Dynamic Distributed Clustering in Wireless Sensor Networks via Voronoi Tessellation Control*

A. Pietrabissa[1,2,3], F. Liberati[2,4]

Abstract

This paper present two dynamic and distributed clustering algorithms for Wireless Sensor Networks (WSN). Clustering approaches are used in WSNs to improve the network lifetime and scalability by balancing the workload among the clusters. Each cluster is managed by a cluster head (CH) node. The first algorithm requires the CH nodes to be mobile: by dynamically varying the CH node positions, the algorithm is proved to converge to a specific partition of the mission area, the generalized Voronoi Tessellation, in which the loads of the CH nodes are balanced. Conversely, if the CH nodes are fixed, a weighted Voronoi clustering approach is proposed with the same load balancing objective: a reinforcement learning approach is used to dynamically vary the mission space partition by controlling the weights of the Voronoi regions. Numerical simulations are provided to validate the approaches.

Keywords: Voronoi partitioning, wireless sensor networks, Markov decision processes, reinforcement learning.

1 Introduction

Wireless Sensor Networks (WSNs) are basically composed by sensor nodes, spread over the monitored area in order to collect the measures of interest, and by data sink nodes, which collect the data transmitted by the sensor nodes, process and aggregate the received measures and convey the elaborated data to a remote data centre. The management of WSNs is a widely-researched topic in the literature, since an efficient operation of WSNs is relevant to many applications, such as, e.g., environmental sensing [1], critical infrastructure monitoring [2], health care monitoring [3], process control monitoring [4], air quality monitoring [5], networked control systems [6]. One of the main challenges in WSN management is the energy consumption of the nodes, which limits the WSN lifetime. A common strategy to maximize the WSN lifetime is to group the WSN nodes into clusters. Data

* This work is partly supported by the by the European FP7 project SWIPE (Space WIreless sensor networks for Planetary Exploration), funded under Grant Agreement n° 312826.

[1] Dept. Of Computer, Control and Management Engineering of the Univ. of Rome "Sapienza", Via Ariosto 25, 00162 Rome, Italy. E-mail: {pietrabissa; liberati}@dis.uniroma1.it

[2] Consortium for the Research in Automation and Telecommunication (CRAT), via Nicotera 25, 00195, Rome, Italy.

[3] Corresponding author; phone: +390677274040, fax: +39064033, email: pietrabissa@dis.uniroma1.it

[4] SMARTEST Research Center, eCampus University, Via Isimbardi 10 - 22060 Novedrate (CO), Italy.

processing operations, such as data filtering, data aggregation and data fusion, are carried out in each cluster to reduce the load of data transmitted over the WSN, and, consequently, the overall energy consumption. The node responsible for gathering and processing the cluster data is the so-called cluster head (CH) node.

WSN clustering can be classified according to many characteristics (see, e.g., [7] and the references therein). In this paper, we are interested in dynamic clustering algorithms with a fixed number of CH nodes, with either fixed or mobile CH and sensor nodes, aimed at balancing the load among the CH nodes during the WSN lifetime. The paper does not take into account other aspects of WSNs which depend on the specific implementation, such as: the presence of algorithms to elect the CH nodes among the sensor nodes, the characteristics of the communication among the nodes and of the implemented routing algorithm, the data processing algorithms within the CH nodes. For the same reason, only the load balancing objective is considered; for instance, to maximize the WSN lifetime the objective might be to balance of the leftover energy among the CH nodes (and not their load only), but the energy consumption depends on the characteristics of the specific WSN implementation, such as the fact that CH nodes act as data relays or not, the energy occurring to move in case of mobile CH nodes, the data processing algorithms within the CH nodes and so on [8].

Generally, clustering is obtained by defining a region of competence for each CH. The first algorithm proposed in this paper deals with mobile CH nodes. We assume that the sensor nodes are associated to the nearest CH node; the objective is then to move the CH nodes in such a way that the set of CH nodes induce a balanced partition of the mission area. A discrete-time, distributed dynamic clustering algorithm is proposed for mobile WSNs, which lets the partition induced by the CH nodes converge to the so-called generalized Centroidal Voronoi Tessellation (CVT), with favorable characteristics in terms of load balancing. The control actions of the algorithm are such that, if the environment is stationary, the positions of the CH nodes are proved to converge to the CVT.

If the CH nodes are fixed, a dynamic clustering algorithm is proposed, in which the CH node regions are varied by opportunely controlling the weights of a weighted centroidal Voronoi tessellation. The system is modelled as a Markov decision process (MDP) and the control actions (i.e., the way the weights are varied) are computed by a reinforcement learning (RL) algorithm with the objective of minimizing a cost representing the load unbalancing among the CH nodes.

The paper is organized as follows: Section 2 presents a literature review and outlines the paper contributions; Section 3 summarized the basic concepts and definitions of Voronoi partitioning, MDP and RL; the proposed dynamic CVT and weighted Voronoi algorithms are analyzed in Sections 4 and 5, respectively; Section 6 collects the simulation results; finally, Section 7 draws the conclusions and outlines the future works.

2 Related works and paper contributions

2.1 Related works

Dynamically modifying the *mission space partitioning* (i.e., the composition of the WSN clusters) can be exploited to ensure balancing and prolong WSN operations. The main degrees of freedom for WSN balancing are dynamic clustering, CH node migration across the mission space and CH node *role* migration among nodes (i.e., dynamic CH node election). A rich variety of algorithms have been proposed for each of these topics, often in combined forms. In this paper, we investigate CH node migration and dynamic clustering concepts. The reader interested in CH node election algorithms is referred to [7,9–11], where the most popular algorithms are explained, such as: LEACH [12], which introduces a random election mechanism, HEED [13], an extension of LEACH including among the election parameters also the residual energy of the node, EECS [14], a LEACH-like algorithm for single-hop WSNs, which introduces a distance-based metric to balance load among cluster head, and many others. EECS is interesting in that it promotes *unequal clustering*, meaning that the clusters far away from the WSN sink (or gateway) node tend to be smaller compared to closer ones (CH nodes far from the sink node spend more energy to transmit data, in single-hop WSNs). As a result [14], EECS manages to prolong network lifetime (the time until the first node runs out of energy) compared to LEACH and HEED. LEACH extensions have been designed to specifically tackle nodes mobility and the associated issue of increased packet loss. In LEACH-M [15], a mobility metric is proposed. In [16], the CH election is based on node residual energy and mobility; node clustering is then based on the connection time with the CH, the distance from the CH, the node residual energy and the node degree of the CH. Another recent extension of LEACH for mobile networks is LEACH-MF [17], which enhances the fuzzy inference system proposed in [18] to improve the CH selection phase for mobile scenarios, by taking into account the residual energy and the moving speed.

Rather than dealing with CH node role migration, in this paper we investigate balancing via i) navigation of the CH nodes in the mission space, in the context of mobile WSNs (Section 4) and ii) via dynamic modification of the cluster extensions, in a context of fixed CH nodes (Section 5).

Several works have been proposed in literature dealing with balancing through CH node mobility. A method is proposed in [19], which overcomes previous strategies based on random movements by investigating three different mobility strategies, one based on movement towards energy-dense regions of the WSN, a second one promoting CH migration towards the WSN regions generating more events and a third one realizing a hybrid approach. The authors show that the hybrid approach is the most effecting in prolonging the operative life of the WSN, up to 75% more with respect to standard algorithms such as LEACH. The CH node mobility algorithm proposed in this paper is based on the concept of Voronoi partitioning [20]. Voronoi WSN partition has been already investigated in some works, since the technique appears well-suited to support information fusion algorithms and transmission power control. Reference [21] presents a distributed approach for explicitly computing the

Voronoi partition of the WSN sensor field (i.e., of the area monitored by the WSN) based on geometrical considerations and aimed at minimizing the energy consumption. Reference [22] presents an offline (i.e., static) distributed algorithm for achieving energy-aware Voronoi WSN partitioning (off-line partitioning is achieved based on the knowledge of the location of nodes deployment). A centralized fuzzy C-means clustering algorithm is presented in [23], in which the fuzzy membership functions are based on the Voronoi partitioning of the WSN, computed relying on a distance metric including both the Euclidean distance and the residual energy of the nodes. Reference [24] presents a Voronoi-based clustering algorithm in which mobile CH nodes are pushed by "virtual forces" computed to minimize the variance of the cluster dimensions and the energy depletion of the CH nodes. The algorithm in [24] requires the explicit computation of the Voronoi diagrams after each iteration.

Researchers have realised that both the CH node location (or election) and the cluster sizes are critical to the WSN lifetime. This is because of the combined effect of the intra-cluster data processing load and the inter-clusters data forwarding load (in multi-hop WSNs), which have both to be sustained by the cluster heads. The greater the cluster size is, the greater will be the data processing load; the closer the CH node is to the base station, the greater will be the amount of data from other CH nodes that the CH node in question will have to forward to the base station. Hence, researchers have studied algorithms to control and balance the cluster sizes. One of the earliest works in this sense is [25], which presents a static clustering which takes into account the interaction between routing and clustering; an optimization problem is defined to find the optimal power allocation strategy. In [26], a genetic algorithm is used to find a static association between each sensor node and a CH node, optimal with respect to the modelled energy consumption; the configuration is kept for the whole network lifetime. Reference [27] proposes a dynamic energy-aware distributed topology control, in which CH nodes are assumed to be fixed and the objective is to control the cluster sizes in order to balance the CH node energy. The algorithm starts by computing the initial partition of the mission area as the Voronoi tessellation induced by the CH node positions. Then, an iterative algorithm changes the partition by moving the vertices of the regions according to heuristic rules taking into account the position and the leftover energy of the adjacent CH nodes. No convergence results to favourable configurations are given. In [28], a dynamic clustering scheme for WSN lifetime optimization is proposed, which requires periodically solving a non-linear programming problem to regulate the radius of each cluster.

2.2 Paper contributions

The main innovation proposed by this paper in the context of WSN clustering is the concept of dynamically controlling the Voronoi partition achieved by the CH nodes, while guaranteeing the convergence to a balanced clustering configuration. Two discrete-time control methodologies are proposed, the former suitable for mobile WSNs, the latter for fixed WSNs. Both algorithms are distributed and require the execution of simple update rules by the CH nodes, without the need of explicitly computing the Voronoi partition at any time-step.

The first algorithm, inspired by the data-sink node election method developed in [9] for fixed WSNs, and by the algorithm for multi-vehicle routing in [29], proposes a new approach to move mobile CH nodes in such a way that, in stationary environments, the network partitioning converges towards a generalized CVT, which takes into account both the position and the load generated by each sensor node. In contrast with the algorithms in [21], [22], [23], [24], the proposed one does not require to explicitly compute the partition at each time-step.

The second algorithm, based on a MDP model of the WSN and on a RL algorithm, proposes a new approach to vary the partition of the mission area in static WSNs. The mission area is partitioned according to a weighted Voronoi tessellation. The resulting clusters are unequal, as in [14], [27], [28], with the key differences that the proposed algorithm i) dynamically sets the weights without the need of solving optimization problems at each time-step, ii) in stationary environments, lets the partition converge to the weighted Voronoi tessellation which minimizes the mean squared error between the load of each CH nodes and the average load of the CH nodes.

3 Preliminaries

Sections 3.1 and 3.2 present some notions on Voronoi partitioning, on MDP and on reinforcement learning. Standard notation is used throughout the paper, with vectors denoted with bold characters, and with $|\cdot|$ denoting the cardinality operator.

3.1 Voronoi Partitioning

Table 1 summarizes the nomenclature used to define the Voronoi partitioning problem.

