A Coalition Formation Game for Distributed Node Clustering in Mobile Ad Hoc Networks

R. Massin[#1], C. J. Le Martret[1], and P. Ciblat[2]

Abstract

In the context of wireless mobile ad hoc networks, node clustering is a well known solution for handling the scalability issue. While existing work focused on unstructured (i.e., flat) networks, this paper investigates a clustering algorithm to handle stable size-restricted clusters for structured (i.e., group-based) networks. In addition, we have identified that the ad hoc network clustering literature lacks a theoretical framework. This paper fills this gap by proposing to use coalition game theory, identifying coalitions to clusters and players to nodes. This theoretical framework allows us to derive a novel generic distributed node clustering algorithm. The algorithm is proved to converge to Nash-stable partitions. It is based on the concept of switch operations, where nodes take decision whether to leave or not their current coalition based on the coalition values. These decisions are made independently on any node individual payoff, meaning that the coalition formation game has a transferable utility. This generic algorithm is then tailored to both structured and unstructured networks, by defining judiciously the value functions and the heuristics dedicated to selecting suitable switch operations. Based on extensive simulations, we show that our proposed solutions outperform the existing ones especially in terms of cluster size and stability.

Keywords—Mobile ad hoc network, distributed clustering, game theory, switch operation, distributed coalition formation algorithm, operational group, cluster stability.

I. Introduction

Ad hoc networks are infrastructure-less multi-hop wireless networks where each node participates in routing by forwarding data for other nodes. Those networks are self-organizing and are

Contact author.

[1] THALES Communications & Security, 4 avenue des Louvresses, 92622 Gennevilliers, France.
Email: raphael-a.massin@thalesgroup.com, christophe.le_martret@thalesgroup.com.

[2] Département Communications et Électronique, Telecom ParisTech, 46 rue Barrault, 75013 Paris, France.
Email: philippe.ciblat@telecom-paristech.fr.

used when usual infrastructure is not available or not suitable, e.g., in wireless sensor networks (WSN), vehicular networks (VANET), public protection and disaster relief (PPDR), or military systems. In order to implement practical large ad hoc networks, gathering nodes in clusters (called *clustering*) was first proposed in the early 80s [1] for HF packet radio networks, and has so far triggered a lot of work in the literature. For instance, it was proposed in the contexts of VANETs [2] and cognitive radio networks [3].

This paper is dedicated to the clustering problem of mobile ad hoc networks. We consider two kinds of ad hoc networks: *i*) *unstructured networks* where there is no special organization of the networks, i.e., all nodes being equal to each other, *ii*) *structured networks* that have an inherent hierarchical structure associated with their *raison d'être*, and in which the nodes belong to *operational groups*, e.g., squad, section. Examples of structured networks are PPDR and military networks. In such networks, we assume that each node belongs to only one operational group. For the sake of readability, an operational group will be simply referred to as *group* in the sequel. The existence of groups raises two major differences with respect to unstructured networks. First, the traffic is strongly dependent on the network hierarchical organization, being mostly concentrated within groups. Second, the nodes from the same group are very likely to move in the same direction. For these two reasons, it is beneficial to design clustering solutions that take into account group information, in order to provide more stable networks as well as better end-to-end quality of service (QoS).

Most existing works about ad hoc network clustering have focused on unstructured networks. For example in [4] the authors propose the lowest identifier (LID) and highest degree clustering (HC) algorithms where nodes with the lowest identifier, respectively the highest degree within their neighborhood, become cluster head (CH). To form the clusters, non-CH nodes affiliate to their neighbor CH with the lowest identifier, respectively the highest degree. The stability of the clusters formed with LID or HC, has been improved in [5] with the least cluster change (LCC) mechanism that only performs re-clustering when multiple CH nodes become neighbors. In the vote-based clustering (VOTE) proposal [6], non-CH nodes join a CH only if the number of its cluster members is below a threshold, thus limiting the cluster size. Based on the knowledge of node location information, another approach [7] attempts to estimate the nodes relative mobilities and to capture the mobility patterns to form stable clusters. For the same purpose, the authors in [8] propose to build clusters using past, current and predicted nodes' positions through the help

of a learning automaton. The clustering algorithm defined in [9] uses a Gauss-Markov model to calculate the velocity and direction of the nodes. The novelty of signal-attenuation aware clustering algorithm (SECA) [10], lies in the introduction of link qualities (based on received signal strength) combined with the nodes relative mobilities to determine if a node becomes CH. Authors in [11] use centralized genetic algorithms and particle swarm optimization to select a stable CH.

By contrast, only a few papers have tackled the problem of clustering structured networks. The authors in [12] introduce the type-based clustering algorithm (TCA). This clustering scheme associates a stability factor to each node and selects as CH the nodes that have the highest stability factor in a radio neighborhood. The stability factor takes group membership into account using the IP subnet of each node. A limitation of TCA lies in the fact that two CH nodes cannot be neighbors. A direct consequence in dense networks is the formation of large clusters. A second example is detailed in [13] which proposes a topology management mechanism for hierarchical group oriented networks, where groups are based on geographical locations. In [14] we have proposed the so-called *distributed clustering based on operational group algorithm*. This algorithm forms size limited clusters whose membership is close to the groups, and thus outperforms other clustering algorithms from the literature.

In addition, we have noted that the overall literature about ad hoc network clustering lacks a theoretical framework. Indeed, most of the proposed solutions are protocol-centric, based on the CH election. They are presented in such different ways that it renders difficult their theoretical analysis (e.g., the study of their convergence), comparisons and extensions. We propose in this paper to use the *coalition game theory* which appears to be relevant for such a framework, and derive our algorithms accordingly. Indeed, modeling the clustering problem as a coalition game sounds natural by identifying clusters to coalitions and nodes to players. This framework allows to extend and generalize the solution from [14] identifying the *node cluster swap* to the *switch operation* used in [19-20]. This also allows us to propose new algorithms for unstructured networks, and to characterize their convergence points as Nash-stable equilibria. Coalition game theory is a branch of game theory used to study the behavior of players when they cooperate among themselves, and can be used more specifically in the context of wireless networks [15]. It has also been proposed for several applications such as: spectrum sensing [16], [17] or spectrum sharing [18] in cognitive networks, interference management in small cell

networks [19], cooperation among roadside units [20] or vehicle to vehicle communications [21] in VANETs, device-to-device communications [22], and cooperative communications in ad hoc networks [23].

Operating a clustered ad hoc network requires to run several processes continuously, in parallel, and distributively. One of these processes corresponds to the clustering. To perform clustering, all nodes need to detect their radio neighborhood and to exchange enough information to form the clusters. This is achieved by protocols that can be either specific to the communication system providers or taken from standards such as the IEEE 802.15.4 one, designed for wireless personal area networks. In our work, we assume that the capability provided by such protocols to share information between nodes such as cluster membership, and link qualities, is available.

Another important process is the intra-cluster *radio resource allocation* (RRA), in charge of the communications between nodes belonging to the same cluster. It is delegated to a specific node in the cluster called the *resource allocator* (RA), which locally decides how to allocate resources inside the cluster, similarly to what is done by base stations in cellular systems. It can be seen as a locally centralized operation, which can be then optimized efficiently to serve the users. However, the intra-cluster RRA performance is directly linked to the cluster topology, e.g., number of users, network connectivity. Therefore, we look for clustering algorithms that can enforce some *constraints* on the cluster topology. To explain these constraints, we model the network as a graph where each node is a vertex, and where an edge between two vertices means that the corresponding nodes are in radio range. We define the cluster constraints as following: *i*) the cluster graph must be connected, meaning that any pair of nodes in a cluster can communicate using the intra-cluster links, *ii*) the cluster size, i.e., the number of cluster members, must be limited to allow efficient RRA, *iii*) the cluster diameter[1] must be limited to prevent increasing too much the delay and cost of communication between cluster members. Notice that there is also an inter-cluster RRA process, in charge of the communications between neighbor nodes from different clusters. This process is fully distributed and thus much less efficient than the intra-cluster RRA. In addition to the lack of results identified for the structured networks in the literature, the clustering algorithms proposed for unstructured networks do not take into account all our constraints at the same time. Finally, the availability of the node locations and velocities

[1]The length of the longest shortest path between any pair of the vertices in the cluster graph.

is often a key underlying assumption of the most recent proposals, which cannot always be guaranteed. We will thus not rely on such assumption in our work.

Thus, the main contributions of this paper are: First, we revisit the clustering problem of ad hoc networks using the coalition game framework, identifying coalitions to clusters and players to nodes. Second, using the coalition game framework, we derive a generic coalition formation algorithm that performs clustering for mobile networks. This algorithm is run at the node level and is fully distributed. Conversely to existing solutions from the literature, this algorithm does not need the use of CH. Convergence of this algorithm to Nash-stable solutions is proved. Third, the generic algorithm is tailored to the unstructured and structured cases. This part encompasses the proposal of several revenue functions and heuristics that are the key design parameters of these algorithms. Last, we compare our solutions with some reference algorithms from the state of the art through extensive simulations. We show that our algorithms outperform the state of the art ones in any case (static or mobile networks, structured or unstructured networks).

The paper is organized as follows. Section II is dedicated to the casting of the clustering problem within the coalition game theory framework and to the design of the generic coalition formation algorithm. Section III adapts the generic algorithm to the unstructured networks, and Section IV to the structured ones. Section V is dedicated to the performance evaluation. Some concluding remarks are given in Section VI.

II. Generic clustering algorithm based on coalition formation theory

This section proposes a generic clustering algorithm for mobile ad hoc networks based upon coalition formation theory. It is generic in the sense that it is agnostic of the network structure. This generic algorithm will be then adapted to unstructured and structured networks in Section III and Section IV respectively.

We begin by introducing the network characteristics considered in this work and then the coalition formation framework that is used to derive the algorithm. We then detail the algorithm that is split into two specific procedures and prove its convergence.

A. Network description

The network considered in this paper is modeled by a graph $\mathcal{G}$ defined by the set of nodes $\mathcal{V}$ and the set of edges $\mathcal{E}$. The number of nodes of $\mathcal{G}$ is $N := |\mathcal{V}|$, where $:=$ stands for *by definition*.

Two nodes i and j are neighbors if $(i,j) \in \mathcal{E}$. The degree of a node is equal to the number of its neighbors. A clustering solution leads to a partition p of $\mathcal{G}$, formed by N_c disjoint clusters noted $\mathcal{S}_k$ with $k \in \{1,\dots,N_c\}$. The graph of a cluster $\mathcal{S}_k$, denoted by *cluster graph* of $\mathcal{S}_k$, is formed by the members of $\mathcal{S}_k$, and the links between them. In structured ad hoc networks, the nodes are organized in groups, and we assume that each node belongs to only one group. The set of groups $\mathcal{O}$ is defined as $\{\mathcal{O}_1,\dots,\mathcal{O}_T\}$ with T the number of groups. We note m_t the size of group $\mathcal{O}_t$, for $t \in \{1,2,\dots,T\}$.

B. Useful definitions related to coalition formation game theory

Coalitional games involve a set of players $\mathcal{N}$ who want to cooperate by forming *coalitions* in order to improve their positions in the game. Let us now recall some definitions from the coalition game theory [15] that are instrumental in the derivation of our clustering algorithm, starting with the notions of coalition and *coalition structure*.

Definition 1: A coalition structure (or coalition partition) is defined as the set $\Pi := \{\mathcal{S}_1,\dots,\mathcal{S}_K\}$ *where* $\mathcal{S}_k \subseteq \mathcal{N}$ *are disjoint coalitions verifying* $\cup_{k=1}^{K}\mathcal{S}_k = \mathcal{N}$.

