Project in Computational Game Theory: Communities in Social Networks

Eldad Rubinstein

November 11, 2012

1 Presentation of the Original Paper

1.1 Introduction

In this section I present the article [1]. The article starts by presenting the term *community* in the context of sociology and graph theory. A community is described as a group of nodes more densely connected with each other than with the rest of the network.

When communities are not allowed to overlap, there are many successful approaches to recover the community structure of a network, such as clustering. However, in real life social networks, communities usually *overlap* each other, so other directions are necessary. Solving this problem directly is not possible, because it would lead to NP-hard problems. Another possible method is assuming a generative model that explains how communities are created. However, developing such models requires reliable data about real world community structure, which is the problem we try to solve in the first place.

The presented paper tries to solve this problem by stating assumptions about the network (as opposed to complete models), and suggesting algorithms that try to recover the communities from the network under these assumptions. The assumptions are based on *ego-centric analysis*, which is a sociological approach that uses information about ties of individuals to gain insight about the entire network.

1.2 Assumptions

The basic setup is as follows: The network is a graph of size n. Each community C is an arbitrary set of nodes, unknown to the algorithm.

The following assumptions are made:

0. Each person participates in up to d communities, where d is constant or small.
1. *Expected Degree Model* — Each node $u \in C$ has an affinity $p_u \in [0,1]$. For $u, v \in C$, The edge (u, v) exists with probability $e_{u,v} = p_u p_v$[1].
2. *Maximality* with gap ε — Nodes outside the community C are less strongly connected to it than community nodes are. For example, if $p_u = \sqrt{\alpha}$ for $u \in C$, then $w \notin C$ has edges to less then $\alpha - \varepsilon$ fraction of nodes in C.
3. Communities explain γ fraction of each person ties.

1.3 First Step: Communities are Cliques

In the following sections, another assumption is made:

For each community C, $\delta k \leq |C| \leq k$ where $\delta > 0$ is some constant and k is arbitrary but known to the algorithm.

First, we assume that all communities are cliques, that is each node u has an affinity $p_u = 1$. Under these assumptions the following theorem is proven:

Theorem 1 *The Clique-Community-Find algorithm outputs each community with probability $\geq 2/3$ in time $O\left(nk/\delta\gamma\right) 2^{\tilde{O}\left(\log^2 d\right)}$.*

Algorithm 1 Clique-Community-Find (informal overview)

1: Pick "starting nodes" uniformly at random
2: For each starting node v, randomly sample $S \subseteq \Gamma(v)$ by including each node with probability p. Proceed only if the sample is large enough.
3: **for all** cliques U of size at most $2pk$ in the induced graph $G(S)$ on S **do**
4: $V' \leftarrow \{$nodes in $\Gamma(v)$ which are connected to all nodes in $U\}$
5: Return high degree vertices from $G(V')$
6: **end for**

[1] Each node might belong to more than one community. In this case each node has a different affinity for each community it belongs to, and the edge (u, v) exists with probability which is at least the maximum of $p_u p_v$ for all the communities u, v are in.

1.4 Communities are Dense Subgraphs

In reality, communities are usually not cliques, so the algorithm presented in the previous section will not necessarily work. Instead, we assume that all affinity values are $p_u = \sqrt{\alpha}$ (even if two nodes u, v belong to more than one community, $e_{u,v} = \alpha$). Using a similar algorithm, it can be proven that:

Theorem 2 *Under the described assumptions ($p_u = \sqrt{\alpha}$), each community can be found with high probability over the graph randomness and with probability 2/3 over the algorithm randomness in time* $O\left(n \cdot (k/\gamma\alpha\delta) \cdot 2^{\tilde{O}\left(\log^2 d\right)}\right)$.

The assumption regarding α can be relaxed even further in order to fit real-life communities, so now we assume $p_u \geq \sqrt{\alpha}$ (for a fixed α).

Theorem 3 *Under the described assumptions ($p_u \geq \sqrt{\alpha}$), each community can be found with high probability over the graph randomness and with probability 2/3 over the algorithm randomness in time* $O\left(n \cdot (k/\gamma\alpha\delta)^T \cdot 2^{\tilde{O}\left(\log^2 d\right)}\right)$.

The reason for the worse running time is the different algorithm used in this case. Previous algorithms might not work here, because $\Gamma(v) \cap C$ is no longer a uniform subset of C. Therefore, in order to prove Theorem 3, the used algorithm has to loop over all sets of nodes $S \subseteq \Gamma(v)$ of certain size T for each sampled node v.

1.5 Communities with Very Different Sizes

We would like to release the assumption about community sizes that was made in Section 1.3 and allow communities of any size. However, sampling might miss some of the small communities. The previous algorithms will still work if they take enough samples, but then they will not be efficient ($1/\delta$ will be too large).