$\mathcal{A}$	Convex Euclidean domain/mission area
$d(\boldsymbol{p}, \boldsymbol{q})$	Euclidean distance between two points $\boldsymbol{p}, \boldsymbol{q} \in \mathcal{A}$
$d_{AW}^{\boldsymbol{w}}(\boldsymbol{p}, \boldsymbol{q})$	Weighted Euclidean distance between two points $\boldsymbol{p}, \boldsymbol{q} \in \mathcal{A}$, with weight vector $\boldsymbol{w}$
η	Density function with discrete support
φ	Density function
$\mathcal{G}$	Set of the generator points of the Voronoi tessellation
m_{φ}	Generalized centroid computed with respect to the density function φ
m_{η}	Generalized centroid computed with respect to the density function η
$\mathcal{P}_{\mathcal{G}}$	Voronoi partition (or tessellation) generated by the generating points in $\mathcal{G}$
$\mathcal{P}_{\mathcal{G}}(\boldsymbol{q})$	Voronoi region associated to generating point $\boldsymbol{q} \in \mathcal{G}$
$\mathcal{P}_{\mathcal{G}}^{\boldsymbol{w}}(\boldsymbol{q})$	Weighted Voronoi region associated to generating point $\boldsymbol{q}$ with weights vector $\mathbf{w}$
$\mathcal{P}_{\mathcal{G}}^{\boldsymbol{w}}$	Weighted Voronoi tessellation generated by the points in $\mathcal{G}$ with weight vector $\mathbf{w}$
$\boldsymbol{w} = \big(w(\boldsymbol{q})\big)_{\boldsymbol{q}\in\mathcal{G}}$	Weights vector of the weighted Voronoi partition

Table 1: Voronoi tessellation nomenclature

Mission space partitioning relies on the definition of Voronoi partition. Let us consider a convex Euclidean domain $\mathcal{A} \in \mathbb{R}^2$, and a set $\mathcal{G}$ of points in $\mathcal{A}$. The Voronoi regions with respect to the set $\mathcal{G}$ is defined as:

$$\mathcal{P}_{\mathcal{G}}(\boldsymbol{q}) = \{\boldsymbol{p} \in \mathcal{A} : d(\boldsymbol{p}, \boldsymbol{q}) \leq d(\boldsymbol{p}, \boldsymbol{q}'), \forall \boldsymbol{q}' \in \mathcal{G}\}, \forall \boldsymbol{q} \in \mathcal{G}, \quad (1)$$

where $d(\boldsymbol{p}, \boldsymbol{q}) = \|\boldsymbol{p} - \boldsymbol{q}\|_2$ is the Euclidean distance between $\boldsymbol{p}$ and $\boldsymbol{q}$. Equation (1) states that a point $\boldsymbol{p} \in \mathcal{A}$ belongs to the Voronoi region $\mathcal{P}_{\mathcal{G}}(\boldsymbol{q})$ if it is such that the distance between $\boldsymbol{p}$ and $\boldsymbol{q}$ is not greater than the distance between $\boldsymbol{p}$ and any other point $\boldsymbol{q}' \in \mathcal{G}$. The Voronoi regions $\mathcal{P}_{\mathcal{G}}(\boldsymbol{q})$ are bounded and convex, and are such that $\bigcup_{q \in \mathcal{G}} \mathcal{P}_{\mathcal{G}}(\boldsymbol{q}) = \mathcal{A}$ and $\bigcap_{\boldsymbol{q} \in \mathcal{G}} \mathcal{P}_{\mathcal{G}}(\boldsymbol{q})$ is a set of zero measure, i.e., they form a partition of the mission space $\mathcal{A}$, referred to as Voronoi partition or tessellation, denoted with $\mathcal{P}_{\mathcal{G}} = \{\mathcal{P}(\boldsymbol{q})\}_{\boldsymbol{q} \in \mathcal{G}}$. The points $\boldsymbol{q} \in \mathcal{G}$ are the so called generating points, or generators, of the Voronoi partition.

Depending on the position of the generating points in the mission space $\mathcal{A}$, specific partitions can be generated. In particular, in Section 4 we are interested in the generalized Centroidal Voronoi Tessellation (CVT), whose generating points are the centers of mass of the Voronoi regions, and which is regarded as an optimal partition corresponding to an optimal distribution of generators [20]:

Definition 1: Let the density function $\varphi: \mathcal{A} \to [0,1]$ be an absolutely continuous spatial distribution, with bounded and convex support in $\mathcal{A}$ [5].The generalized centroid of the set $\mathcal{P}_{\mathcal{G}}(\boldsymbol{q}) \subseteq \mathcal{A}$ with respect to the density function φ is $m_{\varphi} = \underset{\boldsymbol{p} \in \mathbb{R}^2}{\operatorname{argmin}} \int_{\mathcal{P}_{\mathcal{G}}(\boldsymbol{q})} d(\boldsymbol{s}, \boldsymbol{p})^2 \, \varphi(\boldsymbol{p}) d\boldsymbol{p}$. ■

In analogy with Definition 1, in the discrete spatial distribution case, the generalized centroids are defined as follows:

Definition 2. The generalized centroid of a set $\mathcal{P} \subseteq \mathcal{A}$ with respect to a discrete density function $\eta: \mathcal{A} \to [0,1]$ with support given by a finite, discrete set $\mathcal{V} \subseteq \mathcal{A}$ is

$$m_{\eta} = \underset{\boldsymbol{s} \in \mathbb{R}^2}{argmin} \sum_{\boldsymbol{p} \in \mathcal{V} \cap \mathcal{P}} d(\boldsymbol{s}, \boldsymbol{p})^2 \, \eta(\boldsymbol{p}). \quad (2)$$

■

The resulting Voronoi partition is the generalized CVT:

Definition 3: A Voronoi tessellation $\mathcal{P}_{\mathcal{G}} = \{\mathcal{P}_{\mathcal{G}}(\boldsymbol{q})\}_{\boldsymbol{q} \in \mathcal{G}}$ of a convex set $\mathcal{A}$ is called a generalized Centroidal Voronoi Tessellation with respect to the density function φ (η), if each generator $\boldsymbol{q} \in \mathcal{G}$ is equal to the generalized centroid of its partition $\mathcal{P}_{\mathcal{G}}(\boldsymbol{q})$ with respect to φ (η). ■

[5] I.e., there is a bounded and convex subset $\mathcal{Q} \subseteq \mathcal{A}$ s.t. $\varphi(\boldsymbol{p}) > 0$ if $\boldsymbol{p} \in \mathcal{Q}$, and $\varphi(\boldsymbol{p}) = 0$ if $\boldsymbol{p} \in \mathcal{A} \setminus \mathcal{Q}$.

Different Voronoi tessellations can be obtained by defining different distances (see, e.g., [30]), which, however, may present some drawbacks depending on the specific use-case: e.g., multiplicatively weighted Voronoi diagrams, obtained by dividing the Euclidean distance between a point and a generator point by a positive weight associated to the generator, may have disconnected partitions; in power diagrams, where the distance between a point and a generator point is defined as the squared Euclidean distance minus the generator weight, the generator may lie outside its own region. The clustering algorithm in Section 5 makes use of the additively weighted (AW) Voronoi tessellation, where the distance is defined as $d_{AW}^{\boldsymbol{w}}(\boldsymbol{p},\boldsymbol{q}) \coloneqq \|\boldsymbol{p}-\boldsymbol{q}\|_2 - w_{\boldsymbol{q}}$, with $w_{\boldsymbol{q}} \geq 0, \forall \boldsymbol{p} \in \mathcal{A}, \forall \boldsymbol{q} \in \mathcal{G}$. Accordingly, the AW Voronoi regions, which depend on the weights vector, are defined as:

$$\mathcal{P}_{\mathcal{G}}^{\boldsymbol{w}}(\boldsymbol{q}) = \left\{\boldsymbol{p} \in \mathcal{A} : d_{AW}^{\boldsymbol{w}}(\boldsymbol{p},\boldsymbol{q}) \leq d_{AW}^{\boldsymbol{w}}(\boldsymbol{p},\boldsymbol{q}'), \boldsymbol{w} = \left(w_{\boldsymbol{q}'}\right)_{\boldsymbol{q}' \in \mathcal{G}}, w_{\boldsymbol{q}'} \geq 0, \forall \boldsymbol{q}' \in \mathcal{G}\right\}, \forall \boldsymbol{q} \in \mathcal{G}. \quad (3)$$

The AW Voronoi tessellation is suitable for a WSN scenario, given that the generators lie within their region and there are no 'holes' in the regions.

3.2 Markov Decision Processes and Reinforcement Learning

Table 2 summarizes the nomenclature used to define the MDP/RL approach.

$c: \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$	Cost function of the MDP
$C_{\Sigma,\pi}$	Expected discounted total cost under policy π with initial state distribution Σ
ε	Exploration rate
λ_t	Learning rate at time t
$p_{\boldsymbol{x}\boldsymbol{x}'}(\boldsymbol{u})$	Transition probability from state $\boldsymbol{x}$ to state $\boldsymbol{x}'$ when action $\boldsymbol{u}$ is chosen
$\pi: \mathcal{X} \to \mathcal{U}$	Policy
$Q_\pi(\boldsymbol{x},\boldsymbol{u})$	Action-value function for state $\boldsymbol{x}$, action $\boldsymbol{u}$ and policy π
Σ	Initial state distribution
T	Transition probability matrix of the MDP
$\boldsymbol{u} = \pi(\boldsymbol{x}) \in \mathcal{U}$	Action chosen in state $\boldsymbol{x}$ under policy π
$\boldsymbol{u}_t$	Action chosen at time t
$\mathcal{U}$	Action space of the MDP
$V_\pi(\boldsymbol{x},\boldsymbol{u})$	Value function for state $\boldsymbol{x}$ and policy π
$\boldsymbol{w}_{\boldsymbol{x}} = \left(w_{\boldsymbol{x}}(j)\right)_{j \in \mathcal{C}}$	Weight vector in state $\boldsymbol{x} \in \mathcal{X}$
$\boldsymbol{x}_t$	State of the system at time t
$\mathcal{X}$	State space of the MDP

Table 2: MDP/RL nomenclature

A MDP is a discrete-time stochastic control process defined by the tuple $\{\mathcal{X}, \mathcal{U}, T, \Sigma, c, \gamma\}$, where $\mathcal{X}$ is finite state set, $\mathcal{U}$ is finite action set, $T \in [0,1]^{|\mathcal{X}|\times|\mathcal{U}|\times|\mathcal{X}|}$ is the transition probability matrix, Σ is the initial state distribution, $c: \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ is the cost function and $\gamma \in (0,1)$ is the discount factor, weighting immediate vs. future costs. Under the Markovian (or memory-less) property and under the stationary distribution assumption, the transition probabilities are stationary and the generic element $p_{\boldsymbol{x}\boldsymbol{x}'}(\boldsymbol{u})$ of the matrix T describes the probability that the system trajectory transits from state $\boldsymbol{x}$ to $\boldsymbol{x}'$ when action $\boldsymbol{u}$ is chosen. A policy $\pi: \mathcal{X} \to \mathcal{U}$ is a mapping from the state space to the action space, i.e., $\pi(\boldsymbol{x}) = \boldsymbol{u} \in \mathcal{U}, \forall \boldsymbol{x} \in \mathcal{X}$ [6]. In this paper we consider the stationary, infinite-horizon case under the total discounted cost criterion [31], in which a policy is optimal if it minimizes the expected cost:

$$C_{\Sigma,\pi}(\boldsymbol{x}) \coloneqq E_{\pi}\left\{\sum_{t=0,1,\ldots,\infty} \gamma^t c(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1})\right\}, \ \forall \boldsymbol{x} \in \mathcal{X},$$

where $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$ are the state visited and the action chosen at time t, respectively, $E_{\pi}\{\cdot\}$ denotes the expected value under policy π with initial state distribution Σ. The value function $V_{\pi}(\boldsymbol{x})$ is the expected discounted cost starting from $\boldsymbol{x}$ and following policy π thereafter; the action-value function $Q_{\pi}(\boldsymbol{x}, \boldsymbol{u})$ is the expected discounted cost starting from $\boldsymbol{x}$, choosing action $\boldsymbol{u}$ and following policy π thereafter: $Q_{\pi}(\boldsymbol{x}, \boldsymbol{u}) \coloneqq E_{\Sigma,\pi}\left\{\sum_{t=0,1,\ldots,\infty} \gamma^t (\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1})\right\}$.