Within a coalition structure we associate three different quantities to each coalition $\mathcal{S}_k$:

- The *revenue* $u(\mathcal{S}_k) \geq 0$ quantifies the worth of the coalition, with $u(\emptyset) = 0$. The expression of the revenue is a design parameter that depends on the goal of the coalitional game.
- The *cost* $c(\mathcal{S}_k)$ quantifies the cost of cooperation. In this paper, we use the cost to account for the constraints imposed to the coalitions. We have chosen in this work to use hard constraints using the following rule: $c(\mathcal{S}_k) = 0$ if all constraints are satisfied and $c(\mathcal{S}_k) = +\infty$ otherwise. Notice that soft constraints can be used as well, but would require a more complicated set of parameters tailored to the intra-cluster RRA process, which is out of the scope of this paper.
- The *value* $v(\mathcal{S}_k)$ is defined as the difference between the revenue achieved due to the cooperation and the cost of cooperation: $v(\mathcal{S}_k) = u(\mathcal{S}_k) - c(\mathcal{S}_k)$. As a consequence of our choice for c, note that $v(\mathcal{S}_k) = u(\mathcal{S}_k)$ when all the constraints are satisfied, $v(\mathcal{S}_k) = -\infty$ otherwise.

In this paper, the value of a coalition depends only on the members of this coalition. Therefore, the coalition formation games considered here are said to be in characteristic form [15]. Examples of functions are detailed in Sections III and IV.

A transfer of nodes from one coalition to another is called a *switch operation*:

Definition 2: A switch operation $\sigma_{k,\ell}(\mathcal{P})$ *is defined as the transfer of players* $\mathcal{P}$ *from* $\mathcal{S}_k \in \Pi$ *to* $\mathcal{S}_\ell \in \Pi \cup \{\emptyset\}$, $\sigma_{k,\ell}(\mathcal{P}) : \mathcal{S}_k \mapsto \mathcal{S}_k \setminus \mathcal{P}$, *and* $\mathcal{S}_\ell \mapsto \mathcal{S}_\ell \cup \mathcal{P}$.

Note 1: if $\mathcal{S}_\ell = \emptyset$, then $\sigma_{k,\ell}(\mathcal{P})$ leads to the formation of a new coalition $\mathcal{S}_\ell = \mathcal{P}$, thus increasing by one the number of coalitions. In that case the switch operation is noted $\sigma_{k,\emptyset}(\mathcal{P})$.

Note 2: if $\mathcal{P} = \mathcal{S}_k$, then $\sigma_{k,\ell}(\mathcal{P})$ leads to the merge of $\mathcal{S}_k$ with $\mathcal{S}_\ell$, thus decreasing by one the number of coalitions.

To determine if a switch operation improves the coalition structure, we now define the *switch operation gain*.

Definition 3: The switch operation gain $g(\sigma_{k,\ell}(\mathcal{P}))$ *associated with* $\sigma_{k,\ell}(\mathcal{P})$ *is defined as:*

$$g(\sigma_{k,\ell}(\mathcal{P})) := r_{\mathcal{P}}(\mathcal{S}_\ell \cup \mathcal{P}) - r_{\mathcal{P}}(\mathcal{S}_k), \tag{1}$$

with $r_{\mathcal{P}}(\mathcal{S}_k)$ *defined as:*

$$r_{\mathcal{P}}(\mathcal{S}_k) := v(\mathcal{S}_k) - v(\mathcal{S}_k \setminus \mathcal{P}). \tag{2}$$

The $r_{\mathcal{P}}(\mathcal{S}_k)$ quantity can be interpreted as the added value of having players $\mathcal{P}$ in coalition $\mathcal{S}_k$.

Considering only valid partitions (satisfying the constraints), equations (1) and (2) prove that the gain of a switch operation only depends on the values of the two involved coalitions, and not on the way the coalition values are shared among their members. This means that the coalition formation game proposed in this paper is with transferable utility (TU).

Let us now define the *preference relation* used by the players to compare two switch operations.

Definition 4: The preference relation $\succ$ *is defined as a complete and transitive binary relation between two switch operations* $\sigma_{k,\ell}(\mathcal{P}_i)$ *and* $\sigma_{k',\ell'}(\mathcal{P}_j)$ *such that:*

$$\sigma_{k,\ell}(\mathcal{P}_i) \succ \sigma_{k',\ell'}(\mathcal{P}_j) \Leftrightarrow g(\sigma_{k,\ell}(\mathcal{P}_i)) > g(\sigma_{k',\ell'}(\mathcal{P}_j)). \tag{3}$$

Similarly, we also define $\succeq$ *as:* $\sigma_{k,\ell}(\mathcal{P}_i) \succeq \sigma_{k',\ell'}(\mathcal{P}_j) \Leftrightarrow g(\sigma_{k,\ell}(\mathcal{P}_i)) \geq g(\sigma_{k',\ell'}(\mathcal{P}_j))$.

Using our notations, the *Nash-stability* [24] of a partition can be defined as:

Definition 5: A partition $\Pi = \{\mathcal{S}_1 \ldots, \mathcal{S}_K\}$ *is Nash-stable if* $\forall \mathcal{S}_k \in \Pi$, $\forall S_\ell \in \Pi \cup \{\emptyset\}$, $\forall i \in \mathcal{S}_k$, $g(\sigma_{k,\ell}(\{i\}) \leq 0$.

When the partition Π is Nash-stable, it means that there exists no single node switch operation with a strictly positive gain.

C. Generic coalition formation algorithm for clustering

In this section we propose a generic, distributed and asynchronous coalition formation algorithm to cluster the network. In Section III, this algorithm is applied to unstructured networks, and in Section IV, to structured network. To design our algorithm, we consider the coalition game theory framework, identifying coalitions as clusters, and players as nodes. In the sequel, we will use preferably cluster/node terminology.

The proposed algorithm is based upon comparisons of switch operation gains. It is *distributed* since the decision to perform a switch operation is done at each node. Moreover, the decision-making time points are assumed not to be coordinated between nodes and thus the algorithm is *asynchronous*. However, when a node is evaluating the possibility to perform a switch operation, it may consider other nodes of its current cluster. Notice that in our approach, there is no need to consider the notion of CH, unlike most of the solutions from the state of the art. To account for the asynchronism of the decision-making instants, nodes that are involved in a switch operation are set in a *busy* status. Nodes that are not in the busy status are said *available*. Before implementing a switch operation, availability of the involved nodes is checked. Switch operations are done between neighbor clusters; two clusters are neighbors if at least one node in one cluster is a neighbor of at least one node in the other cluster. A detailed example of a distributed and asynchronous model to support our proposed generic clustering algorithm is detailed in Section V-B.

When the network is static, we can prove that the algorithm converges to a stable solution where all constraints are satisfied. When the nodes are mobile, the network topology changes over time. As a consequence, a cluster fulfilling the constraints at one time may not satisfy them after some time. Thus, the algorithm needs to cope with this situation and to react to find a new cluster formation satisfying the constraints. Our generic clustering algorithm can then be summarized as follows. As soon as a node starts a decision-making process, it first checks if the constraints of its current cluster are satisfied. If the constraints are fulfilled, it applies the procedure $\mathbf{P}_1$ to operate the best switch operation described in Section II-C1. If because of the mobility of its members, at least one constraint is violated, it applies another procedure $\mathbf{P}_2$ described in Section II-C2. Each procedure includes a selection of candidate nodes for switch operations which is done according to common sense rules referred to as *heuristics* and noted

$\mathcal{H}_1$ for $\mathbf{P}_1$ and $\mathcal{H}_2$ for $\mathbf{P}_2$.

1) Procedure when constraints are fulfilled ($\mathbf{P}_1$): When a node $i \in \mathcal{S}_k$ starts a decision-making procedure and detects that its cluster does satisfy the constraints, it triggers the procedure $\mathbf{P}_1$ in order to search for a strictly positive gain switch operation and to implement it. Note that the departure (arrival) of nodes from a cluster always conducts to a decrease (increase) of the cluster revenue, respectively. Imposing a strictly positive gain ensures that the revenue increase is strictly larger than the revenue decrease, leading to a strict increase of the network social welfare. The principle of this procedure runs at each node i (summarized in Table I) splits into the three following successive steps:

1) **Selection of candidate switch operations (lines 1-9).** We first use the heuristic $\mathcal{H}_1$ to find the sets of candidate nodes for a switch operation, i.e., that could leave the cluster $\mathcal{S}_k$ to join a neighbor one. We assume that this set, denoted by $\{\mathcal{P}_{1,i}(a)\}_{a=1}^{A_1}$, includes at least the singleton $\{i\}$ itself, hence $A_1 \geq 1$. For each potential candidate set, the switch operation gain is evaluated against all the neighbor clusters. We keep as candidates the switch operations with strictly positive gain.
2) **Selection of one best switch operation (lines 10-11).** If the set of candidate switch operations is not empty, a best switch operation is selected among those with the largest switch operation gain. Otherwise, the procedure is terminated.
3) **Implementation of the switch operation (lines 12-14).** We first check if the identified switch operation can be performed, i.e., if all the nodes involved (actually $\mathcal{S}_k \cup \mathcal{S}_{\ell^*}$) are available. Then, if the condition is met, the selected switch operation is performed, otherwise it is dropped.

Given the network $\mathcal{G}$, the complexity of Procedure 1 is governed by the number of iterations of lines 4-7 in Table I, which is equal to the largest node degree $D(\mathcal{G}) < N$. Thus, the complexity of Procedure 1 is $O(D(\mathcal{G}))$.

2) Procedure when constraints are not fulfilled ($\mathbf{P}_2$): When a node $i \in \mathcal{S}_k$ starts a decision-making procedure and detects that its cluster does not satisfy the constraints because of the mobility of its members, it triggers a procedure to change the cluster topology looking for a new partition matching the constraints. The procedure chooses among three different actions: 1) do nothing, 2) some members of the cluster (including node i) join another cluster, or 3) some members of the cluster (including node i) form a new cluster. Action 1) happens when

TABLE I: Procedure $\mathbf{P}_1$ at node $i \in \mathcal{S}_k$ of the generic clustering algorithm when constraints are satisfied.

// Selection of candidate switch operations
1 Set $\mathscr{M} = \emptyset$.
2 Apply heuristic $\mathcal{H}_1$ to node i to get $\{\mathcal{P}_{1,i}(a)\}_{a=1}^{A_1}$, with $\mathcal{P}_{1,i}(a) \in \mathcal{S}_k$ and $A_1 \geq 1$.
3 **For each** $a \in \{1, \dots A_1\}$ **do**:
4 **For each** $\mathcal{S}_\ell$ neighbor of $\mathcal{S}_k$ **do**:
5 **If** $g(\sigma_{k,\ell}(\mathcal{P}_{1,i}(a))) > 0$ **then**:
6 Set $\mathscr{M} = \mathscr{M} \cup \sigma_{k,\ell}(\mathcal{P}_{1,i}(a))$.
7 **End If**.
8 **End For**.
9 **End For**.
10 **If** $\mathscr{M} \neq \emptyset$ **then**:
// Selection of one best switch operation
11 Find (a^*, ℓ^*) such that $\sigma_{k,\ell^*}(\mathcal{P}_{1,i}(a^*)) \succeq \sigma_{k,\ell}(\mathcal{P}_{1,i}(a))$, $\forall \sigma_{k,\ell}(\mathcal{P}_{1,i}(a)) \in \mathscr{M}$.
// Implementation of the switch operation
12 **If** the nodes involved in $\sigma_{k,\ell^*}(\mathcal{P}_{1,i}(a^*))$ are all available **then**:
13 The nodes in $\mathcal{P}_{1,i}(a^*)$ join $\mathcal{S}_{\ell^*}$.
14 **End If**.
15 **End If** //$\mathscr{M} \neq \emptyset$.

no candidate switch operation is identified. The principle of this procedure runs at each node i (summarized in Table II) splits into the three following successive steps:

1) **Selection of candidate switch operations (lines 1-10).** We first build the potential candidate sets of nodes denoted by $\{\mathcal{P}_{2,i}(a)\}_{a=1}^{A_2}$ according to $\mathcal{H}_2$. When the potential candidate set is not empty, we evaluate the value $r_{\mathcal{P}}$ in (2) of the merge of all the potential candidate sets against all the neighbor clusters and keep those with strictly positive gains.
2) **Selection of one best switch operation (line 11).** The best switch operation selection is one among those with the largest $r_{\mathcal{P}}$ value.

