and outgoing edges coincide.

Let the *neighborhood* $\mathcal{N}_i$ of node v_i be the set of nodes $\{v_j : (v_j, v_i) \in \mathcal{E}\}$.

Let a set of n nodes, each associated to a node in the graph $\mathcal{G}$ and described by the following discrete-time single integrator dynamics:

$$z_i(t+1) = z_i(t) + u_i(t), \quad z_i(0) = z_{i0} \quad (5)$$

where $z_i(t), z_{i0} \in \mathbb{R}$.

Let $\chi(z_{10}, \dots, z_{n0})$ be any function of the initial conditions of all the nodes; the χ-consensus problem consists in finding an input $u_i(\mathcal{N}_i, t)$ that is function of the state of the nodes in the neighborhood of node i and is such that $\lim_{t\to\infty} z_i(t) = \chi(z_{10}, \dots, z_{n0})$ for all $i = 1, \dots n$.

Among the several different problems that can be solved via consensus algorithms, in the following we will review the average-consensus and max-consensus problems.

In the *average* consensus problem the nodes are required to converge to the average of their initial opinions.

Assuming that the graph $\mathcal{G}$ connected and that it is undirected, or directed and balanced, the problem is known to have an asymptotic solution [16], [17] if the following control law is chosen:

$$u_i(\mathcal{N}_i, t) = \tau \sum_{j \in \mathcal{N}_i} (z_j(t) - z_i(t)) \quad (6)$$

where τ is the sampling time, which is assumed to be

$$\tau \leq \frac{1}{max_{i=1}^{n} \sum_{j=1}^{n} w_{ij}} \quad (7)$$

Note that the sampling time τ is indeed a global quantity, and to solve the average-consensus problem the nodes need to have the same τ. For instance if $w_{ij} \in [0, 1]$ then $\frac{1}{n}$ is a good bound for τ; however, this would require each node to know n. Such an issue is overcome by choosing a considerably small value for τ.

In the literature several extensions of the above control law have been provided to grant convergence in finite time: among the others let us report the following by Wang and Xiao [18]:

$$u_i(\mathcal{N}_i, t) = \tau \sum_{j \in \mathcal{N}_i} sign(z_j(t) - z_i(t)) |(z_j(t) - z_i(t)|^{\alpha_{ij}} \quad (8)$$

where $0 < \alpha_{ij} < 1$. The above control law solves the problem in finite time if $\alpha_{ij} = \alpha_{ji}$ for all $i, j = \dots n$.

For the average computation to be fully distributed, each node has to be provided with a stop criterion: in fact, even if a finite time consensus protocol is chosen, the nodes do not typically know such time, which is

dependent on how much the graph $\mathcal{G}$ is connected, hence on global information. A possible solution is that each node, while continuing to execute the consensus algorithm for exactly $T_{max} >> 1$ steps, stores a copy of $z_i(t)$ which is updated until $|z_i(t) - z_i(t-1)| < \epsilon$, with $\epsilon << 1$.

Let us now discuss the $\max$-consensus problem, where the nodes have to converge to the maximum of the initial conditions. The problem, for a connected and undirected (or directed and balanced) graph, is known to have a solution in finite time [16], [17] if the following control law is chosen:

$$u_i(\mathcal{N}_i, t) = \max_{j \in \mathcal{N}_i} z_j(t) \tag{9}$$

With the above control law, the problem is solved in $m \leq n$ steps, where m is the lenght of the diameter of the graph (i.e., the maximum among the minimum paths for every possible couple of nodes, respecting the orientations of the links if the graph is directed). Again, since the nodes do not know m or n, for practical applications a maximum number of iterations $T_{max} > n$ has to be selected.

Regarding the computational complexity, for each node i we have that $|\mathcal{N}_i| \propto n$, hence the complexity is $O(nT_{max})$.

The control laws (6), (8) and (9) easily generalize to the vectorial case, i.e., $z_{i0} \in \mathbb{R}^d$ is a vector. In this case it is possible to execute d average-consensus or max-consensus algorithms in parallel.

For the average consensus the result is that, for sufficiently big T_{max} and small ϵ, the vector $z_i(T_{max}) \in \mathbb{R}^d$ will be the average of the initial vectors, while for the max-consensus each component of $z_i(T_{max})$ will contain the maximum value of the corresponding component of the initial vectors. In the vectorial case, the computational complexity becomes $O(dnT_{max})$ for each node.

In the following we will assume, without loss of generality, that the graph $\mathcal{G}$ has unit weights and is connected and undirected. In fact, several weight balancing techniques have been proposed in the literature for making a directed graph balanced [24], [25], [26]. However, the weight balancing techniques are out of the scope of this work.

4 Distributed K-Means Algorithm

Let n nodes deployed in $\mathbb{R}^2$ and suppose that each node is able to communicate with the other nodes provided that their distance is less than a communication threshold ρ which, for the sake of simplicity, is the same for all the nodes. Each node is endowed with a real vector $x_i \in \mathbb{R}^d$ representing a piece of information or measure. The objective of the distributed k-means algorithm partition the nodes in k clusters minimizing the functional D specified in eq. (1) via local interaction among neighbors.

In the following we will neglect, without loss of generality, the procedure to mitigate the risk of obtaining a local minimum by iterating the algorithm several times and selecting the best result.

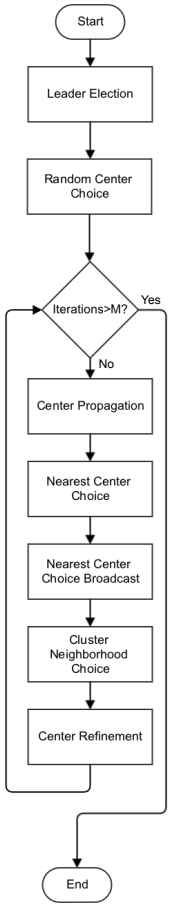

Fig. 2. Distributed k-means algorithm flow-chart

The vector x_i may represent the nodes'position in $\mathbb{R}^2$, resulting in a *position clustering*, or an observation made by node i of a set of variables of interests, such as temperature, humidity, etc., thus the clustering is a *measure clustering*.

