Evolution of Communities with Focus on Stability

Carlos Sarraute and Gervasio Calderon

Grandata Labs
Buenos Aires, Argentina
{charles,gerva}@grandata.com

Abstract. Community detection is an important tool for analyzing the social graph of mobile phone users. The problem of finding communities in static graphs has been widely studied. However, since mobile social networks evolve over time, static graph algorithms are not sufficient. To be useful in practice (e.g. when used by a telecom analyst), the stability of the partitions becomes critical.
We tackle this particular use case in this paper: tracking evolution of communities in dynamic scenarios with focus on stability. We propose two modifications to a widely used static community detection algorithm: we introduce fixed nodes and preferential attachment to pre-existing communities. We then describe experiments to study the stability and quality of the resulting partitions on real-world social networks, represented by monthly call graphs for millions of subscribers.

1 Introduction

In recent years, there has been a growing interest in the scientific community and mobile phone operators to analyze the huge datasets generated by these operators, for scientific and commercial purposes (e.g. as demonstrated by the growth of NetMob attendance).

One of the important tools used to analyze the social graph of mobile phone users is community detection. Within each community, external nodes are susceptible of attraction to become customers and existing customers can be influenced to remain and/or to buy new products or services. This can be done by leveraging mutual influence of nodes [1]. User communities, as smaller units easier to manage, allow the computation of role analysis. The centrality of an actor within a community is the base for role analysis.

The problem of finding communities in static graphs has been widely studied (see [2] for a survey). However, for practical use cases, the detected communities must evolve matching the underlying social graph of communications (for example, to track ongoing marketing campaigns aimed at specific communities). Also, behavior analysis of communities of users over time can be used to predict future activity that interests telecom operators, such as subscriber churn or handset adoption [3]. Similarly, group evolution can provide insights for designing strategies such as early warning of group churn.

The stability of communities is critical to preserve earlier analysis. This is the specific use case we tackle in this paper: tracking evolution of communities in dynamic scenarios with focus on stability.

The rest of the paper is organized as follows. In Section 2 we describe the data sources we used to perform our experiments. In Section 3 we propose two modifications to a widely used static community detection algorithm (the Louvain Method of Vincent Blondel and his team [4]). Then, in Section 4, we describe the experiments performed in order to study the stability and quality of the resulting partitions. These experiments were run on real-world social networks represented by monthly call graphs for millions of subscribers. Section 5 reviews related work, and Section 6 concludes the paper with ideas for future work.

2 Data Sources

Our raw data source is anonymized traffic information from a mobile operator. The analyzed information ranges from January 2012 to January 2013 (same time span as [1]). For each communication it contains the origin, target, date and time of the call or sms, and duration in the case of calls.

For each month T, we construct a social graph $\mathcal{G}_T = < \mathcal{N}_T, \mathcal{E}_T >$. This graph is based on the aggregation of traffic for several months, more concretely $\mathcal{G}_T$ depends on the traffic of three months: T, $T-1$ and $T-2$. The raw aggregation of calls and messages gives a first graph with around 92 M (million) nodes and 565 M edges (on a typical month). Voice communications contribute 413 M edges and text messages contribute 296 M edges to this graph.

Then, we perform a symmetrization of the graph keeping only the edges (A, B) when there are communications from A to B and from B to A. This new graph contains around 56 M nodes and 133 M (undirected) edges, representing stronger social interactions between nodes. Additionally we filter nodes with high degree (i.e. degree greater than 200) since we are interested in the communications between people (and not call centers or platform numbers).

3 Dynamic Louvain Method

Our first experiment to detect evolving communities was to run the original Louvain algorithm [4] on the graphs at time T and $T+1$ and compare the two partitions. This method produced very unstable results. Our second experiment was to run the Louvain algorithm with the modifications by Aynaud and Guillaume [5] (denoted LMAG hereafter) to obtain a more stable evolution.

This is our implementation of the initial step of LMAG. At time $T+1$, the nodes already present at time T are initially assigned to the community they belonged to at time T. New nodes – not present at time T – are assigned to fresh communities (see Algorithm 1). As we show in Section 4, results were still unsuitable in terms of stability.

