Abstract—In this contribution, we present SHARC, a Sharper Heuristic for Assignment of Robust Communities. This algorithm performs distributed network partitioning into communities using epidemic propagation of community labels and the computation of a neighborhood similarity metric. Due to its decentralized nature, SHARC is scalable and well suited for networks where no global knowledge nor node coordination exist, like ad hoc networks. Besides, SHARC is computationally efficient and does not depend on configuration parameters. We validated our approach and compared it to alternative solutions using static and dynamic networks. Results show that SHARC provides a sharper and more robust community assignment and prevents the domination of a single community in both static and dynamic networks.

I. INTRODUCTION

Studies on graph theory and properties analysis of natural, human and social networks have revealed the heterogenous nature of their link density. Inside those networks, there exist groups of vertices, called communities (or clusters or modules), that are defined by a high internal connectivity, and much sparser links to the outside of the group[11]. Those structures have proven to have a key role in the spread of trends and leadership[12] and are likely to also play a central part in the development of ubiquitous interactive exchanges between users of Mobile Social Networks (MoSoNets) or cars using Vehicular Ad Hoc Networks (VANETs) to improve safety and traffic management.

Community structures have no predefined size, shape or diameter. Finding the community assignment, i.e. placing all network vertices in their respective community, is a NP-hard problem, related to graph partitioning[5]. In the scope of this paper, we consider community detection as a network partitioning problem. The number of partitions being unknown in advance, a clustering of the different vertices of the graph is operated based on a set of topology-based characteristics. Atypical vertices may create single-node communities. SHARC aims at providing a fully decentralized solution for this clustering problem. Only a computationally and memory efficient distributed solution can be suitable for operating on the power and energy-limited handheld devices that dynamically form ad hoc networks, without any prior coordination.

The idea is to present here a core algorithm on which we plan to build advanced features described in section II, and to compare this algorithm with other similar contributions.

Although briefly put in perspective with centralized and local-greedy algorithms, we focused our comparison with other distributed algorithms that could also be used in ad hoc networks.

The remainder of this paper is therefore organized as follows. We will first explain in further details the context of our research and give incentives for our contribution in section II. In section III, we propose a more complete definition of the concept of community, associated metrics and a review of community assignment algorithms. We will then introduce the SHARC algorithm in section IV. Next, we will assess the performances of our contribution and compare it to state-of-the-art algorithms on static networks in section V. In section VI, we will extend this comparison to dynamic networks representing two plausible use cases of our algorithm: a city mall and a highway section. We will then present our conclusions in section VII before introducing our future work based on SHARC in section VIII.

II. ROBUST COMMUNITIES FOR MOBILE SOCIAL AND VEHICULAR AD HOC NETWORKS

The overall objective of our research is to use the concept of communities to detect stable and reliable well-connected communication subgraphs in ad hoc networks. Interactive communication can happen on those reliable links (e.g. multimedia exchange, interactive gaming, high priority information) and less stable inter-community links would be reserved for propagation of data with less critical quality of service requirements (in reliability or delay). The field of application is twofold as depicted in Fig. 1 and explained in the following paragraphs.

First, we envision urban users in their daily life, where they alternate between high speed and low speed mobility (while in transportation as opposed to while at home, mall or office). Characteristics of this heterogenous mobility are captured in various models ([20], [9], [13]). Such users are the target audience for mobile social networks, as applications like interactive gaming can help make their transportation more enjoyable, even whether no or unreliable connection to an infrastructure network is available.

A second use case is a pure vehicular ad hoc network (VANET) scenario, where stable communities should be found between set of cars with similar direction, for instance. Those
structure would then help to quickly and reliably propagate safety or traffic-related information to the desired set of cars only. Such scenarios, induce that the assignment protocol should be quick to react and manage dynamic apparition of non-connected components in the network.

In this context, SHARC is a distributed algorithm that partitions the network in communities, trying to optimize the related assignment metrics, such as the modularity [21], later presented in subsection III-B. All detected communities are non-overlapping and nodes not member of any community are considered being their self-community. We have hence, a partition of the network, following the community assignment. Given that SHARC is a truly distributed algorithm, it provides a fully scalable solution in terms of number of nodes for this community-based network partitioning problem.