Model-based and model-free methods exist to solve MDP (i.e., to find an optimal policy). Dynamic programming algorithms (see, e.g., [32]) are model-based methods, since they need a complete environment description (i.e., the transition probabilities must be known in advance) and are able to find an optimal policy iteratively; beside the need of a model, they are not scalable since the state space dimension explodes in practical scenarios (the so-called *curse of dimensionality* [33]).

RL algorithms are model-free methods which converge to an optimal policy under the hypothesis that every state is visited an infinite number of times (see, e.g., [33]); however, RL algorithms are able to fast converge to effective sub-optimal policies in many practical problems. Let the system be in a generic state $\boldsymbol{x} \in \mathcal{X}$; RL algorithms chose an action $\boldsymbol{u} \in \mathcal{U}$ based on a given control rule, and then observe the next state $\boldsymbol{x}' \in \mathcal{X}$ and the cost $c(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ incurred after the transition. Based on the observations, the RL algorithms update the value function estimate $V(\boldsymbol{x})$ or the action-value function estimate $Q(\boldsymbol{x}, \boldsymbol{u})$. Different RL methods exist, which differ by the rule used to decide the control action and by the rule used to update the value (or action-value) function. In this paper (not focused on RL solutions), the widely-used Q-learning algorithm is used, but more complex RL algorithm can be used (e.g., SARSA(λ), as in [34], actor-critic methods [35]). The Q-Learning update rule is the following one:

$$Q(\boldsymbol{x}_t, \boldsymbol{u}_t) \leftarrow \big(1 - \lambda_t(\boldsymbol{x}_t)\big) Q(\boldsymbol{x}_t, \boldsymbol{u}_t) + \lambda_t(\boldsymbol{x}_t) \Big(c(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1}) + \gamma \min_{\boldsymbol{u}' \in \mathcal{U}} Q(\boldsymbol{x}_{t+1}, \boldsymbol{u}')\Big), \tag{4}$$

[6] Policies in which one action per state is chosen with probability 1 are called deterministic policies. Considering deterministic policies only is not a limitation in unconstrained MDPs, since a deterministic optimal policy always exists [35].

where the learning rate $\lambda_t(\boldsymbol{x}_t) > 0$ is the key parameter for the algorithm convergence: if $\sum_{t=1,\dots,\infty} \lambda_t(\boldsymbol{x}_t) = \infty$ and $\sum_{t=1,\dots,\infty} \lambda_t(\boldsymbol{x}_t)^2 < \infty$, the estimate (4) converges to the optimal action-value function as $t \to \infty$ ([35]). In state $\boldsymbol{x}$, the action is then decided based on the current estimate of the state-action value function; the current best action is the one corresponding to the minimum Q value in state $\boldsymbol{x}$. To guarantee a certain degree of *exploration* of the state space set, an ε-greedy rule is followed: the best action is chosen by the controller with probability $1 - \varepsilon$, where $\varepsilon \in (0,1)$ is the *exploration rate*; with probability ε a random action is chosen:

$$\boldsymbol{u}_t = \begin{cases} argmin_{\boldsymbol{u} \in \mathcal{U}_x}\{Q(\boldsymbol{x}_t, \boldsymbol{u})\} \text{ with probability (1-}\varepsilon) \\ rand\{\boldsymbol{u} \in \mathcal{U}\} \text{ with probability } \varepsilon \end{cases}. \quad (5)$$

A large value of ε guarantees that different policies with respect to the current best one are explored, and thus avoids that the system remains stuck in a local minimum. A small value of ε, on the other hand, lets the controller choose the best action based on the current estimates of the action-value function and favors the exploitation of the current best policy.

Several extensions to non-stationary environments have been proposed in the literature. In this paper, the update rule of [36], tailored to non-stationary environments, is used:

$$\begin{aligned} &Q(\boldsymbol{x}_t, \boldsymbol{u}_t) \\ &\leftarrow (1 - \lambda_t)^{1/\pi(\boldsymbol{x}_t)} Q(\boldsymbol{x}_t, \boldsymbol{u}_t) + \left(1 - (1 - \lambda_t)^{1/\pi(\boldsymbol{x}_t)}\right) \left(c(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1}) + \gamma \min_{\boldsymbol{u}' \in \mathcal{U}} Q(\boldsymbol{x}_{t+1}, \boldsymbol{u}')\right). \end{aligned} \quad (6)$$

4 Dynamic Clustering for Mobile WSN

Table 3 summarizes the nomenclature used to define the dynamic Voronoi partitioning algorithm.

$\mathcal{B}_{t_k}(j)$	Multiset collecting all the nodes that have been associated to CH node j up to time t_k
$\mathcal{C}$	Set of the CH nodes
$\mathcal{F}_t = \{\boldsymbol{p}_t(i)\}_{i\in\mathcal{V}}$	Set of sensor node positions at time t
$\hat{\mathcal{G}} = \{\hat{\boldsymbol{q}}(j)\}_{j\in\mathcal{C}}$	Set of the limit reference positions of the generator points (CH nodes)
$\mathcal{G}_t = \{\boldsymbol{q}_t(i)\}_{i\in\mathcal{C}}$	Set of the generator point (CH nodes) positions at time t
$\mu_{t_k}(i,j)$	Multiplicity of sensor node i in $\mathcal{B}_{t_k}(j)$ (i.e., the number of times i appears in $\mathcal{B}_{t_k}(j)$)
$\mathcal{N}_{t_k}(j)$	Set of sensor nodes associated to CH node j at time t_k
$\boldsymbol{p}_t(i)$	Position of sensor node i at time t
$\hat{\boldsymbol{q}}(j)$	Limit reference position of the generator point (CH node) j
$\boldsymbol{q}_t(j)$	Position of the generator (CH node) j at time t
$\bar{r}(i)$	Expected transmission rate of node i
$r_t(i)$	Transmission rate of node i at time t
$r_{t_k}(i)$	Average transmission rate of the sensor node i during round k
$\rho_{t_{k+1}}(j)$	Cumulative transmission rate of the sensor nodes associated to the CH node j up to step t_k
t_k	Time instant of the k-th round
$t_{n(i,j)}$	Time instant when sensor node i was included in $\mathcal{B}_{t_k}(j)$ for the n-th time
$\mathcal{V}$	Set of sensor network nodes
$\mathcal{V}_{\mathcal{G}}$	Partition of the set of sensor network nodes $\mathcal{V}$ according to the generator points in $\mathcal{G}$
$\mathcal{V}_{\mathcal{G}}(j)$	Subset of sensor network nodes in the partition $\mathcal{P}_{\mathcal{G}}(\boldsymbol{q}(j))$

Table 3: Dynamic Voronoi tessellation nomenclature

Let the mission space be a bounded, convex Euclidean domain $\mathcal{A} \subset \mathbb{R}^2$. The sensor network is deployed on the mission area $\mathcal{A}$; let $\mathcal{V}$ be the set of generally mobile network nodes. The position of the node i at time t is defined by the time-varying mapping function $\boldsymbol{p}_t: \mathcal{V} \to \mathcal{A}$, and $\mathcal{F}_t = \{\boldsymbol{p}_t(i)\}_{i\in\mathcal{V}}$ denotes the set of node positions at time t. Finally, let $\mathcal{C}$ denote the set of the CH nodes in the sensor network. The position of the generic CH node j at time t is defined by the mapping function $\boldsymbol{q}_t: \mathcal{C} \to \mathcal{A}$, and $\mathcal{G}_t = \{\boldsymbol{q}_t(j)\}_{j\in\mathcal{C}}$ is the set of CH node positions.

4.1 Dynamic CVT Algorithm

We describe in the following the proposed clustering algorithm for network balancing in presence of mobile nodes and variable transmission data rates. The algorithm relies on the periodical Voronoi partitioning of the network, with each CH node playing the role of a Voronoi region's generator. A proper update rule for the CH node target positions is provided in the following in order to keep the network clustering balanced in time. Target CH node positions are computed periodically; the time scale is then discretized, and, during every round, it is assumed that the CH nodes can reach the target positions by moving on the mission space. The proposed algorithm is distributed, since each CH node takes the control decisions independently of the other CH nodes. The communication among the CH nodes is kept

limited to the communication of their target position at every round, i.e., each time a new target position computation is made.

Let τ_{round} denote the duration of each round, and let t_k be the time instant corresponding to the beginning of round k: $t_k = k\tau_{round}$. We assume that, at time t_k, each node $i \in \mathcal{V}$ is associated to one CH node; thus, we also assume an initial clustering of the network: the initial association of sensor nodes to CH nodes in round 0 can be chosen randomly; however, a reasonable initialization is to associate each sensor node at time t_0 to the nearest CH node. At the beginning of each round k, each sensor node sends to all the CH nodes (e.g., by flooding the network) its position $\boldsymbol{p}_{t_k}(i)$ and its average transmission rate during the k-th round, denoted with $r_{t_k}(i)$. The CH nodes also exchange among them their position set $\mathcal{G}_{t_k}$.

Given the position sets $\mathcal{F}_{t_k}$ and $\mathcal{G}_{t_k}$, each CH node $j \in \mathcal{C}$ computes which are its associated sensor nodes (i.e., the nodes whose distance to j is smaller than the distances to the other CH nodes) and collects them in the neighbor set $\mathcal{N}_{t_k}(j) \coloneqq \left\{i \in \mathcal{V}: d\left(\boldsymbol{p}_{t_k}(i), \boldsymbol{q}_{t_k}(j)\right) \leq d\left(\boldsymbol{p}_{t_k}(i), \boldsymbol{q}_{t_k}(j')\right), \forall j' \in \mathcal{C}\right\}$. Each CH node stores the past and the current neighbor sets in the *multiset* $\mathcal{B}_{t_k}(j) \coloneqq \mathcal{B}_{t_{k-1}}(j) \uplus \mathcal{N}_{t_k}(j)$, with $\mathcal{B}_{t_0}(j) \coloneqq \mathcal{N}_{t_0}(j)$. $\mathcal{B}_{t_k}(j)$ is a multiset in the sense that each node can appear in $\mathcal{B}_{t_k}(j)$ more than once; $\uplus$ denotes the operation of multiset union[7]. The number of times a node i appears in $\mathcal{B}_{t_k}(j)$ is called the *multiplicity* of the node, and is denoted in the following with $\mu_{t_k}(i,j)$. The multiplicity of a node is in practice equal to the number of times the node was associated to the CH node s up to time t_k. With little abuse of notation, let $t_{n(i,j)}$ denote the time instant when node i was associated to $\mathcal{B}_t(j)$ for the n-th time, i.e., raising the multiplicity of node i in $\mathcal{B}_t(j)$ to n. As already specified, each time an element is associated to $\mathcal{B}_{t_k}(j)$, the CH node also keeps trace of the node transmission rate and position. In the following, $\boldsymbol{p}_{t_{n(i,j)}}(i)$ and $r_{t_{n(i,j)}}(i)$, will denote, respectively, the position and the transmission rate of the node i when multiplicity n was gained in $\mathcal{B}_t(j)$.