In our use case (e.g. telecom analysts performing actions on communities), the stability of the partition is our main concern (see stable or natural communities

Algorithm 1: Initial step of Louvain modified by Aynaud and Guillaume

Input: r (percentage of free nodes). $\mathcal{N}_{T+1}$ (nodes). $\mathcal{N}_T \rightarrow \mathrm{C}_T$ (old nodes to communities)
Output: Initial communities assigned to nodes

```
maxOldCommunity ← Max(C_T);
availableComm ← maxOldCommunity + 1;
foreach node x in N_{T+1} do
    oldComm ← FindInOldCommunities(x);
    if oldComm is not NULL and not IsRandomFreeNode(r) then
        initialComms [x] ← oldComm;
    else
        initialComms [x] ← availableComm;
        availableComm ← availableComm + 1;
return initialComms
```

in [2]). With this goal in mind, we propose two modifications to the Louvain method, that give the partition at the previous time step a sort of “momentum”, and make it more suitable to track communities in dynamic graphs.

Before describing them, we introduce some notations. As stated in the previous section, we consider snapshots of the social graph constructed at discrete time steps (in our case every month). Let $\mathcal{G}_T =< \mathcal{N}_T, \mathcal{E}_T >$ be a graph that has already been analyzed and partitioned in communities. Let $\Gamma =< C_1, \ldots, C_R >$ be such partition in R communities. Given a new graph $\mathcal{G}_{T+1} =< \mathcal{N}_{T+1}, \mathcal{E}_{T+1} >$ our objective is to find a partition of $\mathcal{G}_{T+1}$ which is stable respect to Γ.

The first idea is to have a set of *fixed nodes* $\mathcal{F}$. Let $\mathcal{R} = \mathcal{N}_T \cap \mathcal{N}_{T+1}$ be the set of nodes that remain from time T to $T+1$. The set $\mathcal{F}$ is a subset of $\mathcal{R}$, whose nodes are assigned to the community they had at time T. In other words, noting γ the function that assigns a community to each node, we require: $\gamma_{T+1}(x) = \gamma_T(x)\ \forall x \in \mathcal{F}$.

We experimented with different distributions of the fixed nodes, ranging from no fixed nodes ($\mathcal{F} = \emptyset$) to all the remaining nodes ($\mathcal{F} = \mathcal{R}$). For the experimental results, we used a parameter p representing the probability that a node belongs to $\mathcal{F}$ (i.e. $|\mathcal{F}| = p \cdot |\mathcal{R}|$).

The second idea is to add a probability q of “preferential attachment” to pre-existing communities. With probability q, new nodes will be more likely to attach to an existing community at time T instead of attaching to a community formed at time $T+1$. We give the details below.

The Louvain Method [4] is a hierarchical greedy algorithm composed of two phases. During phase 1, nodes are considered one by one and each one is placed in the neighboring community (including its own community) that maximizes the modularity gain. This phase is repeated until no node is moved (that is when the decomposition reaches a local maximum). Phase 2 consists of building the graph between the communities obtained during phase 1. Then, the algorithm starts

Algorithm 2: Fixed nodes and Preferential attachment

```
Input: Probabilities p, q. $\mathcal{N}_T \to \mathcal{C}_T$ (old nodes to communities). Nodes $\mathcal{N}_{T+1}$.
Output: Communities assigned to nodes at this Louvain step.
foreach node $x$ in $\mathcal{N}_{T+1}$ do
    oldComm ← FindInOldCommunities($x$);
    // FIXED NODES (p)
    if oldComm is not NULL and IsRandomFixedNode($p$) then
        Do nothing. Keep the old community;
    else
        foreach neighbor $n$ of $x$ do
            modularityIncrease ← Calculate increase in modularity were x to be
            moved to n's community;
            if $n$ is in old communities then
                bestOldIncrease ← modularityIncrease if bigger;
            else
                bestNewIncrease ← modularityIncrease if bigger;
        // PREFERENTIAL ATTACHMENT (q)
        if $x$ has preferential attachment to old communities (using q factor)
        then
            bestIncrease ← bestOldIncrease if there's at least one "old" neighbor
            else bestNewIncrease;
        else
            bestIncrease ← Max(bestOldIncrease, bestNewIncrease);
        Move $x$ from current community to bestIncrease one, if different.
return nodesToComms
```

phase 1 again with the new graph, in the next hierarchical level of execution, and continues until the modularity does not improve anymore.