3) **Implementation of the switch operation (lines 12-16).** We first check if the identified switch operation can be performed, i.e., if all the nodes involved (actually $\mathcal{P}_{2,i}(a^*) \cup \mathcal{S}_{\ell^*}$) are available. If this condition is met, the selected switch operation is performed, otherwise nodes available in $\mathcal{P}_{2,i}(a)$ form a new cluster.

TABLE II: Procedure $\mathbf{P}_2$ at node $i \in \mathcal{S}_k$ of the generic clustering algorithm when constraints are not satisfied.

// Selection of potential candidate switch operations
1 Apply heuristic $\mathcal{H}_2$ to node i to get $\{\mathcal{P}_{2,i}(a)\}_{a=1}^{A_2}$, with $\mathcal{P}_{2,i}(a) \in \mathcal{S}_k$ and $A_2 \geq 0$.
// Selection of candidate switch operations
2 **If** $A_2 > 0$ **then**:
3 **For each** $a \in \{1, \dots A_2\}$ **do**:
4 Set $\mathscr{M} = \{\sigma_{k,\emptyset}(\mathcal{P}_{2,i}(a))\}$.
5 **For each** $\mathcal{S}_\ell$ neighbor of $\mathcal{S}_k$ **do**:
6 **If** $r_{\mathcal{P}_{2,i}(a)}(\mathcal{S}_\ell \cup \mathcal{P}_{2,i}(a)) > 0$ **then**:
7 Set $\mathscr{M} = \mathscr{M} \cup \sigma_{k,\ell}(\mathcal{P}_{2,i}(a))$.
8 **End If**.
9 **End For**.
10 **End For**.
// Selection of one of the best switch operation
11 Find (a^*, ℓ^*) such that $r_{\mathcal{P}_{2,i}(a^*)}(\mathcal{S}_{\ell^*} \cup \mathcal{P}_{2,i}(a^*)) \geq r_{\mathcal{P}_{2,i}(a)}(\mathcal{S}_\ell \cup \mathcal{P}_{2,i}(a))$, $\forall \sigma_{k,\ell}(\mathcal{P}_{2,i}(a)) \in \mathscr{M}$.
// Implementation of the switch operation
12 **If** the nodes involved in $\sigma_{k,\ell^*}(\mathcal{P}_{2,i}(a^*))$ are all available **then**:
13 Nodes $\mathcal{P}_{2,i}(a^*)$ join $\mathcal{S}_{\ell^*}$.
14 **Else**:
15 The nodes in $\mathcal{P}_{2,i}(a^*)$ that are available form a new cluster.
16 **End If**.
17 **End If**.

Although $\mathbf{P}_2$ is organized in the same lines as in $\mathbf{P}_1$, its differs along the following features:

- Unlike $\mathcal{H}_1$, $\mathcal{H}_2$ may return an empty set for the potential candidate since moving node i (and some of its neighbors) may not improve the fulfillment of the constraints.
- Switch operations are selected using the value of $r_{\mathcal{P}}$ in (2) instead of the switch operation gain g in (1). The reason is the following: when at least one constraint in $\mathcal{S}_k$ is not fulfilled, then $v(\mathcal{S}_k) = -\infty$ and $r_{\mathcal{P}}(\mathcal{S}_k)$ is undefined. As a consequence, the switch operation gain cannot be used. However if there is one $\mathcal{S}_\ell$ such that $\mathcal{S}_\ell$ and $\mathcal{S}_\ell \cup \mathcal{P}$ satisfy the constraints, $r_{\mathcal{P}}(\mathcal{S}_\ell \cup \mathcal{P})$ still exists. Remembering that $r_{\mathcal{P}}(\mathcal{S}_\ell \cup \mathcal{P})$ quantifies the gain associated with the arrival of $\mathcal{P}$ in $\mathcal{S}_\ell$, we propose to use this quantity instead of g.
- When the nodes are not available to implement the selected switch operation, $\mathbf{P}_2$ creates a new cluster instead of dropping the switch. We justify this choice by the fact that dropping the switch would not resolve any of the constraints infringement, whereas creating a new cluster reduces the number of nodes that belong to a cluster not fulfilling the constraints.
- Unlike $\mathbf{P}_1$, the cluster $\mathcal{S}_k$ may not satisfy the constraints after performing $\mathbf{P}_2$, for instance because no switch operation is performed. The constraint infringement resolution may occur by the other nodes of the cluster, when they run Table II.

Like Procedure 1, the complexity of Procedure 2 is $O(\Delta(\mathcal{G}))$.

The important design parameters of this algorithm are: *i*) the heuristics $\mathcal{H}_1$ and $\mathcal{H}_2$ to select the candidate sets of nodes $\mathcal{P}_{1,i}(a)$ and $\mathcal{P}_{2,i}(a)$, and *ii*) the revenue and the associated cost functions used to calculate switch operation gains (either for g or $r_{\mathcal{P}}$). Various heuristics, cluster revenue and cost functions can be chosen depending on the type of network. Several examples are given in Section III dedicated to unstructured networks and in Section IV to structured networks.

D. Convergence properties

When the network topology is fixed and the algorithm is initialized by clusters fulfilling the constraints (the simplest being to set each node as a singleton cluster), then the following result holds:

Result 1: For a fixed network topology and starting from any initial partition Π_0 of $\mathcal{N}$ for which the clusters satisfy the constraints, the cluster formation algorithm maps to a sequence of switch operations which converges in a finite number of iterations to a final partition $\tilde{\Pi}$.

Proof See Appendix A. ∎

Since the candidate set of nodes returned by $\mathcal{H}_1$ includes the node i itself, we directly get:

Result 2: The final partition $\tilde{\Pi}$ achieved in Result 1 is Nash-stable.

Note that the final Nash-stable partition $\tilde{\Pi}$ is in general not unique. The final partition depends on the order in which the network nodes make their switch operation decisions. As soon as the switch operation times are driven by random processes (see e.g., Section V-B), experience shows that different final Nash-stable partitions can be found.

E. On the cluster constraints and revenue function

As mentioned in the introduction, we force some constraints to the clusters in order to ensure an efficient RRA. In addition to the connectivity constraint, we impose a maximum number of nodes per cluster noted n_{max} and a maximum cluster diameter noted d_{max}. Let us note n_k the size (or number of members) of cluster $\mathcal{S}_k$, and $d(\mathcal{S}_k)$ the diameter of its graph. The cluster constraints are noted by:

- $\rho_1(\mathcal{S}_k)$: the cluster graph of $\mathcal{S}_k$ is connected,
- $\rho_2(\mathcal{S}_k)$: $n_k \leq n_{max}$,
- $\rho_3(\mathcal{S}_k)$: $d(\mathcal{S}_k) \leq d_{max}$,

and the cost within a cluster $\mathcal{S}_k$ is defined as:

$$c(\mathcal{S}_k) := \chi(\rho_1(\mathcal{S}_k) \cap \rho_2(\mathcal{S}_k) \cap \rho_3(\mathcal{S}_k)), \tag{4}$$

with $\chi(condition(\mathcal{S}_k)) := 0$ if $condition(\mathcal{S}_k)$ is satisfied, $+\infty$ otherwise.

For the same reason, we want to build clusters that include as many members as possible (limited by the size constraint n_{max}). We will refer this goal to as $\mathbf{G}_1$:

Goal 1 ($\mathbf{G}_1$): Build clusters whose size is maximal, i.e., equal to n_{max}.

To achieve $\mathbf{G}_1$, the revenue function should always allow the merge of two clusters, as long as the constraints are satisfied. Let us call this property Condition 1 that can be expressed as:

Condition 1: The cluster revenue function must verify $g(\sigma_{k,\ell}(\mathcal{S}_k)) > 0$, $\forall(\mathcal{S}_k, \mathcal{S}_\ell) \in \Pi^2$, $k \neq \ell$, as long as $\mathcal{S}_k \cup \mathcal{S}_\ell$ satisfies the constraints.

When a revenue function satisfies Condition 1, then the switch operation consisting in a singleton cluster merging with one of its neighbor singleton clusters, has always a strictly positive gain. Thus, these two singleton clusters are always allowed to merge through the action of the proposed algorithm. Consequently, if the network is initialized with each node set as a singleton cluster (as we recommend and do in Section V), then using a revenue function satisfying

Condition 1 ensures that the network is not blocked in its initial configuration. Experience shows that there are a lot of revenue functions that do not satisfy Condition 1, and for which the network does remain blocked in its initial all singleton clusters state.

III. Clustering algorithm for unstructured mobile ad hoc networks

In this section we specify the generic clustering algorithm to the unstructured network case by designing dedicated revenue function and heuristics. The resulting algorithm is called Clustering with Link Quality (CLQ).

A. *Revenue function*

In unstructured networks, we would like to build stable clusters. To do that, we suggest to build these clusters by gathering nodes with high link capacities between each other. We will refer this goal to as $\mathbf{G}_2$:

Goal 2 ($\mathbf{G}_2$): Build clusters with high link capacities.

In order to achieve $\mathbf{G}_2$, we propose to define the revenue function as the sum capacity of all the intra-cluster links:

$$u_{\text{un}}(\mathcal{S}_k) := \sum_{i \in \mathcal{S}_k} \sum_{j \in \mathcal{S}_k | (i,j) \in \mathcal{E}} \kappa(i,j). \tag{5}$$

where $\kappa(i,j)$ denotes the *capacity* of link (i,j). This revenue function also achieves $\mathbf{G}_1$ since it holds the following result:

Result 3: The revenue function defined in (5) satisfies Condition 1.

Proof See Appendix B. ∎

B. *Heuristics for node selection*

1) $\mathcal{H}_1$ in procedure $\mathbf{P}_1$: We note $\mathcal{H}_1^{\text{un}}$ the corresponding heuristic that returns the set $\{\mathcal{P}_{1,i}(a)\}_{a=1}^{A_1}$. Since in unstructured networks there is not any particular reason to select any other node than itself, i.e., $\{i\}$, $\mathcal{H}_1^{\text{un}}$ is defined as follows: set $A_1 = 1$ with $\mathcal{P}_{1,i}(1) = \{i\}$.