SHARC can be used as a core component of a network topology service. Other services or applications, like interactive gaming or road status information propagation, will be able to rely on it for building communication-efficient solutions.

In this article we base our assignment process on topological information only and the concept of community detected here should not be interpreted as related to any social group. However, we plan to later focus on more long-term related metrics, similar to approaches presented in [16] and [4]. This would allow users with the most in common to be more likely grouped in the same community, as they should have more incentives to exchange.

III. RELATED WORK

In this section, we will present key concepts of the community detection problem. After a clear definition of the community structure, we will review metrics used to assess the detection quality, as some of them will be used for evaluation in section V and section VI. We will also present well-known and recent algorithms for community detection existing in the literature.

A. A definition for communities

Real networks, i.e. issued from social network analysis, or connection graphs of communication networks like the Internet or wireless ad hoc networks, have structures between order and randomness. They are neither lattices nor purely random graphs. On the contrary, they show global and local inhomogeneities in both vertex degree and edge distribution [10].

This discrepancies generate a division of the network into groups within which the connections are dense, but between which they are sparser [11]. Those groups of vertices, whose definition is closely related to the network topology, are called communities, clusters or modules.

Considering the underlying concepts the studied graph is abstracting from, communities can also be envisioned as groups of vertices which probably share common properties and play similar roles within the graph [10].

B. Assessing detection quality

As the definition of a community can vary depending on the usage context, it is important to define detection quality metrics coherently. Here we will present measures that are only related to the topology of the network on which the assignment is performed. They will be used for performance assessment in section V and section VI.

The modularity measure $Q$, introduced by Newman and Girvan in [21] is one of the first metric used to evaluate the quality of a community assignment. It measures how the performed assignment matches the community structure exhibited by the network topology. For a partitioning counting $n_c = |C|$ communities, we define a squared $n_c \times n_c$ matrix $e$ whose elements $e_{ij}$ represent the fraction of edges from a vertex in community $i$ to a vertex in community $j$. As a consequence, the sum of rows, or columns, of $e$, $a_i = \sum_j e_{ij}$ corresponds to the fraction of links connected to $i$. If the community assignment were random, the expected number of intra-community links would be $a_i^2$. Hence the definition of $Q$ given in Equation 1

$$Q = \sum_i (e_{ii} - a_i^2)$$

As it is closely related to the underlying community structure of a network, the maximum achievable value for $Q$ is not absolute but topology dependent. The advantage of this metric is that it does not require a reference assignment to compare to but, unfortunately, it does not quantify the distance to an optimal assignment and does not measure the robustness of the assignment to small network changes [17].

Whenever a natural or pre-defined assignment exists for a network, many other metrics have been developed to estimate a normalized distance between the given and the computed classification. In [5], Danon et al. introduced the normalized
mutual information (NMI) measure based on information theory. This normalized metric evolves between 0 when computed ($C$) and pre-determined ($C'$) assignments are independent, and 1 when they are identical. The NMI is defined as such:

$$I_n(C, C') = \frac{2H(C) - H(C)+H(C')}{H(C)+H(C')}$$

In search for a robustness measurement, Karrer et al. proposed the variation of information [17], based on the difference of the conditional entropies $H(C|C')$ and $H(C'|C)$ of two communities. Those metrics shall be preferred for benchmarks where networks are generated based on pre-determined community assignment, like the tests described in [18].

C. Community detection algorithms

As for many complex problems, centralized solutions have been presented first. Then more memory efficient algorithms, based on local information have been introduced before the recent emergence of fully decentralized solutions, the only ones suitable for wireless ad hoc networks.

1) **Centralized and local algorithms:** The first proposed algorithms aimed at performing community detection relied on a greedy, centralized approach trying to agglomerate nodes in communities or divide the network in such structures. Basically, they iteratively operate until a measure reflecting the assignment is optimized. Most of the time, the modularity measure $Q$ (presented in subsection III-B above) is used. Nodes to group or split are chosen using similarity, betweenness [21] or extremal optimization [7]. Alternative approaches are based on spectral properties of the graph, Laplacian matrix [6] for instance. A good comparison of those centralized algorithms can be found in [5] or in [18].

