Noname manuscript No.
(will be inserted by the editor)

Towards the Dynamic Community Discovery in Decentralized Online Social Networks

Barbara Guidi · Andrea Michienzi · Giulio Rossetti

Received: date / Accepted: date

Abstract The community structure is one of the most studied features of the Online Social Networks (OSNs). Community detection guarantees several advantages for both centralized and decentralized social networks. Decentralized Online Social Networks (DOSNs) have been proposed to provide more control over private data. Several challenges in DOSNs can be faced by exploiting communities. The detection of communities and the management of their evolution represents a hard process, especially in highly dynamic environments, where churn is a real problem. In this paper, we focus our attention on the analysis of dynamic community detection in DOSNs by studying a real Facebook dataset. We evaluate two different dynamic community discovery classes to understand which of them can be applied to a distributed environment. Results prove that the social graph has high instability and distributed solutions to manage the dynamism are needed and show that a Temporal Trade-off class is the most promising one.

Keywords Decentralized Online Social Networks, P2P, dynamic community detection

1 Introduction

Static features, such as clustering coefficient or centrality of Online Social Networks (OSNs) have been largely studied. In particular, the community structure is one of the most studied feature of the OSNs and it has attracted wide attention. The general notion of community refers to the fact that nodes tend to form clusters which are more densely interconnected through social relationships, relatively to the rest of the network. Communities reflect the behaviour of users and a high percentage of shared contents are generated by communities (or groups) of social users. During the last ten years, the increase of the amount of social data produced by social users, has put users inside several privacy issues. Centralized solutions for OSNs have been considered the main weak point in the problem of guarantee a certain level of privacy. To overcome this issue, decentralized solutions, known as Decentralized Online Social Networks (DOSNs), have been proposed. The decentralization includes several benefits, in particular in terms of privacy preserving, but it introduces new challenges that have to be faced. In particular, the problem of data availability is one of the most important ones. Current proposals manage the problem of data availability through a user-centric point of view, and no approaches take into account groups (or communities) of users. However, communities are useful to face other issues concerned DOSNs, such as information diffusion and privacy.

Several studies are proposed to manage the community detection in dynamic environments, such as Mobile Networks or Opportunistic Networks. However these studies manage scenarios in which mobile devices make contact with each other and they consider a community as a group of connected nodes.

By considering the importance of community detection and the high level of dynamism in DOSNs, this work propose a study concerns the need of community discovery algorithms in DOSNs. Our analysis have been conducted by exploiting a real dataset, gathered from

Barbara Guidi, Andrea Michienzi
University of Pisa - Department of Computer Science
Largo B. Pontecorvo, 56127 Pisa, Italy
E-mail: guidi@di.unipi.it, andrea.michienzi@di.unipi.it

Giulio Rossetti
ISTI - CNR
Pisa, Italy
E-mail: giulio.rossetti@isti.cnr.it

Facebook. We analyse two different community detection approach and we analyse the pros and cons of them when applied in a distributed environment. The final goal of our work is to show how centralized community detection differs from the reality and which kind of approach we have to use to develop a distributed community detection algorithm. To understand how centralized analysis differs from a distributed one, we define a set of community change events that permit us to understand the dynamic of the social network. We study dynamic communities from a user-centric point of view, by exploiting the ego network model, to evaluate how frequent the communities change over time and which events are more frequent. All our studies show the need of a distributed approach to manage the problem of the high instability of the social graph over time when we consider the online presence of users, and we show that among the three main classes proposed for Dynamic Community Detection [26], the Temporal Trade-off approach seems to be the most promising one. This work is an extended version of [17]. In the original work the main idea was to study the communities in a dynamic fashion and evaluate the possibility and the method to use dynamic communities to address a specific problem of DOSNs: the one of data availability. With respect to the original work, in this work we generalize and extend the possibility to use dynamic communities as support tool also for other interesting and challenging problems of DOSNs: information diffusion and privacy *in primis*, but also to the other problems. We put more emphasis also in the feasibility of adopting a specific dynamic community detection method and we show that some methods cannot be borrowed from literature because of the information they use to perform the task. We extended this work with a study involving a new community detection algorithm with a logic that is completely different from the one used in the original work. This is meant to show that the community detection problem can be understood in many different ways, thus leading to different and often non compatible results. We performed the same set of analyses with the second algorithm and compared them with the results obtained in the previous work. Then, we also extended the sets of analyses with a study in terms of similarity between communities detected by the two approaches.

In this contribution we pose ourselves two research questions:

1. is it really necessary to consider dynamic approaches when we study problems in context where churn is a real thing? Studying networks statically is much easier because they do not change over time, so the study can be carried out only once. Switching to a dynamic study adds some problems, one of which is addressed by the second research question.
2. Among all the techniques to study dynamic communities, is there any technique that fits better the classical scenario of the DOSNs?

As we will see in the following sections of this paper, node churn is critical in DOSNs, thus requiring us to study these networks in a dynamic fashion. Moreover, we will also show that not all dynamic community detection algorithms fit the scenario of DOSNs, making a class of approaches more suitable for the aim with respect to others.

The important contribution of this work is that, even we consider a specific scenario, such as a DOSNs, our contribution could be applied to other distributed systems (i.e. wireless sensor networks), by taking into account the specific constraints.

This paper is organized as follow. In Section 2 we describe the related work. In Section 3 we introduce the dynamic community analysis in DOSNs. A preliminary analysis is showed in Section 6. Section 4 introduce our DOSN scenario and how dynamic community detection is important to manage main DOSNs' issues. Section 5 show our study by focusing on two of the three dynamic community detection approaches. In Section 6 we introduce our results obtained by analysing a real dataset. Finally, conclusions and future work are presented in Section 7.

2 Related Work

In this section we describe the two fields involved in our work. First of all, we introduce current DOSN proposals by describing their characteristics. Afterwards, we introduce the state of the art in the dynamic community detection field.

2.1 DOSN's approaches

DOSNs [9] have been proposed, mainly, in order to overcome the privacy issues of the centralized OSNs. The decentralization of most of the current proposals is usually implemented by a P2P network.

Diaspora[1], with about 669,000 users, is one of the most successful DOSN proposal currently active and deployed in a decentralized way. A user joining the service must register himself to a so-called *pod*. Pods can be seen as servers containing the information of the users registered to them, and can communicate with

[1] https://joindiaspora.com/

each other thus forming a network. Diaspora also introduces the *aspects* as a mechanism to share information with specific subsets of one's contacts. Each user can define his own aspects which are sets of his friends on the DOSN. A content shared with an aspect will not be seen by users outside it. This mechanisms allow the users to have a high level of privacy both inside, with respect to other users, and outside, with respect to third parties, the DOSN.

PeerSoN [2] is one of the most well-known DOSN after Diaspora. It is implemented as a logical two-tier system. The first tier is used for the lookup service, while the second tier is used as a communication layer between users of the DOSN. The look-up service stores both the meta-data required to find users (IP address) and the data they store (profile information, contents, etc...). In the original work, authors used a Distributed Hash Table (DHT) to implement this layer. A peer, to connect to another peer, gathers the needed information from the look-up service, then directly connects to it.

SafeBook [8] is made of two layers: a P2P overlay implementing lookup services, and a user-centric social overlay implementing the main functionalities of the social network. The social overlay is composed by a set of structures named *Matryoshkas*, which are concentric rings of peers built around each user. Matryoshkas are connected through radial paths from the outer one, through inner ones, up to the core node. These paths are built over the social network itself by using trusted relationships. The social overlay guarantees a trusted data storage, profile retrieval, and an obscure communication through indirection.

LifeSocial [13] is a plugin-based social network. This philosophy makes it highly modular and easily extensible. Data within the service is stored by using a DHT and cryptography. In particular both symmetric and asymmetric techniques are used to provide a high level of privacy. Also messaging between users is handled through the overlay given by the DHT.

A similar approach is Cachet [23], which replicates profiles on the DHT to guarantee the data availability. Read-policies are used to increase the privacy of users by controlling who can read each user's data, while write-policies protects the system from malicious data overwrites. Access policies, both read and write, are enforced by the extensive usage of cryptographic techniques. One original contribution by this work is a social caching algorithm. Furthermore, a gossip-based algorithm is proposed to let peers exchange cached, unencrypted social data.

DiDuSoNet [15], similarly to others, is made of two layers: a lookup overlay and a social overlay. The lookup overlay is implemented with a DHT, instead the social overlay is a Dunbar-based social overlay where connections between nodes correspond to social relations between users in the Dunbar-based ego network and it is used to manage the main social services provided by the system. For instance, to address the problem of data availability, the concept of Point of Storage (PoS) is introduced as a particular instance of a replica-based technique. The number of replicas of each profile is minimized by considering only two replicas.