2) $\mathcal{H}_2$ in procedure $\mathbf{P}_2$ (adaptation to mobility): We note $\mathcal{H}_2^{\text{un}}$ the corresponding heuristic that returns the set $\{\mathcal{P}_{2,i}(a)\}_{a=1}^{A_2}$. Here, at least one constraint in the selected cluster is not satisfied, i.e., connectivity or diameter, since the mobility does not increase the cluster size. We identify three different situations for node $\{i\}$: 1) it has a lot of neighbors and it could be interesting for

it to form a new cluster with its neighbors, 2) it has a few neighbors in the cluster meaning that it is very likely involved in the constraints violation, especially for diameter if it is in the border of the cluster. It would be then beneficial for it to leave the cluster, 3) it is neither in case (1) nor (2) and we suggest doing nothing. From the discussion above, we propose the following heuristic considering the extreme situations for (1) (the most connected) and (2) (the least connected):

- If node i has the highest degree within its cluster graph, then set $A_2 = 1$ with $\mathcal{P}_{2,i}(1)$ defined as all cluster members that are in the $\lfloor d_{max}/2 \rfloor$-hop neighborhood of i. When the diameter constraint is equal to 2, $\mathcal{P}_{2,i}(1)$ includes node i and all its 1-hop neighbors within the cluster, i.e., $\mathcal{P}_{2,i}(1) = \{i\} \cup \{j \in \mathcal{S}_k | (i,j) \in \mathcal{E}\}$.
- If node i has the lowest degree within its cluster graph, then set $A_2 = 1$ with $\mathcal{P}_{2,i}(1) = \{i\}$.
- In the other cases, node i chooses to remain in the cluster and to wait for the action of other cluster members, then set $A_2 = 0$.

Notice that there are as many choices for the heuristics as one can imagine. The heuristics proposed here can thus be changed. However, when designing a heuristic for node selection, one has to keep in mind its computational complexity and the amount of information needed, often leading to a trade-off between complexity and performance.

IV. Clustering algorithm for structured mobile ad hoc networks

In this section we specify the generic clustering algorithm to the structured network case by designing dedicated revenue function and heuristics. The resulting algorithm is called Clustering with Operational Groups (COG). Hereafter, we need the following additional notation: let $m_{t,k}$ be the number of members of group $\mathcal{O}_t$ in the cluster $\mathcal{S}_k$.

A. *Revenue function*

As discussed in Section I, in structured networks, the members of the same group exchange the main part of their traffic within their group, and intra-cluster links benefit from a RRA which is more efficient than the one associated with the inter-cluster links. Therefore, as much as possible, we want to collect the members of the same group into a single cluster. We will refer this goal to as $\mathbf{G}_3$:

Goal 3 ($\mathbf{G}_3$): Build clusters including the highest number of members of the same group.

As already seen in Section II-E, we also want to achieve $\mathbf{G}_1$. We will show that we can define for each of the two goals an adapted revenue function. Therefore, we suggest to define a revenue function for structured networks as the linear combination of these two revenue functions:

$$u_{\text{st}}(\mathcal{S}_k) := u_1(\mathcal{S}_k)\epsilon + u_2(\mathcal{S}_k)(1-\epsilon),\ \epsilon \in (0,1), \tag{6}$$

where

- u_1 is a revenue function adapted to $\mathbf{G}_1$. Since goal $\mathbf{G}_1$ is related to the cluster size, we consider functions depending only on the cluster size:

$$u_1(\mathcal{S}_k) := f_1(n_k), \tag{7}$$

- u_2 is a revenue function adapted to $\mathbf{G}_3$. Since goal $\mathbf{G}_3$ is related to the number of nodes of a group in a cluster, we consider functions depending only on the number of nodes per group in the cluster:

$$u_2(\mathcal{S}_k) := \sum_{t \in \mathcal{I}(\mathcal{S}_k)} f_{2,t}(m_{t,k}) \tag{8}$$

  with $\mathcal{I}(\mathcal{S}_k)$ the set of the indices of groups with at least one member in cluster $\mathcal{S}_k$.

We now want to determine relevant functions for f_1 and $f_{2,t}$. Regarding f_1, we use the following result.

Result 4: Let two different clusters $\mathcal{S}_k$ and $\mathcal{S}_\ell$ satisfying the constraints, and a set of nodes $\mathcal{P} \subset \mathcal{S}_k$ such that

i) $|\mathcal{S}_\ell \cup \mathcal{P}| > |\mathcal{S}_k|$ *and*

ii) $\mathcal{S}_\ell \cup \mathcal{P}$ *and* $\mathcal{S}_k \setminus \mathcal{P}$ *satisfy the constraints.*

If f_1 is strictly convex then u_1 defined by (7) verifies $g(\sigma_{k,\ell}(\mathcal{P})) > 0$.

Proof See Appendix C. ∎

As a corollary, f_1 also verifies Condition 1 (apply Result 4 with $\mathcal{P} = \mathcal{S}_k$). Consequently, we propose to select a strictly convex function f_1 to achieve $\mathbf{G}_1$. Note that this result also holds for unstructured networks since it depends only on the cluster size. As the simplest strictly convex function is the second-order monomial, we propose to use

$$f_1(n_k) := \frac{n_k^2}{n_{max}^2}, \tag{9}$$

where n_{max}^2 normalizes u_1 in $[0,1]$.

Regarding $f_{2,t}$, we use the following result.

Result 5: Let two different clusters $\mathcal{S}_k$ and $\mathcal{S}_\ell$ satisfying the constraints, such that

i) $\mathcal{P}$ *is a subset of* $\mathcal{O}_t \cap \mathcal{S}_k$,

ii) $m_{t,\ell} + |\mathcal{P}| > m_{t,k}$, *and*

iii) $\mathcal{S}_\ell \cup \mathcal{P}$ *and* $\mathcal{S}_k \setminus \mathcal{P}$ *satisfy the constraints.*

If $f_{2,t}$ is strictly convex then u_2 defined by (8) verifies $g(\sigma_{k,\ell}(\mathcal{P})) > 0$.

Proof See Appendix D. ∎

Consequently, the strict convexity of $f_{2,t}$, ensures that a switch operation leading to a larger number of members of the same group in the receiving cluster than in the departing cluster has a strictly positive gain. It will thus contribute to achieving $\mathbf{G}_3$.

The same way as for f_1, we select the simplest second-order monomial function for $f_{2,t}$:

$$f_{2,t}(m_{t,k}) := \frac{m_{t,k}^2}{T m_t^2}, \tag{10}$$

where $T m_t^2$ normalizes u_2 in $[0, 1]$.

The revenue function previously defined holds the following property.

Result 6: The revenue function defined as (6) with (7)-(10) satisfies Condition 1.

Proof See Appendix E. ∎

We now discuss the design of ϵ to control the trade-off between goals $\mathbf{G}_1$ and $\mathbf{G}_3$. Let us first state the following result.

Result 7: Let three different clusters satisfying the constraints be $\mathcal{S}_k$, $\mathcal{S}_\ell$ and $\mathcal{S}_q$, with each of them having some nodes belonging to the group $\mathcal{O}_t$. Let $\mathcal{U}_q \subset \mathcal{S}_q$ that contains some nodes only of group $\mathcal{O}_t$ with $n_{\mathcal{U}} := |\mathcal{U}_q| \leq m_{t,q}$. We assume that $\mathcal{S}_k \cup \mathcal{U}_q$ and $\mathcal{S}_\ell \cup \mathcal{U}_q$ satisfy the constraints. Assuming that $\forall t, |\mathcal{O}_t| \leq n_{max}$ and using the revenue function (6) with (7)-(10), the following properties hold:

i) If $m_{t,k} = m_{t,\ell}$ and $|\mathcal{S}_k| > |\mathcal{S}_\ell|$, then $g(\sigma_{q,k}(\mathcal{U}_q)) > g(\sigma_{q,\ell}(\mathcal{U}_q))$, $\forall \epsilon$.

ii) If $m_{t,k} = m_{t,\ell}$ and $|\mathcal{S}_k| = |\mathcal{S}_\ell|$, then $g(\sigma_{q,k}(\mathcal{U}_q)) = g(\sigma_{q,\ell}(\mathcal{U}_q))$, $\forall \epsilon$.

iii) If $m_{t,k} > m_{t,\ell}$, and $\forall |\mathcal{S}_k|, \forall |\mathcal{S}_\ell|$, then $g(\sigma_{q,k}(\mathcal{U}_q)) > g(\sigma_{q,\ell}(\mathcal{U}_q))$ as soon as $\epsilon < \epsilon^* := \frac{1}{1+T}$.

Proof See Appendix F. ∎

First, when the number of nodes belonging to group $\mathcal{O}_t$ in cluster $\mathcal{S}_k$ and $\mathcal{S}_\ell$ are equal, Result 7 tells that, whatever the value of ϵ: *i)* the algorithm selects the switch of $\mathcal{U}_q$ towards the cluster with the largest number of nodes, thus fulfilling $\mathbf{G}_1$, *ii)* if cluster $\mathcal{S}_k$ and $\mathcal{S}_\ell$ have the same number of nodes, both switch operations are even. Second, when the number of nodes belonging

to group $\mathcal{O}_t$ in cluster $\mathcal{S}_k$ and $\mathcal{S}_\ell$ are different, let say $m_{t,k} > m_{t,\ell}$, then, for any $\epsilon < \epsilon^*$, the algorithm always selects the switch of $\mathcal{U}_q$ towards the cluster which has the largest number of nodes of group $\mathcal{O}_t$, i.e., $\mathcal{S}_k$, regardless of the size of the clusters, and more specifically even if $|\mathcal{S}_k| < |\mathcal{S}_\ell|$. In that condition, it thus always fulfills $\mathbf{G}_3$ over $\mathbf{G}_1$.

B. Heuristics for node selection

We assume that node i running the algorithm in cluster $\mathcal{S}_k$ belongs to group $\mathcal{O}_t$. For structured networks, the heuristic should select the set denoted by $\mathcal{L}$, including node i, gathering the maximum number of members of group $\mathcal{O}_t$, and fulfilling the constraints. Thus, in order to build $\mathcal{L}$, we first identify i^* the node among the neighbors of i in group $\mathcal{O}_t$ that has the highest degree. Then, we select the set of nodes including i^* that has the largest cardinality and which respects the constraints. Note that when the diameter constraint is equal to two, $\mathcal{L}$ is obtained by considering all the neighbors of i^* (which includes i by construction). This approach is used for the two following heuristic used in procedures $\mathbf{P}_1$ and $\mathbf{P}_2$.

1) $\mathcal{H}_1$ in procedure $\mathbf{P}_1$: We note $\mathcal{H}_1^{\text{st}}$ the corresponding heuristic that returns the set $\{\mathcal{P}_{1,i}(a)\}_{a=1}^{A_1}$. It is defined as: $A_1 = 2$, with $\mathcal{P}_{1,i}(1) = \{i\}$ and $\mathcal{P}_{1,i}(2) = \mathcal{L}$.

2) $\mathcal{H}_2$ in procedure $\mathbf{P}_2$ (adaptation to mobility): We note $\mathcal{H}_2^{\text{st}}$ the corresponding heuristic that returns the set $\{\mathcal{P}_{2,i}(a)\}_{a=1}^{A_2}$. It is defined as: $A_2 = 1$, with $\mathcal{P}_{2,i}(1) = \mathcal{L}$.

V. Numerical results

In this section we simulate our proposed algorithms (CLQ and COG) and analyze their performance. In case of unstructured ad hoc networks, CLQ is compared to three standard clustering algorithms named LCC [5] (as an extension of LID [4]), VOTE [6], and SECA [10]. In case of structured ad hoc networks, COG is compared to a naive algorithm called 1-group 1-cluster (1G1C), whose goal is to force all members of a group to be in the same cluster as soon as the constraints can be satisfied. If not, simple mechanisms are carried out to satisfy the constraints. Static and mobile configurations are tested.