Algorithms relying only on local information have also been proposed, in order to improve the memory efficiency of community detection. In [25], the authors use a cached table representing the network structure and allowing information processing at each vertex to be in $O(1)$ time complexity. In their proposal, Wan et al. use a variation of the L-shell algorithm introduced by Bagrow and Bolllt in [2] to iteratively add nodes to a community. They start from the shell origin, until the number of added adjacent vertices goes below a given threshold. In [23], the authors introduce the notions of vertex intensity (sum of weights of all its attached edges) and vertex contribution to a community $C$ (as the ratio of the sum of weights of all its edges to adjacent vertices in community $C$ over its total intensity). They then proceed to a greedy agglomerative algorithm, starting with the edge of highest contribution. All those algorithms, however, require at some point node coordination or global network knowledge, which is not suitable for use in ad hoc networks.

2) **Decentralized algorithms:** Decentralized algorithms appear amongst most recent contributions to the community detection problem. In [22], Raghavan et al. introduce the epidemic label propagation principle: vertices either synchronously or asynchronously beacon their current community identifier to their adjacent nodes and each vertex simply choses the community to which most of its neighbors belong to. While directly inheriting from the vertex community definition from [11], this algorithm is proven to generate label oscillation (two subsets of vertices alternate between two different community labels) when run synchronously [22] and to form monster communities that plague an extended amount of nodes [19].

In order to address the latter effect, Leung et al. refined this algorithm in [19]. Their version includes a hop attenuation factor $\delta$ to limit formation of monster communities and a node preference function (the authors suggest the node degree) weighted by a parameter $m$. Each vertex beacons its community label with a score function of the weighted node preference, the edge weight and the internal community score. The internal community score equals the maximum beaconed score for this label by the adjacent vertices minus the hop attenuation factor $\delta$. Presented experiments show that the parameter combination ($\delta = 0.1, m = 0.0.5$) yields the best results.

The existence of those fixed parameters may not guarantee that this algorithm will perform efficiently whatever the underlying network topology, especially when it is changing, like in dynamic networks with heterogenous mobility. Besides, whether the hop attenuation may limit the creation of monster communities, it also limits the natural span of a label around an arbitrary center. Achieved community assignment might therefore be far from optimal.

IV. SHARC ALGORITHM

As other decentralized community detection algorithms [22] [19], SHARC uses epidemic propagation of a community label (presented in paragraph III-C2) to allow each vertex to independently decide of its community assignment. From the community structure definition, a vertex should be most infected by the label corresponding to its community. Besides, as SHARC is asynchronous, it tends not to produce label oscillation.

Our design objective is to cope with two major flaws of such algorithms. We would like to prevent the creation of monster communities, especially when the inner and inter-community link densities are close; that is make the assignment sharper. We also wish the performances to be resilient to underlying changes in the topology (while keeping the same overall characteristics) or the initialization parameters; that is make the assignment more robust (in the sense of the robustness given by Siegel et al. in [1] and studied for community detection in [17]).

To that end, we will use the concept of common adjacent vertices set, or neighborhood similarity, for heuristic in the community assignment process. This is described in the following paragraphs.

A. Neighborhood similarity measure

While studying relationship networks, sociologist Granovetter isolated the relation between community structures in those
networks and the predominance of closed triads in relationships\cite{12}. If A is a friend of both B and C, then B and C are also very likely to be friends. Triads not respecting this closure rule explain the existence of links between communities.

Let $\mathcal{N}$ be a network of $n$ vertices and $m$ edges. We denote $N(v)$, the neighborhood of vertex $v$, i.e. all the vertices $w \in \mathcal{N}$ that are adjacent to $v$, and $|N(v)|$ its cardinality, i.e. the degree of vertex $v$.

As a consequence of the triadic closure rule, two neighbor vertices, $v$ and $w$ of the network $\mathcal{N}$, belonging to the same community $c$ both have a high link density with other vertices of $c$ and should hence have a more similar set of adjacent vertices than with a given vertex $u$, not belonging to $c$. The size of this common adjacent vertices set, can be used to assess the likelihood of both $v$ and $w$ belonging to the same community. However it needs to consider the ratio of common neighbors over the total node degree, in order not to favor high degree nodes.