Instead, a different algorithm is presented. This algorithm uses the following definition:

Definition For α, $\varepsilon > 0$ an $(\alpha, \alpha - \varepsilon)$-set is a subset of nodes such that 1) every node in the set has edges to at least an α fraction of nodes in the set; and 2) every outside node has edges to at most an $\alpha - \varepsilon$ fraction of nodes in the set.

Theorem 4 *Under the base assumptions (0-3), and if for all communities, $p_u \geq \sqrt{\alpha_{min}}$ and all communities are $(\alpha_C - \varepsilon/8,\ \alpha_C - 7\varepsilon/8)$ sets, then Any-Size-Dense-Community-Find algorithm will output all communities in time*[2] $O\left(n^{\frac{\log(kd/\gamma)}{\alpha_{min}\varepsilon^2}}\right)$.

[2] k is still used to denote the size of the largest community.

Algorithm 2 Any-Size-Dense-Community-Find

$T = \frac{100 \log(kd/\gamma)}{\alpha_{min}\varepsilon^2}$
for $\alpha = 1$ downto α_{min} step $-\varepsilon/4$ **do**
    for all sets of nodes S of size T **do**
        $U \leftarrow \{v : \geq \alpha - \varepsilon/4$ fraction of its edges are to $S\}$
        Return U if it is a $(\alpha, \alpha - \varepsilon/2)$ set
    end for
end for

Any-Size-Dense-Community-Find algorithm does solve the problem of communities with very different sizes, however iterates over many sets, and therefore its running time is not polynomial.

The paper presents another algorithm, that finds *cliques* of very different sizes. It needs the following additional assumptions:

3'. This is a stronger variation of the original Assumption 3. It states that every community a node v belongs to has size at least γ/d times the number of ambient edges incident on the node v. That is, small communities are distinguishable from "noise" edges.

4. *Distinctness* — For $u \in C$, at least a constant factor of the community C does not lie in any other community containing u.

5. *Completeness (Duck Assumption)* — Any set that satisfies all the assumptions of a community in the model is a community. This ensures that Assumption 4 cannot be satisfied by pretending that a certain set in not a community.

Theorem 5 *Given a graph G that satisfies all presented assumptions, cliques of different sizes can be found in it with high probability in polynomial time.*

The algorithm required to prove this theorem works in the following way: it finds large cliques first using sampling. Then it ignores their edges and continues to smaller cliques. The extra assumptions ensure these small communities can be found.

1.6 Relaxing the Assumptions

The presented paper states that Theorems 3 and 4 still hold even if Assumption 1 (expected degree model assumption) is released, but instead we have to require that the following are concentrated around their expectation:

- The number of edges from any node u to any community C (that is, $|\Gamma(u) \cap C|$)

- The degree of each node are concentrated around their expectation.
- The number of nodes that are neighbors of two nodes u, v inside a community C (that is, $|\Gamma(u) \cap \Gamma(v) \cap C|$)

Another assumption that can be (independently) relaxed is Assumption 2, known as the Gap Assumption. The paper shows that even if the assumption does not hold, as it might be the case with real-life graphs, Clique-Community-Find Algorithm (see Section 1.3) will still return sensible answers:

Theorem 6 *Let G be a graph that satisfies all assumptions of Theorem 1 other than the Gap Assumption. An adopted version of Clique-Community-Find Algorithm will be able to find the communities in G, but for every community C the algorithm will return a similar set C' (and not C itself).*

The paper states that Gap Assumption can be also relaxed in a similar way in the setting of Theorem 2 (where $p_u = \sqrt{\alpha}$).

1.7 Sparser Communities

Ideas that are related to those shown in the previous sections can be used to identify sparse communities, in which the probability that two community nodes will be connected is smaller. A different set of assumptions is used here:

0. Assumption 1 (expected degree model).
1. (u, v) exists with probability $B/\sqrt{k}$, where $|C| = k$ for all communities and $B > 10$ is a constant.
2. All edges belong to some community.
3. Communities intersection size is limited — $|C \cap C'| \leq k/20d^2$.[3]

In order to find communities under these assumptions, we transform our sparse G into a dense graph G' that fulfills the assumptions of Theorem 3, that deals with dense and similar size communities.

Theorem 7 *Let a graph G and a set of communities $\mathcal{C}$ be consistent with the Sparse Model. Construct a graph G' on the same set of nodes, where u, v have an edge in G' if and only if they have at least $B^2/2$ length-2 paths in G. Then the pair $(G', \mathcal{C})$ are consistent with the assumptions of Theorem 3 with parameters $(n, k, d, \alpha, \delta, \varepsilon, \gamma) = (n, k, d, 0.9, 1, 0.6, 1/3d)$.*

[3] Recall that d is the maximum number of communities in which each person can participate.