The new target position $\boldsymbol{q}_{t_{k+1}}(j)$ of the CH node j, that is, the point in the region which the CH node j has to reach at time t_{k+1}, is then computed as the *reference point* of the multiset $\mathcal{B}_{t_k}(i)$, defined as the point which minimizes the average weighted squared distance to the sensor nodes in $\mathcal{B}_{t_k}(j)$, with each squared distance being weighted by the respective node's transmission rate:

$$\boldsymbol{q}_{t_{k+1}}(j) = \underset{\boldsymbol{s}\in\mathbb{R}^2}{argmin}\left\{\sum_{i\in\mathcal{B}_{t_k}(j)}\frac{1}{t_k}\sum_{n=1}^{\mu_{t_k}(i,j)} d\left(\boldsymbol{s}, \boldsymbol{p}_{t_n(i,j)}(i)\right)^2 r_{t_{n(i,j)}}(i)\right\}, \forall j \in \mathcal{C}. \quad (7)$$

[7] Multisets, also commonly known as *bags*, are unordered collections of items which may contain duplicates. For example, the multiset {1,1,1,2,3} is equivalent to the multiset {1,2,1,1,3}, but differs from the multiset (also a set in this case) {1,2,3} because of the *multiplicity* of element 1. The multiset $A = A_1 \uplus A_2$ by definition contains only the elements that occur either in A_1 or in A_2, and the multiplicity of each element in A is the multiplicity of that element in A_1 plus the multiplicity of that element in A_2, e.g., {1,1,1,2,3} $\uplus$ {1,2,3} = {1,1,1,2,3,1,2,3}.

(Note that $i \in \mathcal{B}_{t_k}(j)$ in (2) means to visit each node in $\mathcal{B}_{t_k}(j)$ only once, as in usual set operations.) Given that the function to be minimized in (7) is strictly convex in $\mathbb{R}^2$, there is a unique reference point $\boldsymbol{q}_{t_{k+1}}(j)$ for each $j \in \mathcal{C}$. Also, since all the elements of $\mathcal{B}_{t_k}(j)$ belong to $\mathcal{A}$, and $\mathcal{A}$ is convex, it follows that $\boldsymbol{q}_{t_{k+1}}(j) \in \mathcal{A}$.

Different balancing objectives can be pursued by changing the weights of the weighted squared distances in equation (7), i.e., by substituting the transmission rate $r_{t_{n(i,j)}}(i)$ with another characteristic of sensor node i, such as the leftover energy.

4.1.1 Convergence of the reference points to the CVT

We are interested at showing that the sequences of the reference points (7) converge to the generating points of the CVT, i.e., to the generalized centroids, as $k \to \infty$. The property will be proved in the stationary case, i.e., under the following assumptions:

Assumption 1.

a. The sensor nodes (differently from the CH nodes) are fixed, and thus the set of the sensor node positions is time-independent; we will then omit the subscript t, i.e., $\mathcal{F} = \{\boldsymbol{p}(i)\}_{i \in \mathcal{V}}$;
b. The sensor transmission rates have a Poisson distribution with mean $\bar{r}(i), i \in \mathcal{V}$.

Under Assumption 1.a, at each round k the only required communication exchange among the CH nodes concerns their positions $\boldsymbol{q}_{t_k}(j), j \in \mathcal{C}$ and the node transmission rates $r_{t_k}(i), i \in \mathcal{V}$. Under Assumption 1.b, the scenario is then equivalent to a standard Voronoi clustering problem with points on the mission space appearing with exponential distribution. As shown in [29], since the cardinality of $\mathcal{V}$ is finite, the sequences of the positions of the reference points $\boldsymbol{q}_{t_k}(j)$ converge to well-defined limit generation points, denoted as $\hat{\boldsymbol{q}}(j) = \lim_{k \to \infty} \boldsymbol{q}_{t_k}(j), j \in \mathcal{C}$, collected in the limit set $\hat{\mathcal{G}}$; the corresponding limit Voronoi partition and limit Voronoi regions are denoted as $\mathcal{P}_{\hat{\mathcal{G}}}$ and $\mathcal{P}_{\hat{\mathcal{G}}}(\hat{\boldsymbol{q}}(j))$, respectively. The following property holds:

Property 1 [29]. The sequence of the Voronoi partitions $\left\{\mathcal{P}_{\mathcal{G}_{t_k}}\right\}_{k=0,1,2,\dots}$ generated by the sequences of reference points $\left\{\left\{\boldsymbol{q}_{t_k}(j)\right\}_{j \in \mathcal{C}}\right\}_{k=0,1,2,\dots}$, converges, almost surely, to the limit Voronoi partition $\mathcal{P}_{\hat{\mathcal{G}}}$ generated by the limit reference points $\hat{\boldsymbol{q}}(j), j \in \mathcal{C}$. ■

Thanks to Property 1, to study the steady-state properties of the algorithm we just need to check the properties of the limit Voronoi partition $\mathcal{P}_{\hat{\mathcal{G}}}$, i.e., to check that $\mathcal{P}_{\hat{\mathcal{G}}}$ is a generalized centroidal tessellation.

The Voronoi partition $\mathcal{P}_{\mathcal{G}} = \{\mathcal{P}_{\mathcal{G}}(\boldsymbol{q}(j))\}_{j\in\mathcal{C}}$ induces an associated node set partition, denoted with $\mathcal{V}_{\mathcal{G}} = \{\mathcal{V}_{\mathcal{G}}(j)\}_{j\in\mathcal{C}}$, where the sensor nodes are grouped as:

$$\mathcal{V}_{\mathcal{G}}(j) = \mathcal{V} \cap \mathcal{P}_{\mathcal{G}}(\boldsymbol{q}(j)) = \{i \in \mathcal{V} | \boldsymbol{p}(i) \in \mathcal{P}_{\mathcal{G}}(\boldsymbol{q}(j))\}, \forall j \in \mathcal{C}. \quad (8)$$

Also, from Property 1, it follows that, as $k \to \infty$, the network graph is partitioned in $|\mathcal{C}|$ subsets of nodes $\mathcal{V}_{\hat{\mathcal{G}}}(j), j \in \mathcal{C}$, defined as in eq. (8), and that the sequence of node set partitions $\{\mathcal{V}_{\mathcal{G}_{t_k}}(j)\}_{j\in\mathcal{C}}, k = 0,1,\ldots$, converges to the limit node set partition $\{\mathcal{V}_{\hat{\mathcal{G}}}(j)\}_{j\in\mathcal{C}}$.

The main result is given in the following Theorem 1, which demonstrates that the limit Voronoi partition obtained by the proposed algorithm is a generalized CVT:

Theorem 1. Under Assumption 1, for all CH nodes $j \in \mathcal{C}$, the limit reference point $\hat{\boldsymbol{q}}(j) = \lim_{k\to\infty} \boldsymbol{q}_{t_k}(j)$ of the sequence of reference points (7) coincides with the generalized centroids of the limit Voronoi region $\mathcal{P}_{\hat{\mathcal{G}}}(\hat{\boldsymbol{q}}(j))$, computed with respect to the stationary density function η, defined as the spatial distribution of the sensors, weighted by the node average transmission rates:

$$\eta(\boldsymbol{s}) = \begin{cases} \frac{1}{c}\bar{r}(i) \text{ if } \boldsymbol{s} = \boldsymbol{p}(i), \forall i \in \mathcal{V} \\ 0 \qquad\qquad \text{otherwise} \end{cases}. \quad (9)$$

where c is a normalization constant. ■

Proof. Since the average transmission rate of the nodes is stationary by assumption, the distribution η is stationary as well. We are interested in the limit generating points of the limit Voronoi partition, which, given the update rule of eq. (7), and considering that the sensor node positions are constant by Assumption 1.a, are defined as

$$\hat{\boldsymbol{q}}(j) \coloneqq \lim_{k\to+\infty} \boldsymbol{q}_{t_k}(j) = \underset{\boldsymbol{s}\in\mathbb{R}^2}{argmin}\left(\lim_{k\to+\infty} \sum_{i\in\mathcal{B}_{t_k}(j)} \frac{1}{t_k} \sum_{n=1}^{\mu_{t_k}(j,i)} d(\boldsymbol{s}, \boldsymbol{p}(i))^2 r_{t_n(j,i)}(i)\right), \forall j \in \mathcal{C}. \quad (10)$$

By Property 1, eventually, as $k \to \infty$, the position of each generator j converges to the limit point $\hat{\boldsymbol{q}}(j)$ and $\mathcal{V}_{\mathcal{G}_{t_k}}(j)$ converges to the limit set $\mathcal{V}_{\hat{\mathcal{G}}}(j)$. Hence, eventually, the new nodes i in $\mathcal{N}_{t_k}(j)$ will all belong to $\mathcal{V}_{\hat{\mathcal{G}}}(j)$. The first consequence is that the contribution of the terms of the first summation in (10) for $i \in \mathcal{B}_{t_k}(j) \backslash \mathcal{V}_{\hat{\mathcal{G}}}(j)$ will vanish as $k \to +\infty$:

$$\hat{\boldsymbol{q}}(j) = \underset{\boldsymbol{s}\in\mathbb{R}^2}{argmin}\left(\lim_{k\to+\infty} \sum_{i\in\mathcal{V}_{\hat{\mathcal{G}}}(j)} d(\boldsymbol{s}, \boldsymbol{p}(i))^2 \frac{1}{t_k} \sum_{n=1}^{\mu_{t_k}(j,i)} d(\boldsymbol{s}, \boldsymbol{p}(i))^2 r_{t_n(j,i)}(i)\right). \quad (11)$$

Secondly, it holds that $\lim_{k\to\infty} \frac{\mu_{t_k}(j,i)}{t_k} = 1, \forall i \in V_{\hat{\mathcal{G}}}(j)$. Thanks to the fact that the transmission rate distribution is stationary, the average transmission rate is recovered at the limit, and, from equation (11), it follows:

$$\widehat{\boldsymbol{q}}(j) = \underset{\boldsymbol{s}\in\mathbb{R}^2}{argmin} \left\{ \sum_{i\in\mathcal{V}_{\hat{\mathcal{G}}}(j)} d\big(\boldsymbol{s}, \boldsymbol{p}(i)\big)^2 \bar{r}(i) \right\}. \tag{12}$$

Finally, by the definition (9) of η, from equation (12) it holds that (considering also that the normalization constant c of equation (9) does not affect the argmin operator):

$$\widehat{\boldsymbol{q}}(j) = \underset{\boldsymbol{s}\in\mathbb{R}^2}{argmin} \left\{ \sum_{i\in\mathcal{V}_{\hat{\mathcal{G}}}(j)} d\big(\boldsymbol{s}, \boldsymbol{p}(i)\big)^2 \eta\big(\boldsymbol{p}(i)\big) \right\}. \tag{13}$$

By comparing equation (13) and equation (2) of Definition 2, it turns out that the limit reference point $\widehat{\boldsymbol{q}}(j)$ is the generalized centroid of the limit Voronoi region $\mathcal{P}_{\hat{\mathcal{G}}}\big(\widehat{\boldsymbol{q}}(j)\big)$ with respect to the discrete stationary density function η. ■

4.2 Mobile WSN Clustering Implementation

At stage k, the information available to each CH node j are the sensor node positions, the CH node positions, the indexes of its associated sensor nodes, collected in the set $\mathcal{N}_{t_k}(j)$, and the transmission rates of its associated nodes, $r_{t_k}(i), i \in \mathcal{N}_{t_k}(j)$.