Therefore, we propose the following definition of the neighborhood similarity measure $n_{sim}$, that is defined for any vertices $v \in \mathcal{N}$ of degree at least 1, $w \in N(v)$ as:

$$n_{sim}(v, w) = 1 - \frac{|(N(v) \setminus N(w)) \cup (N(w) \setminus N(v))|}{|N(v)| + |N(w)|} \quad (3)$$

The neighborhood similarity linearly varies between 0 when the assessed nodes have no neighbor vertex in common and 1 when the neighborhoods are identical. It is a normalized value which does not favor high or low degree vertices. This measure is also symmetric for any pair of adjacent vertices of the network $\mathcal{N}$.

**B. Community assignment using neighborhood similarity**

As in other epidemic algorithms, SHARC accounts for each neighbor vertex community label $c \in \mathcal{C}$ and assigns any vertex $v$ to the community $C(v)$ with the highest count. However, in our algorithm, neighbors do not contribute equally, as the Equation 4 shall be verified:

$$C(v) = \arg \max_{c \in \mathcal{C}} \sum_{w \in N(v)} n_{sim}(v, w) \quad (4)$$

Each neighboring vertex $w$ contributes to the count of its community label $C(w)$ at node $v$ to the extent of its neighborhood similarity with the vertex $v$. This heuristic therefore strengthens the gap between community counts as it considers both the number of neighboring vertices with this label and their value in terms of neighborhood similarity.

Besides, the count of a community label $c$ at any vertex $a \in \mathcal{N}$ denotes the strength with which $v$ assigned itself to community $C(v) = c$.

We therefore introduce the membership strength of a given vertex $v \in \mathcal{N}$ as the degree-normalized value of the community count computed in Equation 5

$$m_{str}(v, C(v)) = \sum_{w \in N(v)} \frac{n_{sim}(v, w)}{|N(v)|} \quad (5)$$

**C. Algorithm implementation**

SHARC has been designed to be fast and computationally light and to operate without node synchronization, in order to be easily implemented on handheld devices forming wireless ad hoc networks. Also, SHARC relies on simple decentralized operations using local information only. To perform its assignment, each vertex only requires knowledge of the neighborhood of its adjacent vertices (referred as two-hop neighborhood) and their current community label.

In this section we will describe two possible implementations for SHARC: one used for benchmarking the algorithm (as presented in Section V) and the other suitable for deployment on real devices.

1) Implementation for benchmarking: For benchmarking on static or dynamic networks purposes, SHARC has been implemented as a simple single-threaded algorithm, where nodes iteratively proceed to their assignment in a random order until a termination condition is met. Access to neighborhood information was granted using the locally shared memory model of communication\footnote{An implementation for benchmarking of the SHARC protocol is available at http://github.com/gjherbiet/sharc}.

For static networks, this condition was a stabilization of the NMI or Q metric (or a maximum of 50 iterations on all nodes to prevent infinite divergence). For dynamic networks, each node was only processed once per network update, and the simulation terminated as soon as the network evolution was over.

An implementation of the neighborhood similarity measure is presented in Algorithm 1\footnote{General implementation of SHARC for benchmarking is given in Algorithm 2}. An implementation for benchmarking of the SHARC protocol is available at http://github.com/gjherbiet/sharc.

---

**Algorithm 1: Neighborhood Similarity Measure (NS)**

**Input:** Vertex $v$, $N(v)$, $C = \{e| C(w) = c, \forall w \in N(v)\}$

$C(v) = v$;

CommunityScore = $\{\}$;

MaxScore = 0;

for each $w$ in $N(v)$ do

CommunityScore[$C(w)$] += $n_{sim}(v, w)$;

if CommunityScore[$C(w)$] $\geq$ MaxScore or (CommunityScore[$C(w)$] = MaxScore and RandomTieBreaksWon()) then

$C(v) = C(w)$;

MaxScore = CommunityScore[$C(w)$];

Output: ($C(v)$, MaxScore)

