

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

The rhythm of the crowd: Properties of evolutionary community detection algorithms for mobile edge selection

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:

Belli D., Chessa S., Foschini L., Girolami M. (2020). The rhythm of the crowd: Properties of evolutionary community detection algorithms for mobile edge selection. *PERVASIVE AND MOBILE COMPUTING*, 67, 1-15 [10.1016/j.pmcj.2020.101231].

Availability:

This version is available at: <https://hdl.handle.net/11585/802037> since: 2021-02-19

Published:

DOI: <http://doi.org/10.1016/j.pmcj.2020.101231>

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>).
When citing, please refer to the published version.

(Article begins on next page)

This is the final peer-reviewed accepted manuscript of:

Belli, D., Chessa, S., Foschini, L., & Girolami, M. (2020). The rhythm of the crowd: Properties of evolutionary community detection algorithms for mobile edge selection. Pervasive and Mobile Computing, 67

The final published version is available online at
<https://dx.doi.org/10.1016/j.pmcj.2020.101231>

Rights / License:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>)

When citing, please refer to the published version.

The Rhythm of the Crowd: Properties of Evolutionary Community Detection Algorithms for Mobile Edge Selection

Dimitri Belli^a, Stefano Chessa^{a,c}, Luca Foschini^b, Michele Girolami^c

^a*Department of Computer Science University of Pisa, Italy, email: name.surname@di.unipi.it*

^b*Dipartimento di Informatica: Scienza e Ingegneria University of Bologna, Bologna, Italy, email: name.surname@unibo.it*

^c*Istituto di Scienza e Tecnologie dell'Informazione, National Council of Research, Pisa, Italy, email: name.surname@isti.cnr.it*

Abstract

The Multi-access Edge Computing (MEC) paradigm increases the computational capabilities of distributed sensing architectures, such as Mobile CrowdSensing platforms, which are designed to collect heterogeneous data from the crowd by exploiting mobile devices. In this context, our work focusses on the impact of three community detection algorithms to our edge selection strategy. In particular, we study TILES, Infomap, and iLCD which are specifically designed to identify evolving communities of users in dynamic networks. Our analysis is based on the ParticipAct data set that offers real human mobility data. We first measure the quality of the data set during an observation period of 1 year, during which the data set provides the 75% of the expected traces collected by approximately 170 users. We then compare some structural properties of the communities detected, namely Similarity, Forward Stability, Cohesion and Coverage. We conclude our study with a performance analysis of the selected Mobile MECs by varying the community detection algorithms adopted. In particular, we measure the latency and the number of satisfied requests and we show that the average latency obtained with Infomap is slightly lower than that of the other algorithms, while the average number of satisfied requests is higher when we adopt the TILES algorithm.

Keywords: Community Detection; Multi-access Edge Computing; Mobile CrowdSensing; Social Mobility.

1. Introduction

The widespread availability of mobile devices equipped with a rich set of sensors and short-range communication interfaces has enabled the development of participatory and opportunistic Mobile CrowdSensing (MCS) [1]. The MCS paradigm aims to sense the city pulse, leveraging a crowd of citizens freely roaming through public and private spaces along with their smartphones, acting as powerful mobile sensors. At the same time, in recent years, the Multi-access Edge Computing (MEC) approach has been standardized as part of the future 5G networks evolution. MEC roots in the idea to evolve the traditional two-layer cloud-device communication model by adding, close to the base stations, another middle layer. The MEC layer consists of powerful nodes and micro data-centers providing services to nearby users according to the *locality* principle. MEC advantages include latency reduction, bandwidth saving and easing the mobile devices computation by enabling low-cost computation/storage offload from the device to the MEC layer. The combination of the MCS paradigms with the MEC approach [2, 5] is a new research area focusing on the potentialities of humans and on their ties for the purpose of optimizing a data collection campaign. The advantages of such combined approach are manifold. Firstly, the MEC reference architecture allows to deploy edges directly on the sensing region by increasing the reliability of the MCS architecture. More specifically, edges can be considered as an extension of the MCS back-end, they can collect data sensed by end-devices as well as to store the data back to the cloud. Moreover, edges can be activated on-demand according to a specific requirement of the MCS data collection campaign. As a meaningful example, we report the case in which it is required to sense data from a small area of the region for a limited time period. In this case, only the edges deployed on such area will be activated for the data collection. Lastly, edges can act as fixed and mobile (FMEC and M²EC). We envision that specific user's devices will play the role of M²EC during a MCS initiative. Such idea allows to exploit the user's mobility for the purpose of collecting data from the user's gatherings. Under this context, we argue that the selection of the M²ECs assumes a key-role for a MCS data collection campaign. We started investigating in [2, 5 and 46] the impact of social ties and of the community structure for the purpose of the M²EC selection. This paper focusses on that specific issues and in particular we face with the problem of evaluating different community detection algorithms in order to select a representative set of M²ECs. We identify three remarkable community detection algorithms available in literature, namely TILES, Infomap and iLCD. All of them are designed for evolutionary networks, and we analyse the communities discovered in terms of 4 metrics, Similarity, Forward Stability, Cohesion and Coverage. All of such metrics aim at measuring how strong, robust and stable the communities detected are. We compare the metric results by adopting the mobility traces of the ParticipAct MCS living lab [7]. ParticipAct is a MCS experiment that involved about 170

volunteer students from the University of Bologna for more than two years. We first analyse the quality of the data set in terms of the amount of traces collected and of the number of active users. The results obtained from the comparison of the community detection metrics do not highlight the existence of the *optimal* community detection algorithm for the purpose of M²EC selection, rather such results claim for the possibility of designing a more specific algorithm for our scenario. Finally, we analyse the impact of the community detection algorithm to our M²EC selection strategy. To this purpose, we analyse the performance of the M²ECs selected in terms of latency and number of satisfied requests. From our analysis we observe that the Infomap algorithm provides slightly lower values of the latency with respect to the others, while for what concerns the number of satisfied requests is higher when we adopt TILES.

The approach we follow in this work presents some novelties with respect to the current literature. Firstly, we envision the combination of the CrowdSensing paradigm with the Multi-Access Edge Computing, through which specific volunteer's devices can also play the role of Mobile MECs. Such devices act as proxy for devices of the community they join, in order to share sensing tasks as well as to upload sensed data back to the Cloud. Secondly, we assume that devices can exploit the full potentialities of proximity network interfaces, such as WiFi Direct and UltraWide Band (e.g. U1 chipset). Indeed, they offer the possibility of exchanging information with nearby devices in an opportunistic way, by exploiting the locality principle. Finally, we address the problem of selecting which devices can be elected as MEC. Our strategy focusses on the centrality of devices inside their community as a gradient for an optimal selection. It is worth to notice that we adopt evolutionary community detection algorithms specifically designed to detect communities in a dynamic network.

The remainder of the paper is organized as follows. Section 2 reports several methods and techniques commonly adopted in literature for detecting communities, covering both static and dynamic approaches. Section 3 describes our reference scenario and a general-purpose M²EC selection strategy. We also introduce in this section the three community detection algorithms selected and the evaluation metrics. Section 4 presents an analysis of the ParticipAct data set, in which we measure its quality in terms of active users and the amount of reported traces. The Section 4 also includes the comparison of the evaluation metrics, whose effectiveness is discussed in Section 5.

2. Related Works

Without claiming to be exhaustive, in the following we introduce the main techniques and metrics adopted for community detection in dynamic networks. Section 2.1 summarizes three main approaches for detecting communities, while Section 2.2 reviews some key-metrics adopted to measure properties of the communities detected of interest for the purpose of this work.

2.1. Community Detection in Dynamic Networks

Real-world networks of people are fundamentally organized according to community structures that evolve over time. In the recent years, the scientific community has been working on methods that can reveal such structures in complex networks [8, 9] by adopting very different approaches. Authors of [10, 11, 12] provide a systematic review of state-of-the-art algorithms and methods for detecting communities also in dynamic networks. This section reviews some of the recent techniques that can be adopted to detect communities, by selecting those works useful for the purpose of the selection of mobile edges.

As discussed in [12], time is a key element that can be used to detect communities evolving along the time. More specifically, communities detected at time t can only depend either on the current state of the network, namely, non-temporally smoothed, or on the past and the current state of the network, namely, incrementally temporally smoothed, or on the past and the future evolutions of the network, namely, completely temporally smoothed. Furthermore, each of these categories can be subdivided in subcategories based on the technique adopted to face the community discovery problem. The subsequent overview of community detection algorithm follows this approach, giving the priority to online algorithms. This class of algorithms is characterized by the processing of the graph in a piece-by-piece manner, without dealing with the entire input from the start. In other words, such algorithms explicitly leverage information about the network and the community structure at previous time step to identify communities at the current step. In other words, they manage the past to model the present. The rationale is that online algorithms, rather than the offline ones, provide better performance in the network partition process and they are able to optimize the community discovery procedure dynamically as well.

Each of the three community discovery categories correspond to a different definition of community on the basis of the time process considered to detect the optimal community. Consequently, the three categories respectively gather approaches non-temporally smoothed, incrementally temporally smoothed, and completely temporally smoothed.