2.2 Dynamic Community Detection

Community Discovery is a relatively novel, yet intriguing, task in complex network analysis [1,3]. There is no a formal definition of the task that is widely accepted, but, intuitively, its goal is to identify clusters of highly connected nodes. A first definition of community is given by [6], where a community is defined as *a set of entities that share some closely correlated sets of actions with the other entities of the community.*

Up to now, most of the research in this field focused on static networks, modeled by static graphs that do not change over time. Unfortunately, this simplification does not describe well real-world scenarios and the dynamic nature of most complex networks. As for the static case, it is hard to formally define what a dynamic community is. A very abstract definition is proposed in [26], which does not make any assumption on the communities to be found and the method to find them.

Up to date, the two most practical models used to represent dynamic networks are the *Temporal Networks* and the *Network Snapshots*.

Temporal Networks model is the most complex method which provides all possible temporal details. In this approach the information is decomposed into elementary bricks: series of temporally ordered, timestamped, relations. This representation of the network, which is the closest to the real-world scenario, allow a very fine-grained representation of the dynamics.

The idea behind the **Network Snapshots** model is to aggregate data over a discretization of time. Rather than storing every single perturbation of the network, the network is observed at, possibly periodic, instant of times and its state is recorded. This gives us an ordered set of networks, each representing the state of the network as observed at a particular instant of time. Even if this model is less expressive than the previous one, it is also easier to use it as each snapshot can be considered as a standalone network.

2.2.1 Dynamic Community Detection approaches

Dynamic Community Detection Algorithms can be divided into three main classes [26]: *Instant-optimal Community Detection*, *Temporal Trade-off Community Detection*, and *Cross-Time Community Detection*. Each of these three classes corresponds to a different definition of Dynamic community with respect of which information is used to determine communities at a given time instant t.

In the **Instant-optimal Community Detection** class, communities existing at time t only depend on the state of the network at time t. No past or future, with respect to time t, information is used in discovering communities at time t. The network evolution is seen as a series of successive steps, which makes the Network Snapshot a more natural model to work with. In the second class, **Temporal Trade-off Community Detection**, communities defined at a time instant t depend on the actual state of the network and the past information, possibly up to the initial known state. Typically this is done by an iterative procedure which consist of an initial bootstrap and successive updates. During the bootstrap communities are found in the initial state of the network, while during the successive updates communities are updated using the current state of the network, current communities and other past information. Finally, we find in the **Cross-Time Community Detection** class all the methods that use all available information, i.e. past, current and future, to identify communities at instant t.

Dynamic communities show a life-cycle during their evolution. In detail, several studies [24,31,3] have analyzed the dynamic behaviour of social communities, and a list of events are proposed:

- *Birth*: this event is identified when a new community appears for the first time;
- *Death*: a community is vanished: all nodes belonging to the vanished community lose this membership;
- *Growth*: a community increases its size due to the adding of one or more nodes (or relations) to the social graph;
- *Contraction*: some nodes are rejected by a community thus reducing its size;
- *Merge*: two or more existing communities merge into a single one due to changes to the social graph;
- *Split*: a community, as consequence of node/edge vanishing, splits into two or more components;
- *Continue*: a community remains unchanged;
- *Resurgence*: a community vanishes for a period, then comes back without perturbations as if it has never stopped existing. This event can be seen as a fake death-birth pair involving the same node set over a lagged time period (example: seasonal behaviors).

Figure 1 shows a graphical representation of the above events. This toy example captures the eight events that regulates dynamic community life. In the first row Birth and Death; in the second row Growth and Contraction; in the third row Merge and Split; in the fourth row Continue; in the last row Resurgence.

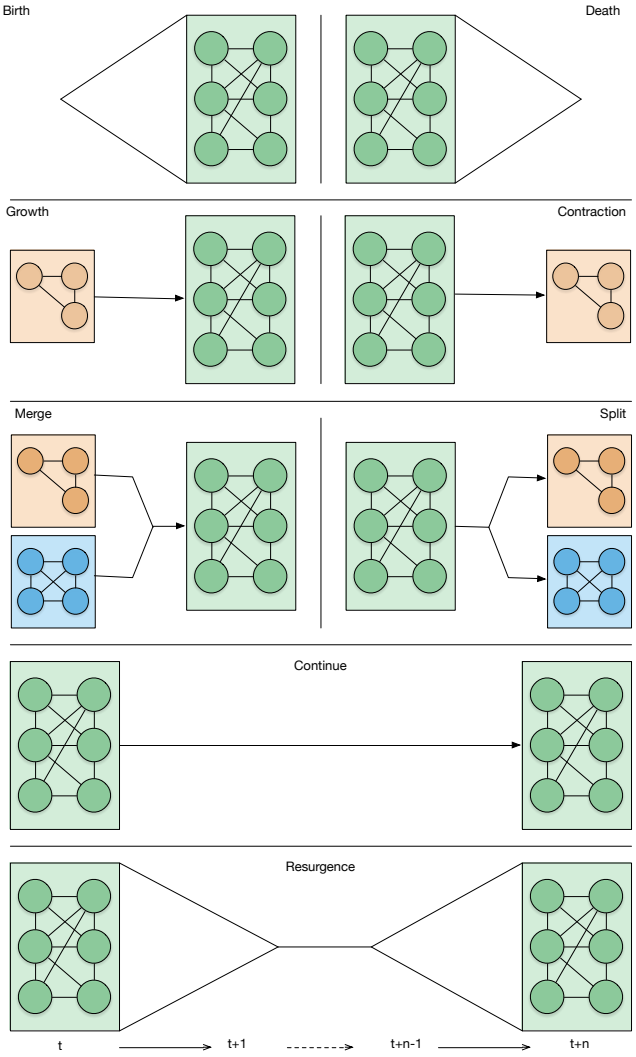

Fig. 1 Graphical representation of Community events.

3 Dynamic community analysis in complex dynamic networks

Dynamic Community Discovery is an interesting novel task in the area of Complex Networks. There are several techniques to address it, each showing peculiar strengths and weaknesses. Up to now, all the studies

concerning Dynamic Community Discovery focus on the a posteriori study of the evolution of the community structure through time and they are executed in a centralized way (low end entities can gather data which is transfered to powerful clusters where the actual computation is made). Clearly, Dynamic Community Discovery is very important and it can be used to better understand how networks evolve through time. However, the analyses are run only when all the data is collected, and there is no waste of computing power because all data is available when the computation starts.

However, distributed systems, such as P2P networks or IoT, require some constraints that were not needed in centralized systems, or in the a posteriori analysis. In detail, problems rise when the community structure itself, or a derivative analysis result, is needed on the fly as the system lives and changes. In this cases, a posteriori analyses are not useful tools because, for instance, results are too old to be used when they are ready. Moreover, there exist distributed systems which needs to change their behaviour on the fly in respect of the results of the analysis. One of the main constraint in these systems concerns the high dynamism of the network they model.

Several approaches proposed to manage the problem of community detection in social networks take into account the evolution of the social graph in term of friendship relationships (or co-authorships [30,29]), or in term of interactions between users (or call graphs [14]).

Focusing on a single user, its friendship relationships do not change so frequently. Instead, interactions of each nature (calls, emails, posts, tweets, etc...) suffer of a different level of dynamism. However, the study of the interactions graph represents a different evaluation of the social graph, because the interaction graph is an abstraction of the social graph that should be represented as a weighted and usually directed graph [16]. In a distributed system, such as a DOSN, an interesting evaluation concerns the study of dynamic community by considering the temporal behaviour of users. As showed in our previous work [28], the static view of an ego network and, as a consequence, its communities are completely different when we consider the time-varying ego network.

The aim of this paper is to show that there is a huge difference in the results when the analyses must be performed on the fly with respect to the a posteriori ones. In detail, we will take the DOSNs as case study and perform some investigations.

In the follow, we describe more in detail our DOSN's architecture by explaining how our architecture is organized. Moreover, we explain the problems of DOSNs, with a special focus on privacy, information diffusion, and data availability. Finally, we give our definition of the events occurred during the normal activity of a DOSN which involve the dynamic communities.

4 DOSN: our scenario

A current trend of DOSNs is the usage of a social overlay [15], which represents in some way the friendship relationships between users, to implement the needed services. The network topology resulting is generally known as a Friend to Friend network (F2F) in which users only make direct connections with people they know, i.e. their friends on the DOSN. Usually, the social graph of each user is referred by using a well-known social network model known as *Ego Network* [21]. The *Ego Network* is a structure built around the ego which represents the user's knowledge of the network. In fact, the Ego Network of a user is made of his direct friends, known as *alters*, and the existing ego-alter and alter-alter relations (Figure 2). Formally, each vertex $u \in V$ can be seen as an *ego* and $EN(u) = (V_u, E_u)$ is the ego network of u where $V_u = \{u\} \cup \{v \in V | (u,v) \in E\}$, $E_u = \{(a,b) \in E | \{a,b\} \subseteq V_u\}$ and E is the set of edges present in the original graph. $N(u) = V_u - \{u\}$ is the set of adjacent nodes of u.