---

**Algorithm 2: Sharper Heuristic for Assignment of Robust Communities (SHARC)**

**Input:** $\mathcal{N}$

while not TerminationCondition() do

$S = \text{Shuffle}(\{v \in \mathcal{N}\})$

for each $v$ in $S$ do

($C(v)$, MaxScore) = NS($v$);

Strength($v$, $C(v)$) = MaxScore / Degree($v$);

---
As an instance), therefore, SHARC is running in overhead similarity measure is based on simple set operations sparse graphs, which should limit this extra overhead. ever, networks exhibiting community properties are usually messages shall include the list of neighbor identifiers. How- than for other distributed protocols, especially because beacon overhead generated by the SHARC protocol may be higher payload of this packet is given for reference in Fig. 2. The packet, or piggybacked to an existing beacon message (for real devices, each wireless network node periodically updates its community before sending a beacon message containing the node identifier, the updated community label and the list of identifiers of its neighbors. Processing of all received beacons during a period allows to compute all relevant information for the next update. Typical period values are in the range of a couple of seconds.

This beacon can be sent as a one-hop broadcast applicative packet, or piggybacked to an existing beacon message (for instance if the devices use 802.11 in ad hoc mode). The payload of this packet is given for reference in Fig. 2. The overhead generated by the SHARC protocol may be higher than for other distributed protocols, especially because beacon messages shall include the list of neighbor identifiers. However, networks exhibiting community properties are usually sparse graphs, which should limit this extra overhead.

In both implementations, the computation of the neighbor- hood similarity measure is based on simple set operations and can be implemented in linear time (using sorted lists for instance). Therefore, SHARC is running in $O(\Delta^2)$, where $\Delta$ is the network average degree.

V. COMPARATIVE BENCHMARK

In this section, we will first compare our contribution with other similar algorithms also relying on epidemic label propagation. Then we will put our results in perspective with local and centralized algorithms, hardly applicable to the ad hoc network use case, in order to assess the gap with our decentralized approach. All those algorithms were briefly introduced in section III.

The benchmarking process is closely based on the one presented by Lancichinetti and Fortunato in [18], which can be used as reference. The experiments were performed using the framework presented in paragraph IV-C1 which is publicly available.

A. Girvan-Newman (GN) experiment

In this experiment, first presented in [21], we generated random networks composed of $N = 128$ nodes, pre-assigned in four different communities composed of $n_c = 32$ nodes. The average network degree is set to $\langle k \rangle = 16$. The node degrees follow a power law distribution of exponent $\tau_1 = -2$. Edges are created so that each node $n$ of degree $k_n$, has $(1 - \mu_t)k_n$ links with other nodes of its community and $\mu_t k_n$ links outside its community. Sharpness of the assignment is assessed by increasing the value of the mixing factor $\mu_t$ until the community structures are no longer properly defined. Results were collected on 10 different runs of each algorithm, on 10 different networks with similar properties, for a total of 100 simulations per algorithm, per value of $\mu_t$.

The Fig. 3(a) presents the average (with standard deviation as error bars) achieved NMI by the different distributed algorithms presented in paragraph III-C2 for increasing values of $\mu_t$. Dotted lines present the maximum and minimum values. Those results show SHARC is able to maintain higher value for average NMI, even when the community structures become less clear cut ($\mu_t > 0.4$). The asynchronous epidemic label propagation algorithm [22] performs the poorest and presents the widest variation in its results, making it unreliable for correctly detecting communities in any case. The synchronous epidemic label propagation algorithm [22] and the algorithm presented by Leung et al. [19] perform equivalently in average, with a wider variance in results for the latter.

When communities are not well-defined any more ($\mu_t > 0.7$), SHARC results are comparable with those of other algorithms.

A look at the evolution of the maximum community size, presented in Fig. 3(b) shows that SHARC good performances are explained by its ability to longer bound the size of the communities to meaningful values. Thanks to the neighborhood similarity metric, the assignment performed by SHARC is sharper.

It it also interesting to look at the spread of metrics on the different runs. It appears that SHARC presents less standard deviation in its results. In particular, it does not produce very low results in some configurations. SHARC is hence robust against particular network configurations and initialization parameters.