Focusing on non-temporally smoothed algorithms, they have been originally designed for static communities, and they have also been used to detect evolutionary communities [13, 14]. These community discovery algorithms analyse the evolution of the network step-by-step, by considering for each step an optimal partition to be matched across the different time-steps. Authors of in [15], for instance, focus on finding stable communities at each network snapshot through a

bootstrap resampling procedure guided by significant clustering. Another example that considers evolving network as represented by a series of static snapshots is presented in [16]. Here, communities are detected through the Louvain algorithm [17] and matched using a similarity metric. The main advantage of this approach is the possibility of using the detecting community algorithm at each step in parallel and reduce computational costs. The drawback is in term of instability in the community detection due to the changes in the evolution of community structures at different run of the algorithm. A more advanced solution is implemented with Infomap [18] algorithm based on the map equation [19]. Infomap detect communities by minimizing the map equation. More specifically, the goal of Infomap is to find communities of nodes such the paths connecting the nodes is the shortest. Paths are discovered by a random walker and every step of the walker is coded with a sequence of codes. The random walker stops its process, when it is not possible to further minimize the length of such coded-paths.

As regard incrementally temporally smoothed algorithms, the community discovery phase iteratively processes the evolution of the network along with or without the community discovery process in the previous n -steps, so to detect communities in the last step. The iterative process can either find communities for the initial state of the network [20] or find communities at the current step using both present and past information [21]. An example of community detection algorithm for this category is given by [22] in which the authors propose a dynamic modularity-based clustering procedure for network partition in which changes come as a stream. In [23], instead, the authors introduce a game-theoretic approach for community detection where each node choses the connections with other nodes on the basis of the maximization of a utility function. The procedure creates communities at each snapshot by simulating a decision-making process that end when the game reaches the Nash equilibrium [24]. The previous examples discover communities by updating the network partition at previous step using a global approach. In any case, belong to this category also the community discovery algorithms that implement a local approach over the previous assumptions and following a set of rules. Example of community detection algorithm for dynamic networks that use the local approach and are based on a set of rules are TILES [25] and iLCD [26]. The first is an online algorithm that follows a local topology perturbation procedure to detect communities in dynamic networks. The second uses a multi agent system to handle the evolution of communities over time. The advantage of these approaches is mainly in term of optimization of the time of community discovery since the algorithms take advantage of existing communities in the past period to identify those in the current one. The drawback is in term of difficulties to parallelize the process of community detection because each iteration depends on the previous one.

Finally, in completely temporally smoothed algorithms, the community discovery phase does not consider the different steps of the network evolution independently, and the community discovery procedure is performed in a single process. As in the previous category, also in this case we can identify subcategories on the basis of the method adopted. Some methods neither allow to nodes to switch communities nor allow to the communities themselves to appear or disappear while the network evolves [27, 28]. Other methods consider that communities do not show homogeneity along time and, consequently, they compute the best partition on average over a time period [29, 30]. And other use stochastic block models to allow nodes to switch between communities while maintaining fixed the number of communities and their density along time [31-33]. In addition to the above, we can add a subcategory that gathers those methods that allow both node switching and community evolution simultaneously [34-37]. The algorithms in this category do not suffer of the problem of instability and have more potential to deal with local anomalies that affect the community discovery procedure. However, the drawback is in terms of the fixed number of communities discovered and all of them lack operations such as merge or split. In addition, they are not able to handle real-time community detection because the past network partitions might be different to the ones of the subsequent steps.

2.2. Properties of Communities

The goodness of community discovery techniques can be evaluated by considering the application of metrics for community discovery in static or dynamic graphs based on the properties of the communities themselves. Concerning the metrics to evaluate clustering techniques in static networks there can be single-criterion or multi-criterion scores. Belong to the single-criterion score well-known metrics that consider the number of edges entering the cluster (i.e. the community) such as many variations of Modularity [3], metrics that consider the number of edges needed to be removed to disconnect nodes in a cluster from the rest of the network, such as the Edges cut [6], metrics that consider the fractions of all possible edges leaving the cluster, such as the Cut Ratio [6], and so forth. Differently, belong to the multi-criterion score evaluation metrics such as Conductance and Expansion [38]. While the first metric measures the fraction of the total edge volume that points outside the cluster, the second measures the number of edges per nodes that point outside the cluster. All these metrics provide insights of how communities are well-expressed on the basis of their dimension. Evaluation metrics for dynamic graphs must necessarily take into account also the temporal aspect other than the simple distance between nodes. Several variations of the metrics in use for static graphs (such as Modularity, Conductance and Expansion) have been adapted to evaluate the goodness of the clusters in dynamic graphs [47]. Most of such techniques consider the measure that characterize the cluster at the next snapshot with respect to its variation to the current one, providing effective solutions to test the community discovery technique in use. However, no one of them fit our necessity

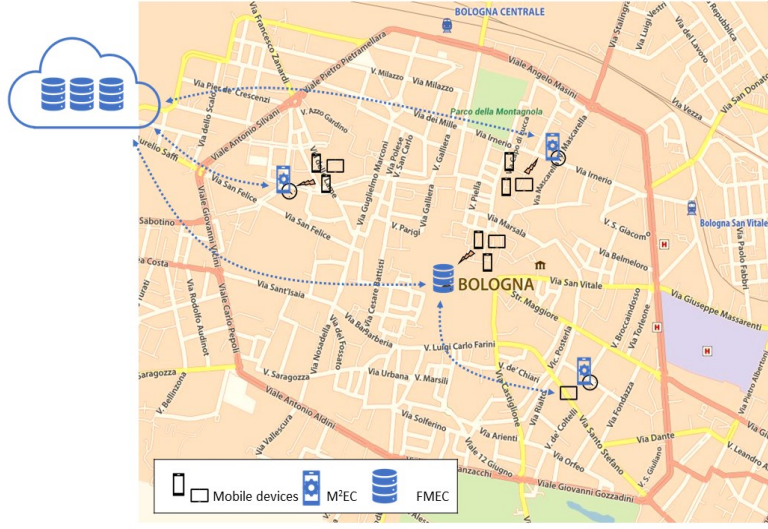


Fig. 1. The System architecture for a Mobile CrowdSensing platform.

of finding a comprehensive method to test the goodness of community discovered (for mobile edge selection) because they do not consider cross and intra layer dimensions together as well as the connection between community nodes and the coverage that the communities detected provide to the overall dimension of the network. In order to overcome such shortages, we have developed a set of metrics for measuring the effectiveness of community detection algorithms that consider all the previous aspects, complementary with each other for the accurate assessment of the communities identified. Details about these metrics are given in the following section.

In the following sections, we consider three remarkable community detection algorithms, namely Infomap, TILES, and iLCD. They are based on different techniques, but they all provide communities that evolve with the time. We present an overview of such algorithms in Section 3.1 by highlighting pros and cons in the community discovery procedure. It is worth to notice that our intention is not to rank such algorithms, rather we analyse their effectiveness for the selection of edges in a Cloud-based infrastructure. To this purpose, we analyse such algorithms on the basis of 4 evaluation metrics that we consider representative of the problem we address.

3. Selecting Mobile Edges in a Social Context

MCS architectures can be extended with edge nodes acting as proxy to and from the Cloud. As described in [4], the main idea is to design a dynamic architecture composed by mobile MECs (M^2EC s) able to collect information and to disseminate information from and to the crowd. M^2EC s are user's devices directly interacting with other devices in proximity, by exploiting short-range network interfaces. As for example, a M^2EC can collect data from its neighborhood by using the Wi-Fi Direct or Ultra WideBand interfaces. Data collected from M^2EC s are, in turn, directly uploaded to the Cloud by means of e.g. a broadband connection or transferred to what we refer to as Fixed MECs (FMECs) that represent traditional static MEC nodes. Hence, FMECs are stationary devices and/or micro data centres directly connected to the Cloud acting as intermediate storage units for M^2EC roaming around. Figure 1 shows an overview of our reference scenario.