A. Simulation setup

The nodes are deployed in a 1.5 km $\times$ 1.5 km square area as explained in Section V-C (resp. V-D) for the unstructured networks (resp. for the structured networks). We assume that

the radio range is equal to $d_{\text{ref}} = 250$ m and thus two nodes i and j such that $d_{i,j} > d_{ref}$ do not communicate. The average SNR of the link (i, j) is defined by $\Gamma(i, j) := -10\alpha \log(d_{i,j}/d_{\text{ref}})$, with $\alpha = 4$. The cluster diameter constraint is set to $d_{max} = 2$. All the results are obtained by averaging over 100 different random networks. The simulation duration is fixed to 5000 units of time. The unit of time could be, e.g., one second or one millisecond, depending on the performance of the radio access scheme. In mobile configurations, the duration of the warm-up period is set to 500 units of time. This warm-up period accounts for the transient phase for the algorithm to converge to its permanent phase. During this period, the nodes move according to the scenario but no performance metric is measured to disregard the initial transient effects.

B. Distributed asynchronous model

We specify here the way distributed and asynchronous decision-making is handled in the simulation. Remember that a node $i \in \mathcal{S}_k$ can be either in a *Waiting* or *Busy* (i.e., not available) state. According to these states, the simulation behaves as the following:

- **Waiting state**: node i is not involved in current operations done by the clustering algorithm. However, it is learning information needed to run the clustering algorithm. It stays in this state during a duration δ_w which is randomly chosen according to an exponential distribution with parameter λ. It may leave this state in the two following cases:
  1) After a duration δ_w, it checks its cluster constraints. Then two cases may occur:
     a) If the constraints are satisfied, then it applies the procedure $\mathbf{P}_1$ of the clustering algorithm described in Table I. If it decides to move to another cluster, it then switches to the *Busy* state to account for the time needed to operate this change.
     b) If the constraints are not satisfied, then it applies the procedure $\mathbf{P}_2$ of the clustering algorithm described in Table II. If it decides to perform a change in the cluster, it then switches to the *Busy* state to account for the time needed to operate this change.
  2) When the node i becomes involved in a cluster modification decided by another node j and enters into the *Busy* state.
- **Busy state**: it means that the node is involved in a cluster modification initiated by itself or another node. The duration of this state is deterministic and set equal to δ_b to account for the time spent to exchange information between nodes inside the cluster. This is applicable to our

algorithms COG and CLQ and it is the price to pay for having fully distributed algorithms. For the reference algorithms from the literature that are all based on CH election, there is no need for such exchange of information (only the CH is involved) and we will then set $\delta_b = 0$. Notice that if the nodes involved in this cluster modification are in the *Busy* state (because already involved in other cluster modifications simultaneously), then this cluster modification is canceled and the node i goes back to *Waiting* state.

The above description can be modeled as a state machine as depicted in Fig. 1. Notice that we assume that the time required for running the procedures $\mathbf{P}_1$ and $\mathbf{P}_2$ is zero. Unless otherwise stated, we set: $\lambda = 5$, $\delta_b = 0.5$ for COG and CLQ, and $\delta_b = 0$ for reference algorithms.

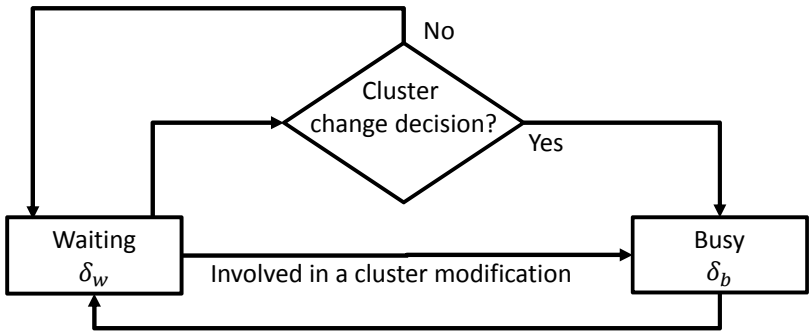

Fig. 1: State machine model for asynchronous clustering algorithm.

C. Performance for unstructured ad hoc networks

In this section, we compare our proposed algorithm CLQ with algorithms from the state of the art: the well-known reference LCC [5], VOTE which forces the cluster size to be less than a target threshold [6], and the recent SECA which takes the link quality into account [10]. Unless otherwise stated, we set $n_{max} = 20$ in VOTE and CLQ. The design parameters for SECA (defined in [10]) are $w_{\text{cd}} = 0.2$, $w_{\text{M}} = 0.4$, $w_{\text{SL}} = 0.4$, and $q_d = 0.1$. We consider a Gaussian channel to evaluate (5) setting $\kappa(i,j) = \log_2(1 + \Gamma(i,j))$.

We start by considering static configurations where the nodes are deployed randomly following a uniform distribution. In Fig. 2, we plot the number of iterations required by CLQ to converge to its final Nash-stable partition. Even if the proposed coalition formation algorithm requires slightly more iterations than the reference algorithms, this number remains low (< 20).

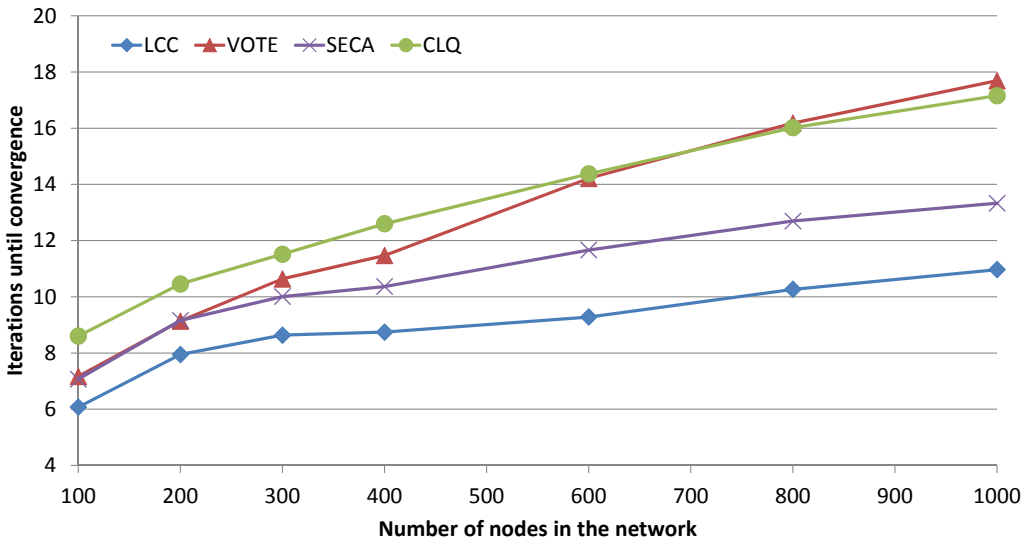

Fig. 2: Iterations required to converge vs. N in static unstructured networks.

In Fig. 3, we display the CLQ convergence duration vs. δ_b when $N = 100$. We observe that the duration grows linearly w.r.t. δ_b. As reference, we also plot the values for the other algorithms (LCC, VOTE, SECA) for which $\delta_b = 0$ (first points of Fig. 2). Decreasing δ_b enough in CLQ, i.e., by optimizing the exchange protocol, would allow to obtain almost the same convergence duration as the reference algorithms.

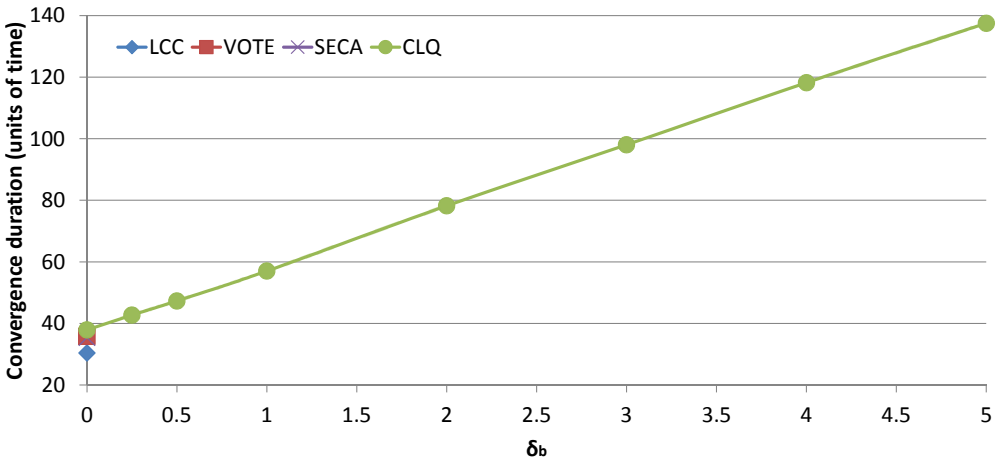

Fig. 3: Convergence duration vs. δ_b, in static unstructured networks for $N = 100$, when $\lambda = 5$ units of time.

In Fig. 4, we plot the cluster size vs. N. It is a non-decreasing function in the number of nodes in the networks. More precisely, the cluster size is limited by $n_{max} = 20$ for CLQ and VOTE, and increases linearly for LCC and SECA (that do not control the cluster size).

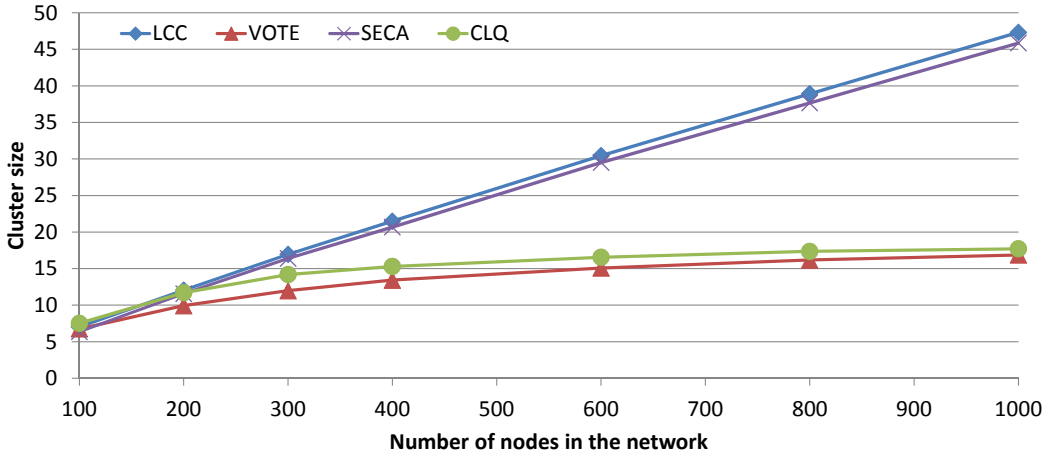

Fig. 4: Cluster size vs. N in static unstructured networks.

In Fig. 5, we show the distribution of the cluster size with $N = 300$. In this figure, the notation '$>$' on the x axis means that the cluster size is strictly greater than $n_{max} = 20$. Our simulations also show that CLQ builds *i*) the lowest number of singletons, which is of great interest since singleton clusters are inefficient for network performance, and *ii*) the highest number of maximum size clusters, thus achieving $\mathbf{G}_1$. Moreover CLQ always satisfy the cluster size constraints whereas LCC and SECA lead to very large clusters. For example, when $N = 1000$, the average size of the clusters formed by LCC is 47, with a lot of 200 node clusters, which is a very bad case for intra-cluster RRA efficiency.

In Fig. 6, we plot the intra-cluster link capacity vs. N. We observe that CLQ always ensures the highest link capacity compared to the state-of-the-art, and thus achieves $\mathbf{G}_2$. Compared to SECA which was also designed to maximize the link capacity, CLQ brings about $+18\%$ gain.

We now consider mobile networks for which nodes moves between waypoints, whose coordinates are updated once every $\tau = 20$ units of time. Between waypoints, nodes follow a uniform rectilinear motion with a speed limited by v_{max}. Their coordinates are updated once every second. In order to verify that small clusters should be more stable than large ones as expected, we simulated CLQ and VOTE for two values of the maximum size: $n_{max} \in \{10, 20\}$.