In practice, to compute the new target positions $\boldsymbol{q}_{t_{k+1}}(j), j \in \mathcal{C}$, there is no need to solve the optimization problem (7) at each stage, or to store all the values of the past CH node positions and transmission rates, since iterative algorithms exist, as, for instance, the MacQueen's *k*-means method, which eventually converges to the same minimizer of (12) (see [20,37]). In the considered scenario, the iterative algorithm of Table 4 is executed at every stage k by every CH node $j \in \mathcal{C}$ to compute the target points $\boldsymbol{q}_{t_{k+1}}(j)$.

Step 0	Initialization at time $t = t_0 = 0$ (round 0): a. $\boldsymbol{q}_{t_0}(j) = \mathbf{0}$ b. $\rho_{t_0}(j) = 0, \forall j \in \mathcal{C}$ c. $k = 1$
Step 1	At time t_k (round k), for all $j \in \mathcal{C}$, compute the neighbor set $\mathcal{N}_{t_k}(j)$ and initialize: a. $\boldsymbol{q}_{t_{k+1}}(j) = \boldsymbol{q}_{t_k}(j)$ b. $\rho_{t_{k+1}}(j) = \rho_{t_k}(j)$
Step 2	For all $j \in \mathcal{C}$ and for all $i \in \mathcal{N}_{t_k}(j)$, update a. $\boldsymbol{q}_{t_{k+1}}(j) \leftarrow \frac{\rho_{t_{k+1}}(i)}{\rho_{t_{k+1}}(j)+r_{t_k}(i)} \boldsymbol{q}_{t_{k+1}}(j) + \frac{r_{t_k}(i)}{\rho_{t_{k+1}}(j)+r_{t_k}(i)} \boldsymbol{p}(i)$ b. $\rho_{t_{k+1}}(j) \leftarrow \rho_{t_{k+1}}(j) + r_{t_k}(i)$
Step 3	For all $j \in \mathcal{C}$, a. Move towards $\boldsymbol{q}_{t_{k+1}}(i)$ and transmit the new position to the other CH node b. Update $k \leftarrow k + 1$
Step 4	Go to Step 1

Table 4: Dynamic CVT algorithm

At the end of step 2, $\boldsymbol{q}_{t_{k+1}}(j)$ is the new target position of CH node j and $\rho_{t_{k+1}}(j)$ is the cumulative transmission rate of the nodes associated to CH node j up to step t_k. To execute the algorithm of Table 4 it is sufficient that each CH node j stores the current positions of the CH nodes (to compute the neighbor set at step 1), the last reference point $\boldsymbol{q}_{t_k}(j)$ and the last cumulative load $\rho_{t_k}(j)$. The initial conditions are $z_{t_0}(s) = 0$ and $\rho_{t_0}(s) = 0, \forall s \in \mathcal{G}$.

In non-stationary environments (e.g., when the transmission rates are time-varying or if the network graph is time-varying due to sensor mobility and/or due to the occurrence of node failures), the distribution φ_η is non-stationary, and, according to Definition 2, the generalized CVT is time-varying as well. In this case, Step 1 of Table 4 can be modified to weight the new points (i.e., the node associations at step k) more than the old ones, in an exponential averaging fashion. Step 1.b of the iterative algorithm becomes:

Step 1	b. $\rho_{t_{k+1}}(j) = \alpha \rho_{t_k}(j)$

Table 5: Step 1.b of the dynamic CVT algorithm for non-stationary environments

where α is a constant real number between 0 and 1 which weights the past points: at time t_k, the weight of the node association occurred at time $t_j \leq t_k$ is reduced by a factor α^{k-j}. Depending on the dynamics of the distribution, by tuning the parameter α this new rule might be able to 'follow' the variations of the traffic rate distribution and/or of the sensor node positions. Note that the stationary algorithm is obtained by setting $\alpha = 1$.

5 Dynamic Weighted Clustering for Fixed WSN

In this Section, a new idea is presented to let fixed (non-mobile) CH nodes cope with variations of the network, e.g., in terms of node transmission rates. The idea is to control the width of the areas covered by the generators (i.e., by the CH nodes) by varying the generator weights in response to the variations of the transmission rates of the node.

5.1 Dynamic Weighted Clustering MDP Model

The system is modelled as a discrete-time MDP, defined by the tuple $\{\mathcal{X}, \mathcal{U}, T, c\}$, under the assumption that sensor node position is fixed and that the sensor transmission rates are stationary and exponentially distributed with mean $\bar{r}(i), \forall i \in \mathcal{V}$.

An additive weighted metric is used, where the weights are used to vary the neighbor sets of the CH nodes. The sensor node association to the CH nodes depends on the weighted distance, i.e., on a weights vector $\boldsymbol{w} = \left(w(j)\right)_{j\in\mathcal{C}}$; the neighbor sets are defined accordingly:

$$\mathcal{N}^{\boldsymbol{w}}(j) \coloneqq \left\{ i \in \mathcal{V} \,\middle|\, j = \underset{j'\in\mathcal{C}}{argmin}\left(d\left(\boldsymbol{q}_{t_k}(j'), \boldsymbol{p}_{t_k}(i)\right) - w(j)\right)\right\}, j \in \mathcal{C}. \tag{14}$$

State space $\mathcal{X}$. Let $\boldsymbol{x}_{t_k} \coloneqq \left(s_{t_k}^{min}, w_{t_k}(1), w_{t_k}(2), \dots, w_{t_k}(|\mathcal{C}|)\right)$ denote the system state at time t_k, where:

- $w_{t_k}(j) \in [0, w^{max}], j \in \mathcal{C}$, is the weight associated to the CH node j at time t_k;
- $j_{t_k}^{min}$ is the *least loaded* CH node at time t_k, i.e.: $j_{t_k}^{min} \coloneqq \underset{j\in\mathcal{C}}{\text{argmin}} \sum_{i\in\mathcal{N}^{w_{t_k}(j)}} r_{t_k}(i)$.

A maximum weight value w^{max} is defined to limit the state space; clearly, w^{max} must be selected in oder not to affect the control potential. To obtain a discrete and finite state space, the set of admissible values is then quantized and defined as:

$$\mathcal{Q} \coloneqq \left\{0, \frac{1}{q|\mathcal{C}|}\, w^{max}, \frac{2}{q|\mathcal{C}|} w^{max}, \dots, \frac{q|\mathcal{C}|-1}{q|\mathcal{C}|} w^{max}, w^{max}\right\}, \text{ with } q \in \mathbb{N}, \tag{15}$$

where $\frac{w^{max}}{q|\mathcal{C}|}$ is the quantization interval. Notice that all the weights are equal if $w(j) = \frac{1}{|\mathcal{C}|} w^{max}, \forall j \in \mathcal{C}$ (in such case, the weighted tessellation corresponds to the standard one). A feasible weight vector $\boldsymbol{w}$ is defined as the vector of the weights of the CH nodes such that their sum is w^{max}; the set of feasible weights is then defined as

$$\mathcal{W} \coloneqq \left\{ \boldsymbol{w} = \left(w(j)\right)_{j\in\mathcal{C}} \middle| w(j) \in \mathcal{Q}, \forall j \in \mathcal{C}, and\ \sum_{j\in\mathcal{C}} w(j) = w^{max} \right\}. \tag{16}$$

With this approximation, the state space is defined as the following finite set:

$$\mathcal{X} \coloneqq \{\boldsymbol{x} = (j, \boldsymbol{w}) | j \in \mathcal{C}, \boldsymbol{w} \in \mathcal{W}\}. \tag{17}$$

Note that different states may exist with the same weights vector, since the CH node loads vary with time. With little abuse of notation, the least loaded CH node, the weights vector and the weight associated to the j-th generator in state $\boldsymbol{x}$ will be denoted with $j_{\boldsymbol{x}}^{min}$, $\boldsymbol{w_x}$ and $w_{\boldsymbol{x}}(j)$, respectively. Correspondingly, let $\mathcal{N}^{w_x}(j)$ be the set of the nodes associated to the CH node j in state $\boldsymbol{x}$.

Action space $\mathcal{U}$. In the generic state $\boldsymbol{x}$, the available actions are the ones which increase the weight of the least loaded CH node $j_{\boldsymbol{x}}^{min}$ and, at the same time, decrease the weight of a CH node. By defining $\boldsymbol{\delta}_j$ as a null vector of $|\mathcal{C}|$ components but the j-th element equal to one, the action space when the system is in state $\boldsymbol{x}$ is defined as the following set of $|\mathcal{C}|$ actions:

$$\mathcal{U}_{\boldsymbol{x}} = \left\{\boldsymbol{u} = \boldsymbol{\delta}_{j_{\boldsymbol{x}}^{min}} - \boldsymbol{\delta}_j, j \in \mathcal{C}\right\}. \tag{18}$$

As the controller chooses the action $\boldsymbol{u}$ in state $\boldsymbol{x}$, the next state $\boldsymbol{x}'$ is one of the states in $\mathcal{X}$ such that $\boldsymbol{w}_{\boldsymbol{x}'} = \boldsymbol{w}_{\boldsymbol{x}} + \frac{w^{max}}{q|\mathcal{C}|}\boldsymbol{u}$. Note that, the action $\boldsymbol{u} = \boldsymbol{\delta}_{j_{\boldsymbol{x}}^{min}} - \boldsymbol{\delta}_j$ with $j = j_{\boldsymbol{x}}^{min}$ is a null-valued vector, i.e., it equals to doing nothing.

Transition matrix T. Given that the transmission rates of the nodes are exponentially distributed, there is a probability that, at time t_{k+1}, any of the CH nodes is the least-loaded one. The transition probabilities $p_{\boldsymbol{x}\boldsymbol{x}'}(\boldsymbol{u})$ are then positive if $\boldsymbol{x}'$ is such that $\boldsymbol{w}_{\boldsymbol{x}'} = \boldsymbol{w}_{\boldsymbol{x}} + \frac{w^{max}}{q|\mathcal{C}|}\boldsymbol{u}$, null otherwise. If action $\boldsymbol{u} = \boldsymbol{\delta}_{j_{\boldsymbol{x}}^{min}} - \boldsymbol{\delta}_j$ with $j = j_{\boldsymbol{x}}^{min}$ is chosen, the next weights vector is equal to the current one; nonetheless, a state transition may occur if the sensor node loads vary in such a way that the least loaded CH node changes. Since the transition probabilities depend on the statistical characteristics of the traffic and are not easily computed, and to cope with the curse of dimensionality, a model-free RL approach is proposed in the following.

Cost function c. The main objective of the algorithm is to balance the load among the CH nodes by varying the weights vector and, consequently, the neighbor sets. Let $R_{\boldsymbol{x}}(s)$ denote the expected load of CH node s in state $\boldsymbol{x}$, and let $\bar{R}_{\boldsymbol{x}}$ be the average expected load of the CH nodes on state $\boldsymbol{x}$, i.e: $R_{\boldsymbol{x}}(j) = \sum_{i \in \mathcal{N}^{w_x(j)}} \bar{r}(i)$ and $\bar{R}_{\boldsymbol{x}} = \frac{1}{|\mathcal{C}|}\sum_{j \in \mathcal{C}} R_{\boldsymbol{x}}(j)$. The proposed state-dependent cost function $c: \mathcal{X} \to \mathbb{R}_{\geq 0}$, evaluating the load balance among the CH nodes, is then the mean squared error

$$c(\boldsymbol{x}) = \sum_{j \in \mathcal{C}} (R_{\boldsymbol{x}}(j) - \bar{R}_{\boldsymbol{x}})^2. \tag{19}$$

The expected load $R_{\boldsymbol{x}}(j)$ depends on the state $\boldsymbol{x}$ only thanks to the assumption of a stationary environment.