It is also possible to consider a *hybrid clustering* problem where vector x_i contains both the position of the node and other sensorial information (eventually scaled to rank the importance of the different infomation), hence the clustering will divide the nodes into groups depending on both position and measures.

Let us make the following assumption.

Assumption A1: For each $t \geq 0$ the subgraph induced by each cluster $S_j(t)$ is connected.

Remark 2: The above assumption is indeed strong, and is more likely to be verified for position clustering. In fact, in this case the assumption tends to be verified as ρ grows. In general, especially for measure and hybrid clustering, the subgraph induced by cluster S_j will be likely to be disconnected in several subclusters $s_{jh} \in \sigma(S_j)$, where $\sigma(S_j)$ is the set of subclusters that

compose cluster S_j.

As will be shown in the next subsection, the disconnectedness of a cluster S_j causes the failure of the distributed k-means algorithm and a more complex procedure has to be implemented.

In the following we will provide a distributed k-means algorithm that relies heavily on assumption A1; in the next section we will provide a distributed procedure to overcome such a limitation.

4.1 Proposed Algorithm under Assumption A1

Let us now describe the proposed Algorithm 1, for which a flow chart is provided in Figure 2. In the following we will assume that Algorithm 1 is executed synchronously by each node and that the nodes exchange the necessary information with their neighbors or with a subset of the nodes in their neighborhood.

Let average-consensus$(z_i, \mathcal{N}_i, T_{max}, \epsilon)$ and max-consensus$(z_i, \mathcal{N}_i, T_{max})$ be the average-consensus and max-consensus distributed algorithms, respectively, where z_i is the (possibly vectorial) initial condition for node i, $\mathcal{N}_i$ is the neighborhood of node i and T_{max}, ϵ are parameters used to terminate the consensus and are the same for each node.

The initialization phase of the algorithm is as follows. In order to choose the initial random centers, a leader is elected and is in charge to choose the centers.

Supposing that each node has a unique identifier i, the nodes challenge on their identifiers via max-consensus. The node i^* whose identifier is the greatest is elected as leader and chooses the centers $c_{i^*h}(0)$, for $h = 1, \ldots, k$. The other nodes choose $c_{ih}(0) = 0$, for $h = 1, \ldots, k$.

After the initialization phase, the main cycle is iterated M times; at every step t it is composed of the following phases:

1) **Center Propagation**: each node selects a vector $z_i^t(0) \in \mathbb{R}^{kd}$, and a max-consensus procedure is executed, so that the resulting $z_i^t(T_{max}) \in \mathbb{R}^{kd}$ is the stack vector containing all the k centers.
2) **Nearest Center Choice**: each node chooses the index $k_i^*(t)$ of the nearest among the current centers and updates a vector $\mu_i(t) \in \mathbb{R}^d$ that represents its choice.
3) **Nearest Center Choice Broadcast**: each node provides its choice of $k_i^*(t)$ to the nodes in its neighborhood $\mathcal{N}_i$.
4) **Cluster Neighborhood Choice**: each node selects, among its neighbors in $\mathcal{N}_i$, those nodes j that share the same choice of $k_i^*(t)$. As a result a new neighborhood $\mathcal{N}_i^c(t) \subseteq \mathcal{N}_i$ is obtained.
5) **Center Refinement**: The new neighborhood represents a subgraph $\mathcal{G}^c(t)$ of $\mathcal{G}$, where each node is connected only to nodes with the same center choice. During this phase a an average consensus on the nodes'observations x_i over $\mathcal{G}^c(t)$ is performed in order to obtain $c_{ik_i^*}(t)$.

Proposition 1: Under assumption A1, if graph $\mathcal{G}$ is connected Algorithm 1 solves the k-means distributed problem exactly. Moreover the computational complexity is $O(kdnT_{max}M)$.

Proof: If $\mathcal{G}$ is connected then an average or max consensus procedure (possibly with vectorial state for each node) over $\mathcal{G}$ will succeed for sufficiently high T_{max} and sufficiently small ϵ.

Let us consider a generic step t of the main cycle. During the center propagation phase each node obtains via max-consensus a vector containing the current k centers (for the first step the random centers selected

Algorithm 1: Distributed k-means algorithm: i-th node

Data: M, T_{max}, ϵ,$\mathcal{N}_i$
Result: $c_{i1}(M), \ldots, c_{ik}(M)$, $\mathcal{N}_i^c(M)$, $k_i^*(M)$

/* Initialization */
$\mu_{ij} = 0$ for all $j = 1, \ldots, k$;
$c_{ij}(0) = 0$ for all $j = 1, \ldots, k$;
$\mathcal{N}_i^c(0) = \mathcal{N}_i$;
$k_i^*(0) = 0$;

/* Leader Election */
$i^* =$ max-consensus $(i, \mathcal{N}_i, T_{max})$;
if $i = i^*$ **then**
  node i chooses random centers $c_{ij}(0) = 0$ for all $j = 1, \ldots, k$;
end

/* Main Cycle */
for t= 1 **to** M **do**
  /* Center Propagation */

$$z_i^t(0) = \begin{cases} 0 & \text{if } t = 1 \text{ and } i \neq i^* \\ [c_{i1}(0), \ldots, c_{ik}(0)]^T & \text{if } t = 1 \text{ and } i = i^* \\ \mu_i(t) \otimes c_{ik_i^*(t)}(t) & \text{if } t > 1 \end{cases}$$

  $z_i^t(T_{max}) =$ max-consensus $(z_i^t(0), \mathcal{N}_i, T_{max})$;
  /* Nearest Center Choice */

$$\mu_{ij}(t) = \begin{cases} 1 & \text{if } j = \arg\min_{h=1}^k |c_{ih}(t) - x_i| \\ 0 & \text{else} \end{cases}$$