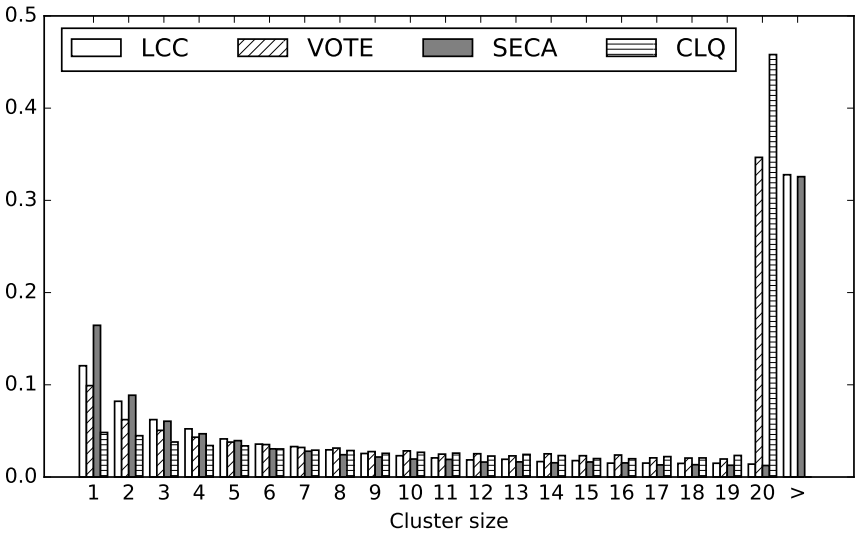

Fig. 5: $p(|\mathcal{S}_k|)$ vs. $|\mathcal{S}_k|$ in static unstructured networks for $N = 300$.

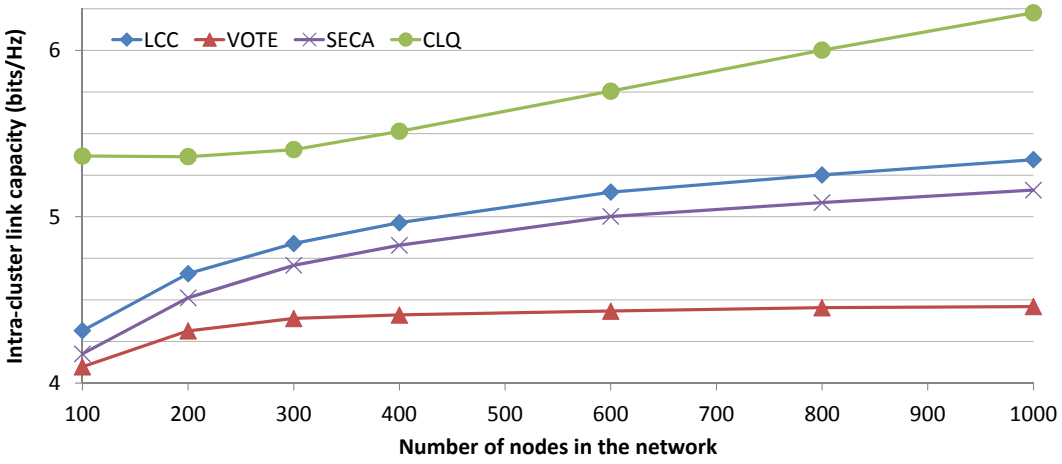

Fig. 6: Intra-cluster link capacity vs. N in static unstructured networks.

We denote by CLQ n and VOTE n when the cluster size constraint is $n_{max} = n$.

In Fig. 7, we display the cluster life time vs. v_{max}. These curves first confirm that building smaller clusters improves their stability. The cluster life time for CLQ 10 is on average twice the one induced by CLQ 20 (this gain decreases with v_{max}). VOTE also benefits from a lower cluster size: if $n_{max} = 10$, then the cluster life time is on average 40% higher than if $n_{max} = 20$. Additionally, regardless of v_{max}, the most stable clusters are obtained by CLQ. The algorithms LCC and SECA achieve similar performance, and VOTE 10 and VOTE 20

are the worst performers. The good performance of CLQ is a consequence of its capability to achieve $\mathbf{G}_2$: maximizing the sum capacity of intra-cluster links is only possible if on average, each intra-cluster link has a high capacity, which also means that this link is robust in presence of mobility.

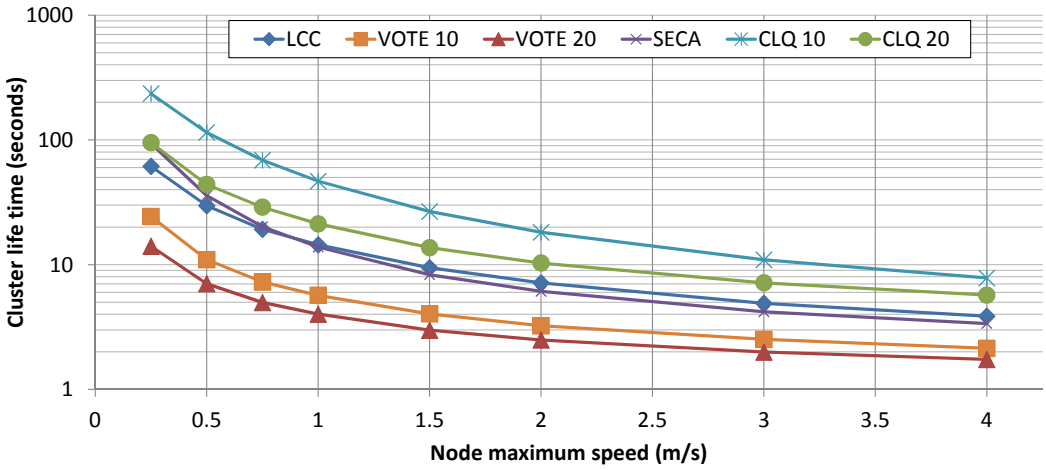

Fig. 7: Cluster life time vs. maximum node speed in 100 node mobile unstructured networks.

D. Performance for structured ad hoc networks

In this section we present the simulation results for COG. We set the group size $m_t = 10$. We denote by COG n the algorithm when the cluster size constraint is $n_{max} = n$. The maximum number of groups T is equal to 100, when $N = 1000$. Consequently, following Result 7, we set $\epsilon = 10^{-5} < \epsilon^* = 1/(1+T) \simeq 9.9\ 10^{-3}$ in (6), to favor $\mathbf{G}_3$ over $\mathbf{G}_1$.

Within the deployment area, the nodes follow a modified RPGM model [25]. At the beginning of the simulation, a randomly located virtual center $A_t(0)$ is associated with each group t, and all group t members are deployed randomly in a disk of radius d_{ref} centered at $A_t(0)$. In mobile networks, time is split into fixed duration intervals of Λ seconds. At time $k\Lambda$ (i.e., the beginning of interval number $k \in \mathbb{N}$), the virtual center $A_t(k+1)$ of group t is randomly selected, and the coordinates of group t members at time $(k+1)\Lambda$ are randomly chosen in the disk of radius d_{ref} centered at $A_t(k+1)$. Then during time interval $[k\Lambda, (k+1)\Lambda)$, each node follows a uniform rectilinear motion with a speed limited to a maximum v_{max}. Notice that when new coordinates are selected for a virtual center $A_t(k)$, if the distance between $A_t(k)$ and the deployment area

boundaries is smaller than d_{ref}, then a new location is drawn to make sure that no group member is placed outside of the deployment area.

Inasmuch as the literature is very poor in the context of structured ad hoc network, we have decided to create a naive centralized algorithm called *one group-one cluster* and denoted by 1G1C in order to exhibit the potential interest of our solution for static networks. The 1G1C algorithm is described as follows: *i*) for each group $\mathcal{O}_t$, find the node with the highest degree in the subgraph of $\mathcal{O}_t$ and build a cluster including this node and its 1-hop neighbors, *ii*) for each node not yet member of a cluster, assign it to an existing cluster while making sure that the constraints are satisfied, and build a new singleton cluster otherwise.

Let us first focus on static configurations. In Fig. 8, we plot the cluster group diversity (CGD), defined as the average number of groups per cluster, versus N. We see that when n_{max} is close

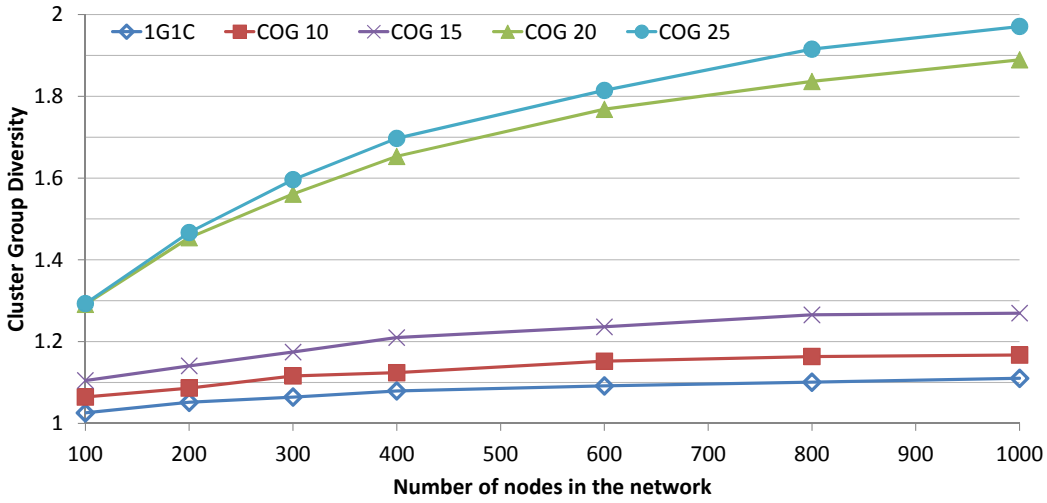

Fig. 8: Cluster Group Diversity vs. N in static structured networks.

to the size of the group, the corresponding COG have similar behavior as 1G1C which almost forces one cluster to be composed by one group only, and so achieves $\mathbf{G}_3$. In contrast, when n_{max} is larger than the group size, the corresponding COG allows to merge more than one group within one cluster and so achieves $\mathbf{G}_1$ too. Thus, with COG, intra-group communications are mainly intra-clusters, which ensures a good user QoS.

Let us now consider inter-group communications. When COG is allowed to build clusters including several entire groups, the number of inter-cluster links for these communications is

decreased, thus improving user QoS. To assess this gain, we consider p_x^{A} the number of paths with x inter-cluster links in the partition formed by algorithm A. Let us define $\delta_x^{\mathrm{A}} := p_x^{\mathrm{A}} - p_x^{\mathrm{1G1C}}$. Fig. 9 plots the values of δ_x^{A} achieved by the algorithms COG 10, COG 15, COG 20 and COG 25, for $N = 300$. This figure confirms our expectations. When two groups can be included in a single cluster ($\mathrm{A} \in \{\mathrm{COG}\ 20, \mathrm{COG}\ 25\}$), then *i*) the number of paths with zero or one inter-cluster links is increased, and logically *ii*) the number of paths with more than two such links is decreased.