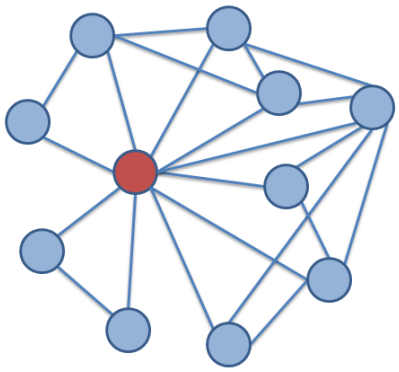

Fig. 2 This image shows an example of an Ego Network. The ego, the red node, is connected to all its alters, the blue nodes. Also the relations between alters are included in the Ego Network.

In a DOSN, the Ego Network model reflects the local and limited knowledge that each user has about the whole network. A F2F network can be formally represented by using an *Ego Network* to model the social graph and we assume a one-to-one mapping between the users of the OSN and the nodes of the DOSN [15].

4.1 Problems in DOSNs

Decentralization brings major benefits mostly to users in terms of privacy, but also introduces new interesting challenges and problems to be addressed by the developers. We now briefly discuss this problems:

- *Dynamism.* In a DOSN there are two types of dynamism: a social and an infrastructure dynamism. The social dynamism concerns social relationships which can change due to the variation of relations between users, i.e. users forming new relations or breaking existing ones, and of the total number of users registered on the DOSN. This kind of dynamism is present also in centralized OSNs, but in DOSNs it has a major impact on the service delivered because it affects the structure of the social overlay. The infrastructure dynamism is related to the underlying overlay network. Users arbitrarily decide when they go online or offline in the system, accordingly nodes in the underlying network representing users may appear or disappear. In different time instants, the available connections of the overlay may change in term of active links and nodes. Now, even though both types of dynamism are present in DOSNs, social dynamism is less dangerous because it is a quite rarer effect. Moreover, in this case, it is more common to create new links rather than destroying existent ones [20]. This high level of dynamism must thoughtfully be taken into account when developing the service and, most of the times, very specific algorithms are needed.
- *Data availability/persistence.* Without a central entity providing data, proper storing methods must be designed to let it always be available. Data availability is a real hard problem for every distributed environment. One of the main used technique is the replication [32]. Several proposed solutions have exploited the social overlay to store data among nodes in the networks. A current trend is to use trustworthiness to choose replica nodes because of the need of a high level of privacy inside the system, such as in My3 [22]. In [15,11] a friendship-based replication schema is proposed. A friendship-based replication schema chooses replica nodes by taking into account the friendship relationships between users. Indeed, consider an ego node e, only its friend nodes can be chosen to be its replica nodes.
- *Scalability.* Scalability is a crucial property of large scale systems. Simply mapping a social graph onto a distributed network can be very expensive due to the number of social links for each node, so the cost of mirroring the social network links into distributed network links can be high. It can also be very inefficient. As discussed previously, most of the social links are inactive (i.e. two friends who rarely interact online).
- *Topology.* Nodes should be connected according to their social connections in order to cluster friends in the overlay network. This should facilitate operations as information diffusion or data storage. As a downside, this would limit the availability and robustness of data access if a user has only few online friends.
- *Information diffusion.* This issue is related to how to deal with updates, i.e. new content generated by users. A key feature for a successful DOSN is the one of making the service the closest possible to a real-time service to supply the freshest possible informations to the users. For this reason it is very important to design an effective mechanism that implements Information diffusion. In centralized OSNs, the spread of new content to other users is granted for free in such system thanks to the uniqueness of this repository because all users can only receive informations through it. In decentralized systems, users have a limited knowledge of the network and communication between users' devices happen on the overlay. In DOSNs based on a Social Overlay, users can directly communicate each other if there is a social link connecting them [5], and, since data can only travel through social links, there is the need of specific information dissemination strategies to spread information over the DOSN.
- *Privacy.* While having an increased level of privacy was the main motivation to move to a decentralized implementation of the service, maintaining an overall high level of privacy is a complex problem for DOSNs. In the centralized version, privacy was granted by the service provider, as long as its servers are secure from attacks. The major downside is the fact that the service provider itself could maliciously violate the users' privacy by exploiting their data, for instance by selling them to third parts. In distributed services this is no more possible because, in principle, each user decides who can access their data. Current DOSNs typically make use of encryption, both symmetric and asymmetric, to preserve the privacy of a user with respect to other users.

In this paper, we focus our attention on the usage of community discovery to manage three of the main problems of DOSNs: Data availability, Information diffusion and Privacy.

4.2 Dynamic Community Analysis to manage DOSN issues

Community structure is of great interest in the study of complex networks. In addition to the interest by the algorithmic point of view, it is also interesting because it can uncover hidden properties if properly carried out. A relatively novel usage of community structure is the one to improve systems that can be modeled through a network by actively exploiting it in the implementation of the system. The usage of community in DOSNs represents a promising and uncovered solution. As discussed in Section 3, DOSNs suffer of a high level of dynamism and for this reason, we are interested in studying how communities evolve during the online activity of the system due to the online/offline of users to understand which events could happen and the frequency of them. Communities in DOSNs can be applied to manage the problem of data availability by implementing a new content-based replication technique to address data availability. For sake of clarity, a content based point of view concerns the problem of finding groups of users which are interested into the same content to minimize the number of replicas [17]. The presence of densely connected groups of nodes can be exploited to increase the level of data availability and to minimize the replicas. A possible approach could be to exploit the community structure to store at least one replica of the whole profile or of interest content for the users belonging to the community [17]. Furthermore, community structure can be exploited to manage the information diffusion issue by discovering groups of users which share the same interested and guide the information among the discovered communities. Finally, in terms of privacy, community structure can be exploited to study problem of privacy, and to guarantee a high level of privacy.

Some studies about community detection in dynamic P2P networks have been presented. Many of these approaches are basically just a distributed version of the Label propagation [25] approach. For instance, in [4] authors propose a revised label propagation divided in five phases, each of which has a different rule to update the labels of nodes. A simpler approach is presented in [18], where the rule to update the labels of the nodes is based on a similarity metric. Moreover, in [19], authors propose a distributed approach for local dynamic community detection and three implementation variants. In this case, the distributed nature of the algorithm induces a very weak consistency among the nodes of the network. Contrary to the presented works, the distributed approach we consider is missing in literature should be a pure Temporal Trade-off approach, which can be implemented by exploiting a P2P networks, and by exploiting a super-peer approach, when super-peer nodes can build and manage the evolution of communities.

4.2.1 How community change events affect DOSNs

In this study we refer to the events proposed in [30] and we do not consider the event *survive*, usually referred as *growth* and *shrink*, due to the fact that this event gives little information about the evolution of the communities in the network.

Considering the problems concerned DOSNs and, in detail the proposed community-based replication technique explained in Sec. 4.2, the events *birth*, *death*, *split* and *merge* can affect the level of data availability. Moreover, these events can affect the diffusion of the information. *Birth* events are critical, especially with respect to the data availability, and they are one of the main issue that has to be faced. Indeed, a newly formed community may have no information about the most fresh contents created by the ego and nodes inside such communities and it must find a way to retrieve the information. *Death* events, reported to give us more information about node churn in such dynamic context, are no concern in a replication technique because offline nodes do not need any content. Instead, they have a huge impact in the information diffusion problem: when a community disappears a community-based routing technique must adapt accordingly. Finally, *Merge* and *split* events are important because, in the former case, nodes that belong to different communities converge in the same community, so they should merge the available information, both social and routing, and probably a few replicas of data can be dropped. In the latter case, splitted communities suggest that communities may become more distant over time, so the content may need to be redistributed and replicated over the newly formed communities.

5 Our Dynamic Community Study

A real interest in studying the dynamic community in distributed environments is to understand how the network changes and in particular, after defining what we intend as community, how the community evolves during the time.

As explained in section 2.2.1, the current dynamic community detection approaches are three: the Instant-optimal Community Detection, the Temporal Trade-off Community Detection, and the Cross-Time Community Detection. Each of them have some peculiarities, both in the way communities are discovered and the result

itself. This fact, ultimately pose the systems developers a big question: which is the best dynamic community detection method considering our scenario? We already know that there is no approach that is clearly and always better than the others, but rather they show better results according to their strengths. Before moving into the study of the different Dynamic Community Discovery approaches, we make some considerations on the feasibility of the application of this methods in our scenario. Both the Instant-optimal and the Temporal Trade-off approaches are applicable because communities are evaluated considering only the current state of the network in the first case, and considering also past information in the latter case. However, the Cross-time Community Discovery approach can not be useful in this kind of environment because, since communities are used within the functioning of the system, it must be possible to discover them while the system lives. This fact makes the third approach unusable because it also requires all the future possible information which is not available. For this reason, in the following sections, we present the two approaches we used to study the dynamic nature of communities.