M^2EC s extend the MCS architecture with a further opportunistic data retrieval layer, which may remarkably increase the performance of a MCS campaign. Under this context, we consider two metrics of particular interest: latency and the number of satisfied requests. The latency measures the time required in order to propagate a task. A task is an action that the user or its device are required to complete, such as collecting sensorial information from people roaming in a specific region or retrieving feedbacks from users after a social event. The lower the latency, the more quickly data can be retrieved. On the other hand, the number of satisfied requests measures how many tasks are successfully completed. More specifically, such metric assesses the capacity of a MCS architecture in completing tasks along with the time. Such metrics can be optimized when M^2EC s are selected in an efficient way. More precisely, the M^2EC s selected should be able to interact with a high number of user's devices so that to share and collect information. To this purpose, we consider as a crucial point the strategy to be used for the selection of M^2EC s. In [4] we already proposed a generic strategy for electing M^2EC s, such strategy aims at detecting those devices able to interact with other devices, so that to increase the probability of disseminating tasks to the crowd as well as collecting data from the crowd. We generalize such strategy by considering also the possibility of selecting a varying number of M^2EC as a percentage of the total number of devices. The strategy can be summarized in 3 steps:

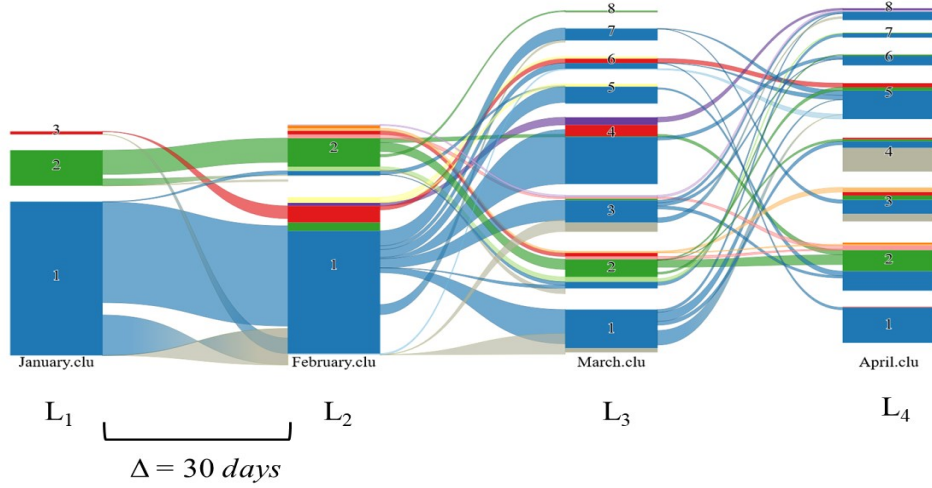


Fig. 2. Example of evolution of communities from January to April.

1. Periodically detecting communities in the network of devices;
2. measuring the centrality of the members of every community;
3. computing the number of M²ECs as a percentage of all candidate devices. The elected k M²ECs will act as mobile edges and will opportunistically collect data from the other devices in the next period, until the selection strategy will refresh the M²ECs.

In this work, we focus on the impact of the community detection described in the first step. To this purpose, we further analyse how to efficiently detect communities by using algorithms designed to capture the dynamism of the network.

A community can be defined as a set of devices among which there exist robust links lasting along the time. The idea is to identify ties that, with high probability, are useful to exchange information directly, in a peer-to-peer fashion. Communities of our scenario are supposed to be not static, rather they evolve with the time. This is the case of communities detected by analysing human location traces during meaningful time period. Three key factors affect the human mobility [39, 40]. Mobility is influenced by the social ties among people. In fact, people tend to move toward location in which they can meet friend or acquaintances. People tend to roam around few locations, this is the case of commuters that routinely visit their home and their working place. Finally, people tend to move for short and repetitive paths instead of jumping to far locations [41, 42].

The next section introduces three community detection algorithms specifically designed to capture evolutionary communities, we refer to [12] for in-depth review of community detection algorithms both for static and dynamic networks.

3.1. Evolutionary Community Detection Algorithms

The goal of the selection strategy introduced in Section 3 is to select a set of M²ECs acting as representative nodes for the communities identified. To this purpose, we adopt TILES [25], Infomap [18, 19, 43] and iLCD [26]. In order to clarify how we use such algorithms, we consider a network of contacts as a temporal graph [45]. A temporal graph describes the structure of the graph G at layer L . In this work, we refer to the layer L as a time window of width Δ (expressed in days) during which it is possible to observe the evolution of the network. The structure of a generic layer L is $G_L = (V_L, E_L)$ where $V_L = \{n_1 \dots n_k\}$ is the set of nodes observed at layer L , and $E_L = \{e_{ij} : (n_i, n_j) \in V_L\}$ are the edges among nodes active at layer L . An edge connecting two nodes describes the existence of a connection between them. More specifically, an edge can describe the co-location of node n_i and n_j in the same place at the same time, or an edge can model the existence of a network link between nodes n_i and n_j . As clarified in Section 4, we consider an edge connecting a pair of nodes as a co-location event, therefore e_{ij} if and only if node n_i is in proximity of n_j at the same time.

Since the network G evolves layer after layer, it is highly probable that also communities evolve accordingly. We show in Figure 2 an example of such evolution. The figure describes with an alluvial graph the evolution of the communities during 4 layers, January to April 2014 (obtained with the Alluvial Generator tool provided by Infomap¹). The graph shows the communities detected with Infomap during 4 months, January to April 2014. We first detect the communities layer after layer ($\Delta=30$ days). The example shows the existence of communities remaining stable along the time. As for example, communities with identifier 1 and 2 span during L_1 (January) and L_2 (February) as shown with the blue and green ribbons. Differently, other communities appear in L_3 (March) and they evolve to other communities in L_4 (April).

¹ Tool available at: <https://www.mapequation.org/alluvial/>

The rest of this section describes the three algorithms that we select for the detection of communities, namely TILES, Infomap and iLCD.

Tiles

The Temporal Interactions a Local Energy Strategy (TILES) [25] is an online community discovery algorithm that explores the flow of interactions between nodes over time through a domino effect strategy based on a label propagation procedure. TILES tracks the changes in the neighbour of those nodes that produce a variation in the interaction flow. The label propagation procedure is activated whenever a new interaction is produced by a given node. Each node is part of a community with two possible level of involvement: core or peripheral member. If a node is involved in at least a 3-clique, it is a core member. Differently, if a node is the neighbour of a core member, then it is considered as a peripheral one. Only core nodes can elect peripheral nodes to core nodes during the phase of label propagation. In detail, TILES takes as input an empty graph (G), a flow of streaming (S) for the arcs, a temporal threshold of observation (T), and a time to live (TTL) for the interactions. The flow of streaming is composed by co-location traces with the temporal information of the beginning and end interaction between pairs of nodes. The TTL affects the overall stability of the observations processed. The output is a chronologically ordered sequence of community sets, each of which representing the partition of the network at the end of the time interval T . At each step TILES reads a new interaction from the flow of streaming S and adds it to the graph G . Then, it may or may not make changes to the community structure based on the following scenarios:

- Both nodes appear for the first time in G . In this case no action is required until a new interaction is produced by the flow of streaming S .
- Either a node occurs for the first time and the other one is already labelled as peripheral, or both nodes are labelled as peripheral. As in the previous scenario, no action is required until a new interaction is produced by S .
- A node occurs for the first time in G and the other one is labelled as core node. In this case the new node gains the label of peripheral node.
- Both nodes already exist in G . In this case there are two possible sub-scenarios:
 - If the two nodes have no neighbours in common, both are labelled as peripheral nodes.
 - If the two nodes have at least one neighbour in common, all community members are re-labelled and the changes propagated to their neighbourhood.

Infomap

Infomap is a flow-based community detection algorithm that implements and optimizes the network partitions of the map equation [18, 19, 43, 44]. The method takes advantage of the duality between the discovery of community structures within a network and the minimization of the movements in the path of a random walker within such network. The movements of the random walker have associated an information cost for each module in which the network is split. Infomap measures just the theoretical limit given by the map equation, similarly to the Louvain modularity method [17]. More specifically, the algorithm splits the graph into modules by assigning a module at each node, and then it gathers neighbour nodes in super-modules recursively. To model the information flow, the Infomap runs a random walker that encodes every step with a code i.e. entering in a module, visiting a node, leaving a module etc. Therefore, it is possible to describe the set of movement of the random walker as a code.

The algorithm is applicable for finding two-level, multi-level and overlapping communities in several kind of networks such as weighted, directed, and multidimensional ones. In detail, Infomap works as follows. Initially, each node is associated to a module. Then, a random procedure moves each node in those neighbour modules that return the biggest decrease of map equation. If the map equation returns no decreasing associate to a neighbour module, there is no movement and the module remains in its original position. This procedure is repeated in several random sequential orders until any movement generates no diminishing in the map equation. Then the network is rebuilt following a hierarchical reconstruction.

iLCD

The intrinsic Longitudinal Community Detection (iLCD) is a meta-algorithm that considers the dynamics of a network to detect strongly overlapping communities in a temporal region [26]. iLCD takes as input a list of edges (i.e. co-location traces) chronologically ordered and start creating the network. The creation of a new community is determined by the minimal community pattern. This pattern is generally a clique of 3, 4, or more nodes. Each time a new edge is added to the network, the algorithm verifies if it enables to form a minimal community or not. Once the minimal pattern is detected, new nodes are integrated in the community based on two characteristic values. In detail, iLCD characterizes each community with two approximated values that are estimated and re-assigned each time the structure of a community is modified: the mean number of second neighbours (EMSN), and the mean number of robust second neighbour (EMRSN). The first represents the mean number of neighbours inside of a community a node can access with a path of length 2 or less. The second represents the mean number of neighbours inside of a community that can be accessed by at least two different paths of length 2 or less. The new node is integrated inside the community if the number of its neighbours at rank 2 inside the community is greater than EMSN, and the number of its robust neighbours at rank 2 is greater than EMRSN.