We construct a set $\mathcal{P} \subseteq \mathcal{N}_{T+1}$ such that $|\mathcal{P}| = q \cdot |\mathcal{N}_{T+1}|$. For every node x, we consider its neighbors that belong to a community existing at time T, that is the set $A(x) = \{z \in \mathcal{N}_{T+1} \,|\, (x,z) \in \mathcal{E}_{T+1} \land \gamma_{T+1}(z) \in \Gamma_T\}$. During phase 1 of the first iteration of the algorithm (i.e. during the first hierarchical level of execution), the inner loop is modified. For all node $x \in \mathcal{N}_{T+1}$, if $x \in \mathcal{P}$ and $A(x) \neq \emptyset$ then place x in the community of $A(x)$ which maximizes the modularity gain (whereas if $A(x) = \emptyset$ proceed as usual).

The two methods ("fixed nodes" and "preferential attachment") apply at every Louvain step and, since they are related, may be described as a whole (see Algorithm 2).

4 Experimental Results

In our experiments we computed the social graph (constructed as described in Section 2). Since we are interested in the real-world application of our method we preferred to evaluate it on real data.

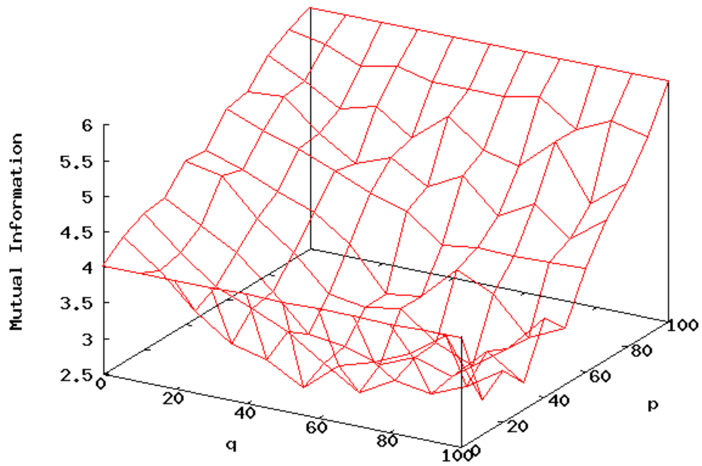

Fig. 1. Mutual Information as a function of p and q (expressed as percentages).

Given two months T and $T+1$, we calculated a partition in communities of $\mathcal{G}_T$ using the Louvain Method (with the modification of [5]) that we note $\Gamma =< C_1, \ldots, C_R >$; and a partition of $\mathcal{G}_{T+1}$ using our dynamic version of the Louvain Method, with different values of the parameters p and q. Let $\Gamma' =< C'_1, \ldots, C'_S >$ be the partition of $\mathcal{G}_{T+1}$. We are interested in comparing Γ and Γ' in terms of stability and quality of the partition. To this end, we measure: (i) the mutual information between the two partitions; (ii) the number of matching communities (i.e. such that the proportion of nodes in common is greater than a parameter r); (iii) the final modularity of Γ' (as defined in [6]).

The number of matching communities is computed as follows: for each community $C'_j \in \Gamma'$, we evaluate whether there is a community $C_i \in \Gamma$ such that $|C_i \cap C'_j| > r \cdot |C_i|$ and $|C_i \cap C'_j| > r \cdot |C'_j|$, where r is a fixed parameter verifying $r > 0.50$ (for instance we used $r = 0.51$). In that case, we say that C'_j matches C_i (another criterion is Jaccard similarity: intersection size divided by union size, see [2] and [1]). The matching communities are of particular interest, because C'_j can be considered as the evolution of C_i (although the community may have grown or shrank) and can be individually followed by a human analyst.