In first place we present the instant-optimal study, which is a common way to study dynamic communities due to the intuitive reasoning behind it. Then we move to a temporal trade-off one, which is still quite easy to understand, but very closer to a real-world distributed application.

In the Instant-optimal study we analysed the community evolution at periodic time instants. In particular we extracted a Network Snapshots representation of the network from our dataset, then we extracted communities for each of this snapshots using a static community discovery algorithm taken from literature. This process yields a set of communities for each of the snapshot. Finally, to evaluate the dynamism of the community structure, we need to match this communities using some metric. After choosing such a metric from the literature, we redefined the community events relevant to us, according to our scenario.

Considering the fact that Community Discovery is a complex task, we believe that re-evaluating communities from scratch whenever they are needed may be a waste of time and computing power. Therefore in the Temporal Trade-off study we analysed the network with an approach in which communities are updated, rather than computed from scratch, every time a new entity joins the network. In this case, we only had to choose an algorithm in the Temporal Trade-off approach capable of handling these updates. There is, in principle, no need to redefine the events describing a dynamic community life cycle, and therefore no need to define a similarity metric to match communities found at different time slots. This is because the Dynamic Community Discovery algorithm itself is able to record whenever one of the main event happens.

5.1 The instant-optimal approach

Given its simplicity, the first technique to analyze our dataset was an Instant-Optimal Community Discovery technique. In our approach, also due to the fact that our dataset is represented with a Temporal Networks model, we decided to periodically extract communities and match them at a second time.

We use DEMON[7] to discover static community for two main reason: a definition of community similar to the one adopted in this paper and the theoretical linear, with respect to the number of nodes, time complexity. In the follow, we describe more in detail how DEMON works.

5.1.1 DEMON

DEMON, acronym for Democratic Estimation of the Modular Organization of a Network, is a static community discovery algorithm that falls in the Model-based approach. The novelty of the algorithm is the fact that communities are discovered in a "democratic" fashion: each node of the network proposes a set of communities based on its local view. Then these local communities are merged together to build the global communities.

The algorithm uses two useful concepts: the Ego Network, the same we presented in section 4, and the Ego Minus Ego. The ego minus ego of a node u can be obtained from its ego network by simply removing node u itself and all its incident edges. The ego minus ego of a user u is therefore the set of its alters with the interactions between them.

The algorithm proceeds in two steps: firstly with the extraction of local communities, lastly with the merging of local communities up to the global communities.

To extract local communities, the algorithm computes the ego minus ego for each node and, on the obtained smaller networks, it performs a community discovery algorithm. To discover communities in the ego minus ego network, the authors chose the Label propagation algorithm [**?**]. In the Label propagation algorithm, each node is given a label which represent a community membership. Labels are propagated iteratively, until convergence is met. There is convergence when nodes do not change labels after a propagation. The steps of the algorithm are shown in algorithm 1. DEMON discovers local communities, in the sense that these communities are the ones proposed by each

node, based only on the node's view of the network. These communities based on local views, are then to be merged in order to build global communities.

Algorithm 1 Label propagation steps

1: Assign to each node in the network a different label.
2: Set $t = 1$.
3: Shuffle the set of nodes V.
4: For each node $v \in V$, update the label of v with the most frequent label among its neighbours. Ties are broken uniformly at random.
5: If labels did not change from last iteration, or t reached a maximum established value, stop the algorithm. Otherwise set $t = t + 1$ and go to step 3.

5.1.2 The application of the algorithm

We represent an ego network through time e as a set of n snapshots $(EG_1^e, EG_2^e, ..., EG_n^e)$. Each of this snapshot represents the state of the ego network at the corresponding instant of time, i.e. considering only online nodes. At each snapshot of an ego network e at time i, identified as EG_i^e, is associated a set of communities $C = (C_i^1, C_i^2, ..., C_i^m)$, which represents the community structure present at time i. In this paper a community is identified with nodes that are densely linked to each other, directly or through other nodes. We are interested in evaluating the evolution of communities in term of the community change events explained in detail in [30]. For sake of readiness, communities events are merge, split, death, and birth.

Communities are extracted at each time snapshot independently using DEMON, then communities belonging to adjacent snapshots are matched according to a similarity metric. To evaluate the similarity between communities, we use a revised version of the similarity metric proposed in [30]. In detail, consider an ego network e and two snapshot EG_i^e and EG_j^e, the revised similarity metric is introduced by the Eq. (1),

$$sim(C_{i-1}^p, C_i^q) = \frac{|V_{i-1}^p \cap V_i^q|}{max(|V_{i-1}^p|, |V_i^q|)} \quad (1)$$

where C_i^q is the community q included in EG_i^e and C_{i-1}^p is the community p included in EG_{i-1}^e. Instead, V_{i-1}^p is the set of nodes contained in C_{i-1}^p and V_i^q is the set of nodes contained in C_i^q.

Thanks to this similarity metric, each community in a time instant i is compared with each community of the time instant $i - 1$.

Moreover, we need to redefine all the possible community change events (merge, split, death, birth) to be applied in a DOSN according to the proposed similarity metric. We propose our definition of the four events:

- *Birth*: we say that a community C_i^p is born at time i if, given the set of communities $C_{i-1}^* = \{C_{i-1}^1, C_{i-1}^2, \cdots, C_{i-1}^k\}$ at time $i-1$, $\forall C_{i-1}^j \in C_{i-1}^*$, we have that $sim(C_i^p, C_{i-1}^j) = 0$. This means that all the communities discovered at the previous time instant $(i-1)$ do not share any node with C_i^p.
- *Death*: we say that a community C_{i-1}^p is dead at time i if, given the set of communities $C_i^* = \{C_i^1, C_i^2, \ldots, C_i^k\}$ at time i, $\forall C_i^j \in C_i^*$, we have that $sim(C_{i-1}^p, C_i^j) = 0$. This means that all the communities discovered at time i do not share any node with C_{i-1}^p.
- *Merge*: we say that a set of communities $C_{i-1}^* = \{C_{i-1}^1, C_{i-1}^2, \ldots, C_{i-1}^k\}$ merge into a community C_i^p if, for each community $C_{i-1}^j \in C_{i-1}^*$, we have that $sim(C_{i-1}^j, C_i^p) \geqslant k$, where k is the similarity threshold defined in [30]. This means that k% of mutual friends between C_i^p and each community in C_{i-1}^* are included in C_i^p.
- *Split*: we say that a community C_{i-1}^p splits into a set of communities $C_i^* = \{C_i^1, C_i^2, \ldots, C_i^n\}$ if, for each community $C_i^j \in C_i^*$, we have that $sim(C_i^j, C_{i-1}^p) \geqslant k$ where k is the similarity threshold as described in [30]. This means that a community C_i^p is divided in a set of community identified by C_{i-1}^*.

5.2 The Temporal trade-off approach

The choice of the algorithm to use for this analysis is very important, because we want to have a view of the community structure which is the closest possible to a real world case. Therefore, the chosen algorithm should be light and quick in updating communities. These two properties are highly desirable in environments where entities have low computational power or highly dynamic such as sensor networks or mobile networks. Among the many present in literature, belonging the temporal trade-off class, we chose TILES [27]. The choice was driven mainly by the logic behind the algorithm, which can be roughly summarized as: each time there is a perturbation on the network, update communities locally with respect to the perturbation. In the follow, we propose an overview of the algorithm.

5.2.1 TILES: an online algorithm for dynamic communities

TILES falls in the Temporal Trade-off CD because communities existing at a certain moment in time t depend on the current state of the network and the existing communities up to time t. The authors suppose the presence of an interaction streaming source. Each

time a new interaction is produced or expired, the graph is modified and the memberships into communities of the surroundings of the two endpoints of the edge are reevaluated. Nodes inside a community can have two different roles: core and peripheral. A node involved in at least one triangle with other nodes in the same community is a core node for that community. Instead peripheral nodes are neighbours of core nodes but they are not core nodes themselves. Clearly, the algorithm can output overlapping communities as nothing forbids a node of being part of two or more community cores. The algorithm outputs a chronologically ordered sequence of sets of communities, each of which represents the partition of the network at the end of each interval of duration τ. The pseudo code of the algorithm is shown in Algorithm 2. A queue is used to store the edges so that the expired ones are always the first ones to be popped.