The last step that iLCD performs is the merging of the communities that are close with each other. Two communities are merged if they have a certain ratio of nodes in common. The similarity between two communities is given by a customizable threshold value. The higher the value the more intricate the communities.

3.2. Metrics for Mobile Edge Selection

We consider four metrics for measuring the effectiveness of the community detection algorithms, namely, Similarity (S), Forward Stability (F_s), Cohesion (C_h) and Coverage (C). Metrics are computed by considering the communities detected at layer i , namely $L_i = \{C_1^i \dots C_n^i\}$.

Similarity measures how much communities detected at layer L_i overlap. More specifically, the metric measures the similarity score among all the communities detected in the same layer, it is given by:

$$S(L_i) = \frac{1}{k} \sum_{\forall (C_x, C_y) \in L_i} J(C_x, C_y)$$

Where $k = \binom{n}{2}$ is the number of pair-wise combinations of n distinct communities found at layer L_i . The similarity S is therefore bound between $[0,1]$, the higher S the more communities resemble in each layer. Conversely the lower S , the more communities differ in each layer. For the purpose of the selection of M²EC, we are interested in algorithms with a low similarity score, so that each M²EC interacts with distinct nodes.

Forward Stability measures how much communities detected in one layer are still detected in the upcoming layers. More specifically, given $L_i = \{C_1^i \dots C_n^i\}$, the metric measures the average maximal similarity of each community in L_i with respect to the communities found in the forward layers, namely $L_{i+1} \dots L_{i+k}$. The maximal similarity for a community is given by detecting the max Jaccard score between such community and the compared ones. Forward Stability is given by:

$$Fs(L_i) = \frac{1}{m} \sum_{\forall (i \in L_i)} \max [J(C_x^i, C_y^j)], \forall j > i$$

where $|L_i| = m$ and J is the Jaccard index. The higher F_s the more communities found in a layer are still detected in the upcoming layers. We are interested in algorithms with a high value of F_s , so that to select a M²EC that is a good representative for a community layer after layer.

The Cohesion metric measures how much communities are tightly connected. To this purpose, we use the network density as measure of connectivity of a community, it is given by:

$$Ch(L_i) = \frac{1}{m} \sum_{\forall (x \in L_i)} D(C_x^i)$$

where $|L_i| = m$ and $D(C_x^i)$ is the density of community C_x^i . Cohesion is ranged in the interval $[0,1]$, it values 1 when all the communities of a layer are cliques, 0 if they are completely disjoint. Communities with high value of Cohesion are more likely well connected, therefore its representative M²EC has more probability to collect and share information with the members. Conversely, communities with a low Cohesion score are disjoint and its M²EC has low chances to interact with the members. We are interested in algorithms detecting communities well connected.

Finally, we measure the Coverage of a layer as the ratio between the number of distinct nodes that join at least one community with respect to the total number of nodes detected in the whole data set. Coverage measures how many nodes the algorithm is able to assign to a community. For the purpose of this work, we are interested in algorithms with a high score of Coverage, since we are interested in selecting a set of k M²ECs acting as representative for all the nodes in the data set. The coverage of a layer is given by:

$$C(L_i) = \frac{1}{N} \sum_{\forall (x \in L_i)} |C_x^i|$$

where N is the total number of nodes detected in the whole data set.

4. Experimenting Community Detection Algorithms with ParticipAct

We now analyse the metrics defined in Section 3.2 for each of the three algorithms. To this purpose, we start our analysis with a description of the data set used, namely ParticipAct [7]. We first review how the data set has been collected and then we analyse its quality in terms of users and traces collected (see Section 4.1). We then focus on the community detection algorithms by first analysing the structure of the communities they return, in terms of average cardinality and number of communities detected layer after layer. Finally, we present the results of the evaluation metrics.

The analysis we present in Section 4.2 is obtained by varying two orthogonal dimensions, namely the duration of the data set and the width Δ of the layer. Concerning the duration, we consider each of the 12 months in 2014. For what concerns

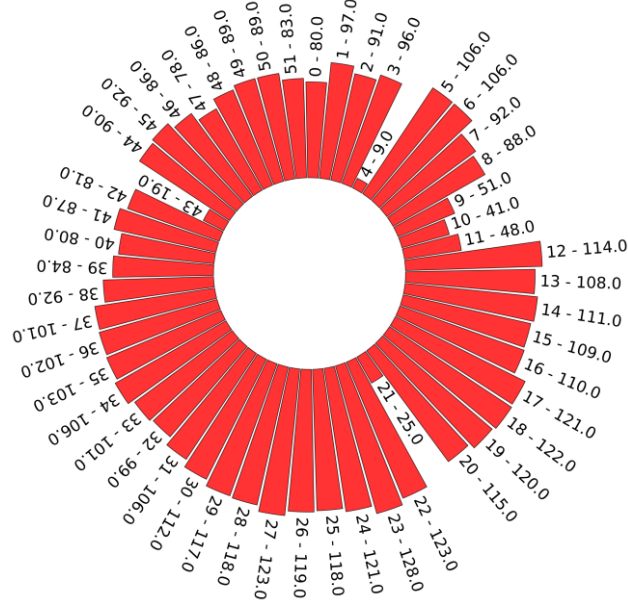


Fig. 3. Aggregated number of active users in 2014.

the width of the layers, we consider three settings $\Delta = 1$, $\Delta = 2$ and $\Delta = 7$ days. The combination of duration and width allows us to explore several configurations of the ParticipAct data set.

4.1. The Data set in Brief

ParticipAct is a CrowdSensing project designed to demonstrate the effectiveness of the MCS paradigm with a real-world data collection campaign. ParticipAct's users are mainly students from the University of Bologna that accepted voluntarily to join the project. In order to increase the recruitment of the students, the organizers of ParticipAct offered a smartphone with a MCS app pre-installed and a SIM card with a pre-paid traffic plan to be used for the purpose of the project. As a result, ParticipAct involved about 170 students in the period December 2013 to February 2015. The number of users vary along the time and according to the recruitment process. The data collected comprise the results of the CrowdSensing tasks submitted to users, their performance in answering the tasks, the success rate of the tasks as well as the mobility traces. The latter describes the geographic position of users, such information is obtained by exploiting the Google location APIs. For each of the participants, the location is obtained by merging information from Wi-Fi Hot Spot coordinates, GPS and base station of the cell phone. Mobility in ParticipAct is characterized by university students roaming in Emilia-Romagna region and, primarily inside the Bologna city². For the purpose of this work we analyse the so-called co-location traces. Co-location traces track the proximity between users along the time, more specifically they report the identifier of the devices in proximity with a timestamp. Proximity is obtained by detecting those devices in the range of 500 meters for a time window of δ seconds. More specifically, mobility traces in ParticipAct are generally collected at 2.5 minutes' rate. Therefore, every 2.5', all the devices are expected to upload their position. We set the width of our time window $\delta=2.5'$. Therefore, if we detect devices A and B in proximity at time t , then we start a co-location between A and B at such time until A and B are no longer in proximity for at least δ time.

The sampling rate of the mobility traces might be not accurate for a number of reasons. Some notable examples are: devices might be switched off, users deliberately disable GPS or WiFI connection or devices are out of the range of any network. Moreover, such traces provide the information we need in order to measure the connectivity between a M²EC and nodes joining to its communities, since such traces report those nodes that can interact in a specific time period.

The mobility of the ParticipAct's users is affected by some temporal constraints driven by as daily scheduling of lessons, examination periods and break periods (e.g. Easter, Summer and Christmas holidays) during which the mobility changes. We first analyse the number of users resulting active during the 52 weeks of 2014. An active user reports its position at least once during a week. It is worth to notice that the amount of traces is affected by several factors such as the battery

TABLE I USERS AND TRACES REPORTED DURING 2014.

| | Users | Current traces | Expected traces | Ratio |
|---------|-------|----------------|-----------------|-------|
| average | 98 | 42782 | 56617 | 0.75 |
| min | 7 | 3332 | 4032 | 0.22 |
| max | 131 | 61497 | 75456 | 0.84 |

² 44°30'27"00 N, 11°21'5"04 E

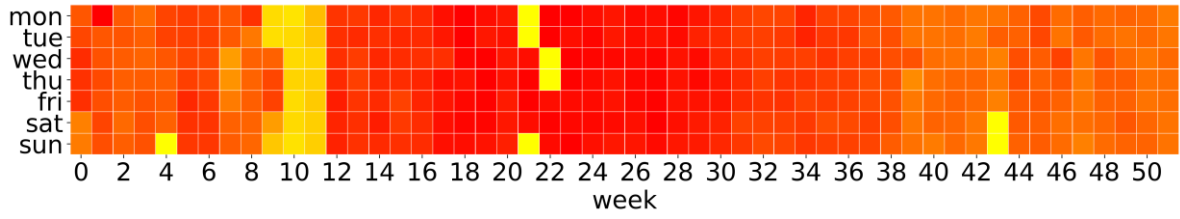


Fig. 4. Aggregated number of active users during the 52 weeks of 2014.