Different balancing objectives can be pursued by changing the cost (19), e.g., by considering the leftover energies of the sensor nodes.

5.2 Dynamic Weighted Clustering RL Algorithm

The optimal policy for the MDP defined in the former Section is pursued on-line by means of RL algorithms. Since the objective of the paper is not focussed on finding new RL algorithms, the Q-learning algorithm is proposed as a simple, popular and effective algorithm for stationary environments. In non-stationary environments, the algorithm in [36] is proposed, which requires minor modifications to the Q-learning update rule to be implemented.

For both the stationary and the non-stationary cases, at stage k, the information available to each CH node j are the sets of the positions of the sensor nodes and of the CH nodes, $\mathcal{F}_{t_k}$ and $\mathcal{G}_{t_k}$, respectively, the current weight vector $\boldsymbol{w}_{t_k}$, the indexes of its associated nodes, collected in the set $\mathcal{N}^{\boldsymbol{w}_{t_k}}(j)$, and the transmission rates of its associated nodes computed over the last stage, $r_{t_k}(i), i \in \mathcal{N}^{\boldsymbol{w}_{t_k}}(j)$. Then, each CH node j computes its current load as the sum of the transmission rates of its associated nodes, i.e., $R_{t_k}(j) = \sum_{i \in \mathcal{N}^{\boldsymbol{w}_{t_k}}(j)} r_{t_k}(j), j \in \mathcal{C}$, and communicates it to the other CH nodes. Every CH node then computes the current cost as $c(\boldsymbol{x}_{t_k}) = \sum_{j \in \mathcal{C}} \left(R_{t_k}(j) - \bar{R}_{t_k}\right)^2$, where $\bar{R}_{t_k} = \frac{1}{|\mathcal{C}|} \sum_{j \in \mathcal{C}} R_{t_k}(j)$ is the average current load of the CH nodes, and the index $j_{t_k}^{min}$ of the least-loaded CH node. Thus, each CH node knows the current state $\boldsymbol{x}_{t_k} = (j_{t_k}^{min}, \boldsymbol{w}_{t_k})$.

The CH node $j_{t_k}^{min}$ is the controller node at stage k; it decides the action $\boldsymbol{u}_{t_k} \in \mathcal{U}_{\boldsymbol{x}_{t_k}}$, based on an ε-greedy policy, and communicates the new weights $\boldsymbol{w}_{t_{k+1}} = \boldsymbol{w}_{t_k} + \frac{w^{max}}{q|\mathcal{C}|} \boldsymbol{u}_{t_k}$ to all the other CH nodes. Finally, all the CH nodes update their neighbour sets $\mathcal{N}^{\boldsymbol{w}_{t_{k+1}}}(j), j \in \mathcal{C}$, according to the additively weighted distance, and update the values of the Q functions according to the Q-learning rule (4).

Step 0	Initialization at time $t = t_0 = 0$ (round 0), for all $j \in \mathcal{C}$: a. $Q(\boldsymbol{x}, \boldsymbol{u}) = 0, \forall \boldsymbol{x} \in \mathcal{X}, \forall \boldsymbol{u} \in \mathcal{U}$ b. $\pi(\boldsymbol{x}) = \text{rand}\{\boldsymbol{u} \in \mathcal{U}_{\boldsymbol{x}}\}$ c. $w_{\boldsymbol{x}}(j) = \frac{w^{max}}{\lvert\mathcal{C}\rvert}$ d. $k = 1$
Step 1	At time t_k (round k), for all $j \in \mathcal{C}$: a. Compute $\mathcal{N}^{\boldsymbol{w}_{t_k}}(j)$ and $R_{t_k}(j) = \sum_{i \in \mathcal{N}^{\boldsymbol{w}_{t_k}}(j)} r_{t_k}(i)$ b. Transmit $R_{t_k}(j)$ to the other CH nodes
Step 2	Once received the R_{t_k}'s, for all $j \in \mathcal{C}$: a. Retrieve the state $\boldsymbol{x}_{t_k} = \left(j_{t_k}^{min}, \boldsymbol{w}_{t_k}\right)$ with $j_{t_k}^{min} \coloneqq \underset{j' \in \mathcal{C}}{\text{argmin}}\, R_{t_k}(j')$ c. Compute $\bar{R}_{t_k}(j) = \frac{1}{\lvert\mathcal{C}\rvert} \sum_{j \in \mathcal{C}} R_{t_k}(j)$ and $c_{t_k}(\boldsymbol{x}) = \sum_{j \in \mathcal{C}} \left(R_{t_k}(j) - \bar{R}_{t_k}(j)\right)^2$ d. If $k > 1$, update $Q\left(\boldsymbol{x}_{t_{k-1}}, \boldsymbol{u}_{t_{k-1}}\right)$ with update rule (4)
Step 3	CH node $j_{t_k}^{min}$: a. Chooses $\boldsymbol{u}_{t_k}$ with ε-greedy rule (5) b. Computes $\boldsymbol{w}_{t_{k+1}} = \boldsymbol{w}_{t_k} + \frac{w^{max}}{q\lvert\mathcal{C}\rvert} \boldsymbol{u}_{t_k}$ c. Transmits $\boldsymbol{w}_{t_{k+1}}$ to the other CH nodes $j \in \mathcal{C}$
Step 4	For all $j \in \mathcal{C}$: a. Update $k \leftarrow k + 1$ and go to Step 1

Table 6: Weighted CVT algorithm

Even if each CH node computes and stores the Q-tables independently, and even if, at every stage, only one of the CH nodes is the controller, the Q-learning algorithm properties still hold since the Q-function estimates coincide, being based on the same information set.

In non-stationary environments, the update rule (6) is used in Step 2.d of the algorithm to try to follow the time-varying distribution of the traffic rate of the sensor nodes:

Step 2	d. If $k > 1$, update $Q\left(\boldsymbol{x}_{t_{k-1}}, \boldsymbol{u}_{t_{k-1}}\right)$ with update rule (6)

Table 7: Step 2.d of the weighted CVT algorithm for non-stationary environments

6 Simulations

Numerical simulations were executed to test the presented algorithms against a static clustering strategy. The objective was to evaluate the algorithm characteristics and no specific WSN was modeled.

Three clustering algorithms were implemented. A reference static algorithm was considered, in which the CH nodes are fixedly placed in the position corresponding to the generators of the CVT of the mission area $\mathcal{A}$ (*static clustering*); let the set of the positions of the CH nodes with static clustering

be denoted as $\mathcal{G}_{\mathcal{A}}^{CVT}$. The dynamic CVT algorithm of Table 4 and Table 5 was also implemented, with mobile CH nodes and with initial CH node positions $\mathcal{G}_{t_0} = \mathcal{G}_{\mathcal{A}}^{CVT}$ (*dynamic CVT clustering*). Finally, also the dynamic CVT algorithm of Table 6 and Table 7 was implemented, with fixed CH nodes with positions $\mathcal{G}_{t_k} = \mathcal{G}_{\mathcal{A}}^{CVT}, \forall k = 0,1, \ldots$ (*dynamic weighted clustering*).

Three scenarios were simulated: with static nodes and stationary distribution of the average transmission rates of the sensor nodes (*stationary scenario*); with static nodes and time-varying transmission rate distribution (*non-stationary scenario*); with mobile nodes and stationary rate distribution (*mobile scenario*). The parameter α of the dynamic CVT algorithm was set equal to 0.7 in the non-stationary and mobile scenarios. The learning rate of the RL algorithm was selected as $\lambda_{t_k}(\boldsymbol{x}_{t_k}) = 1/n_{t_k}(\boldsymbol{x}_{t_k})$, where $n_{t_k}(\boldsymbol{x}_{t_k})$ is the number of times that the current state was visited up to time t_k. The other RL algorithm parameters were set as described in Table 8.

Parameter	Stationary scenario	Non-stationary and mobile scenarios
q	20	20
w^{max}	17.7	41.3
γ	0.95	0.75
ε	0.8	0.8

Table 8: RL parameters

In the first scenario, $|\mathcal{V}| = 625$ fixed sensor nodes were regularly positioned over an area $\mathcal{A}$ of 26×26, as shown in Figure 1 a). The number of CH nodes was $|\mathcal{C}| = 4$ and the number of simulated rounds was $K = 300$. The sensor node transmission rate distribution was stationary during the simulation runs. The distribution was exponential with mean $\bar{r}(i)$ depending on the position $y(i)$ of the sensor node i on $\mathcal{A}$: $\bar{r}(i) = \frac{1}{c} f_{\mathcal{N}}(y(i)|\mu, \sigma^2)$, where $c = \frac{1}{|\mathcal{V}|}\sum_{i \in \mathcal{V}} \bar{r}(i)$ is a normalization constant, $f_{\mathcal{N}}$ denotes the probability density function of a normal spatial distribution on the Euclidean plane, and $\mu = (7.25,7.25)$ and $\sigma^2 = 36$ are the mean and variance of the distribution, respectively (i.e., in this scenario, the normal distribution of the mean transmission rate was centered in the lower-right quadrant of $\mathcal{A}$). Figure 1 a) shows the initial CH node positions $\mathcal{G}_{\mathcal{A}}^{CVT}$, inducing a CVT for the sensor node positions (i.e., without weighting each node with its transmission rate). The upper plot of Figure 1 d) shows that this configuration does not guarantee a balanced load among the CH nodes, with an average cost during the simulation of about 0.35; note that the cost variations during the simulations are due to the fact that the node transmission rates are not constant. Figure 1 b) shows the final positions of the CH nodes as well as their trajectories during the simulation performed with the dynamic CVT algorithm: the cluster of the lower-left CH node (i.e., the CH node closer to the sensor nodes with the largest transmission rates) is reduced, while the cluster of the upper-right CH node (i.e., the CH node closer to the sensor nodes with the smallest transmission rates) is increased. The middle plot of Figure 1 d) shows

that this configuration manages to improve the load balance, with a cost which is rapidly lowered in the first 50 rounds and which then stabilizes at about 0.06 (the cost is about 16.5% with respect to the static simulation). Figure 1 c) shows the final partition obtained by varying the weights vector in the simulation performed with the dynamic weighted algorithm: again, the cluster of the lower-left CH node is reduced, while the cluster of the upper-right CH node is increased. The lower plot of Figure 1 d) shows that also this configuration improves the load balance, with a cost which is lowered in the first 50 rounds, and which then stabilizes at about 0.09 (the cost is about 25.7% with respect to the static simulation). During the first 50 rounds the cost oscillates because the RL algorithm is exploring the state space, and sometimes unfavorable actions are chosen. We note that, in this scenario, the weighted dynamic algorithm performances are similar to the dynamic CVT algorithm ones, even if the CH node positions are fixed.