Algorithm 2 Tiles

```
1: Get new edge from source
2: Put the new edge in a priority queue
3: Remove expired edges and update communities
4: Add the new edge to the graph and update communities
```

By considering the Algorithm 2, when an edge (u, v) is added to the network we may have 4 distinct cases:

1. both u and v appear in the network for the first time. In this case, no action is performed because there is no new triangle;
2. one node is in a peripheral community and the other appears for the first time or is in a peripheral community as well. Again, no action is performed because nodes in peripheral communities do not propagate community memberships unless it exists a node w such that u, v and w form a triangle. In this second case a new community should be created;
3. one node is a core node for a community and the other appears for the first time. The appearing node inherits the peripheral community membership of the other node;
4. both u and v are core nodes for two different communities. Here we have two different possibilities:
   (a) u and v have no common neighbours, the two nodes propagate each other the peripheral community membership;
   (b) u and v have common neighbours, communities memberships are reevaluated and, possibly, changes are propagated.

When an edge (u, v) is removed from the network, communities shared by u and v, the roles of u and v and their neighborhood must be reevaluated. Two are the scenarios that can happen:

1. The original community is still made of one component, only reevaluate the roles at close range with respect to the removed edge;
2. The original community splits into two or more communities: each of the new communities is considered as a new community and all the roles are reevaluated.

In the reevaluation phase, the clustering coefficient of each node within the specific community is computed and the node remains core if the clustering coefficient is > 0, while they turn peripheral nodes if the clustering coefficient is equal to 0.

5.2.2 The application of the algorithm

The algorithm returns a set of communities for each of the requested observations. Thanks to the nature of the algorithm, this communities do not need to be successively matched using some metric as in the Instant-optimal analysis. In fact the evolution of the communities, their events, is recorded as the algorithm runs. This trait of the algorithm lifts us also from the need of defining a similarity metric and the need of giving new definitions of the community events, as the events are automatically detected by the algorithm itself.

5.3 Comparison between the two approaches

We now briefly underline the positive and negative traits of both of the presented approaches.

The first presented approach is the instant-optimal approach, which is very intuitive and pretty straightforward. The only requirement to use this approach is to choose a community discovery algorithm, to discover communities at each time instant, and to choose a similarity metric, to match the identified communities. Both the problems, static community detection and set similarity, are well studied problems, so in literature we can find a very large amount of already implemented and tested approach to choose from. Another important fact to consider is that each timeslot can be processed independently. Thus, this means that the discovery of communities can be naturally made parallel. This is the same for the matching phase: each matching can be performed on its own, when the communities from both timeslots has been discovered. Another good point of using this approach is that communities are discovered from scratch at each time slot. which prevents the phenomenon of drifting. This phenomenon happens when communities at a time instant t are computed by

updating the ones discovered at earlier time instants to approximate the ones at time t. Depending on how communities are updated, it may happen that the updates lead to different communities with respect to the ones present. Generally speaking, in case we have a drifting effect, we observe a huge number of tiny communities which are actually fragments of the real community.

The instant-optimal analysis can be a really powerful data mining tool, but when it comes to distributed environments it may be not the best choice, mainly because it requires the exact knowledge of the status of the network at a given time. This translates into having synchronization, or strong consistency, which is impossible in truly distributed environments (see FLP theorem [12]). Another important fact is that communities discovered with the instant-optimal analysis do not take into account time. Instead, in our Temporal Trade-off community detection technique, as we saw in section 5.2, communities at a given time instant t are influenced by both current and past information. Taking into account time may potentially lead to some effects, such as avalanche effect, large community drifting with respect to the static case and high instability of the process. While this is true, it does not mean that they are negative or undesirable effects. It is up to the analyst to choose whether to give great importance to time or not. Clearly, if we are interested in finding the optimal partition at a given time instant, without giving any relevance to time, those are all negative effects that have to be mitigated somehow. Instead, if we give a special meaning to the time at which entities or nodes join or leave the network, then those are desirable effects. The drifting of a community may translate into a finer grained definition of community which is also based on time information. In addition we cannot underestimate the fact that in our Temporal Trade-off approach there is no matching phase between communities belonging to adjacent timeslots. In fact, trying to match communities that are different each other can be a waste of time, while matching two sets of highly overlapping communities can lead to guessing the correct matching. Temporal Trade-off eliminates these two potentially dangerous problems by keeping track of each community throughout time.

6 A case study: Facebook

To evaluate the dynamics in real OSNs, we retrieved a real dataset, gathered by a Facebook application, called SocialCircles![2].

In this section we present the dataset we used for our experiments, and we also present SocialCircles![3], the Facebook application that allowed us to retrieve all the needed data for this analysis.

6.1 The Facebook application SocialCircles!

SocialCircles! was a Facebook application that showed to registered people interesting facts about their Facebook ego network. The application was deployed in 2014 and has gone under maintenance on the 1st of May 2015 due to the change of the Facebook APIs which were substantially reduced in size.

As described in [10], SocialCircles! was able to retrieve the following sets of information from registered users:

Topology and profile information For each registered users, we obtained its friends and the friendship relationships existing between them or, in other words, the ego network of the registered users. We were also able to retrieve the profile informations about the registered users.

Interactions By analyzing interactions, such as *posts*, *comments*, *likes*, *tags*, and *photos*, between users registered to the application and their friends, we could estimate the strength of their interactions. By aggregating all this informations, it is also possible to weight the links connecting each pair of users and study the associated graph.

Online presence It was not trivial to collect temporal information since the Facebook API did not permit it directly, even while the application was online. The online presence of users was approximated by monitoring the chat status of registered users in Facebook periodically. Each time the chat was monitored, a user can be flagged with 0 if he is offline, 1 if he is in the active state and 2 if he is idle (online, but no action performed in the last few minutes).

During its life, the application was able to build two datasets: the first one containing more than 300 hundred ego networks and 15 days of temporal session of users, instead the second one contains 240 ego networks and 32 consecutive day of temporal information. For our analysis we use the second one which is also the most recent one. In details. our dataset contains 240 users monitored and their complete ego networks (for a total of 78.129 users). For each of the registered users we were able to gather their profile and ego network, and

[2] `https://www.facebook.com/SocialCircles-244719909045196/`

[3] `https://www.facebook.com/SocialCircles-244719909045196/`

the interactions between them and the alters. Moreover, we also obtained temporal information about the total 78.129 users for 32 consecutive days, by sampling all the registered users and their friends every 5 minutes, for 32 days (from the 9^{th} March to the 10^{th} April 2015).

Since the time aspect is the cornerstone of our research, we preliminary analyzed the temporal information contained in our dataset. This preliminary analysis is aimed to understand a general trend of online/offline behaviour of the users on a reference OSN.

We start by recalling that, in our dataset, time is modeled in a discrete way to represent the online/offline status of the users. In particular, each day of the monitored period consists of a finite number of time slots (i.e., 288 time slots each of 5 minutes), for a total number of 9251 time slots in the whole monitored period. This choice of the time granularity was driven by the fact that the machine storing the data was not always able to cope with finer granularities. In fact, for smaller choice of granularities, it sometimes happened that a new set of requests were issued while the reply for old ones were not yet stored on the application database. For the sake of our analysis, we do not make any distinction if the user is in the active status (status=1) or in the idle status (status=2). This decision was taken because if a user is online, active or idle, his device can still help the P2P network in delivering the service. Thanks to these facts we were able to build a temporal track for each user: an array of 9251 positions of boolean values. For each of the 9251 positions, the value of this array is set to 1 (or true) only if the corresponding user is found active or idle at the time slot with id equals to i, it is set to 0 (or false) otherwise. With the temporal tracks, we had the possibility to lead some experiments regarding the online/offline behaviour of all the users on the OSN.

Figure 3 shows the number of online users for each time slot. The figure shows that there is a clear periodic pattern, probably reflecting the day/night cycle. By analyzing the amount of users online for each time slot, we can see that we have at most around 18000 online users, roughly 23% of the total amount, and at least 3000, 3.8% of the total amount of users.

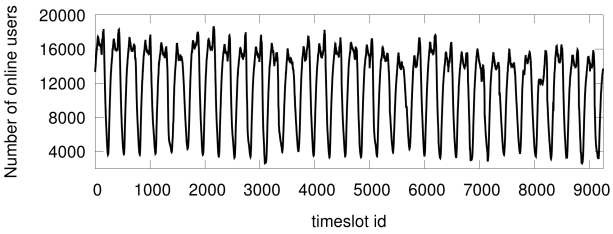

Fig. 3 Online users count during the observed period

	Min	Max	Mean	Std. Deviation
Number	1	26	9.49583333	4.401405174
Size	4	1894	99.3878894	141.2894853

Table 1 Statistical measures on number and size of static communities

6.2 Preliminary community evaluation

An important aspect of our research is the topology of the social network. As we saw, it is very important to exploit it at our advantage, so, understanding it, is a primary concern. The main information we wanted to mine from the dataset, is the possible presence of the community structure. To this aim, we extracted the communities by executing a community discovery algorithm on the social graph.

We start by recalling that the social graph is a graph where the nodes are the users and the edges represent, generally speaking, some sort of relationship between users. Thanks to SocialCircles!, we had access to friendship relationships of each registered user. In detail, we got to know, for each registered user, the friends of the registered user, and the friendship relationships between pairs of users sharing at least a common ego. Summing it up, we had access to all friendship relationships inside each registered user's ego network.