  /* Nearest Center Choice Broadcast */
  each node provides $k_i^*(t) = \arg\min_{h=1}^k |c_{ih}(t) - x_i|$ to the neighbors in $\mathcal{N}_i$;
  /* Cluster Neighborhood Choice */
  each node selects $\mathcal{N}_i^c(t) \subseteq \mathcal{N}_i$ based on $k_j^*(t)$ for each $j \in \mathcal{N}_i$;
  /* Center Refinement */
  $c_{ik_i^*(t)}(t) =$ average-consensus$(x_i, \mathcal{N}_i^c, T_{max}, \epsilon)$;
end
return $c_{i1}(M), \ldots, c_{ik}(M)$, $\mathcal{N}_i^c(M)$, $k_i^*(M)$;

by the leader are obtained). As a result of the Nearest Center Choice, Nearest Center Choice Broadcast and Cluster Neighborhood Choice phases, each node selects a neighborhood $\mathcal{N}_i^c(t) \subseteq \mathcal{N}_i$.

During the Center Refinement phase an average consensus on the nodes'observations x_i over $\mathcal{G}^c(t)$ is performed in order to obtain $c_{ih_i^*}(t)$. Under assumption A1, an average consensus on the nodes'observations x_i over $\mathcal{G}^c(t)$ is such that the state of each node converges to the average of the observations of the nodes in its cluster, hence to the centroid of the cluster.

For the computational complexity of the proposed algorithm note the most expensive operation in the main cycle is the center propagation, where a vectorial max-consensuses is executed with initial values of size kd, hence the center propagation has a complexity $O(kdnT_{max})$. As a result the algorithm has a complexity $O(kdnT_{max}M)$.

□

Let us conclude the section with the following remark

Remark 3: According to Remark 1, there is the risk that some centers are not chosen by the agents. In this case the proposed distributed k-means algorithm overwrites each of these centers with a vector of zeros.

This is due to the particular choice of $z_i^t(0)$; in fact, besides the leader at the first time step, each node chooses just a center and the vector $z_i^t(0)$ is filled with zeros for the not chosen centers. If a center is not selected by any node, then the corresponding components of $z_i^t(0)$ will be zero for each node.

Such an issue is however easy to fix: if a center c_j is not chosen, then each node will have a vector $z_i^t(0)$ such that $c_j = 0$. In this case the leader may select another random center and a propagation can be implemented via max consensus.

5 Coping with Disconnected Clusters

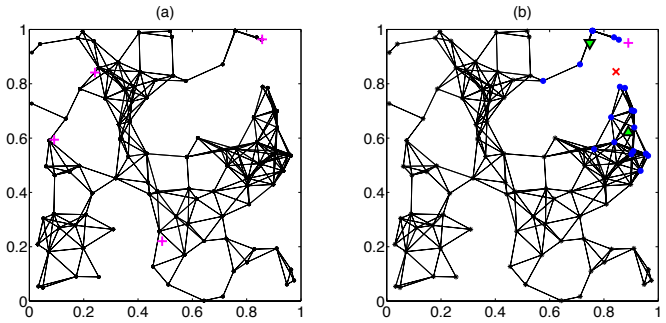

Fig. 3. Example of fail of the distributed k-means algorithm when assumption A1 is not verified.

The above algorithm heavily relies on assumption **A1** for the correct refinement of the centers. While the assumption tends to be verified as the communication threshold ρ grows, for small values of ρ it might not be verified, as shown by the example in Figure 3. Specifically, Figure 3.(a) shows an instance of the problem for $n = 100$ and $\rho = 0.15$, where the positions of the nodes are in $[0, 1]^2$. The purple crosses in Figure 3.(a) represent the initial random centers selected by the leader. Figure 3.(b) reports in blue the nodes that are nearest to the upper-left most center of Figure 3.(a). The red x represents the exact centroid of the nodes in blue; however note that the nodes are decomposed in 2 subgraphs. As a result, the refinement phase produces a different result for the two disconnected components (the green triangles facing upwards and downwards), and the center transmission phase generates an incorrect centroid (i.e., the maximum for each coordinate), represented by the purple cross in Figure 3.(b).

The above issue arises especially when the measure is not only position, i.e., for measure and hybrid clustering. In the following we will provide an although computationally expensive procedure to obtain the correct centroid.

5.1 Centroid Computation for Disconnected Clusters

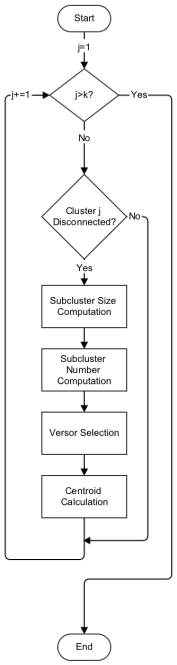

Fig. 4. Flow-chart of the proposed algorithm for the centroid computation when Assumption A1 does not hold true.

Figure 4 provides a flow chart of the proposed distributed centroid computation algorithm (Algorithm 5). Specifically, each node belonging to a sub-cluster computes the sub-cluster size using Algorithm 2 over $\mathcal{G}^c(t)$, the number of sub-clusters (Algorithm 3), and a unit versor e of the same size of the number of subclusters describing the fact that the node i belongs to a given subcluster (Algorithm 4). Then a vectorial max-consensus

procedure is applied and each node performs a weighted average of the centroids of each sub-cluster, where the weights are the sub-cluster sizes. As a result, the exact centroid of the cluster is obtained for each node in the subcluster.

Let us first provide a distributed algorithm to calculate the number of nodes in the network.