consumption of the MCS app, any crash of the MCS app and the network / GPS coverage of the smart phone. The combination of such factors can decrease remarkably the amount of traces reported by a user. Figure 3 reports the number

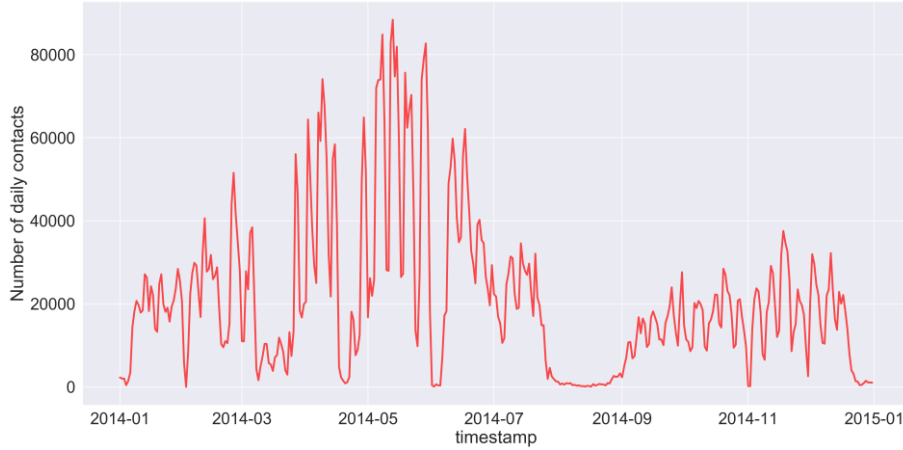


Fig. 5. Daily number of co-location traces in 2014.

of users active during the 52 weeks. The figure shows for each week the number of active users. A first consideration is that the number of users varies along the time. In particular, we observe some weeks during which the number of users reduces. This is the case of early weeks 4, 9 to 11, 21 and 43. As previously observed, such behaviour depends from different factors. However, as a general trend, the number of active users remains always acceptable, as also reported in Table I. Figure 4 further clarifies how the number of users vary along the week's days (from Monday to Sunday). From the heatmap, we do not observe any daily pattern in terms of active users. However, we observe some weeks during which the number of active users decrease (the yellow tiles). We finally study the variation of the co-location traces along the time, as shown in Figure 5. From the figure we observe that contacts among users vary according to the time period. We observe a general increase of the contacts from January up to June, after which contacts slightly decrease reaching the absolute minimum in August. The last part of the year is characterized by a slow new increase of contacts, from September to late November after which we observe a new decrease in December. We finally show in Figure 6, a geographical-aggregated representation of the coverage of the ParticipAct data set.



Fig. 6. Geographical representation of ParticipAct.

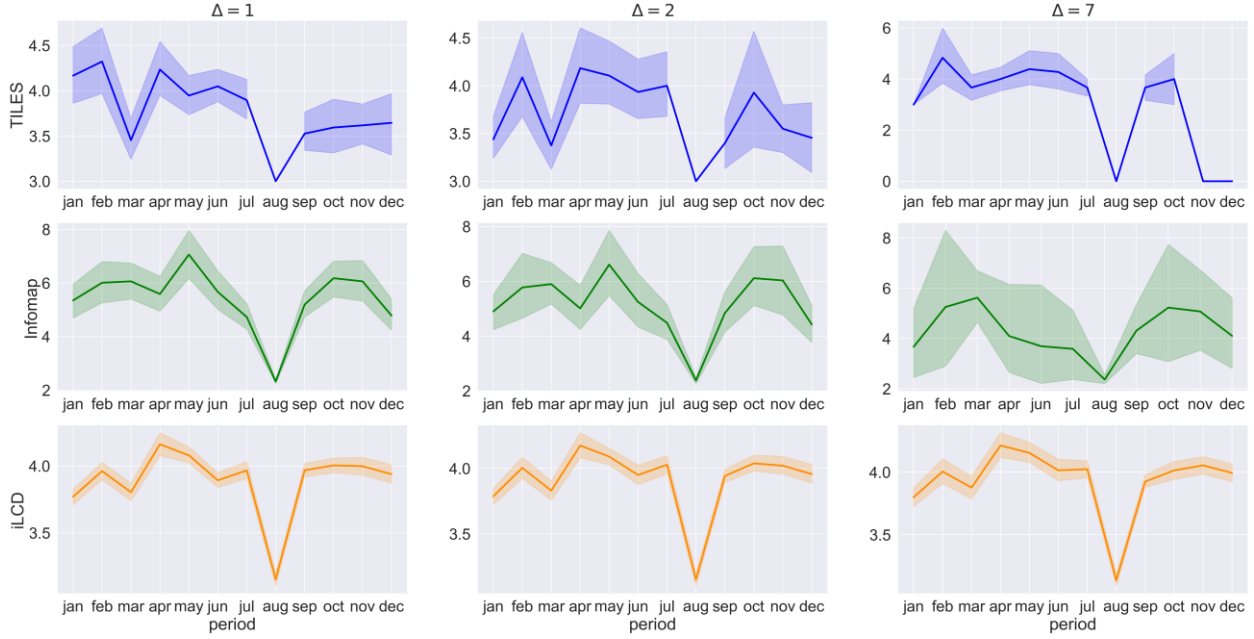


Fig. 7. Cardinality of the communities detected with different algorithms and with different values of Δ .

4.2. Analysis of Metrics

We now analyse the performance of the community detection algorithms previously introduced. The results are presented by showing the results by varying Δ in the range 1, 2 and 7 days. We start by studying the average cardinality of communities identified by the three algorithms on a monthly basis, as shown in Figure 7. Each row reports the results for the three algorithms, while the columns show the results by varying Δ . The curves also show an error band for each line which represents the confidence interval set to 90%.

The three algorithms detect communities whose cardinality vary along the months. As a general trend and for all values of Δ , we observe that the three algorithms capture one of key features of the ParticipAct data set, namely the academic year scheduling (as shown in Figure 5). More precisely, the scheduling of lessons and breaks has the effect of increasing/decreasing the social events among users. A meaningful example can be observed during the month of August. Such reduction is not caused by the low number of active users. In fact, from Figures 4 and 5, we observe that the number of active users still remain high during the weeks 31 – 36 (August 2014). More probably, users change their common social habits, by interacting with people not part of the ParticipAct experiment. Therefore, communities in ParticipAct tend to shrink. This can happen when users move back to their homes, travel abroad and, more generally, when they stop interacting with other ParticipAct’s users. Such behaviour is also confirmed by the lower number of communities detected in August for the three algorithms with respect any other month.

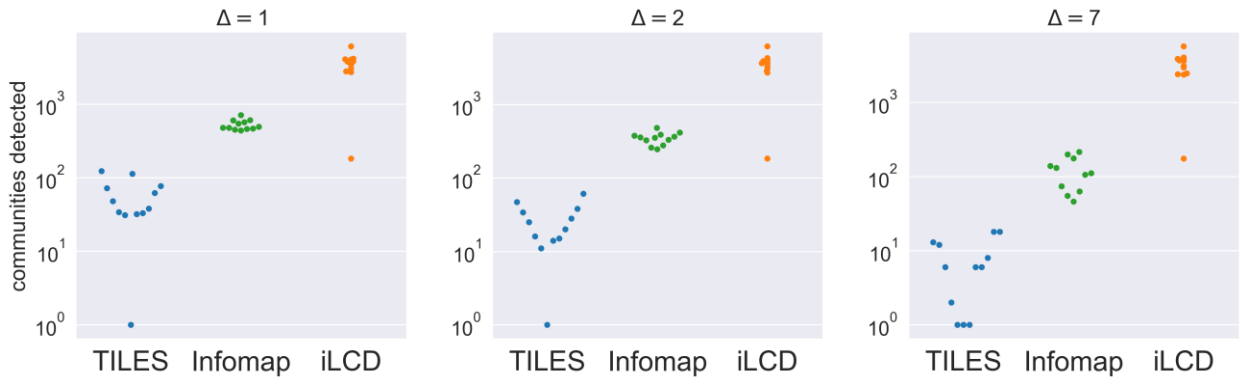


Fig. 8. Number of communities detected.

Moreover, the variation of the parameter Δ has the effect of extending the width of the layer. As for example, with $\Delta = 7$, the community detection algorithms return communities every 7 days. We observe different behaviours of the algorithms when changing Δ . TILES highlights the pattern we reported in Figure 7, according to which some periods of

the year are characterized by a low number of contacts. During such periods, TILES does not identify any strong communities as the result of a decrease number of contacts. More specifically, none of the contacts available during a layer create triangle-structures of the graph giving rise to a community. The results obtained with Infomap show also such trend, but Infomap still detects some small communities during periods with a low number of contacts. Finally, we observe that the variation Δ does not affect the results obtained from iLCD.

We further analyse in Figure 8 the community detection algorithms by showing the number of communities detected, on a logarithmic scale and by varying Δ . The results are obtained by computing the monthly average of the number of communities detected. From the figure, it emerges that iLCD detects the highest number of communities, with an average of 3387 communities detected every month. TILES and Infomap detect a far lower number of communities with respect to iLCD, TILES returns an average of 30 communities every month while Infomap an average of 330 communities.