Since the presence of the ego brings a lot of noise to the results due to the fact that it drastically reduces distances and makes the network clusterized, communities are, in general, not extracted from the ego network itself. For this goal, the ego-minus-ego subgraph is used. After the extraction of the ego-minus-ego graphs for each registered used, we proceeded in analyzing the community structure in this small, yet significant views.

As a preliminary analysis, we computed some statistical measures on the number and size of the dynamic communities to compare them with the static communities to demonstrate the need of a dynamic analysis to model the real life. Table 1 reports the computed statistical measures for the static communities. This table show us that the static view of the network has a strong structure with respect to the static communities. In particular, we see that, on average, each ego network has 10 communities, each made of 100 nodes.

6.3 Dynamic community discovery introduction

Considering the scope of this study, we decided to define two analysis frameworks: a periodical one (see section 5.1) and an evolutionary one (see section 5.2). The aim

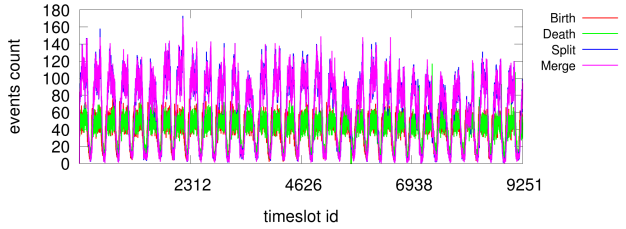

Fig. 4 Community events for each time slot of all dynamic communities

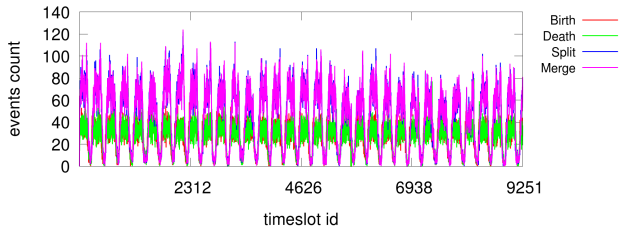

Fig. 5 Community events for each time slot of selected dynamic communities

of this work is to show that different community discovery algorithms lead to different results, possibly quite distant each other. Indeed, while the two kinds of analyses aim to the common result of discovering communities, they are conceived to run in different situations. The periodic one is a more canonical kind of analysis, extremely useful for an a posteriori study of the network. Since static community discovery algorithms are used for the dynamic periodic community discovery, it enables a very meaningful comparison between static and dynamic communities. Not requiring ad hoc algorithms also incentive this kind of analysis. Instead, an evolutionary analysis, while at first glance may seem hard to understand, it is more natural when communities are needed on the fly. It also enables a more efficient way to discover communities since they can be updated rather than being recomputed from scratch.

We computed the community events as described in section 3 considering two different sets of communities:

- All: in this case we considered all the communities of all ego networks during the observed period of time of 32 days;
- Selected: consider only the communities in the time slots where the related ego was offline (inter-arrival session slots).

With this differentiation we aim to capture a generic, global view of the dynamism of the network in the first case, and a more specific, critical view in the second case. It is very important to understand how the network evolves in time.

6.4 Instant-optimal dynamic community discovery results

To carry out the periodical analysis, the first task to be accomplished was how to inflate topological information with the temporal information in our possession. Since for this first analysis we needed a series of Network Snapshots, the most intuitive way was to use the temporal tracks as obtained in section 6.1. In detail, for each registered user, starting from the ego-minus-ego defined in section 6.2 we extracted a snapshot for each of the 9251 snapshots. The i^{th} snapshot is built by removing from the ego-minus-ego all the nodes, and the incident edges, which are found offline (i^{th} position of the temporal track equal to 0). This way, each snapshot is made only of the online users and the relationships between them.

Communities are then discovered on each snapshot of each registered user using DEMON. As last step, to discover the events occurring, we tried to match communities extracted from a dynamic ego network with the ones extracted from the dynamic ego network of the previous time instant. This choice is driven by the fact that, if we want to use communities in our system, it is important to know the moments a community is available or not. This way, if a community disappears, even for just one time slot, the event is recorded. The similarity metric used for this matching is the one presented in 5.1.2. The similarity threshold to detect a merge or a split event is set to 0. For merge events, as additional constraint, we also want that, for each of the source communities, the destination community is the one with the highest similarity compared to the other ones in the same time slot. For split events we have a dual constraint: for each of the destination communities, the source community is the one with the highest similarity compared to the other ones in the same time slot. This additional constraints are used to give value to the matching of two communities only if the similarity is the highest recorded.

The very first step was to have a general idea of the community structure. Table 2 reports some statistical measures of all dynamic communities. Just by analyzing these results, we can say that the network is, as expected, very shattered and not even close to the static view. When considering the number of communities, the high value of standard deviation with respect to the average, suggests that in some particular time slots some ego networks have no community at all. We see that in the static case, table 1 we have a lower maximum value and a higher average with respect to the dynamic case, which suggests that it is very unlikely to have a dynamic ego network that is similar to the

	Min	Max	Mean	Std. Deviation
Number	0	104	2.28143443952	3.75809047
Size	4	452	17.6435637388	22.10944505

Table 2 Statistical measures on number and size of all dynamic communities

static one. Also the size statistics confirms this fact: static communities tend to be larger than the dynamic ones. We can explain the difference in the two results by recalling the fact that we have at most less than a forth of the users online, as reported in figure 3.

To better understand how the events are arranged during the observed time, we decided to make some plots. Figure 4 shows the arrangements of the events when considering all communities of all time slots while Figure 5 shows the events for the selected communities. Both the figures show that there is a temporal pattern in the results, suggesting that the behaviour follows a daily cycle, confirming the results in figure 3. Moreover, on the peaks, the number of merge/split events are roughly double the number of death/birth events, while in the nadirs the number of merge/split events are slightly less than the number of death/birth events. By taking a closer look at the arrangements of the events, we may also observe that peaks and nadirs of merge and split events are slightly moved on the right with respect to the ones of birth and death events, which means that, before observing a variation on the number of split and merge events, we should see a variation in the number of birth and death events. It is also worth noticing that, as expected, at each drop of the events corresponds a peak in deaths, which probably means that we are approaching the night time slots. Dually, at each increase of events, we usually see a peak of birth events, which should corresponds to the time slots where people wake up. Another important result is that the two graphs look similar which is sign that the network behaves in the same way both when the ego is online or offline. This is of interest in the sense that all the analysis can be done regardless that an ego is online or not.

Since the events follow a daily cycle, we are interested to see how this events are related to the presence of users on the network. From a comparison between figures 4 and 5 with figure 3 we can see that the more users are online, the more events are observed in the network. This could be very useful to manage the problem of data availability. Indeed, it means that, in a community-based replication technique, choosing the replicas when there are less users on the network is somewhat easier because the network is more stable in terms of communities, while, on the other hand, when there are a lot of users online, we need to handle more community events, especially split and merge events.

Finally, to get a more generic trend of the dynamism of the communities, we analysed how much communities belonging to the same ego network discovered in adjacent time slots are similar to each other. To measure the similarity between this pairs of sets of communities, we choose a similarity metric taken from literature: F1-score.

F1-score is presented as a metric to evaluate the set of communities X obtained as result of a community discovery algorithm based on a given set Y of ground truth communities which is based on the concepts of precision and recall. Precision and recall are two metrics used to measure the similarity between two sets of items and they are defined as follows: given two sets of labeled nodes, $x \in X$ representing an identified community and $y \in Y$ representing a ground truth community

- The **Precision** is defined as the percentage of nodes in x which have labels that are also present in y. In formula: $P = \frac{|x \cap y|}{|x|}$
- The **Recall** is instead defined as the percentage of nodes in y which have labels that are also present in x. In formula: $P = \frac{|x \cap y|}{|y|}$

To evaluate the similarity between the two sets of communities, each of the identified communities x is matched to a single ground truth community y based on the number of common nodes in the two communities. This matching procedure creates pairs of communities (x, y), with $x \in X$ and $y \in Y$, which will be used to evaluate the similarity. The quality of each pair is evaluated separately using the F1-measure which is obtained by combining precision and recall as follows: F1-measure$=\frac{2 \times precision \times recall}{precision + recall}$.

Once we assign a quality for each pair, the F1-score is simply defined as the average of the F1-measure of the identified pairs.

In our study we are not evaluating the results of a novel community discovery algorithm, but we are interested in evaluating the dynamism of the network at a community structure level. For this purpose, we evaluate the similarity of the communities identified at each time step t with the ones, considered as ground, identified at the previous time step $t-1$. Computing the F1-score between adjacent, in time, sets of communities should give us an insight on how much communities affectively change over time in term of the nodes present in each community.