The mutual information for two partitions of communities (see [7, 3] for definitions[1]) is computed as:

$$MI(\Gamma, \Gamma') = \sum_{i=1}^{R} \sum_{j=1}^{S} P(C_i, C'_j) \log \frac{P(C_i, C'_j)}{P(C_i) \cdot P(C'_j)} \ .$$

[1] Since nodes can change between time T and $T+1$, we only consider the intersection $\mathcal{N}_T \cap \mathcal{N}_{T+1}$ for the mutual information computation.

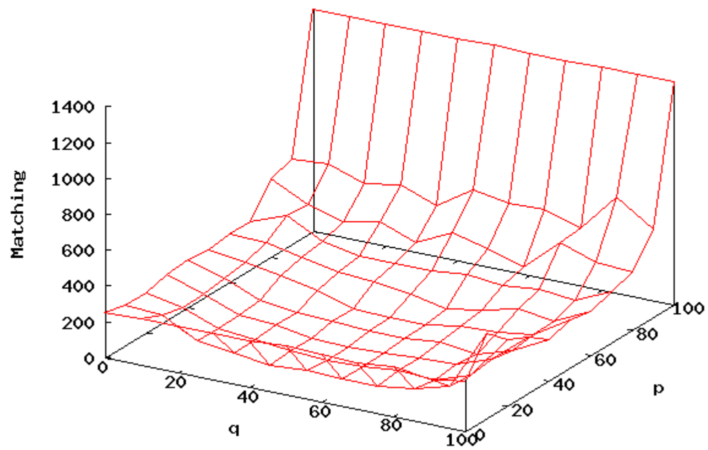

Fig. 2. Matching communities as a function of p and q.

To analyze the effect of p and q, we made those parameters vary from 0 to 1. The baseline, for $p = 0$ and $q = 0$, corresponds to the Louvain Method with the modifications of [5].

Fig. 1 shows the effect on the mutual information between the two partitions. We can clearly observe that the mutual information increases as p increases and reaches its maximal values at $p = 100\%$. The effect of varying q is not so clear since it produces fluctuations of the mutual information without a marked tendency.

Fig. 2 shows the number of matching communities (according to our criterion). In this graph we see that the number of matching communities increases dramatically when p approaches 100%. The effect of varying q is again not clearly marked although the increase of q produces higher matching communities for smaller values of p.

Fig. 3 shows the effect on the modularity of the new partition. We can observe that the modularity decreases slightly as p increases for small values of q. For greater values of q (closer to 100%), varying p produces fluctuations with a decreasing tendency.

As a conclusion, we can see that increasing the probability p of fixed nodes has a clear effect on increasing the mutual information between the two partitions and the number of matching communities. The trade-off with quality is good, since the decrease in modularity is relatively low.

By contrast, increasing the probability q of preferential attachment to pre-existing communities has no clear effect on mutual information or matching communities. It does not seem advisable to use this second modification to generate evolving communities.

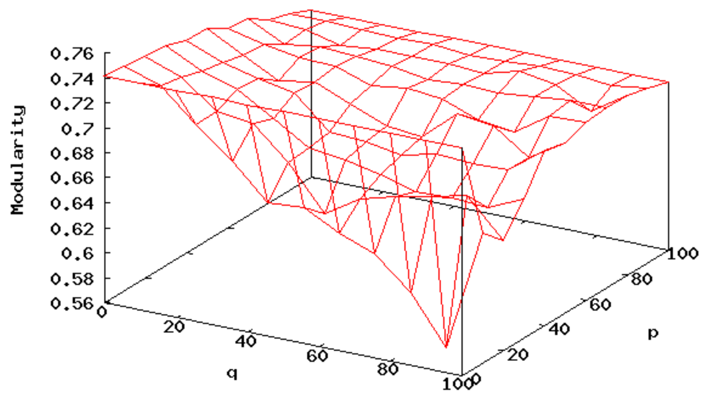

Fig. 3. Modularity as a function of p and q.