We now analyse the evaluation metrics described in Section 3.2. Figure 9 shows, for every value of Δ , the distribution of the results of the evaluation metrics by aggregating the results from the 12 months (January to December 2014). Each graph reports a set of 3 box plots, one for every algorithm. As introduced in Section 3.2, our goal is to detect communities with low Similarity score but high scores of Stability, Cohesion and Coverage, so that the selection of the k M²ECs results

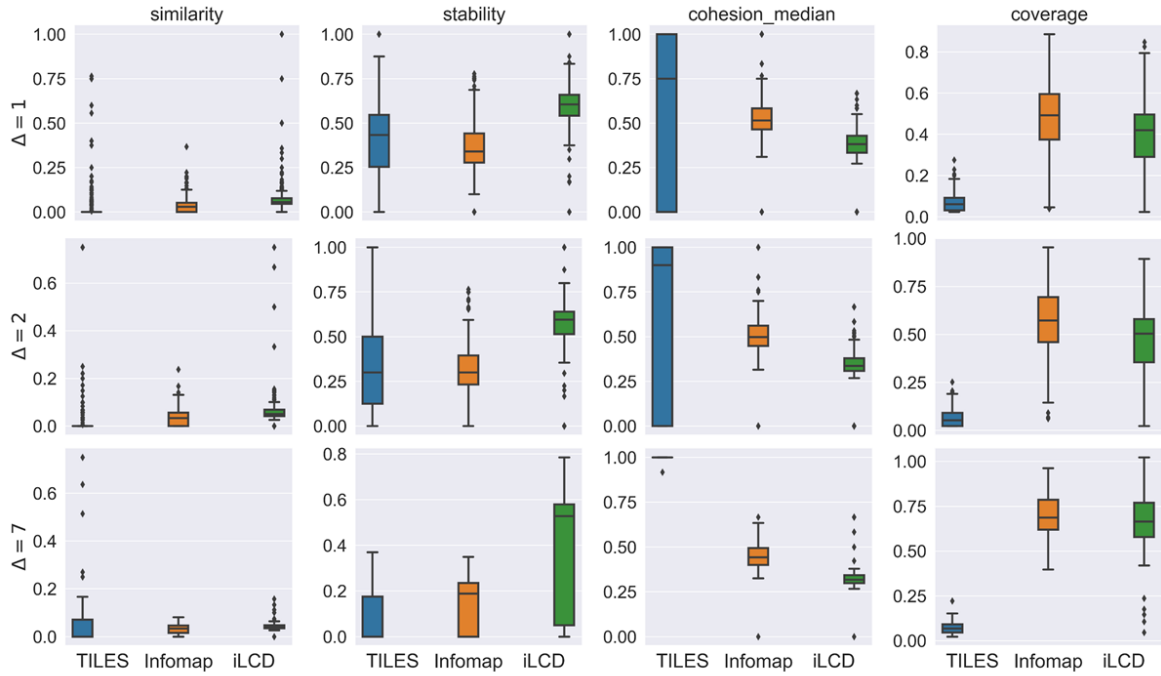


Fig. 9 Evaluation metrics for the three algorithms and different values of Δ .

effective along with the time. Under this context, we observe that the comparative analysis shown in Figure 9 does not provide an optimal algorithm that always outperforms the others. More specifically, we report in Table II the mean, standard deviation and 75th percentile of the evaluation metrics, with $\Delta = 1$ (similar values are obtained for different values of Δ). For every metric, we highlight the algorithm that minimize or maximize the metrics according to our objectives: minimizing Similarity and maximizing Stability, Cohesion and Coverage. TILES provides the best results for Similarity and Cohesion, Infomap provides the best results for the Coverage while iLCD provides the best results for Stability.

TABLE II STATISTICS OF THE EVALUATION METRICS, $\Delta = 1$

| | | Similarity | Stability | Cohesion | Coverage |
|---------|-----------------------------|------------|-----------|----------|----------|
| TILES | Mean | 0.03 | 0.4 | 0.57 | 0.06 |
| | Standard deviation | 0.11 | 0.23 | 0.43 | 0.04 |
| | 75 th percentile | 0 | 0.54 | 1 | 0.09 |
| | Median | 0 | 0.43 | 0.75 | 0.06 |
| Infomap | Mean | 0.03 | 0.35 | 0.51 | 0.47 |
| | Standard deviation | 0.04 | 0.15 | 0.14 | 0.18 |
| | 75 th percentile | 0.05 | 0.44 | 0.58 | 0.59 |
| | Median | 0.02 | 0.33 | 0.51 | 0.49 |
| iLCD | Mean | 0.07 | 0.58 | 0.37 | 0.39 |
| | Standard deviation | 0.08 | 0.15 | 0.14 | 0.18 |
| | 75 th percentile | 0.07 | 0.65 | 0.42 | 0.49 |
| | Median | 0.05 | 0.6 | 0.38 | 0.41 |

We further observe that the metrics we computer vary according to the months. Such variability depends very much from the month during which the metrics are computed. As a meaningful example, we report in the left-side of Figure 10 a comparison between the 3 algorithms concerning the Similarity metric with $\Delta = 1$, January to December. In this case, our goal is to obtain communities with low values of Similarity. TILES outperforms the other algorithms in some months (e.g. January, April, September to December) but in February, March, May, June and July Infomap results with the lowest value of Similarity. A different behaviour can be observed with the Stability metric, as shown in the right-side of Figure 10. In this case, it is more evident that iLCD provides the highest values of Stability during all the year.

We finally evaluate the performance of the strategy presented in Section 3. In particular, we analyse the performance of the M²ECs selected, by varying the community detection algorithm used and by varying the number of M²ECs, in the range 5% to 25% of the total number of devices. We consider two metrics for measuring the performance, namely latency and number of satisfied request. we analyse the latency and the percentage of the number of satisfied requests. The latency metric measures the time interval between the request generation (e.g. a task to be shared) and the time at which such request is assigned to a Mobile MEC, while the percentage of satisfied requests measures the amount of information successfully retrieved from nodes and assigned to the Mobile MEC for the final upload to the Cloud. We measured such metrics by exploring the different values of Δ we used for detecting communities, namely $\Delta = 1, 2$ and 7. For the purpose of this analysis we present results obtained with $\Delta = 2$ since they more clearly highlight differences between the community detection algorithms. Results obtained with other values of Δ follow a similar trend. Figure 11 shows the results obtained, where on the columns we report the 3 algorithms (TILES, Infomap, iLCD) and on the rows the results for latency and number of satisfied requests. From the figure, we observe that the trends for latency and requests satisfied is similar for the 3 community detection algorithms. The average latency obtained with Infomap is slightly lower than that of the other algorithms, especially when we adopt the eigenvector score in order to rank the M²EC candidates. While, concerning the average number of requests satisfied it is higher when we adopt the TILES algorithm. In this last case, we observe that the betweenness score always outperforms the eigenvector with all the community detection algorithms.

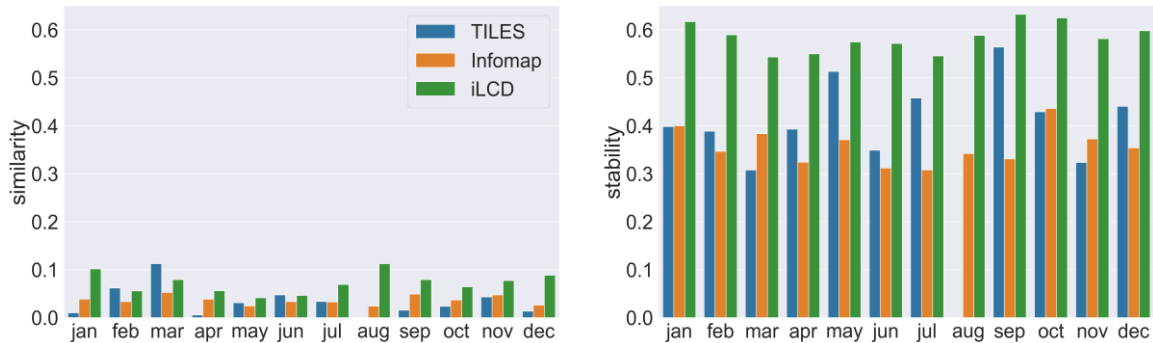


Fig. 10 Similarity and Stability for every month.