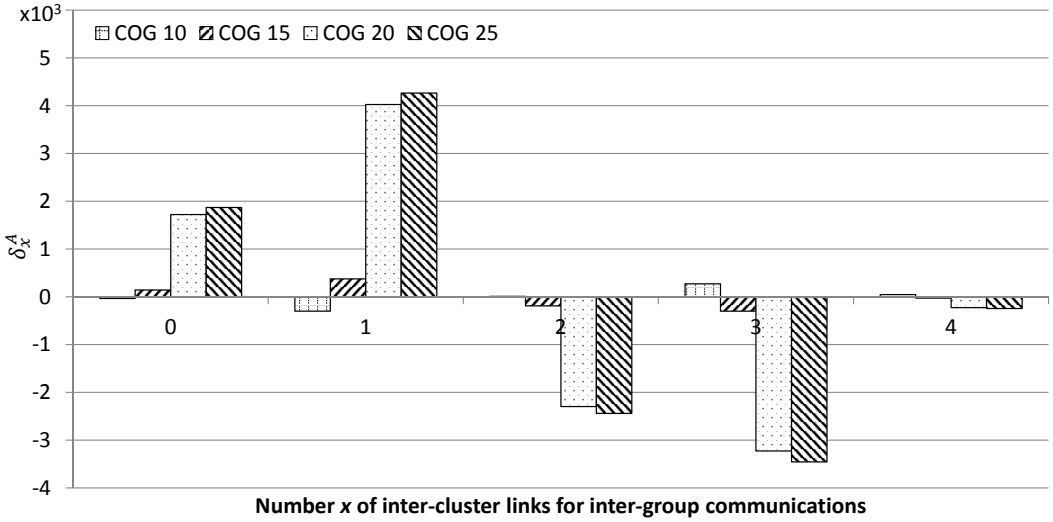

Fig. 9: Difference in number of paths with x inter-cluster links inter group communications between 1G1C and COG (COG 10, COG 15, COG 20, COG 25) in static structured networks for $N = 300$.

We now consider mobile configurations. In Fig. 10, we plot the cluster life time versus v_{max} for different values of $n_{\max}$. When n_{max} if lower than twice the group size, the clusters are mainly composed of a single group and so are very stable: the clusters formed at the beginning of the simulation are nearly never modified. This is an expected consequence of our node deployment scheme which ensures that within a group, the nodes are almost always at two radio hops. When n_{max} is at least equal to twice the group size, then some clusters include two groups. During the simulation, the groups move independently, leading to two kinds of cluster modifications: *i*) switch operations resulting from procedure $\mathbf{P}_1$, and *ii*) mobility adaptations performed during procedure $\mathbf{P}_2$ to enforce the cluster constraints. When the node speed increases, the number of

these modifications also increases, thus reducing the cluster life time.

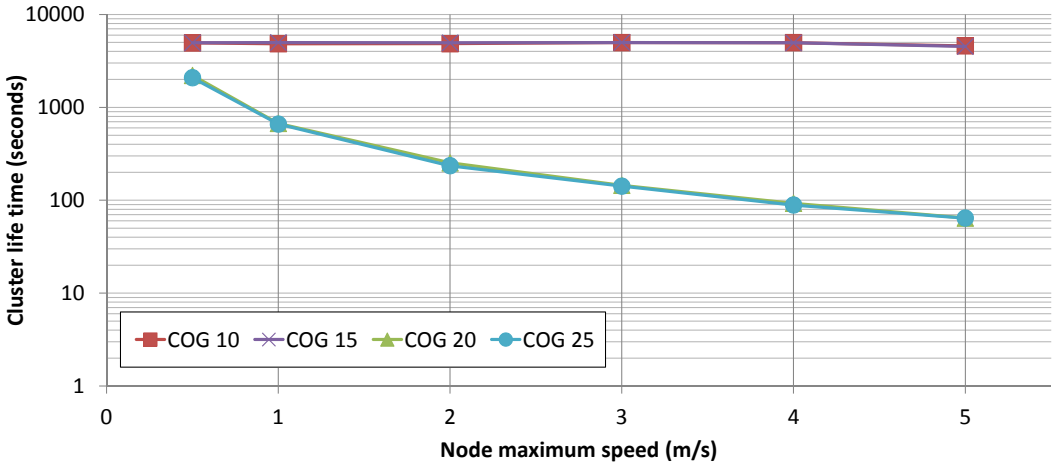

Fig. 10: Cluster life time vs. maximum node speed in mobile structured networks.

VI. Conclusion

We introduced a novel generic distributed node clustering algorithm for mobile ad hoc networks based on the coalition formation game framework. We proved that the proposed algorithm converges to a Nash-stable solution while simultaneously satisfying some practical constraints, such as the cluster size, the cluster diameter, etc. Designing judiciously *i*) the revenue function used to evaluate the worth of a cluster, and *ii*) the heuristics to select the nodes moving between clusters, we derived two practical algorithms, CLQ and COG, adapted to two different kinds of networks: unstructured and structured ad hoc networks respectively. We conducted extensive simulations to evaluate the performance of the proposed algorithms. In unstructured networks, CLQ outperforms the state of the art clustering algorithms LCC, VOTE and SECA in both static and mobile conditions in terms of the cluster size, intra-cluster link capacity and cluster stability. In structured networks, COG is capable of building stable clusters taken into account the network hierarchical structure allowing to gather more than one group in the same cluster. Compared to a naive approach, it leads to a reduction of long multi-hop inter-cluster links, and thus to a better QoS.

APPENDIX A
PROOF OF RESULT 1

Since the clusters of the initial partition fulfill the constraints, $\mathbf{P}_1$ is operated at the first decision-making time of the algorithm. After $\mathbf{P}_1$ execution, the resulting clusters satisfy the constraints by construction. Thus, by recurrence, the algorithm will always run $\mathbf{P}_1$. Let us now define the *social welfare* Ψ of a partition Π as $\Psi(\Pi) := \sum_{k=1}^{K} v(\mathcal{S}_k)$. Let us consider the nth decision-making occurrence of $\mathbf{P}_1$ with the corresponding partition Π_n and social welfare $\Psi_n := \Psi(\Pi_n)$. We get $\Psi_n = \Psi_{n-1} + g(\sigma_n)$. Since we consider switch operations with only strictly positive gain, we have $\Psi_n > \Psi_{n-1}$. After any switch operation the social welfare strictly increases, meaning that the same partition can never be visited twice. Furthermore, since there is a finite number of partitions, the algorithm converges to a final partition $\tilde{\Pi}$ after a finite number of iterations.

APPENDIX B
PROOF OF RESULT 3

Let clusters $\mathcal{S}_k$ and $\mathcal{S}_\ell$, such that $\mathcal{S}_k \cup \mathcal{S}_\ell$ satisfies the constraints. Using (1) and (2) the gain associated with the merge of $\mathcal{S}_k$ with $\mathcal{S}_\ell$ is thus equal to: $g(\sigma_{k,\ell}(\mathcal{S}_k)) = u_{\text{un}}(\mathcal{S}_k \cup \mathcal{S}_\ell) - u_{\text{un}}(\mathcal{S}_k) - u_{\text{un}}(\mathcal{S}_\ell)$. From (5), we have $u_{\text{un}}(\mathcal{S}_k \cup \mathcal{S}_\ell) = \sum_{i \in \mathcal{S}_k} \sum_{j \in (\mathcal{S}_k \cup \mathcal{S}_\ell) | (i,j) \in \mathcal{E}} \kappa(i,j) + \sum_{i \in \mathcal{S}_\ell} \sum_{j \in (\mathcal{S}_k \cup \mathcal{S}_\ell) | (i,j) \in \mathcal{E}} \kappa(i,j)$. It can be decomposed as $u_{\text{un}}(\mathcal{S}_k \cup \mathcal{S}_\ell) = \sum_{i \in \mathcal{S}_k} \left[\sum_{j \in \mathcal{S}_k | (i,j) \in \mathcal{E}} \kappa(i,j) + \sum_{j \in \mathcal{S}_\ell | (i,j) \in \mathcal{E}} \kappa(i,j) \right] + \sum_{i \in \mathcal{S}_\ell} \left[\sum_{j \in \mathcal{S}_k | (i,j) \in \mathcal{E}} \kappa(i,j) + \sum_{j \in \mathcal{S}_\ell | (i,j) \in \mathcal{E}} \kappa(i,j) \right]$, which after simplification reduces to: $u_{\text{un}}(\mathcal{S}_k \cup \mathcal{S}_\ell) = u_{\text{un}}(\mathcal{S}_k) + u_{\text{un}}(\mathcal{S}_\ell) + 2 \sum_{i \in \mathcal{S}_k} \sum_{j \in \mathcal{S}_\ell | (i,j) \in \mathcal{E}} \kappa(i,j)$, leading to $g(\sigma_{k,\ell}(\mathcal{S}_k)) = 2 \sum_{i \in \mathcal{S}_k} \sum_{j \in \mathcal{S}_\ell | (i,j) \in \mathcal{E}} \kappa(i,j)$. Since $\kappa(i,j) > 0$, we deduce that $g(\sigma_{k,\ell}(\mathcal{S}_k)) > 0$ which concludes the proof.

APPENDIX C
PROOF OF RESULT 4

Since we assume that the constraints are satisfied, $v = u_1$. From (1) and (2), we have $g(\sigma_{k,\ell}(\mathcal{P})) = u_1(\mathcal{S}_\ell \cup \mathcal{P}) - u_1(\mathcal{S}_\ell) - u_1(\mathcal{S}_k) + u_1(\mathcal{S}_k \setminus \mathcal{P})$. Using (7), we get: $g(\sigma_{k,\ell}(\mathcal{P})) = f_1(n_\ell + |\mathcal{P}|) - f_1(n_\ell) - f_1(n_k) + f_1(n_k - |\mathcal{P}|)$. Since $n_\ell + |\mathcal{P}| > n_k$, let us introduce $\delta := n_\ell - n_k + |\mathcal{P}| > 0$. The gain now writes: $g(\sigma_{k,\ell}(\mathcal{P})) = A - B$, with $A := f_1(n_\ell + |\mathcal{P}|) - f_1(n_\ell)$ and $B := f_1(n_\ell - \delta + |\mathcal{P}|) - f_1(n_\ell - \delta)$. Since $f_1(x), \forall x \in [0 + \infty)$ is a strictly increasing convex

function, it thus verifies $\forall \mu > 0, y \geq 0, x > y \Leftrightarrow f_1(x+\mu) - f_1(x) > f_1(y+\mu) - f_1(y)$, and thus $A > B$, which concludes the proof.

APPENDIX D
PROOF OF RESULT 5

Following the same lines as proof of Result 4, we get $g(\sigma_{k,\ell}(\mathcal{P})) = u_2(\mathcal{S}_\ell \cup \mathcal{P}) - u_2(\mathcal{S}_\ell) - u_2(\mathcal{S}_k) + u_2(\mathcal{S}_k \setminus \mathcal{P})$. Using (8) and after simplifications we obtain: $g(\sigma_{k,\ell}(\mathcal{P})) = f_{2,t}(m_{t,\ell} + |\mathcal{P}|) - f_{2,t}(m_{t,\ell}) - f_{2,t}(m_{t,k}) + f_{2,t}(m_{t,\ell} - |\mathcal{P}|)$. Since $m_{t,\ell} + |\mathcal{P}| > m_{t,\ell}$, let us introduce $\delta := m_{t,\ell} - m_{t,k} + |\mathcal{P}| > 0$. The rest of the proof is strictly identical to the one of Result 4 replacing f_1 by $f_{2,t}$, n_k by $m_{t,k}$, and n_ℓ by $m_{t,\ell}$.