Algorithm 2: Distributed Network Size Computation: i-th node

Data: $T_{max}, \mathcal{N}_i$
Result: Network Size NS_i

```
/* Initialization                                  */
TemporaryID_i = i;
NS_i = 0;
while TemporaryID_i > 0 do
    /* Execute a max consensus on node
       TemporaryID_i                               */
    ConsensusID_i =
    max-consensus(TemporaryID_i, N_i, T_max);
    /* If the node is the leader set its
       TemporaryID to 0                            */
    if ConsensusID_i == i then
        TemporaryID_i = 0;
    end
    /* The node adds 1 to the network
       size counter                                */
    NS_i += 1;
end
return NS_i;
```

Proposition 2: If the graph $\mathcal{G}$ is connected and $T_{max} \geq n$, then Algorithm 2, executed simultaneously by all the n nodes is such that at the end of the procedure for all i $\text{NS}_i = n$. Moreover the computational complexity is $O(n^2 T_{max})$.

Proof: If A is connected then a max consensus protocol converges to the max of the initial conditions in no more than n steps. Within Algorithm 2 the max consensus is iterated exactly n times, hence at the end of the procedure $\text{NS}_i = n$ for all nodes.

The computational complexity of the max consensus algorithm is $O(nT_{max})$ and it is iterated n times, hence the complexity of Algorithm 2 is $O(n^2 T_{max})$. □

The above algorithm can be also adopted to compute the size of the subclusters.

Corollary 1: If Algorithm 2 is applied considering the graph $\mathcal{G}^c(t)$, then the output for the i-th node is equal to the size $\text{SCS}_{ij}(t)$ of the subcluster $s_{hj}(t) \in \sigma(S_j(t))$ the i-th node belongs to at time step t.

Proof: If the algorithm is executed over $\mathcal{G}^c(t)$ and at time step t the ith node belongs to a subcluster $s_{hj}(t)$ of cluster $S_j(t)$, then such cluster is a disconnected component of $\mathcal{G}^c(t)$ and by Proposition 2 the size of such component is obtained. □

In the following we will refer to this modification of Algorithm 2 as the Distributed Subcluster Size Computation Algorithm for i-th node.

Let us now provide an algorithm to calculate the number of subclusters that compose cluster a $S_j(t)$. To this end let $\phi_{ij}(t) = 1$ if node i belongs to cluster j at time step t and $\phi_{ij}(t) = 0$ otherwise.

Algorithm 3: Distributed Subcluster Number Computation: i-th node, cluster $S_j(t)$

Data: $T_{max}, \mathcal{N}_i, \mathcal{N}_i^c(t)$, $\phi_{ij}(t)$
Result: Subcluster Number $\text{SCN}_{ij}(t)$

```
/* Initialization                                  */
if φ_ij(t) == 1 then
    /* The node chooses the maximum ID
       in the subcluster S_j                       */
    TemporaryID_i = max-consensus(i, N_i^c(t), T_max);
else
    TemporaryID_i = 0;
end
SCN_ij(t) = 0;
while TemporaryID_i > 0 do
    /* Execute a max consensus on
       current node IDs                            */
    ConsensusID_i =
    max-consensus(TemporaryID_i, N_i, T_max);
    /* If the node is in the same
       subcluster of the leader                    */
    /* the node sets its ID to 0                   */
    if ConsensusID_i == TemporaryID_i then
        TemporaryID_i = 0;
    end
    /* The node adds 1 to the subcluster
       number counter                              */
    SCN_ij(t) += 1;
end
return SCN_ij(t);
```

Proposition 3: If the graph $\mathcal{G}$ is connected and $T_{max} \geq n$, then Algorithm 3, executed simultaneously by all the n nodes at a step t with respect to cluster $S_j(t)$ is such that $\text{SCN}_{ij}(t)$ is equal to the number of subclusters that compose $S_j(t)$. Moreover the computational complexity of the algorithm is $O(|\sigma(S_j(t))| n T_{max})$

Proof: If $\mathcal{G}$ is connected then a max consensus protocol converges to the max of the initial conditions in no more than n steps. Within Algorithm 3 the max consensus is iterated exactly $|\sigma(S_j(t))|$ times, hence at the end of the procedure $SCN_j(t) = |\sigma(S_j(t))|$.

Note that a consensus algorithm is iterated $|\sigma(S_j(t))|$ times, hence the computational complexity of the algorithm is $O(|\sigma(S_j(t))| n T_{max})$. □

Applying the above algorithms, each node is able to calculate the size of the network, the number of subclusters that compose the cluster the node belongs to, and the size of such subcluster.

Each node is also able to obtain the centroid of the subcluster they belong to, by executing an average consensus algorithm on their observations over the graph $\mathcal{G}^c(t)$.

Such information can be used by each node to construct an appropriate vector to be used for a vectorial max consensus procedure aimed at letting each node now the centroid and the size of each subcluster. Such a consensus procedure allows each node to determine via a weighted average the correct centroid of the cluster it belongs to.

In order to to this, we need to let each node in $S_j(t)$ decide to which subcluster it belongs. This can be modeled as the selection of a $|\sigma(S_j(t))| \times 1$ unit versor $e_i(t)$ such that $e_i(t)$ is selected by and only by the nodes belonging to the same subcluster. The procedure is outlined in Algorithm 4.

Algorithm 4: Distributed Versor Selection: i-th node, cluster $S_j(t)$

Data: $T_{max}, \mathcal{N}_i, \mathcal{N}_i^c(t)$, $\phi_{ij}(t)$
Result: versor $e_i(t)$

/* Initialization */
if $\phi_{ij}(t) == 1$ **then**
  $\text{TemporaryID}_i = \text{max-consensus}(i, \mathcal{N}_i^c(t), T_{max})$;
  /* The node chooses the first unit versor */
  $e_i(t) = e^1$;
else
  /* The node chooses a zero vector */
  $e_i(t) = 0$;
  $\text{TemporaryID}_i = 0$;
end
$h = 1$;
while $h \leq n$ **do**
  /* Execute a vectorial max consensus on $\text{TemporaryID}_i e$ */
  $\text{VectorialConsensusID}_i = \text{max-consensus}(\text{TemporaryID}_i e, \mathcal{N}^i, T_{max})$;
  /* If the node is in S_j and the h-th component of $\text{VectorialConsensusID}_i$ changes the node updates its versor $e_i(t)$ */
  if *TemporaryID*$_i > 0$ *and* *TemporaryID*$_i \neq$ *VectorialConsensusID*$_{ih}$ **then**
    $h+ = 1$;
    $e_i(t) = e^h$;
  end
end
return e(t);

Proposition 4: If the graph described by A is connected and $T_{max} \geq n$, then Algorithm 4, executed simultaneously by all the n nodes with respect to cluster $S_j(t)$ is such that for all nodes i in a subcluster $s_{hj} \in S_j$ chose the same versor $e_j(t) = e^h$ and there is no node $f \notin s_{hj}$ that chooses e^h.