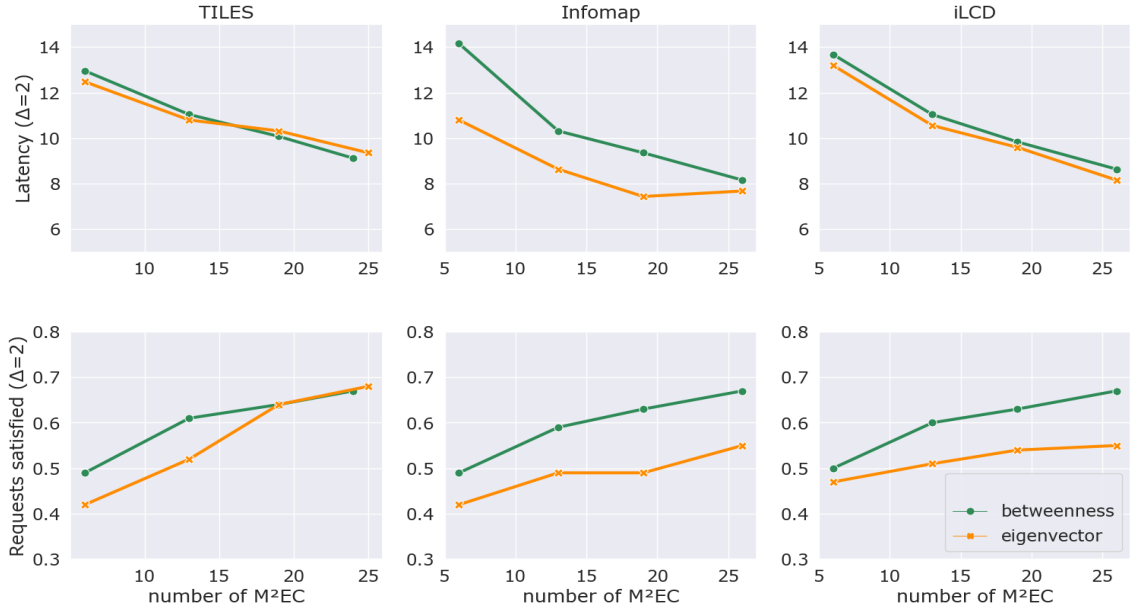


Fig. 11 Comparison of the performance of the M²EC Selection strategy with three community detection algorithms.

5. Discussion and Conclusions

The combination of Mobile Edge Computing with CrowdSensing architectures is a promising approach. We argue that CrowdSensing data collection campaigns will become popular in the next years, since they offer the possibility of collecting heterogeneous data with high temporal resolution and such data will be useful to better understand complex human dynamics. However, the convergence of the two approaches is still in their youth. We discuss in this section three key-aspects claiming for a further investigation.

5.1. Toward an Edge-Oriented Community Detection Algorithm

The results presented in Section 4 show that none of the considered algorithms can be adopted as the optimal ones, as all the algorithms do not perform well under at least one metric. It should be noted that this evaluation is dependent on the ParticipAct data set used for the experimentation. However, the data set had been deeply analysed and validated in comparison against other popular mobility CrowdSensing data sets in our previous work [7]. Furthermore, the fact that different algorithms can reach a good performance under different metrics suggests that another algorithm, possibly combining the best aspect of the considered ones, may actually achieve good performances on all the metrics.

From the point of view of the problem that motivated this work, namely the selection of M²ECs, all the four metrics (Similarity, Stability, Cohesion and Coverage) are important as they denote a specific, different aspect. In particular, since a M²EC selected on the base of the communities detected in a period will act as data collector for the next period, its ability to collect data will depend on the existence (and thus Stability) of the community it represents in the next period. The Cohesion metric instead represents how strong are the ties between the members of a community. The stronger they are and the most likely is that the selected M²EC will meet often the other devices in the community, a fact that will likely reduce the latency.

The last two metrics, Similarity and Coverage, are instead related. In particular, if two communities detected in a period are mostly overlapping (Similarity), then it is unnecessary to select two M²ECs for each of them, as just one may cover both communities in the next period. This motivates the need for a low Similarity score among the detected communities, while Coverage is an overall metric that indicates how many devices remain out of the detected communities (and thus for them the CrowdSensing platform should provide alternative ways of data collection, either by FMECs or broadband communications).

5.2. Optimizing the Mobile MEC Selection

From the point of view of the MCS platform it is also important to keep the number of M²ECs low, because they are personal devices of users and they experience higher costs in terms of energy consumption and memory overhead while acting as M²EC. For this reason, in order to limit the number of selected M²ECs without scarifying the ability to collect data from most of the devices (i.e. the coverage guaranteed by the M²ECs), the selection algorithms [46] generally sort the detected communities in decreasing order and starts selecting the M²ECs as representatives of the **largest** communities first, in the attempt to reach a good coverage with a small number of selected devices. A way to see the effect of Similarity and Coverage, is given by the analysis of the Contribution as shown in Figure 12. The contribution measures how many unique nodes a M²EC acting as representative for a community can provide. More specifically, the contribution can be defined as the number of devices in the community that are not already covered by the communities of the previously selected M²ECs. Figure 12 shows the contribution of each M²EC, selected starting from the largest communities, for communities obtained with TILES, Infomap and iLCD. It is seen the effect of the different behaviour of the three algorithms. In particular, Infomap, that finds one very large community and many small communities, can reach a high coverage already with one M²EC, while iLCD and especially TILES (that finds the smaller communities) need more M²ECs. However, since the Cohesion of the communities identified by Infomap is generally smaller, this means that the M²EC identified by Infomap are, in general, less efficient in collecting data from the other devices.

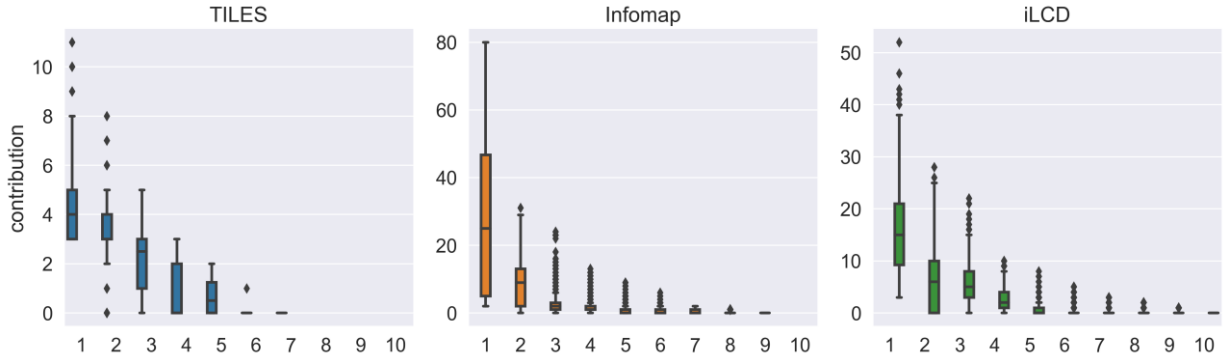


Fig. 12 Distribution of the contribution of the three algorithms.

5.3. The Impact of Mobile MEC on sensing architectures

The architecture we present in Section 3 has been designed with the goal of extending the Cloud potentialities with intermediate devices acting as proxy between user's devices and the Cloud. We consider that the Mobile MECs play a crucial role, since they provide two extra features: network optimization and infrastructure reliability. We argue that the results of our work moves toward a better comprehension of the adoption. In particular, we argue that our research contributes to:

- investigating the use of short-range network communications: the architecture we present in Section 3 assumes that user's devices interact through short-range network interfaces, such as WiFi Direct. We study its possible adoption in a concrete scenario as a viable, cost-effective alternative to broadband interfaces such as LTE and 5G. They are generally considered as the primary networking facility through which devices upload sensed data back to the Cloud. However, we argue that such interfaces might not always represent the ideal choice. Firstly, users have to consume their own traffic plan in order to interact with a MCS architecture. Although such plans are becoming cheaper and cheaper, users might not be motivated in still supporting a MCS initiative at they own costs. Secondly, broadband interfaces rely on an always-on network infrastructure. This requirement might not hold during crowded events or unpredictable events. For these reasons, the adoption of short-range communications can complement traditional network interfaces.
- studying how to efficiently select Mobile MECs: we propose a MEC selection strategy designed to identify central devices. We refer to centrality as a measure of the importance of a device in its community. To this purpose, we study the impact of the communities detected to the performance of the MECs selected and we discuss the need for more specialized algorithms specifically designed for our reference scenario.
- Designing a reliable data collection architecture: Mobile MECs play an important role in our CrowdSensing architecture, since they are part of the network infrastructure. MECs provide two core features: connectivity to/from the Cloud and computing capabilities. Such features increase the reliability of a full distributed architecture and they open the possibility of exploiting them as service providers, as discussed in the following recent works.