APPENDIX E
PROOF OF RESULT 6

Let us consider two clusters $\mathcal{S}_k$ and $\mathcal{S}_\ell$ such that $\mathcal{S}_k \cup \mathcal{S}_\ell$ satisfies the constraints. From (1) and (2), the gain associated with the merge of $\mathcal{S}_k$ with $\mathcal{S}_\ell$ writes: $g(\sigma_{k,\ell}(\mathcal{S}_k)) = u(\mathcal{S}_k \cup \mathcal{S}_\ell) - u(\mathcal{S}_k) - u(\mathcal{S}_\ell)$. Using (6) we get: $g(\sigma_{k,\ell}(\mathcal{S}_k)) = \big[f_1(n_k + n_\ell) - f_1(n_k) - f_1(n_\ell)\big]\epsilon + (1-\epsilon)A(\mathcal{S}_k, \mathcal{S}_\ell)$, with $A(\mathcal{S}_k, \mathcal{S}_\ell) := \sum_{t \in \mathcal{I}(\mathcal{S}_k \cup \mathcal{S}_\ell)} f_{2,t}(m_{t,k} + m_{t,\ell}) - \sum_{t \in \mathcal{I}(\mathcal{S}_\ell)} f_{2,t}(m_{t,\ell}) - \sum_{t \in \mathcal{I}(\mathcal{S}_k)} f_{2,t}(m_{t,k})$. From (9), we have $g(\sigma_{k,\ell}(\mathcal{S}_k)) = \frac{2\epsilon n_\ell n_k}{n_{max}^2} + (1-\epsilon)A(\mathcal{S}_k, \mathcal{S}_\ell)$. Let us define $\mathcal{J} := \mathcal{I}(\mathcal{S}_k) \cap \mathcal{I}(\mathcal{S}_\ell)$. When $\mathcal{J} = \emptyset$ then $A(\mathcal{S}_k, \mathcal{S}_\ell) = 0$ and $g(\sigma_{k,\ell}(\mathcal{S}_k)) > 0$. When $\mathcal{J} \neq \emptyset$ we obtain after simplification $A(\mathcal{S}_k, \mathcal{S}_\ell) = \sum_{t \in \mathcal{J}} \big[f_{2,t}(m_{t,k} + m_{t,\ell}) - f_{2,t}(m_{t,k}) - f_{2,t}(m_{t,\ell})\big]$. Inserting (9) and (10) in the previous expressions leads to: $g(\sigma_{k,\ell}(\mathcal{S}_k)) = \frac{2\epsilon n_\ell n_k}{n_{max}^2} + \frac{2}{T}\sum_{t \in \mathcal{J}} \frac{m_{t,k} m_{t,\ell}}{m_t^2}$. Since all the terms in the later expression are strictly positive, $g(\sigma_{k,\ell}(\mathcal{S}_k)) > 0$, which concludes the proof.

APPENDIX F
PROOF OF RESULT 7

We study the sign of $\Delta := g(\sigma_{q,k}(\mathcal{U}_q)) - g(\sigma_{q,\ell}(\mathcal{U}_q))$ as a function of ϵ. From (1), we have $\Delta = r_{\mathcal{U}_q}(\mathcal{S}_k \cup \mathcal{U}_q) - r_{\mathcal{U}_q}(\mathcal{S}_\ell \cup \mathcal{U}_q)$. Since the constraints are fulfilled, $v = u$, then using (2) we get: $r_{\mathcal{U}_q}(\mathcal{S}_x \cup \mathcal{U}_q) = u(\mathcal{S}_x \cup \mathcal{U}_q) - u(\mathcal{U}_q)$, where x stands for k or ℓ. From revenue (6), and (7)-(10), we obtain: $u(\mathcal{S}_x) = \frac{n_x^2}{n_{max}^2}\epsilon + \frac{(1-\epsilon)}{T m_t^2}(K + m_{t,x}^2)$, where K is a constant that depends on the groups different from $\mathcal{O}_t$. Likewise, we get: $u(\mathcal{S}_x \cup \mathcal{U}_q) = \frac{(n_x + n_\mathcal{U})^2}{n_{max}^2}\epsilon + \frac{(1-\epsilon)}{T m_t^2}(K + (m_{t,x} + n_\mathcal{U})^2)$. After simplification, we obtain: $r_{\mathcal{U}_q}(\mathcal{S}_x \cup \mathcal{U}_q) = \frac{n_\mathcal{U}(2n_x + n_\mathcal{U})}{n_{max}^2}\epsilon + \frac{(1-\epsilon)n_\mathcal{U}}{T m_t^2}(2m_{t,x} + n_\mathcal{U})$, which leads to

$\Delta = D_1\epsilon + D_2(1-\epsilon)$ with $D_1 := \frac{2n_{\mathcal{U}}(n_k - n_\ell)}{n_{max}^2}$, and $D_2 := \frac{2n_{\mathcal{U}}}{Tm_t^2}(m_{t,k} - m_{t,\ell})$. The study of the sign of Δ leads to the following three cases: *i*) $m_{t,k} = m_{t,\ell}$ and $n_k > n_\ell$. Then $D_2 = 0$ and $D_1 > 0$. Consequently $\Delta > 0$ whatever the value of ϵ. *ii*) $m_{t,k} = m_{t,\ell}$ and $n_k = n_\ell$. Then $\Delta = 0$ whatever the value of ϵ. *iii*) $m_{t,k} \neq m_{t,\ell}$. We assume without loss of generality that $m_{t,k} > m_{t,\ell}$ and thus $D_2 > 0$. If $n_k \geq n_\ell$, $D_1 > 0$ and $\Delta > 0$ regardless of the value of ϵ. If $n_k < n_\ell$, then $D_1 < 0$. We can write $\Delta = (D_1 - D_2)\epsilon + D_2$ which is a linear function of ϵ with negative slope. Thus $\Delta > 0 \Leftrightarrow \epsilon < \epsilon_0 := \frac{D_2}{D_2 - D_1}$. We now search the parameters $n_k, n_\ell, m_t, m_{t,k}, m_{t,\ell}, n_{\mathcal{U}}$, leading to the smallest value of ϵ_0 denoted by ϵ^*. When D_2 is fixed, we have to minimize D_1, which is obtained by setting $n_k = 0$, $n_\ell = n_{max}$, leading to $\epsilon_0 = \frac{D_2}{D_2 + 2n_{\mathcal{U}}/n_{max}}$. Minimizing ϵ_0 is then equivalent to minimizing D_2, which is obtained by setting $m_{t,k} = 1$, $m_{t,\ell} = 0$ and $m_t = n_{max}$, leading to $\epsilon^* = \frac{1}{1+T}$.

REFERENCES

[1] D. Baker and A. Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Transactions on Communications*, vol. 29, no. 11, pp. 1694–1701, Nov. 1981.

[2] C. Cooper, D. Franklin, M. Ros, F. Safaei, and M. Abolhasan, "A comparative survey of VANET clustering techniques," *IEEE Communications Surveys Tutorials*, 2016.

[3] A. Asterjadhi, N. Baldo, and M. Zorzi, "A cluster formation protocol for cognitive radio ad hoc networks," in *IEEE European Wireless Conference*, Apr. 2010, pp. 955–961.

[4] M. Gerla and J. Tsai, "Multicluster, mobile, multimedia radio network," *Journal of Wireless Networks*, vol. 1, no. 3, pp. 255–265, Jul. 1995.

[5] C. C. Chiang, H. K. Wu, W. Liu, and M. Gerla, "Routing in clustered multihop, mobile wireless networks with fading channel," in *IEEE Singapore International Conference on Networks (ICCNS)*, 1997.

[6] F. Li, S. Zhang, W. Wang, X. Xue, and H. Shen, "Vote-based clustering algorithm in mobile ad hoc networks," in *Proceedings of International Conference on Information Networking (ICOIN)*, Busan, South Korea, Feb. 2004.

[7] Y. Zhang and J. Ng, "A distributed group mobility adaptive clustering algorithm for mobile ad hoc networks," in *IEEE International Conference on Communications (ICC)*, Beijing, China, May 2008, pp. 3161–3165.

[8] X. Gao, "A mobility-based clustering algorithm in MANETs utilizing learning automata," in *32nd IEEE Chinese Control Conference (CCC)*, Jul. 2013, pp. 6398–6402.

[9] M. Cai, L. Rui, D. Liu, H. Huang, and X. Qiu, "Group mobility-based clustering algorithm for mobile ad hoc networks," in *17th IEEE Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Aug. 2015, pp. 340–343.

[10] X. Tan, Z. Xiong, and Y. He, "Signal attenuation-aware clustering in wireless mobile ad hoc networks," in *Journal of Networks*, vol. 8, no. 4, Apr. 2013, pp. 796–803.

[11] N. Keerthipriya and R. S. Latha, "Adaptive cluster formation in MANET using particle swarm optimization," in *3rd IEEE International Conference on Signal Processing, Communication and Networking (ICSCN)*, Mar. 2015, pp. 1–7.

[12] H. Wu, Z. Zhong, and L. Hanzo, "A cluster-head selection and update algorithm for ad hoc networks," in *IEEE International Conference on Communications (GLOBECOM)*, Dec. 2010, pp. 1–5.

[13] D. Camara, C. Bonnet, and N. Nikaein, "Topology management for group oriented networks," in *IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, Honolulu, Hawai, Sep. 2010.

[14] R. Massin, C. J. Le Martret, and P. Ciblat, "Distributed clustering algorithm in dense group-based ad hoc networks," in *16th IEEE Mediterranean Ad Hoc Networking Workshop (MedHocNet)*, Jun. 2016.

[15] W. Saad, Z. Han, M. Debbah, A. Hjørungnes, and T. Basar, "Coalitional game theory for communication networks," *IEEE Signal Processing Magazine*, vol. 26, no. 5, pp. 77–97, Sep. 2009.

[16] ——, "Coalitional games for distributed collaborative spectrum sensing in cognitive radio networks," in *IEEE International Conference on Computer Communications (INFOCOM)*, Apr. 2009, pp. 2114–2122.

[17] W. Saad, Z. Han, R. Zheng, A. Hjørungnes, T. Basar, and H. V. Poor, "Coalitional games in partition form for joint spectrum sensing and access in cognitive radio networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 2, pp. 195–209, Apr. 2012.

[18] R. Umar and W. Mesbah, "Coordinated coalition formation in throughput-efficient cognitive radio networks," *Wireless Communications and Mobile Computing*, vol. 16, no. 8, pp. 912–928, 2016.

[19] Z. Zhang, L. Song, Z. Han, and W. Saad, "Coalitional games with overlapping coalitions for interference management in small cell networks," *IEEE Transactions on Wireless Communications*, vol. 13, no. 5, pp. 2659–2669, May 2014.

[20] W. Saad, Z. Han, A. Hjørungnes, D. Niyato, and E. Hossain, "Coalition formation games for distributed cooperation among roadside units in vehicular networks," *IEEE Journal on Selected Areas in Communications*, Jan. 2011.

[21] F. Chiti, R. Fantacci, E. Dei, and Z. Han, "Context aware clustering in VANETs: a game theoretic perspective," in *IEEE International Conference on Communications (ICC)*, Jun. 2015, pp. 6584–6588.

[22] Y. Cai, H. Chen, D. Wu, W. Yang, and L. Zhou, "A distributed resource management scheme for D2D communications based on coalition formation game," in *IEEE International Conference on Communications Workshops (ICC)*, Jun. 2014.

[23] M. W. Baidas and A. B. MacKenzie, "Altruistic coalition formation in cooperative wireless networks," *IEEE Transactions on Communications*, vol. 61, no. 11, pp. 4678–4689, Nov. 2013.

[24] A. Bogomolnaia and M. O. Jackson, "The stability of hedonic coalition structures," *Games and Economic Behavior*, vol. 38, pp. 201–230, 1998.

[25] X. Hong, M. Gerla, G. Pei, and C. Chiang, "A group mobility model for ad hoc wireless network," in *ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM)*, Aug. 1999, pp. 53–60.