Moreover the computational complexity of the algorithm is $O(|\sigma(S_j(t))|n^2 T_{max})$.

Proof: Since the nodes initially perform the vectorial max consensus with initial opinion equal to e^1 multiplied by the value of their subcluster leader, the consensus vector is equal to e^1 multiplied by the maximum value of the subcluster leaders. Therefore the nodes in such subcluster keep their choice of $e_i(t)$ while the others select e^2. When the main cycle is executed again the result of the vectorial max consensus is be a vector in the form

$$\text{VectorialConsensusID}_i = ID_1(t)e^1 + ID_2(t)e^2$$

where $ID_h(t)$ is the h-th greatest index among the subcluster leaders at step t. Again, the nodes in the subclusters containing node $ID_1(t)$ and $ID_2(t)$ will keep their choice, while the others will select e^3. Iterating the procedure n times the final output of the vectorial max consensus is in the form

$$\text{VectorialConsensusID}_i = \sum_{j=1}^{n} ID_j(t)e^j$$

Hence, for all $s_{hj} \in \sigma(S_j)$, each node in s_{hj} selects the correct versor.

Note that the vectors used for consensus are of size $|\sigma(S_j(t))|$ hence the vectorial consensus has a complexity $O(|\sigma(S_j(t))|nT_{max})$. In the worst case, such a procedure is executed n times hence the complexity is $O(|\sigma(S_j(t))|n^2 T_{max})$. □

Let us provide a distributed algorithm for the calculation of the centroid of a disconnected cluster:

Algorithm 5: Distributed Centroid Calculation: i-th node, cluster $S_j(t)$

Data: $\text{SCS}_{ij}(t), \text{SCN}_{ij}(t), e_i(t), T_{max}, \mathcal{N}_i, \mathcal{N}_i^c(t)$, $\phi_{ij}(t), c_i^c(t), [c_1(t), \dots c_{NS}(t)]^T$
Result: Centroid $c_j^*(t)$

if $c_i(t) \neq c_i^c(t)$ **then**
  /* Execute a vectorial max consensus on $c_i^c(t)$ and $\text{SCS}_i(t)$ */
  $[c_1^d(t), \text{SCS}_{i1}(t), \dots c_n^d(t), \text{SCS}_{i,\text{SCN}_{ij(t)}}(t)]^T =$
  $\text{max-consensus}(e_i(t) \otimes \begin{bmatrix} c_i^c(t) \\ \text{SCS}_{ij}(t) \end{bmatrix}, \mathcal{N}_i, T_{max})$;
  /* The centroid is given by the weighted average of subcluster centroids using the $SCS_{ij}(t)$ as weights */

$$c_j^*(t) = \frac{\sum_{j=1}^{\text{SCN}_{ij(t)}} \text{SCS}_{ij}(t) c_j^d(t)}{\sum_{j=1}^{NS} \text{SCS}_{ij}(t)};$$

end
return $c_j^*(t)$;

Proposition 5: If graph $\mathcal{G}$ is connected and $T_{max} > n$ and the cluster $S_j(t)$ is disconnected, then the output of Algorithm 5 is the correct centroid of cluster $S_j(t)$. Moreover the computational complexity of the algorithm is

$$O((d+1)|\sigma(S_j(t))|nT_{max})$$

Proof: The correctness of the centroid follows from propositions 1–4.

The computational complexity follows from the fact that the size of the vectorial max consensus is

$$(d+1)|\sigma(S_j(t))|$$

□

To conclude the section, let us discuss the computational complexity of the centroid computation and of the distributed k-means algorithm when assumption A1 does not hold true.

Proposition 6: The computational complexity of the centroid computation algorithm for all clusters is

$$O(k\max\{(d+1),n\}|\sigma(S_j(t))|nT_{max})$$

Proof: In the worst case, the procedure in Algorithm 5 has to be performed for each cluster.

For each cluster $S_j(t)$ the Versor Selection algorithm has greater computational complexity than Subcluster Size and Subcluster Number computation procedures; the computational complexity for a given cluster is therefore:

$$O(\max\{(d+1),n\}|\sigma(S_j(t))|nT_{max})$$

Considering all the k clusters the proof is complete. □

Proposition 7: The computational complexity of the distributed k-means algorithm when assumption $A1$ does not hold true is

$$O(k\max\{(d+1),n\}|\sigma(S_j(t))|nT_{max}M)$$

Proof: The proposition is proven noting that, when assumption $A1$ does not hold true, the most expensive operation in main cycle the disconnected centroid calculation. □

Remark 4: Since $|\sigma(S_j(t))| \propto n$, if $d > n$, the lack of assumption A1 causes an increase of computational complexity of a factor n. If conversely $d < n$, then the computational complexity is increased by a factor $\frac{n^2}{d}$.

6 Simulation Results

In this section we provide an example of application of the proposed distributed k-means algorithm, with respect to a position clustering, a measure clustering and a hybrid clustering.

We consider a situation where $n = 30$ nodes are embedded in $[0,1]^2$, $\rho = 0.25$ for all nodes and $k = 3$.

Figure 5 shows an example of position clustering, where the value of each $x_i(0)$ is a 2×1 vector containing the node position. Specifically the upper plots in the figure show the clustering obtained with respect to the spatial distribution of the nodes (the different colors and shapes represent different clusters, while the centers are reported with purple crosses), while the lower plot shows the evolution of the functional D during the execution of the algorithm.