References

- [1] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, P. Bouvry, "A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities", *IEEE Communications Surveys & Tutorials*, 21(3), pp. 2419-2465, 2019.
- [2] P. Bellavista, S. Chessa, L. Foschini, L. Gioia, M. Girolami, "Human-enabled edge computing: Exploiting the crowd as a dynamic extension of mobile edge computing", *IEEE Communications Magazine*, 56(1), pp. 145-155, 2018.
- [3] Newman, Mark EJ. "Modularity and community structure in networks." *Proceedings of the national academy of sciences* 103.23 (2006): 8577-8582.
- [4] D. Belli, S. Chessa, A. Corradi, G. Di Paolo, L. Foschini, M. Girolami, "Selection of mobile edges for a hybrid crowdsensing architecture", *IEEE Symposium on Computers and Communications*, 2019.
- [5] P. Bellavista, D. Belli, S. Chessa, L. Foschini, A social-driven edge computing architecture for mobile crowd sensing management, *IEEE Communications Magazine*, 57(4), pp. 68-73, 2019.
- [6] Leskovec, Jure, Kevin J. Lang, and Michael Mahoney. "Empirical comparison of algorithms for network community detection." *Proceedings of the 19th international conference on World wide web*. 2010.
- [7] S. Chessa, M. Girolami, L. Foschini, R. Ianniello, A. Corradi, and P. Bellavista, "Mobile crowd sensing management with the ParticipAct living lab," *Pervasive and Mobile Comput.*, vol. 38, pp. 200-2014, 2017.
- [8] S. Qiao, N. Han, Y. Gao, R. H. Li, J. Huang, J. Guo, X. Wu, "A fast parallel community discovery model on complex networks through approximate optimization", *IEEE Transactions of Knowledge and Data Engineering*, 30(9), pp. 1638-1651, 2018.
- [9] F. Folino, C. Pizzuti, "An evolutionary multiobjective approach for community discovery in dynamic networks", *IEEE Transactions of Knowledge and Data Engineering*, 26(8), pp. 1838-1852, 2013.
- [10] M. Coscia, F. Giannotti, D. Pedreschi, "A classification for community discovery methods in complex networks", *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 4(5), pp. 512-546, 2011.
- [11] S. Parthasarathy, Y. Ruan, V. Satuluri, "Community discovery in social networks: Applications methods and emerging trends", in *Social Networks Data Analytics*, Springer, Boston, MA, pp. 79-113, 2011.
- [12] G. Rossetti, R. Cazabet, "Community discovery in dynamic networks: a survey", *ACM Computing Surveys (CSUR)*, 51(2), 35, 2018.
- [13] A. Bóta, M. Krész, A. Pluhár, "Dynamic communities and their detection", *Acta Cybernetica*, 20(1), pp. 35-52, 2011.
- [14] R. F. Bourqui, F. Gilbert, P. Simonetto, F. Zaidi, U. Sharan, F. Jourdan, "Detecting structural changes and command hierarchies in dynamic social networks", in *International Conference on Advances in Social Network Analysis and Mining*, IEEE, pp. 83-88, 2009.
- [15] M. Rosvall, C. T. Bergstrom, "Mapping change in large networks", *PLoS one*, 5(1), e8694, 2010.
- [16] N. İlhan, Ş. G. Ögüdücü, "Predicting community evolution based on time series modeling", *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Network Analysis and Mining* pp. 1509-1516, 2015.
- [17] V. D. Blondel, J. D. Guillaume, R. Lambiotte, E. Lefebvre, "Fast unfolding communities in large networks", *Journal of Statistical Mechanics: Theory and Experiment*, 10, 2008.
- [18] M. Rosvall, D. Axelsson, C. T. Bergstrom, "The map equation", *The European Physical Journal Special Topics*, 178(1), pp. 13-23, 2009.
- [19] D. Edler, L. Bohlin, M. Rosvall, "Mapping higher-order network flows in memory and multilayer networks with Infomap", *Algorithms*, 10(4), 112, 2017.
- [20] J. Shang, L. Liu, F. Xie, Z. Chen, J. Miao, X. Fang, C. Wu, "A real-time detecting algorithm for tracking community structure of dynamic networks", *arXiv preprint arXiv:1407.2683*, 2014.
- [21] K. Henderson, T. Eliassi-Rad, "Applying latent dirichlet allocation to group discovery in large graph", in *Proceedings of the 2009 ACM Symposium on Applied Computing*, pp. 1456-1461, 2009.
- [22] R. Görke, P. Maillard, C. Staud, D. Wagner, "Modularity-driven clustering of dynamic graphs", in *International Symposium on Experimental Algorithms*, Springer, Berlin, Heidelberg, pp. 436-448, 2010.
- [23] H. Alvari, A. Hajibagheri, G. Sukthankar, "Community detection in dynamic social networks: A game-theoretic approach", in *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mapping*, IEEE Press, pp. 101-107, 2014.
- [24] J. F. Nash, "Equilibrium points in n-person games", *Proceedings of the National Academy of Sciences*, 36(1), pp. 48-49, 1950.
- [25] G. Rossetti, L. Pappalardo, D. Pedreschi, F. Giannotti, "TILES: an online algorithm for community discovery in dynamic social networks", *Machine Learning*, 106(8), pp. 1213-1241, 2017.
- [26] R. Cazabet, F. Amblard, C. Hanachi, "Detection of overlapping communities in dynamical social networks", in *IEEE Second international Conference on Social Computing*, pp. 309-314, 2010.
- [27] T. Aynaud, J. L. Guillaume, "Multi-step community detection and hierarchical time segmentation in evolving networks", in *Proceedings of the 5th SNA-KDD Workshop*, 2011.
- [28] D. Duan, Y. Li, Y. Jin, Z. Lu, "Community mining on dynamic weighted directed graphs", in *Proceedings of the 1st ACM International Workshop on Complex Networks meet Information & Knowledge Management*, pp. 11-18, 2009.
- [29] L. Gauvin A. Panisson, C. Cattuto, "Detecting the community structure and activity patterns of temporal networks: A non-negative tensor factorization approach", *PLoS one*, 9(1), e86028, 2014.
- [30] V. Miele, C. Matias, "Revealing the hidden structure of dynamic ecological networks", *Royal Society Open Science*, 4(6):170251, 2017.
- [31] T. Herlau, M. N. Schmidt, M. Mørup, "Completely random measures for modelling block-structured sparse networks", in *Advances in Neural Information Processing Systems*, pp. 4260-4268, 2016.
- [32] K. S. Xu, A. O. Hero, "Dynamic stochastic blockmodels for time-evolving social networks", *IEEE Journal of Selected Topics in Signal Processing*, 8(4), pp. 552-562, 2014.
- [33] C. Matias, T. Rebafka, F. Villers, "A semiparametric extension of the stochastic block model for longitudinal networks", *Biometrika*, 105(3), pp. 665-680, 2018.
- [34] M. B. Jidia, C. Robardet, E. Fleury, "Communities detection and analysis of the dynamics in collaborative networks", in *2007 2nd International Conference on Digital Information Management*, IEEE, vol. 2, pp. 744-749, 2007.
- [35] T. Viard, M. Latapy, C. Magnien, "Computing maximal cliques in link streams", *Theoretical Computer Science*, 609, pp. 245-252, 2016.
- [36] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, "Community structure in time-dependent, multiscale, and multiplex networks", *Science*, 328(5980), pp. 876-878, 2010.
- [37] A. S. Himmel, H. Molter, R. Niedermeier, M. Sorge, "Enumerating maximal cliques in temporal graphs", in *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*, IEEE Press, pp. 337-344, 2016.

- [38] Hartmann, Tanja, Andrea Kappes, and Dorothea Wagner. "Clustering evolving networks." *Algorithm Engineering*. Springer, Cham, 2016. 280-329.
- [39] Chiara Boldrini, Andrea Passarella, "HCMM: Modelling spatial and temporal properties of human mobility driven by users' social relationships, *Computer Communications*, Volume 33, Issue 9, 2010, Pages 1056-1074.
- [40] J. Silvis, D. Niemeier, R. D'Souza, Social networks and travel behavior: report from an integrated travel diary, in: 11th International Conference on Travel Behaviour Research, Kyoto, 2006
- [41] M. Gonzalez, C. Hidalgo, A. Barabasi, Understanding individual human mobility patterns, *Nature* 453 (7196) (2008) 779–782.
- [42] Pappalardo, L., Simini, F., Rinzivillo, S., Pedreschi, D., Giannotti, F., & Barabási, A. L., "Returners and explorers dichotomy in human mobility. *Nature communications*", 6, 8166, 2015.
- [43] Martin Rosvall, Carl T. Bergstrom, "Maps of random walks on complex networks reveal community structure", *Proceedings of the National Academy of Sciences* Jan 2008, 105 (4) 1118-1123.
- [44] Ulf Aslak, Martin Rosvall, and Sune Lehmann, "Constrained information flows in temporal networks reveal intermittent communities", *Phys. Rev. E* 97, 062312 (2018)
- [45] Nicosia, V., Tang, J.K., Mascolo, C., Musolesi, M., Russo, G., & Latora, V. (2013). Graph Metrics for Temporal Networks. *ArXiv*, abs/1306.0493.
- [46] D. Belli, S. Chessa, L. Foschini and M. Girolami, "A Probabilistic Model for the Deployment of Human-Enabled Edge Computing in Massive Sensing Scenarios," in *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2421-2431, March 2020, doi: 10.1109/JIOT.2019.2957835.
- [47] Javadi, Saeed Haji Seyed, Pedram Gharani, and Shahram Khadivi. "Detecting community structure in dynamic social networks using the concept of leadership." *Sustainable interdependent networks*. Springer, Cham, 2018. 97-118.