B. LFR experiment

The LFR experiment, as presented by Lancichinetti and Fortunato in [18], can be seen as a generalization of the Girvan-Newman experiment. In this model, nodes degree still follow a power law distribution of exponent $\tau_1 = -2$ but the sizes of the communities also varies according to a power law distribution of exponent $\tau_2 = -1$. In our experiments, average node degree was set to $\langle k \rangle = 20$, maximum degree to 50. We used a larger network size ($N = 1000$ nodes) and two different community configurations: $S$ (small communities of size between 10 and 50 nodes) and $B$ (big communities composed of between 20 and 100 nodes). The mixing parameter $\mu_t$ is still used to make the communities more or less clear-cut. This test can be considered as more demanding for the algorithms, due to the increased network size and the wider distribution in the community sizes.

On both tests, SHARC maintains a higher NMI level whatever the current value of the community mixing factor $\mu_t$. This means that SHARC is able to perform an assignment closest to the pre-defined assignment than all main other distributed algorithms. This is especially noticeable for very high values of $\mu_t$. 

| Neighbor ID 1 |
| Community Label |
| Beacon Timestamp |
| Sender ID |

Fig. 2. Example of beacon message to be used with SHARC.
In this demanding range ($\mu_t \geq 0.6$), SHARC minimum achieved NMI values are most of the time higher than the other distributed algorithms best results. Besides, the variance in SHARC results is very low, thanks to the effect of the neighborhood similarity metric.

Results on both Girvan-Newman and LFR benchmarks tend to prove that SHARC is well suited for sharp and robust community detection, even in large networks and when the communities have no pre-defined size or when they are not clear-cut.

C. Comparison with local and centralized algorithms

Although they are not directly applicable to ad hoc networks, we want to compare the performance of our contribution with centralized and local algorithms used for community detection. This allows to put our work in perspective with more generic solutions recognized as valuable in the literature for this problem. For this purpose, we will use the maximum modularity $Q$ achieved on the Zachary karate club [26] network. This metric and this network are the most used for comparison between algorithms. The corresponding results are shown in Table I.

It appears that SHARC achieves results comparable to, or better than, other distributed algorithms, as shown in the previous section. Besides, our algorithm outperforms all the early centralized community detection algorithms, that were introduced by Newman [3] and Donetti et al. [6].

Compared to local algorithms, i.e. greedy algorithms that mostly use local information to perform their community assignment, the performance gap lies between 1% and 33%, when compared to the most efficient algorithm. However, as explained in section III those local algorithms assume, at some point of their execution, a global knowledge of the network.

![Evolution of Normalized Mutual Information for test “GN”](image1)

![Evolution of Maximum community size for test “GN”](image2)

![Evolution of Normalized Mutual Information for test “LFR, 1000 nodes, small communities”](image3)

![Evolution of Normalized Mutual Information for test “LFR, 1000 nodes, large communities”](image4)

**Fig. 3.** Evolution of NMI (a) and Maximum community size (b) values for the GN benchmark

**Fig. 4.** Evolution of NMI for the LFR benchmark with small (a) and large (b) communities

**TABLE I**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Year</th>
<th>max. $Q$</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newman [3]</td>
<td>2004</td>
<td>0.381</td>
<td>Centralized</td>
</tr>
<tr>
<td>Donetti et al. [6]</td>
<td>2004</td>
<td>0.412</td>
<td>Centralized</td>
</tr>
<tr>
<td>Wang et al. [25]</td>
<td>2007</td>
<td>0.4188</td>
<td>Local</td>
</tr>
<tr>
<td>Wan et al. [23]</td>
<td>2008</td>
<td>0.488</td>
<td>Local</td>
</tr>
<tr>
<td>Tian et al. [23]</td>
<td>2009</td>
<td>0.564</td>
<td>Local</td>
</tr>
<tr>
<td>Raghavan et al. [22]</td>
<td>2007</td>
<td>0.4151</td>
<td>Distributed</td>
</tr>
<tr>
<td>Leung et al. [19]</td>
<td>$m = 0.1, \delta = 0.05$</td>
<td>2008</td>
<td>0.3718</td>
</tr>
<tr>
<td>SHARC</td>
<td>2009</td>
<td>0.4151</td>
<td>Distributed</td>
</tr>
</tbody>
</table>
This assumption makes them incompatible for a use with networks where no global knowledge nor node coordination exist, like ad hoc networks. This results corroborates the general observation that decentralized algorithms, based on a given heuristic like SHARC, offer a sub-optimal solution while comparing with results from centralized algorithms.