Figure 6 shows the result of a measure clustering where $x_i(0)$ is a scalar value of each node and is obtained from the map in the left-lowermost plot in the figure, depending on the position of the nodes. Specifically, $x_i(0)$ assumes values between 0 (blue region) and 1 (red region); note that there are several peaks of different magnitude in different positions of the map. The upper plots in the figure show the clustering obtained from a spatial point of view (the different colors and shapes represent different clusters); in this case the clustering depends only on the measured value and the nodes do not show a clear spatial clustering pattern.

The lower central plot shows the clustering obtained from a measure point of view. The abscissa represents the measure and the ordinate represents the step of the distributed k-means algorithm. Again, different colors and shapes represent different clusters, while the centers are reported with purple crosses. According to the plot, the algorithm is indeed effective in partitioning the nodes according to the measured value. The lower rightmost plot shows the results in terms of minimization of the function D.

Note that, while in the previous case there was a chance to have connected clusters, in this case the assumption A1 is not verified and hence the proposed centroid calculation algorithm is required.

For the hybrid clustering we choose for each node i a 3×1 vector $x_i(0)$ containing both the position and the scalar measure of the previous example. In this way the nodes will be partitioned according to both the position and the measure. Since we chose unitary weight for the two coordinates and for the measure, as a result the clustering is much more similar to a position clustering than a measure clustering. Of course other choices for the composition of $x_i(0)$ are possible, yielding to different results. Note that, also in this case the proposed centroid calculation algorithm is required for the correct functioning of the distributed k-means.

Note further that, as shown by the center lowermost figure, one of the initial centers is selected with a measure of about 0.85 while the nodes have very distant values: the center is therefore not selected by any node, and as a result of the proposed schema the new center becomes the point $[0,0,0]^T$. When a new iteration of the algorithm is performed, the center in zero attracts some of the nodes; hence the proposed algorithm is also robust with respect to this type of singular case.

7 Conclusions and Future Work

In this paper we provide a fully distributed implementation of the k-means clustering algorithm considering a

set of nodes each equipped with a piece of information or measure. The proposed algorithm, in its simplest fashion, is unfortunately very likely to fail when the partitions do not have a direct relation with the topological clusters, resulting in sparse groups of agents. To solve this issue, an although expensive distributed procedure is provided.

REFERENCES

[1] M. K. Ng, "K-means-type algorithms on distributed memory computer," Int. J. of High Speed Computing, pp. 75Ð91, Jun. 2000.

[2] I. Dhillon and D. Modha, A data-clustering algorithm on distributed memory multiprocessors," in Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence, 2000, pp. 245Ð260.

[3] Y. Zhang, Z. Xiong, J. Mao, and L. Ou, "The study of parallel K- means algorithm," in Proc. 6th World Congress on Intelligent Control and Automation, vol. 2, Dalian, China, Jun. 21-23, 2006, pp. 5868Ð5871.

[4] Vendramin, L., Campello, R. J. G. B., Coletta, L. F., and Hruschka, E. R. (2011, January). Distributed Fuzzy Clustering with Automatic Detection of the Number of Clusters. In International Symposium on Distributed Computing and Artificial Intelligence (pp. 133-140). Springer Berlin Heidelberg. ISO 690

[5] Patel, S., Patel, V., and Jinwala, D. (2013). Privacy Preserving Distributed K-Means Clustering in Malicious Model Using Zero Knowledge Proof. In Distributed Computing and Internet Technology (pp. 420-431). Springer Berlin Heidelberg.

[6] Monti, S., Tamayo, P., Mesirov, J., and Golub, T. (2003). Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. Machine learning, 52(1-2), 91-118.

[7] Topchy, A. P., Law, M. H., Jain, A. K., and Fred, A. L. (2004, November). Analysis of consensus partition in cluster ensemble. In Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on (pp. 225-232). IEEE.

[8] Goder, A., and Filkov, V. (2008, June). Consensus Clustering Algorithms: Comparison and Refinement. In ALENEX (Vol. 8, pp. 109-117).

[9] W. Kowalczyk and N. Vlassis, "Newscast EM," in Proc. of Advances in Neural Info. Processing Systems. Vancouver, B.C., Canada: MIT Press, Dec. 13-18, 2005, pp. 713Ð720.

[10] D. Gu, "Distributed EM algorithm for Gaussian mixtures in sensor networks," IEEE Tran. on Neural Networks, vol. 19, no. 7, pp. 1154Ð 1166, Jul. 2008.

[11] J. Wolfe, A. Haghighi, and D. Klein, "Fully distributed EM for very large datasets," in Proc. of 25th Intl. Conference on Machine Learning, Helsinki, Finland, Jul. 6-8, 2008, pp. 1184Ð1191.

[12] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed expectation-maximization algorithm for density estimation and classification using wireless sensor networks," in Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing, Las Vegas, NV, Mar. 30-Apr. 4, 2008.

[13] Forero, P. A., Cano, A., and Giannakis, G. B. (2008, May). Consensus-based k-means algorithm for distributed learning using wireless sensor networks. In Proc. Workshop on Sensors, Signal and Info. Process., Sedona, AZ.

[14] J. MacQueen, "Some methods for classification and analysis of multivariate observations". In: 5th Berkeley symposium on mathematics, statistics and probability, pp. 281Ð296, 1967.

[15] P. Brucker, "On the complexity of clustering problems". Optimization and operations research (Proc. Workshop, Univ. Bonn, Bonn, 1977) (Lecture Notes in Economics and Mathematical Systems 157 ed.). Berlin: Springer

[16] R. Olfati-Saber and R. M. Murray, Consensus Problems in Networks of nodes with Switching Topology and Time-Delays, *IEEE Trans. on Autom. Control*, Vol. 49, No. 9, pp. 1520–1533, 2004.

[17] R. Olfati-Saber, J. A. Fax and R. M. Murray, Consensus and cooperation in networked multi-node systems, *Proceedings of the IEEE*, vol 95 (1), pp. 215–233, 2007.