Table 1: Summary of the Community Finding Algorithms

case no.	extra / different assumptions?	probability of edges in communities	communities sizes must be similar?	running time
1.	No	Cliques	Yes	Polynomial
2.	No	$p_u = \sqrt{\alpha}$	Yes	Polynomial
3.	No	$p_u \geq \sqrt{\alpha}$	Yes	Polynomial
4.	No	$p_u \geq \sqrt{\alpha}$	No	Quasi-Poly
5.	Extra	Cliques	No	Polynomial
6.	Different	Sparse	Yes	Polynomial

2 Open Questions

One can think of the several areas of possible further research following the article:

- Releasing the base assumptions in more cases, especially the Expected Degree Model Assumptions and the Gap Assumption.
- Polynomial algorithm for dense communities with different sizes. The paper presents two algorithms for communities with very different sizes. The first is not polynomial, and the second adds more restrictions. The question is whether some of these restrictions can be relaxed.
- Implementation of the suggested algorithms. This might involve using suitable heuristics in order to find cliques or dense subgraphs.
- Testing on real-world data. The questions are will this data satisfy the assumptions, and what will be the running time (possibly using heuristics).
- Adapting the algorithms to a dynamic setting. For instance, a model in which there is an initial graph that is built according to the assumptions, and then each node connects to each neighbor of a neighbor with some probability.

3 Understanding the Proof of Theorem 1

In this section we will try to fill some of the missing details in the proof of Theorem 1[4]. The proof of this theorem states that *with high probability, the subsampling gives a sample S of size at most thrice the expectation.* We will try to bound that probability.

The paper hints that Chernoff Bound should be used. Chernoff Bound is a known probabilistic inequality that can be stated as follows:

Let $X_1, \ldots, X_n$ be independent random variables. They need not have the same distribution. Assume that $0 \leq X_i \leq 1$ always, for each i. Let $X = X_1 + \cdots + X_n$ and $\mu = \mathbf{E}[X] = \mathbf{E}[X_1] + \cdots + \mathbf{E}[X_n]$. Then for any $\varepsilon' \geq 0$,

$$\mathbf{Pr}\left[X \geq (1+\varepsilon')\mu\right] \leq \exp\left(-\frac{\varepsilon'^2}{2+\varepsilon'}\mu\right) \text{ and } \mathbf{Pr}\left[X \leq (1-\varepsilon')\mu\right] \leq \exp\left(-\frac{\varepsilon'^2}{2}\mu\right)$$

In our case, we can define X_i for each $v_i \in \Gamma(v)$ such that $X_i = \begin{cases} 1 & v_i \in S \\ 0 & v_i \notin S \end{cases}$.

Therefore,

$$\mathbf{E}[|S|] = \mathbf{E}[X_1 + \cdots X_n] = \mathbf{E}[X] = \mathbf{E}[X_1] + \cdots + \mathbf{E}[X_n] = \mu$$

Using Chernoff Bound (we choose $\varepsilon' = 2$),

$$\mathbf{Pr}\left[|S| \geq 3\mathbf{E}[|S|]\right] \leq \exp\left(-\mathbf{E}[|S|]\right) = \exp\left(-\frac{deg(v)}{\delta k} \cdot \frac{\log(12d/\varepsilon\delta\gamma)}{\varepsilon}\right)$$

It is known that:

- $d \geq 1$, because d denotes the maximum number of communities in which each person can participate.
- $0 < \varepsilon, \delta, \gamma < 1$, because $\varepsilon, \delta, \gamma$ all represent fractions.
- $deg(v) \geq \delta k$, because δk is the minimum community size, v is a member of at least one community and all communities are cliques in this setting.

Therefore, assuming log is the natural logarithm,

$$\mathbf{Pr}\left[|S| \geq 3\mathbf{E}[|S|]\right] \leq \exp\left(-\log(12)\right) = \frac{1}{12}$$

[4]It is numbered Theorem 1 in the original paper as well.

4 Implementing Clique-Community-Find

We would like to make the first step towards testing the suggested algorithms, which is to implement Clique-Community-Find Algorithm (see Section 1.3) and test it on toy data. The source code (in Python) is attached to the project.

Two toy data graphs are used (see Figures 1 and 2):

1. 100 nodes, 3 non-overlapping cliques. In addition, random (noise) edges are added with probability 0.01 per edge.
2. Similar to the previous graph, but 10 nodes in each clique overlap.