Figures 6 and 7 show the 25^{th}, 50^{th}, and 75^{th} percentiles of F1-score obtained respectively for all communities and selected communities. At a first glance we can observe that figure 6, the one which considers all

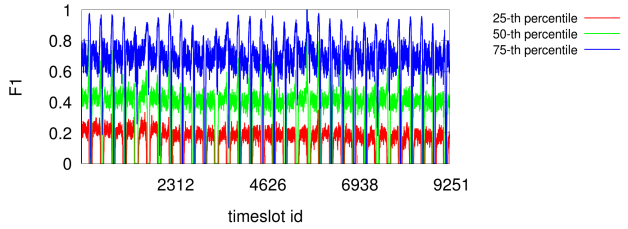

Fig. 6 F1-score for each time slot of all dynamic communities

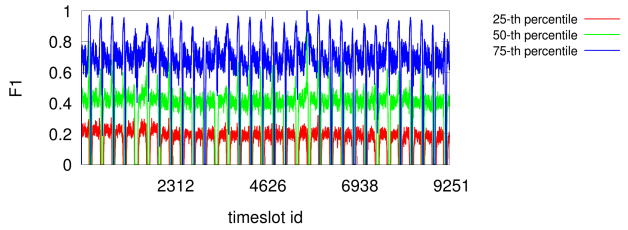

Fig. 7 F1-score for each time slot of selected dynamic communities

communities, shows values a little more stable. Both the figures again show a clear day/night cyclic pattern with peaks which correspond to evening hours and nadirs immediately after, during night. During the peaks we observe a very high amount of similarity reaching 0.6 on average and over 0.9 for the 75^{th} percentile, meaning the the network is very stable in terms of communities. The fact is that these peaks are very short and only last for few timeslots, corresponding to less than two hours. Nadirs are also quite short in time lasting no more than five hours, and often it happens that also the 75^{th} percentile reaches 0. This means that during nighttime the community structure is mostly absent, so it can't be actively used. Finally, during the other hours of the day, we can see that the similarity is quite low, in fact we observe that on average the F1-score is just above 0.4.

6.5 Temporal Trade-off dynamic community discovery results

At this point, we switched our interest in algorithms capable of updating communities on the fly. There are plenty of different community discovery algorithms in literature, each with strengths and weaknesses. We only had two requirements for the algorithm to use for this analysis. Obviously, we only took in consideration algorithms with a definition of community quite similar to the one given by DEMON. Since definitions of community may vary a lot from algorithm to algorithm, the constraint was necessary to get results comparable to the one obtained for the other analysis. The other constraint was related to the nature of the system we plan to use the result of the algorithm. Since we want to know and use the community structure for important functionality of distributed systems, we had to choose an algorithm which can be possibly implemented in a P2P fashion. Considering these constraints, the choice fell on TILES.

To be used, TILES requires that the network is represented with the Temporal Network model. While it was possible to exactly represent our network according to the Network Snapshot model, due to the fact that the temporal information in our possession was already discretized, it wasn't possible for the needed model. The problem is that we had no information about the actual order in which nodes joined or left the OSN. So, to obtain a Temporal Network representation of our dataset we had to make some modifications to our dataset. TILES, the algorithm choice for the analyses, requires an edge streaming source but the temporal information in our possession is node based. To realize the transformation we proceeded in the following way: if a node maintains its state from a time slot to the following one, no action is performed. Otherwise, if a node joins the network, a new edge addition is added to the interaction source for each of its online friends. The exact order in which these interactions, created when a node switches its state from offline to online, are added to the source is completely random, but their time label is the same. Having the same time label lets us, when extracting the results, to always consider communities where all the interactions from the same time slot have been processed. Dually, when a node leaves the network, switching its state from online to offline, a new edge deletion is added to the interaction source for each of its online friends. Offline neighbours are ignored because the interaction with them have already been deleted when they went offline. Again, the order in which this set of interactions is added to the source is random, but they all have the same time label. The time label of each interaction (addition or deletion) is the id of the current time slot scanned. For the sake of clarity, after analyzing all the interactions with id $\leq x$, we have the same network of the $x - th$ snapshot.

Once obtained the complete list of ordered and time labeled interactions, we were able to extract them using TILES. Since we have rather meaningful temporal information about both the joining and the leaving of users, we decided to use the version of the algorithm with explicit removal of interactions. In the explicit version of the algorithm, edges are deleted only when the source emits a specific interaction deletion. In the vanilla version of the algorithm, it was possible to set a time to leave *ttl* as input parameter. An interaction

Measure	Min	Max	Mean	Std. Deviation
Number	0	74	4.041437412	4.1011431095
Size	3	336	7.609	9.55

Table 3 Statistical measures on number and size of dynamic communities

is deleted after *ttl* amount of time after its insertion. The observation threshold was set to 1, so that communities are extracted once for each time slot and the results can be compared to the ones obtained from the previous analysis technique. Finally, we recall that, in the Temporal Trade-off Community Detection analyses, community events are detected as communities are reevaluated. Thus, there is no need to define any similarity metric, nor to match community anyhow to detect community events.

In a preliminary study, we simply observed the size and the number of dynamic communities. Table 3 shows some statistical values on number and size of communities as returned by TILES. In particular, we see again that the minimum number of communities is 0, meaning that there is at least one timeslot during which there are no communities at all. A lower maximum value together with an higher mean value suggest us that the community structure is even more shattered with respect to the previous study. In particular, we see that communities tend to be smaller, with less than half nodes, but almost double in number.

Figure 8 shows the registered events considering all communities of all timeslots, while figure 9 only consider the selected communities. The two figures seem quite similar, the main difference between the two is that the number merge and split events do not change substantially, while death and birth event counts are lower by 200. If we compare these figures with the ones produced by the previous analysis (figure 4 and figure 5) we see great changes. First of all, while the death count is again comparable to the birth count, merge and split counts are quite different each other. Split events have become quite rare, in fact only few of these events, during all observation, are detected. In turn, we observe some merge events which peaks are less emphasized with respect to the previous analyses. The second major change we observe is in the effective number of events. We recall that in the previous analyses we observed around 75 merge and split events plus 40 death and birth events. In these new analyses merge events are again around 75 each timeslot, we already said that split events are rare, but, unexpectedly, birth and death registered events are between 500 and 700 during the central part of the day. During the night, death and birth events drop below 100, while merges often reach 0. These opposing results can be explained in a great number of small communities, probably just triangles, assembling and disassembling at a quick pace. The relatively small amount of communities registered during the simulation also suggest us that all these communities hardly live long enough to be registered at the end of the timeslot. This effect can be considered as interference as these communities has a lifetime so short that can't be used anyhow. The only property remained almost unchanged is the periodic temporal pattern.

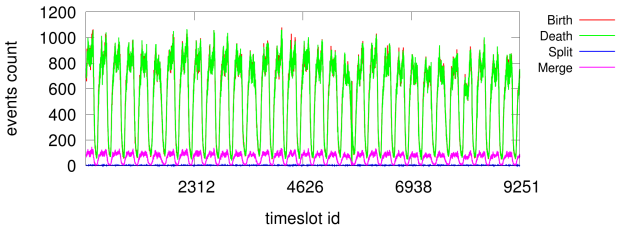

Fig. 8 Community events for each time slot of all dynamic communities

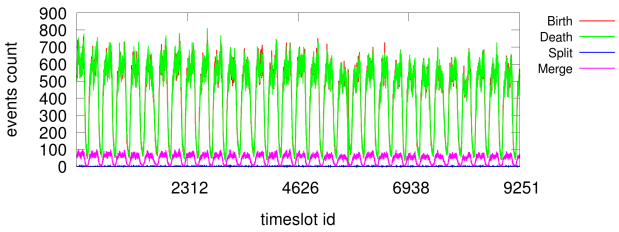

Fig. 9 Community events for each time slot of the selected dynamic communities

Then, once again, we evaluated the overall stability of the community structure with F1-score. Figures **??** and **??** show the 25^{th}, 50^{th}, and 75^{th} percentiles of F1-score obtained respectively for all communities and selected communities. The daily pattern guided by the succession of day and night is still present and follows the same pattern as the previous analyses: high similarity values during the day, a considerable peak in the late evening, then a nadir in the night. We can also observe that, overall, the similarity is significantly increased, in fact the 25^{th} percentile is now around 0.4, while the 50^{th} percentile is just below 0.6 during the day, which corresponds to an increase of 0.2 on the F1-score value. We also observe an increase of about 0.1 to the F1-score of the 75^{th} percentile. While we observe a considerable increase in the similarity metric of these dynamic communities, with respect to the ones discovered in the previous analysis, we still have to say that

the community structure is pretty unstable. If we consider the events results combined with this similarity results, we suppose that with further investigation we can find a subset of these communities which are core communities, namely communities which are resilient to churn and have a lifespan much longer than the typical one.