[18] L. Wang and F. Xiao , Finite-Time Consensus Problems for Networks of Dynamic nodes, *IEEE Trans. on Autom. Control*, Vol. 55, n. 4, pp. 950 - 955, 2010.

[19] Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs (1988)

[20] Maulik, U.; Bandyopadhyay, S., "Performance evaluation of some clustering algorithms and validity indices," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.24, no.12, pp.1650,1654, Dec 2002

[21] D.L. Davies and D.W. Bouldin, "A Cluster Separation Measure," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 1, pp. 224-227, 1979.

[22] J.C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters," J. Cybernetics, vol. 3, pp. 32-57, 1973.

[23] X.L. Xie and G. Beni, "A Validity Measure for Fuzzy Clustering," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 13, pp. 841-847, 1991.

[24] C.N. Hadjicostis and A. Rikos, "Distributed strategies for balancing a weighted digraph", 20th Mediterranean Conference on Control Automation, 2012.

[25] B. Gharesifard and J. Cortes, "Distributed strategies for generating weight-balanced and doubly stochastic digraphs", European Journal of Control, 2012.

[26] K. Cai and H. Ishii, "Average consensus on arbitrary strongly connected digraphs with dynamic topologies", American Control Conference, 2012.

[27] R. D. Nowak, "Distributed EM algorithms for density estimation and clustering in sensor networks," IEEE Trans. on Signal Processing, Vol. 51, no. 8, pp. 2245Ð2253, Aug. 2003.

[28] Chen, W. P., Hou, J. C., and Sha, L. (2004). Dynamic clustering for acoustic target tracking in wireless sensor networks. Mobile Computing, IEEE Transactions on, 3(3), 258-271.

[29] Younis, O., and Fahmy, S. (2004). HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. Mobile Computing, IEEE Transactions on, 3(4), 366-379.

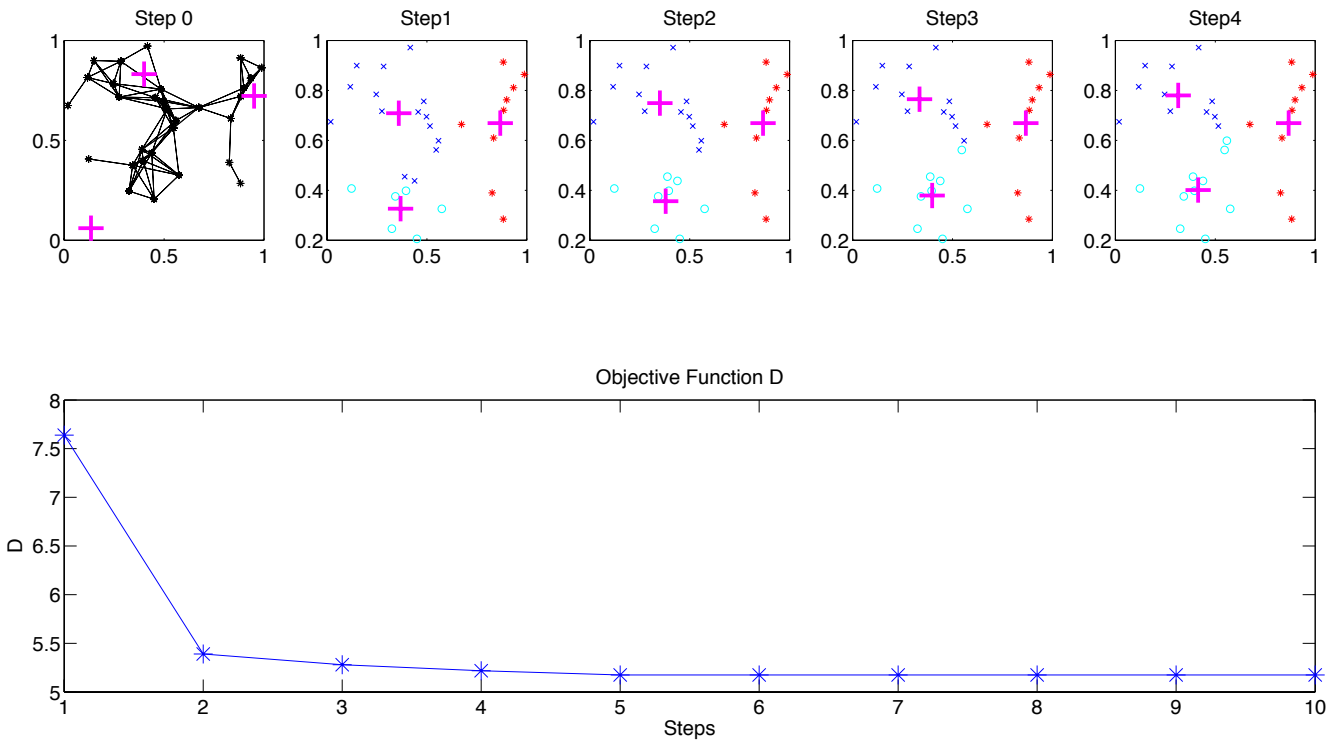

Fig. 5. Example of Position Clustering.