The implemented algorithm performs well on both data sets and consistently recovers the exact cliques using the following parameters:

$$k = 100,\ \delta = 0.3,\ \varepsilon = 0.3,\ \gamma = 0.8,\ d = 4$$

References

[1] Sanjeev Arora, Rong Ge, Sushant Sachdeva, and Grant Schoenebeck. Finding overlapping communities in social networks: toward a rigorous approach. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pages 37–54, New York, NY, USA, 2012. ACM.

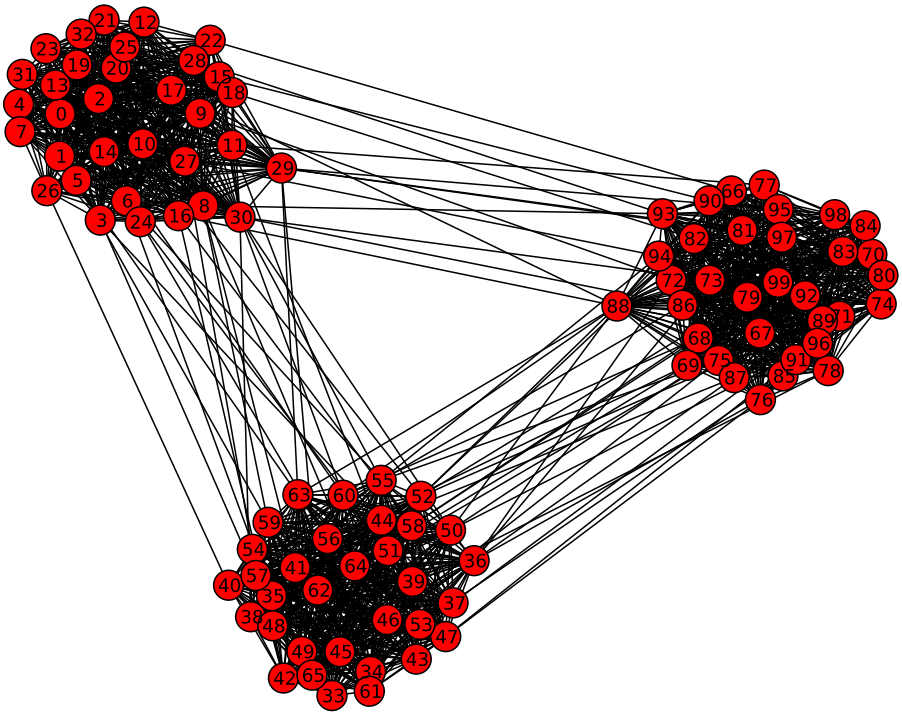

Figure 1: Toy Data without Overlapping Cliques

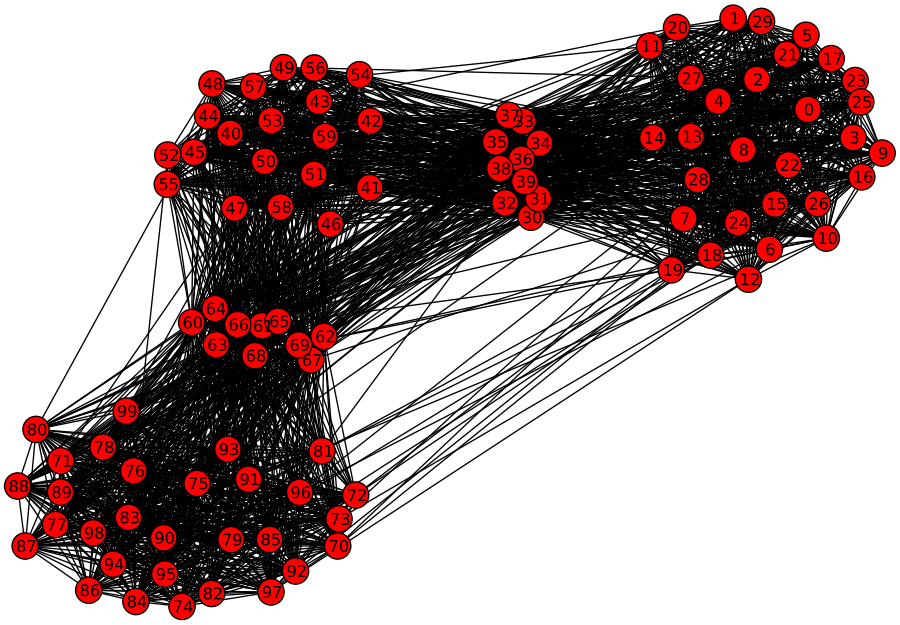

Figure 2: Toy Data with Overlapping Cliques