However, this comparison is based on one network only and some pitfalls of the presented algorithms may only appear when tested in other conditions, like when communities become less clear cut. Therefore, a detailed comparison would require a simulation campaign as complete and thorough as the one presented in the previous sections.

VI. EVALUATION ON DYNAMIC NETWORKS

We will now compare SHARC with other distributed community detection algorithm on dynamic networks.

A. Experimentation setup

We propose to evaluate SHARC, along with the asynchronous and Leung epidemic label propagation algorithms on two different scenarios, reflecting mobility of human users in two urban contexts: a shopping mall and a highway section. Both derive from the Human Mobility Model, introduced by Hogie in [14]. Their particular characteristics are summed up in Table II.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mall</th>
<th>Highway</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>99</td>
<td>80</td>
</tr>
<tr>
<td>Playground (m)</td>
<td>300 × 300</td>
<td>750 × 900</td>
</tr>
<tr>
<td>Duration (s)</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>Iterations</td>
<td>480</td>
<td>480</td>
</tr>
<tr>
<td>Iterations per second</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Mobility speed (km/h)</td>
<td>[1; 10]</td>
<td>[70; 140]</td>
</tr>
<tr>
<td>Avg. transmission range (m)</td>
<td>43</td>
<td>46</td>
</tr>
<tr>
<td>Avg. node degree</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Avg. link lifetime (s)</td>
<td>48</td>
<td>3.4</td>
</tr>
<tr>
<td>Avg. link density (m⁻²)</td>
<td>5 × 10⁻³</td>
<td>6 × 10⁻⁴</td>
</tr>
</tbody>
</table>

In this model, a mobile node n first chooses a destination spot \( d(n) \) as the closest element not contained in a list of recently visited spots \( V S(n) \). The node then move towards its destination spot, with a certain degree of random variation in its direction. If all spots have been visited, then \( V S(n) \) is emptied. The mobility traces were generated using MadHoc [15], a MANET simulator providing an advanced propagation model, with lognormal shadowing and terrain modeling: spots occupy a given surface on the simulation area and also attenuate signal propagation.

In the shopping mall scenario, spots stand for shops, uniformly distributed over the mall area. This scenario generates highly dense wide areas and allows to study the behavior of protocols in dense network conditions. An example of this scenario is depicted in Fig. 5, where vertices colors represent the current community attributed by SHARC, and their size the membership strength defined in subsection IV-B.

The highway section environment is characterized by nodes moving at a very high speed between a few number of spots (2 in our tests). We have chosen this particular scenario as it produces topologies where most links are organized into chains of nodes moving in opposite directions, which favors domination of a single community. As nodes move very fast, link duration is also very short. This is therefore a very aggressive scenario to test the robustness of the algorithms.

We implemented all tested algorithms in GraphStream [8], a dynamic graph library written in Java, complementary to MadHoc. For each scenario, we have generated four different dynamic networks with similar properties. For each network, we performed 25 runs in our simulation framework, deriving from the implementation made for static networks. We used different initialization seed for each run, leading to a statistical analysis over 100 different simulations per scenario.

B. Performance evaluation

Based on the setup described in subsection VI-A we will present and analyze average, minimum and maximum values of the modularity \( Q \), of the number of communities and the maximum community size during simulations. Standard deviation is not represented in order to preserve graphs clarity. As no pre-defined assignment exists for those networks, we could not use the NMI as quality metric.