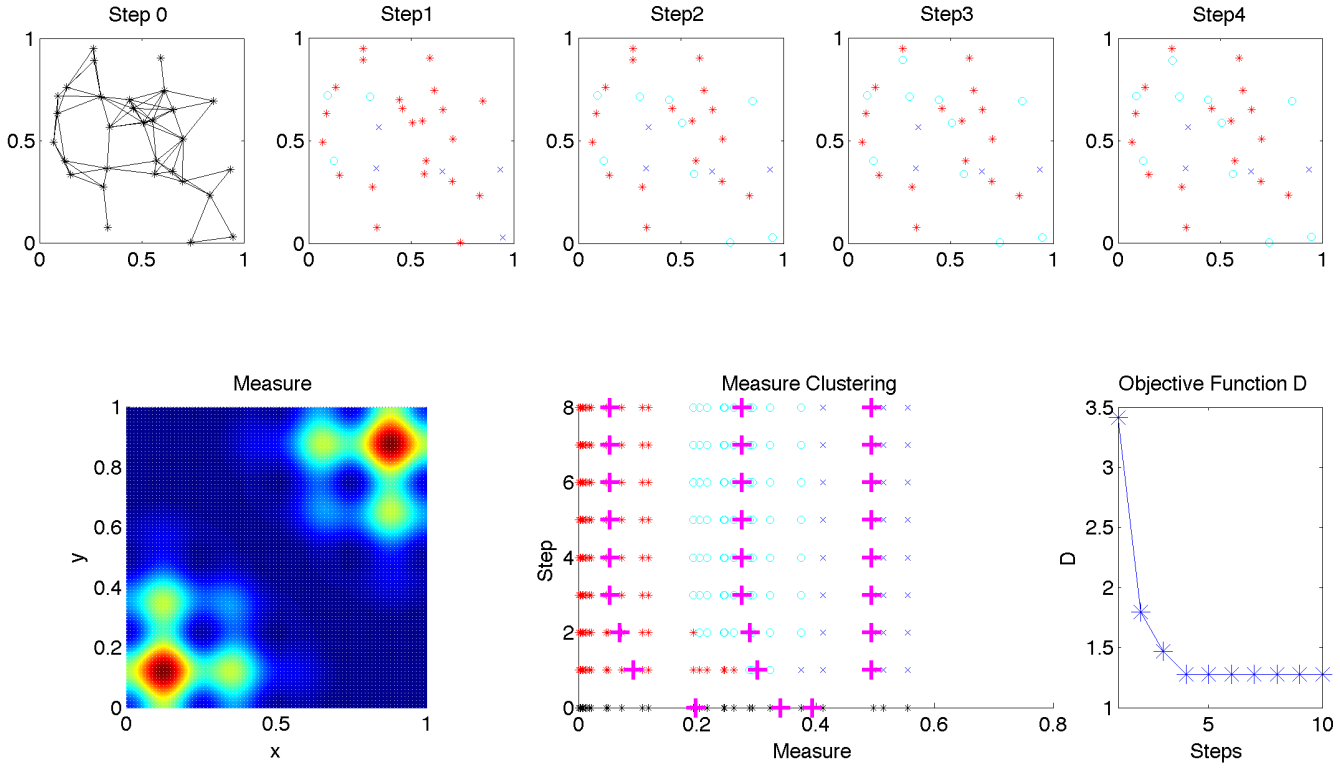

Fig. 6. Example of Measure Clustering.

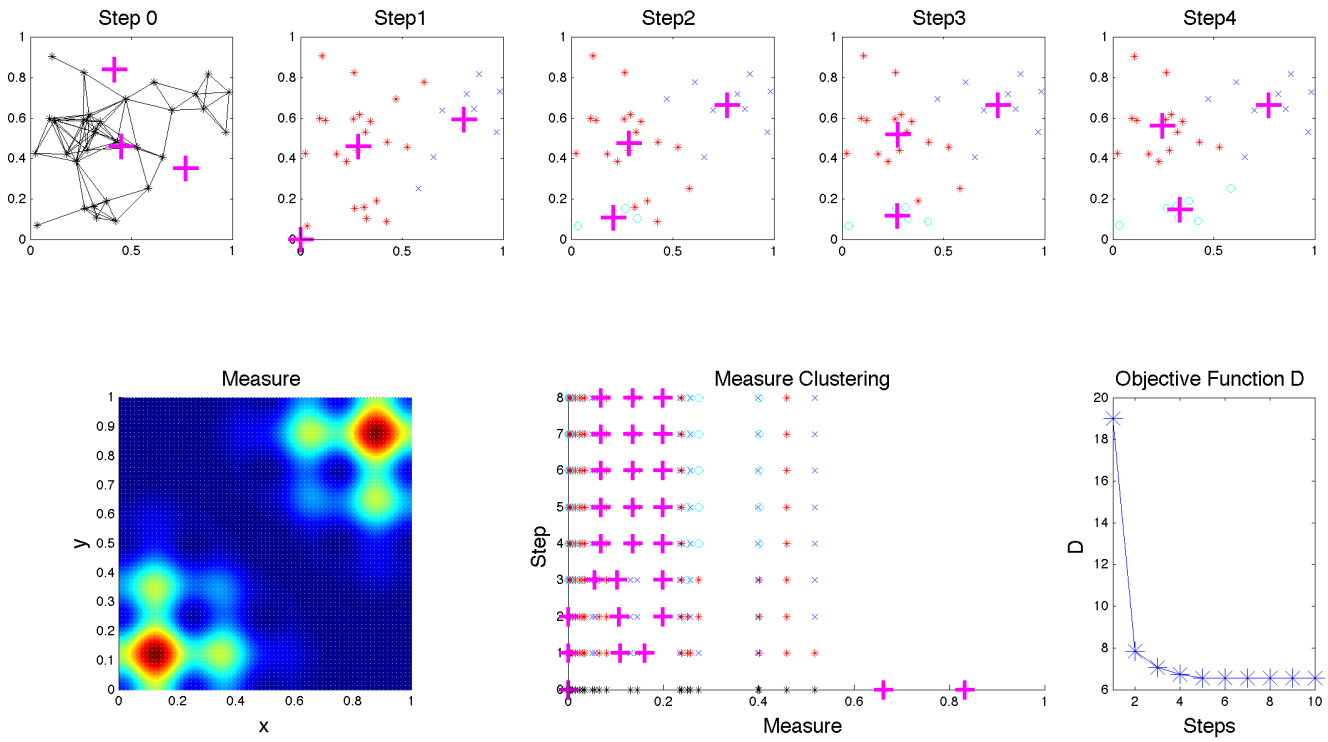

Fig. 7. Example of Hybrid Clustering.