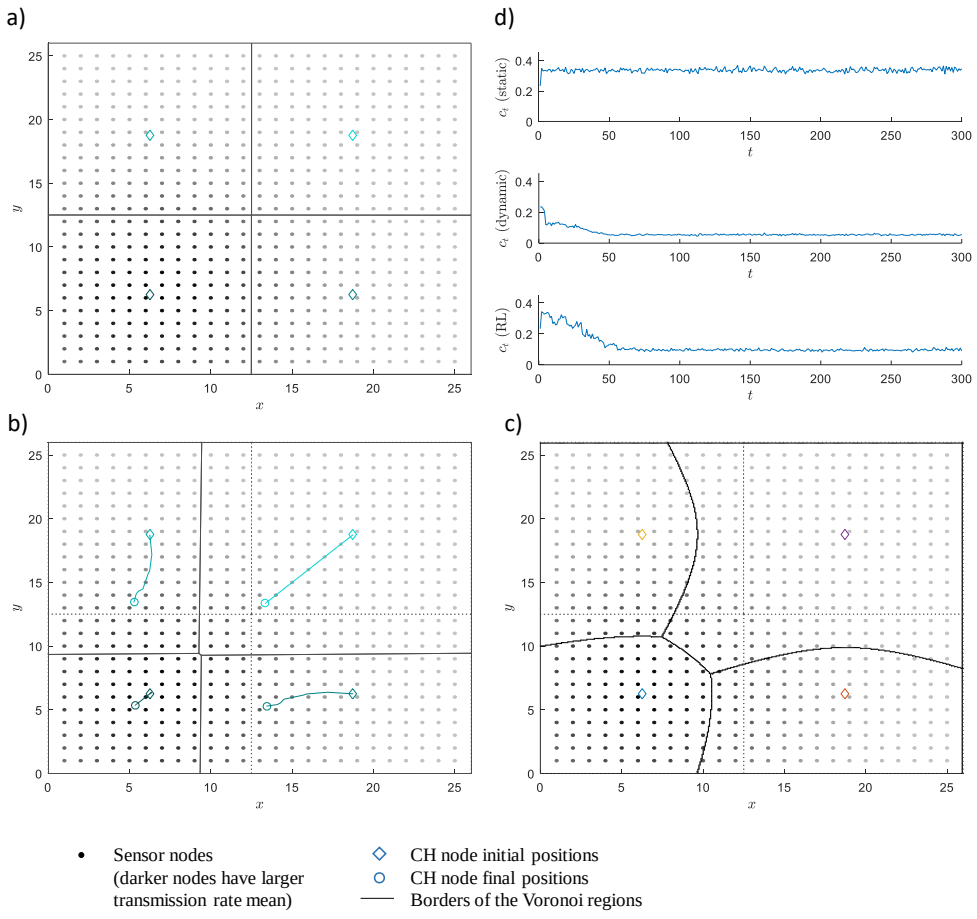

Figure 1: stationary scenario; a) initial configuration; b) final configuration with dynamic CVT algorithm; c) final configuration with dynamic weighted algorithm; d) cost dynamics during the simulation.

In the non-stationary scenario, 800 fixed sensor nodes were randomly positioned over an area $\mathcal{A}$ of 45×45 , as shown in Figure 2 a). The number of CH nodes was $|\mathcal{C}| = 9$ and the number of simulated rounds was $K = 400$. The sensor node transmission rate distribution was time-varying during the first

$K_1 = 300$ rounds. Initially, at time $t_0 = 0$, all the sensor nodes transmits with equal mean rate $\bar{r}_{t_0}(i) = \frac{1}{|\mathcal{V}|}, \forall i \in \mathcal{V}$; at the end of the simulation, the distribution was exponential with normally distributed intensity: $\bar{r}_{t_K}(i) = \frac{1}{c} f_{\mathcal{N}}\left(y_{t_K}(i)|\mu, \sigma^2\right)$, with $\mu = (22{,}22)$ and $\sigma^2 = 49$ (i.e., the final transmission rate distribution was centered in $\mathcal{A}$). The mean rates of the sensor nodes were linearly varied from round 0 to round K_1 as $\bar{r}_{t_k}(i) = \bar{r}_{t_0}(i) + \frac{k}{K_1}\left(\bar{r}_{t_{K_1}}(i) - \bar{r}_{t_0}(i)\right), k = 0{,}1, \dots, K_1$, and then remained constant during the last rounds, i.e., $\bar{r}_{t_k}(i) = \bar{r}_{t_K}(i), k = K_1, K_1 + 1, \dots, K$. Figure 2 a) shows the initial CH node positions $\mathcal{G}_{\mathcal{A}}^{CVT}$, inducing a CVT for the sensor node positions. The upper plot of Figure 2 d) shows that, initially, when the mean transmission rates of the sensor nodes are uniform, this configuration guarantees a balanced load among the CH nodes; however, as the rate distribution changes, the load balance degrades and the cost grows up to about 0.22 in the final rounds. Figure 2 b) shows the final positions of the CH nodes as well as their trajectories during the simulation performed with the dynamic CVT algorithm: the cluster of the center CH node (i.e., the CH node closer to the sensor nodes with the largest transmission rate) is reduced, while the other clusters are increased. The middle plot of Figure 2 d) shows that the CH node trajectories manage to vary the Voronoi regions in order to keep the load balanced during the simulation rounds, with an average cost of about 0.03 (the cost is about 12.6% with respect to the static simulation). Figure 2 c) shows the final partition obtained by varying the weights vector in the simulation performed with the dynamic weighted algorithm: again, the cluster of the middle CH node is reduced, while the other clusters are increased. The lower plot of Figure 2 d) shows that the variations of the weights vector during the simulation rounds manage to keep the load balanced, with an average cost of about 0.04 (16.8% with respect to the static simulation).

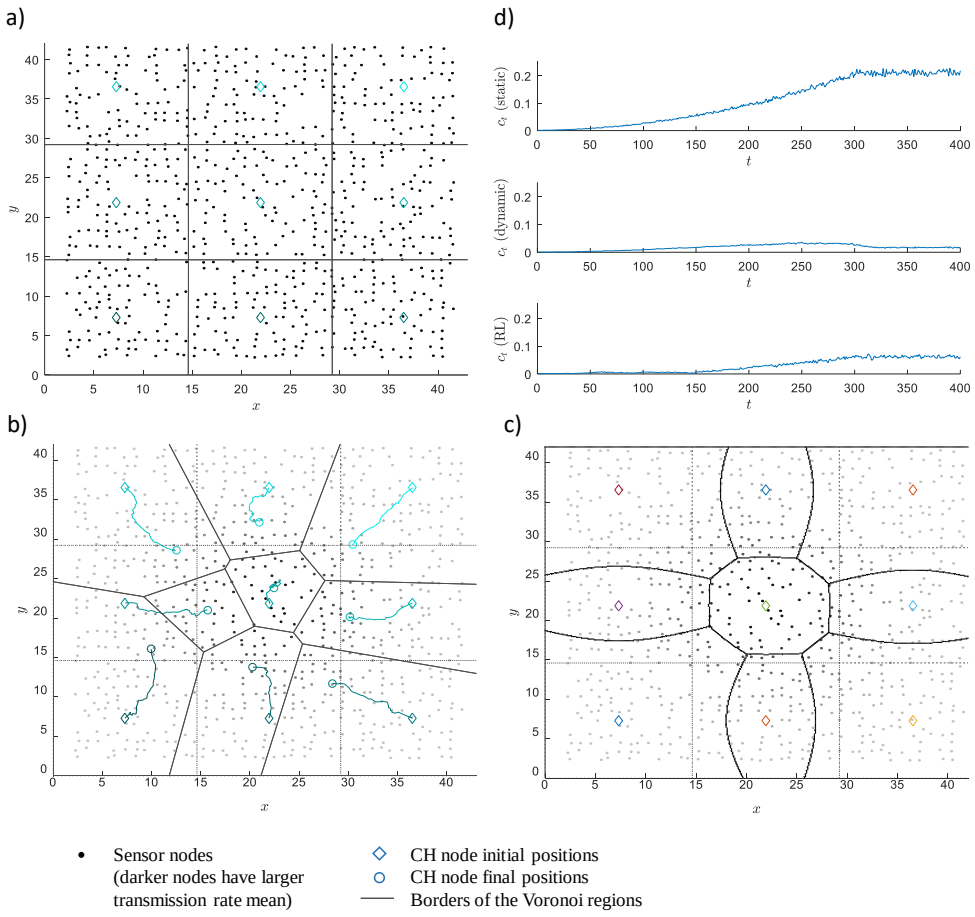

Figure 2: non-stationary scenario; a) initial configuration; b) final configuration with dynamic CVT algorithm; c) final configuration with dynamic weighted algorithm; d) cost dynamics during the simulation.

The mobile scenario presents mobile sensor nodes, initially randomly positioned over an area of 45×45, as shown in Figure 3 a). The number of CH nodes was $|\mathcal{C}| = 9$ and the number of simulated rounds was $K = 400$. The sensor node transmission rate distribution was stationary during the simulation runs: $\bar{r}(i) = \frac{1}{|\mathcal{V}|}, \forall i \in \mathcal{V}$ during the whole simulation. From round 0 to round $K_1 = 300$ the sensor nodes move over a larger mission area of 63×63: the final node position $y_{t_{K_1}}(i)$ was randomly computed as $y_{t_{K_1}}(i) = \text{rand}(1,1.5) \cdot y_{t_0}(i)$, where $\text{rand}(1,1.5)$ is a random number extracted from a uniform distribution between 1 and 1.5. The positions of the sensor nodes were linearly varied from round 0 to round K_1 as $y_{t_k}(i) = y_{t_0}(i) + \frac{k}{K_1}\left(y_{t_{K_1}}(i) - y_{t_0}(i)\right), k = 0,1,\dots,K_1$, and then remained constant during the last rounds, $y_{t_k}(i) = y_{t_K}(i), k = K_1, K_1 + 1, \dots, K$. Figure 3 a) shows the initial CH node positions $\mathcal{G}_{\mathcal{A}}^{CVT}$, inducing a CVT for the initial sensor node positions. The upper plot of Figure 3 d) shows that, initially, since the mean transmission rates of the sensor nodes are uniform, this configuration guarantees a balanced load among the CH nodes; however, as the nodes start moving, the load balance degrades and the cost grows up to about 0.02 in the final rounds. Figure 3 b) shows the final positions of the CH nodes as well as their trajectories during the simulation performed with the

dynamic CVT algorithm: as the nodes spread over the mission area the CH nodes move away from one another to cover a larger area. The lower plot of Figure 3 d) shows that the CH node trajectories manage to vary the Voronoi regions to keep the load balanced during the simulation rounds, with an average cost of about 0.002 (20% with respect to average cost in the static simulation) and a final cost of about 0.004 (17.8% with respect to final cost in the static simulation).

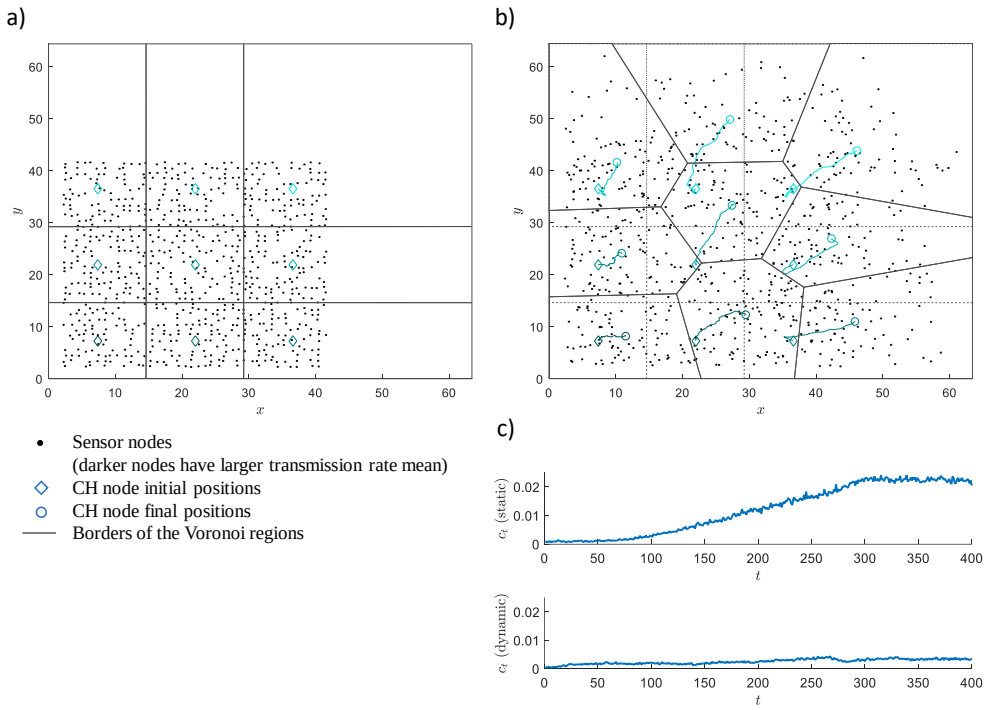

Figure 3: mobile scenario; a) initial configuration; b) final configuration with dynamic CVT algorithm; c) cost dynamics during the simulation.