On the shopping mall scenario (see Fig. 6), all assessed algorithms achieve fairly similar average results in terms of modularity \( Q \), especially during the first 300 iterations. After this state, we observe a decrease in the modularity value for the asynchronous and Leung algorithms. This is due to runs...
Fig. 6. Comparison of SHARC with other distributed community detection algorithms on dynamic network *mall*

Fig. 7. Comparison of SHARC with other distributed community detection algorithms on dynamic network *highway*

![Graphical illustration of the wandering community effect](image)

Fig. 8. Graphical illustration of the *wandering* community effect

...with poor results, as shown by the plot of minimum achieved value. On the contrary, SHARC does not exhibit this behavior. This plummet is due to apparition of *monster* communities as the simulation evolves for the asynchronous and Leung algorithms, as depicted in the two rightmost graphs of Fig. 6. The use of the *neighborhood similarity measure* prevents this effect for SHARC and the size of communities is bounded to a correct value.

We also observe that all algorithms suffer from what we denote as the *wandering* community effect (illustrated in Fig. 8): if a set of vertices of the same community split into two disconnected subsets, they may keep being assigned to the same community by the algorithms and propagate this label to other communities, making the label diversity plummet. This plummet is due to apparition of *monster* communities, as the simulation evolves.

As expected, the *highway section* scenario (Fig. 7) is a grueling test case for all algorithms. They all finally converge to a single community containing all the network vertices. We observed that this is due to the *wandering* community effect conjugated with the high number of sporadic links between nodes moving in opposite directions. This causes convergence of crossing nodes to the same community id, before this id is propagated to nodes going in the same direction.

The impact of those sporadic connections may be lessened in a real-world environment as beacon packets may not have the time to be correctly sent and/or received. However, this test case speaks for the introduction of a link-stability aware community assignment algorithm as well as dedicated mechanism to cope with the *wandering* community effect.

Despite this effect, SHARC is the algorithm that maintains the highest average and maximum modularity $Q$, as it limits, as much as possible, the *monster* community effect.

**VII. Conclusion**

In this paper, we have introduced a new distributed algorithm, SHARC, in order to partition networks where no global knowledge nor node coordination exist, using the vertices community concept. Our objective is to detect stable and reliable well-connected communication subgraphs in mobile social and vehicle-based ad hoc networks.

Our contribution aims at addressing current shortcomings of the existing distributed algorithms used with this kind of networks. According to results from our extensive simulation campaign, SHARC performs community assignment with a better accuracy than those of equivalent solutions. Besides, the introduction of the *neighborhood similarity measure* makes the assignment realized by SHARC sharper and more robust to initial condition changes. For instance, when communities are not clear-cut, SHARC achieves a stable value of the modularity metric $Q$ or the normalized mutual information (NMI). Besides, standard deviation in the results is way lower for the algorithm we propose. This means that this one is more...
resilient to initialization parameters and to underlying network topology changes.

Finally, SHARC has shown to prevent the community domination effect. In all studied cases, especially with dynamic networks and aggressive scenarios, it yielded the best results, by preventing as much as possible the creation of monster communities. This virtuous behavior could benefit from the introduction of a link-stability criterion to prevent sporadic connections to interfere with the assignment process.

Besides, we have excerpted the wandering community phenomenon, that shall be addressed by all algorithms performing community detection on dynamic networks.

VIII. FUTURE WORK

The present article has proven the validity of our approach for distributed network partitioning using community detection. We therefore plan to extend our contribution to be even more suited to its primary application case: dynamic wireless ad hoc communication networks.

In next iterations of our research, we will therefore introduce the notion of link-stability in our assignment process. Our objective is to detect the robust community structures and avoid unstable links for communication and community assignment. For this purpose, SHARC shall be adapted for weighted networks, in order to consider quality and stability of communication links as weight of the network edges.

Then we will introduce some incentives to group in the same community neighboring nodes based on longer-term metrics, reflecting their membership of a common social group or their sharing of interests.

As seen on some dynamic scenarios, the wandering community effect can impair the community assignment evaluation. Implementing a counter-measure to prevent this is also under investigation.

Another important aspect of our research work will also be to consider the effects of lower protocol layers and communication medium. We want to estimate the impact on SHARC performances of message losses, due to interferences or collisions between packets, of transmission delay, of physical media limitations amongst other characteristics.

Besides, we plan to rely on even more realistic mobility models, or use some real-life traces and connection data, to ensure the performances of our protocol in close-to-deployment conditions.

REFERENCES