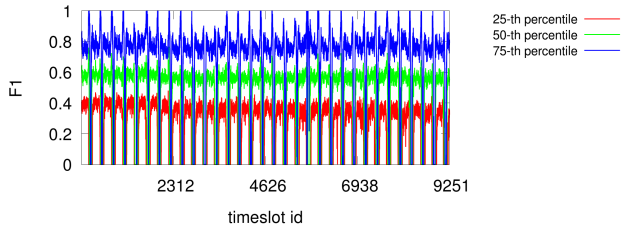

Fig. 10 F1-score for each time slot of all dynamic communities

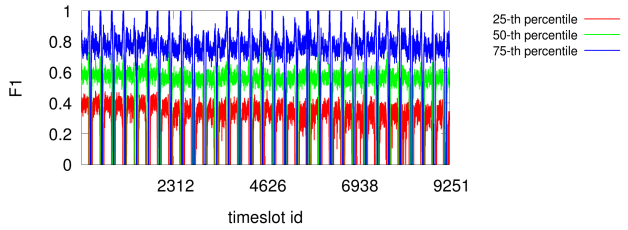

Fig. 11 F1-score for each time slot of selected dynamic communities

7 Conclusion and Future works

In this paper we faced the application of dynamic community discovery to manage DOSNs challenges. The paper is led by the two research questions introduced in section 1. Considering the first research questionwe firstly propose a static analysis of communities in DOSNs to have a general view of the static case. Results of our analysis showed that dynamic community discovery is needed in a distributed environment, such as a DOSN, because the static community discovery does not provide a real view of the changes happened in the social graph. Indeed we observe that the network is very shattered, not even close to the static view with respect to both presented frameworks. This difference clearly suggests us the need a distributed algorithm able to manage the dynamism of communities. As a consequence, we investigated the second research question. We focus our attention on two kind of dynamic community discovery approaches: the Instant-optimal Community Detection and the Temporal Trade-off Community Detection, proposing an analysis framework for each approach. Moreover, we also proposed a set of community change events which are important in our scenario, described in this paper. We analysed the arrangement and the frequency of these events by exploiting a real Facebook dataset gathered by our Facebook application (SocialCircles). The community change events introduced in this paper have a temporal pattern that is similar to the temporal user behaviour. We analysed the differences between the two proposed approaches by considering both the community change events and the similarity between the communities expressed by the F1-score. As concerns the second research question, we analysed in detail the pros and cons of the two approaches mentioned above, coming to the conclusion that the Temporal Trade-off Community Discovery represents the best choice in DOSNs. In the future, we plan a deep analysis of the instability of the social graph due to the online/offline status of users. In particular, we plan to develop a distributed algorithm to detect the dynamic community by exploiting the Temporal Trade-off Community Discovery approach, which can be used in DOSNs to manage several problems, such as data availability, information diffusion, and privacy.

References

1. Thomas Aynaud, Eric Fleury, Jean-Loup Guillaume, and Qinna Wang. Communities in evolving networks: definitions, detection, and analysis techniques. In *Dynamics On and Of Complex Networks, Volume 2*, pages 159–200. Springer, 2013.
2. S. Buchegger, D. Schioberg, L.H. Vu, and A. Datta. Implementing a P2P Social Network - Early Experiences and Insights from PeerSoN. In *Second ACM Workshop on Social Network Systems (Co-located with EuroSys 2009)*.
3. Rémy Cazabet and Frédéric Amblard. Dynamic community detection. In *Encyclopedia of Social Network Analysis and Mining*, pages 404–414. Springer, 2014.
4. Andrea E. F. Clementi, Miriam Di Ianni, Giorgio Gambosi, Emanuele Natale, and Riccardo Silvestri. Distributed community detection in dynamic graphs. *CoRR*, abs/1302.5607, 2013.
5. Marco Conti, Andrea De Salve, Barbara Guidi, and Laura Ricci. Epidemic diffusion of social updates in dunbar-based dosn. In *European Conference on Parallel Processing*, pages 311–322, 2014.
6. Michele Coscia, Fosca Giannotti, and Dino Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 4(5):512–546, 2011.
7. Michele Coscia, Giulio Rossetti, Fosca Giannotti, and Dino Pedreschi. Demon: a local-first discovery method for overlapping communities. In *SIGKDD international conference on knowledge discovery and data mining*, pages 615–623. IEEE ACM, 2012.

8. L. A. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *Comm. Mag.*, 47(12), December 2009.
9. Anwitaman Datta, Sonja Buchegger, Le-Hung Vu, Thorsten Strufe, and Krzysztof Rzadca. Decentralized online social networks. In *Handbook of Social Network Technologies and Applications*, pages 349–378. Springer, 2010.
10. Andrea De Salve, Marco Dondio, Barbara Guidi, and Laura Ricci. The impact of user's availability on on-line ego networks: a facebook analysis. *Computer Communications*, 73:211–218, 2016.
11. Andrea De Salve, Barbara Guidi, Paolo Mori, and Laura Ricci. Distributed coverage of ego networks in f2f online social networks. In *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences*, pages 423–431, 2016.
12. Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
13. K. Graffi, C. Gross, P. Mukherjee, A. Kovacevic, and R. Steinmetz. Lifesocial.kom: A p2p-based platform for secure online social networks. In *Peer-to-Peer Computing*, pages 1–2. IEEE, 2010.
14. Derek Greene, Donal Doyle, and Padraig Cunningham. Tracking the evolution of communities in dynamic social networks. In *Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '10, pages 176–183, 2010.
15. Barbara Guidi, Tobias Amft, Andrea De Salve, Kalman Graffi, and Laura Ricci. Didusonet: A p2p architecture for distributed dunbar-based social networks. *Peer-to-Peer Networking and Applications*, pages 1–18, 2015.
16. Barbara Guidi, Marco Conti, and Laura Ricci. P2p architectures for distributed online social networks. In *High Performance Computing and Simulation (HPCS), 2013 International Conference on*, pages 678–681. IEEE, 2013.
17. Barbara Guidi, Andrea Michienzi, and Giulio Rossetti. Dynamic community analysis in decentralized online social networks. In *International European Conference on Parallel and Distributed Computing (Euro-Par), LSDVE Workshop*, 2017.
18. G. J. Herbiet and P. Bouvry. Sharc: Community-based partitioning for mobile ad hoc networks using neighborhood similarity. In *2010 IEEE International Symposium on "A World of Wireless, Mobile and Multimedia Networks"*, pages 1–9, 2010.
19. Pan Hui, Eiko Yoneki, Shu Yan Chan, and Jon Crowcroft. Distributed community detection in delay tolerant networks. In *Proceedings of 2Nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture*, pages 1–8, 2007.
20. Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 177–187, New York, NY, USA, 2005. ACM.
21. Marsden, P. Egocentric and sociocentric measures of network centrality. *Social Networks*, 24(4):407–422, 2002.
22. Rammohan Narendula, Thanasis G Papaioannou, and Karl Aberer. My3: A highly-available p2p-based online social network. In *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, pages 166–167. IEEE, 2011.
23. Nilizadeh, S. and Jahid, S. and Mittal, P. and Borisov, N. and Kapadia, A. Cachet: a decentralized architecture for privacy preserving social networking with caching. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, CoNEXT '12, pages 337–348. ACM, 2012.
24. Gergely Palla, Albert-László Barabási, and Tamás Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, 2007.
25. Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007.
26. Giulio Rossetti and Rémy Cazabet. Community discovery in dynamic networks: a survey. Technical report, 2017.
27. Giulio Rossetti, Luca Pappalardo, Dino Pedreschi, and Fosca Giannotti. Tiles: an online algorithm for community discovery in dynamic social networks. *Machine Learning Journal*, 2016.
28. Andrea De Salve, Barbara Guidi, and Laura Ricci. Evaluation of structural and temporal properties of ego networks for data availability in dosns. *Mobile Networks and Applications*, pages 1–12, 2017.
29. Mansoureh Takaffoli, Reihaneh Rabbany, and Osmar R Zaïane. Community evolution prediction in dynamic social networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, pages 9–16, 2014.
30. Mansoureh Takaffoli, Farzad Sangi, Justin Fagnan, and Osmar R Zäiane. Community evolution mining in dynamic social networks. *Procedia-Social and Behavioral Sciences*, 22:49–58, 2011.
31. Mansoureh Takaffoli, Farzad Sangi, Justin Fagnan, and Osmar R Zaïane. Modec-modeling and detecting evolutions of communities. In *5th International Conference on Weblogs and Social Media (ICWSM)*, pages 30–41. AAAI, 2011.
32. Matthias Wiesmann, Fernando Pedone, André Schiper, Bettina Kemme, and Gustavo Alonso. Understanding replication in databases and distributed systems. In *Distributed Computing Systems, 2000. Proceedings. 20th International Conference on*, pages 464–474. IEEE, 2000.