7 Conclusions

The proposed approaches to WSN clustering manage to dynamically partition the mission space with the objective of balancing the load of the cluster head nodes. Two distributed iterative algorithms are proposed. In case the CH nodes are mobile, the first algorithm dynamically controls the positions of the CH nodes, considered as the generator points of a Voronoi partition, in such a way that the partition converges to a generalized Centroidal Voronoi Tessellation, with favorable load balancing characteristics. In case the CH nodes are fixed, the second algorithm is proposed, based on reinforcement learning. The algorithm dynamically controls the weights of the additively weighted Voronoi partition generated by the positions of the CH nodes, with the aim of balancing their load. In both the proposed algorithms, the main innovation is that they guarantee the convergence towards a balanced network partition without the need of solving optimization programs or of explicitly computing the Voronoi diagram at any time-step. The algorithms were validated by numerical simulations.

Current and future work is aimed at three main objectives:

i. The first objective is to take into account also energy-balancing objectives by considering the WSN energy-related characteristics – such as the energy depletion due to the transmitted traffic or to the node mobility, or the impact of energy harvesting approaches (see, e.g., [38]). The first algorithm can be modified by defining appropriate weights for equation (7), and the second algorithm can be modified by defining different costs with respect to the ones in equation (19).

ii. The second objective is to enhance the algorithm performances in non-stationary environments, e.g., by means of model-based control design to make use of predictions of the node mobility and/or of the traffic dynamics.

iii. The third objective is to apply the proposed algorithms to real use-cases, therefore taking into account all the technology-dependent implementation issues not investigated in this paper.

References

[1] M.F. Othman, K. Shazali, Wireless Sensor Network Applications: A Study in Environment Monitoring System, Procedia Eng. 41 (2012) 1204–1210. doi:10.1016/j.proeng.2012.07.302.

[2] A. Vidács, R. Vida, Wireless Sensor Network Based Technologies for Critical Infrastructure Systems, in: E. Kyriakides, M. Polycarpou (Eds.), Intell. Monit. Control. Secur. Crit. Infrastruct. Syst. Stud. Comput. Intell., Springer Berlin Heidelberg, 2015: pp. 201–316. doi:10.1007/978-3-662-44160-2_11.

[3] A. Pietrabissa, C. Poli, D.G. Ferriero, M. Grigioni, Optimal planning of sensor networks for asset tracking in hospital environments, Decis. Support Syst. 55 (2013) 304–313. doi:10.1016/j.dss.2013.01.031.

[4] G. Zhao, Wireless Sensor Networks for Industrial Process Monitoring and Control: A Survey, Netw. Protoc. Algorithms. 3 (2011). doi:10.5296/npa.v3i1.580.

[5] T.-C. Yu, C.-C. Lin, C.-C. Chen, W.-L. Lee, R.-G. Lee, C.-H. Tseng, S.-P. Liu, Wireless sensor networks for indoor air quality monitoring., Med. Eng. Phys. 35 (2013) 231–5. doi:10.1016/j.medengphy.2011.10.011.

[6] S. Manfredi, Design of a multi-hop dynamic consensus algorithm over wireless sensor networks, Control Eng. Pract. 21 (2013) 381–394. doi:10.1016/j.conengprac.2012.12.001.

[7] M.M. Afsar, M.-H. Tayarani-N, Clustering in sensor networks: A literature survey, J. Netw. Comput. Appl. 46 (2014) 198–226. doi:10.1016/j.jnca.2014.09.005.

[8] Xiaojun Zhai, Hongyuan Jing, T. Vladimirova, Multi-sensor data fusion in Wireless Sensor Networks for Planetary Exploration, in: 2014 NASA/ESA Conf. Adapt. Hardw. Syst., IEEE, 2014: pp. 188–195. doi:10.1109/AHS.2014.6880176.

[9] A. Pietrabissa, F. Liberati, G. Oddi, A distributed algorithm for Ad-hoc network partitioning based on Voronoi Tessellation, Ad Hoc Networks. 0 (2016) 1–12. doi:10.1016/j.adhoc.2016.03.008.

[10] D. Jia, H. Zhu, S. Zou, P. Hu, Dynamic Cluster Head Selection Method for Wireless Sensor Network, IEEE Sens. J. 16 (2016) 2746–2754. doi:10.1109/JSEN.2015.2512322.

[11] P. Suri, R.K. Bedi, S.K. Gupta, Review paper on various clustering protocols used in Wireless Sensor

Network (WSN), in: 2015 Int. Conf. Electr. Electron. Signals, Commun. Optim., IEEE, 2015: pp. 1–4. doi:10.1109/EESCO.2015.7253700.

[12] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, Proc. 33rd Annu. Hawaii Int. Conf. Syst. Sci. (2000). doi:10.1109/HICSS.2000.926982.

[13] O. Younis, S. Fahmy, HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks, IEEE Trans. Mob. Comput. 3 (2004) 366–379. doi:10.1109/TMC.2004.41.

[14] M. Ye, C. Li, G. Chen, J. Wu, EECS: an energy efficient clustering scheme in wireless sensor networks, PCCC 2005. 24th IEEE Int. Performance, Comput. Commun. Conf. 2005. (2005). doi:10.1109/PCCC.2005.1460630.

[15] G.S. Kumar, P.M. V Vinu, K.P. Jacob, Mobility Metric based LEACH-Mobile Protocol, in: 2008 16th Int. Conf. Adv. Comput. Commun., IEEE, 2008: pp. 248–253. doi:10.1109/ADCOM.2008.4760456.

[16] L. Shen, S. Deng, J. Li, Mobility-based clustering protocol for wireless sensor networks with mobile nodes, IET Wirel. Sens. Syst. 1 (2011) 39–47. doi:10.1049/iet-wss.2010.0084.

[17] J.-S. Lee, C.-L. Teng, An Enhanced Hierarchical Clustering Approach for Mobile Sensor Networks Using Fuzzy Inference Systems, IEEE Internet Things J. (2017) 1–1. doi:10.1109/JIOT.2017.2711248.

[18] J.-S. Lee, W.-L. Cheng, Fuzzy-Logic-Based Clustering Approach for Wireless Sensor Networks Using Energy Predication, IEEE Sens. J. 12 (2012) 2891–2897. doi:10.1109/JSEN.2012.2204737.

[19] M. Ma, Y. Yang, Clustering and load balancing in hybrid sensor networks with mobile cluster heads, in: Proc. 3rd Int. Conf. Qual. Serv. Heterog. Wired/wireless Networks - QShine '06, ACM Press, New York, New York, USA, 2006: p. 16. doi:10.1145/1185373.1185395.

[20] Q. Du, V. Faber, M. Gunzburger, Centroidal Voronoi Tessellations: Applications and Algorithms, SIAM Rev. 41 (1999) 637–676. doi:10.1137/S0036144599352836.

[21] W. Alsalih, K. Islam, Y. Núñez-Rodríguez, H. Xiao, Distributed voronoi diagram computation in wireless sensor networks, in: Proc. Twent. Annu. Symp. Parallelism Algorithms Archit. - SPAA '08, ACM Press, New York, New York, USA, 2008: p. 364. doi:10.1145/1378533.1378597.

[22] J. Chen, C. Kim, F. Song, A Distributed Clustering Algorithm for Voronoi Cell-Based Large Scale Wireless Sensor Network, in: 2010 Int. Conf. Commun. Mob. Comput., IEEE, 2010: pp. 209–213. doi:10.1109/CMC.2010.230.

[23] K.S. Nithyakalyani, S. Suresh, An approach to data Aggregation in wireless sensor network using Voronoi fuzzy clustering algorithm, J. Sci. Ind. Res. (India). 72 (2013) 287–293.

[24] G.S. Tewolde, W. Sheng, Energy aware adaptive clustering in wireless sensor networks, in: 2011 IEEE Int. Conf. ELECTRO/INFORMATION Technol., IEEE, 2011: pp. 1–6. doi:10.1109/EIT.2011.5978632.

[25] T. Shu, M. Krunz, S. Vrudhula, Power balanced coverage-time optimization for clustered wireless sensor networks, in: Proc. 6th ACM Int. Symp. Mob. Ad Hoc Netw. Comput. - MobiHoc '05, ACM Press, New York, New York, USA, 2005: p. 111. doi:10.1145/1062689.1062705.

[26] D. Lee, W. Lee, J. Kim, Genetic Algorithmic Topology Control for Two-Tiered Wireless Sensor Networks, in: 2007: pp. 385–392. doi:10.1007/978-3-540-72590-9_53.

[27] Joongheon Kim, Jihoon Choi, Wonjun Lee, Energy-Aware Distributed Topology Control for Coverage-Time Optimization in Clustering-Based Heterogeneous Sensor Networks, in: 2006 IEEE 63rd Veh. Technol. Conf., IEEE, 2006: pp. 1033–1037. doi:10.1109/VETECS.2006.1682991.

[28] Joongheon Kim, Wonjun Lee, K. Eunkyo, Doe-Wan Kim, Hyeokman Kim, Coverage-time optimized dynamic clustering of networked sensors for pervasive home networking, IEEE Trans. Consum. Electron. 53 (2007) 433–441. doi:10.1109/TCE.2007.381712.

[29] A. Arsie, K. Savla, E. Frazzoli, Efficient Routing Algorithms for Multiple Vehicles With no Explicit Communications, IEEE Trans. Automat. Contr. 54 (2009) 2302–2317. doi:10.1109/TAC.2009.2028954.

[30] A. Okabe, B. Boots, K. Sugihara, S.N. Chiu, Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd Edition, John Wiley & Sons, New York, 2009.

[31] M.L. Puterman, Markov Decision Processes, John Wiley & Sons, Inc., New York, 1994. doi:10.1002/9780470316887.

[32] D.P. Bertsekas, Dynamic Programming deterministic and stochastic models, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.

[33] R.S. Sutton, A.G. Barto, Reinforcement Learning : An Introduction, (2012).

[34] A. Pietrabissa, S. Battilotti, F. Facchinei, A. Giuseppi, G. Oddi, M. Panfili, V. Suraci, Resource Management in Multi-Cloud Scenarios via Reinforcement Learning, in: 34th Chinese Control Conf., Hanzhou (China), 2015: p. in print.

[35] R.S. Sutton, A.G. Barto, Reinforcement learning: an introduction, MIT Press, Cambridge, MA, 1998.

[36] M. Kaisers, Addressing Environment Non-Stationarity by Repeating Q-learning, 17 (2016) 1–31.

[37] Q. Du, T.W. Wong, Numerical studies of MacQueen's k-means algorithm for computing the centroidal Voronoi tessellations, Comput. Math. with Appl. 44 (2002) 511–523. doi:10.1016/S0898-1221(02)00165-7.

[38] A. Frezzetti, S. Manfredi, M. Pagano, A design approach of the solar harvesting control system for wireless sensor node, Control Eng. Pract. 44 (2015) 45–54. doi:10.1016/j.conengprac.2015.07.004.