5 Related Work

Many related papers have investigated the importance of communities, and served us as reference. Foundational concepts and examples for communities detection may be found in [6]. For instance, the interactions between the major characters of the novel "Les Misérables" by Victor Hugo can be viewed as a graph with 77 nodes, and the characters organized as communities (the example is very intuitive and easy to grasp for the fans of the novel). Another classic example is Zachary's karate club, an organization with 34 nodes that split into two separate clubs in real life. Each new part can be mapped almost perfectly to the two main communities of friends detected in the original club. The paper also analyzes algorithms, such as shortest-path and random walk. Of course, the graphs considered in [6] have a much smaller scale than the ones considered in this work.

To perform community detection in graphs with 92 million nodes (see Section 2), efficient algorithms are required. We based our research on the Louvain Method algorithm originally published in [4]. As discussed in Section 3, it was modified in [5] to get a dynamic algorithm. However, that algorithm still lacks stability. We used the implementation of [5] as baseline to evaluate our Dynamic Louvain Method.

A thorough study of the history and the state of the art in communities detection (up to 2010) can be found in [2]. In particular the author discusses ideas on the roles of vertices within communities. We have implemented a classification of nodes as leaders, followers and marginals within each community. Our leaders correspond to "central vertices", but we don't compute boundary vertices, which could be useful.

Besides the static analysis, the report discusses communities evolution (dynamic communities), although it points out that "the analysis of dynamic communities is still in its infancy." It suggest that it would be desirable "to have a unified framework, in which clusters are deduced both from the current structure of the graph and from the knowledge of the cluster structure at previous times." We have implemented that idea, since we use the previous history of community structure throughout the whole algorithm (according to the p and q parameters), and not only during the nodes initialization (as in [5]).

Applications of "communities evolution" are not only to be found in mobile social networks. In [8], the authors study the evolution of scientific collaboration networks. In the analysis of exchange markets [7], the dynamics of currency exchanges (viewed as a dynamic graph of currency pairs) have been studied. For instance, modifications in the currency exchange communities effectively reflect the Mexican peso crisis of 1994. The scale is also much smaller (only 11 currencies being analyzed). A similarity with our work is that financial markets are one of the few fields where a detailed time evolution is readily available. In our case we have data from telecom companies that span a wide range of time (several months), and has fine grained resolution (day, hour, minute and second of each call or message).

6 Conclusion and Future Steps

The detection of evolving communities is a subject that still requires further study from the scientific community. We propose here a practical approach for a particular version of this problem where the focus is on stability. The introduction of fixed nodes (with probability p) increases significantly the stability of successive partitions, at the cost of a slight decrease in the final modularity of each partition.

As future steps of this research, we plan to: (i) study the evolution of communities with finer grain, using smaller time steps; (ii) evaluate the proposed method on publicly available datasets, to facilitate the comparison of our results; (iii) refine the matching criteria, and consider additional events in the evolution of dynamic communities (such as birth, death, merging, splitting, expansion and contraction [3]).

References

1. Bin Wu, Qi Ye, Shengqi Yang, and Bai Wang. Group CRM: a new telecom CRM framework from social network perspective. In *CNIKM'09*, pages 3–10. ACM, 2009.
2. Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.
3. Derek Greene, Dónal Doyle, and Pádraig Cunningham. Tracking the evolution of communities in dynamic social networks. In *ASONAM'10*, pages 176–183. IEEE, 2010.
4. Vincent Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

5. Thomas Aynaud and Jean-Loup Guillaume. Static community detection algorithms for evolving networks. In *WiOpt'10*, pages 513–519. IEEE, 2010.
6. Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.
7. Daniel J Fenn, Mason A Porter, Peter J Mucha, Mark McDonald, Stacy Williams, Neil F Johnson, and Nick S Jones. Dynamical clustering of exchange rates. *Exchange Organizational Behavior Teaching Journal*, 2012.
8. Gergely Palla, Albert-László Barabási, and Tamás Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, 2007.