Ego-Splitting Framework: from Non-Overlapping to Overlapping Clusters

Alessandro Epasto
Google Research NY
aepasto@google.com

Silvio Lattanzi
Google Research Zurich
silviol@google.com

Renato Paes Leme
Google Research NY
renatoppl@google.com

ABSTRACT

We propose *ego-splitting*, a new framework for detecting clusters in complex networks which leverage the local structures known as ego-nets (i.e. the subgraph induced by the neighborhood of each node) to de-couple overlapping clusters. Ego-splitting is a highly scalable and flexible framework, with provable theoretical guarantees, that reduces the complex overlapping clustering problem to a simpler and more amenable non-overlapping (partitioning) problem. We can scale community detection to graphs with tens of billions of edges and outperform previous solutions based on ego-nets analysis.

More precisely, our framework works in two steps: a local ego-net analysis phase, and a global graph partitioning phase. In the local step, we first partition the nodes' ego-nets using a partitioning algorithm. We then use the computed clusters to split each node into its *persona* nodes that represent the instantiations of the node in its communities. Finally, in the global step, we partition the newly created graph to obtain an overlapping clustering of the original graph.

KEYWORDS

Overlapping clustering; ego-nets; large-scale graph algorithms.

1 INTRODUCTION

Detecting the clustering structure of real-world networks has emerged as an important primitive in a wide range of data analysis tasks such as community detection [19], event detection [39], spam detection [3], computational biology [22], link prediction [31] and many others. As a result, the study of the topology of real world networks and of their clustering (or community[1]) structure is central in modern network analysis. In particular, in recent years, several models have been introduced to capture the community structure of social networks [25, 27, 28] and numerous empirical studies analyzed the community structures at a macroscopic [21, 28, 29] and microscopic [13, 17] levels. One of the main observations in this line of work is the lack of a clear macroscopic community structure in real world networks. For instance in [29], Leskovec et al. give an empirical evidence that at global level it is rare to observe medium-sized communities (around 100 nodes) with clear clustering properties.

In addition, Abrahao et al. [1] collected empirical evidence showing that real-world communities are rarely detected by commonly used algorithms. In particular, real clusters overlap with each other and haves many edges crossing cluster boundaries. As a result, real world graphs do not exhibit a clear clustering structure at the macroscopic level.

In sharp contrast with these findings, it has been observed that while the community detection problem is hard at a macroscopic level, it becomes simple at a microscopic level [13, 17]. This is especially true when we restrict our attention to local structures know as *ego-nets* (a.k.a. ego-networks) which consist of the subgraph induced over the neighborhood of a single node in the graph. Intuitively, this happens because, even if a node is part of many communities, if we restrict our attention to a node and one of her neighbors, there is only one or a limited number of communities in which the two nodes interact, which present a clearer structure at the level of the neighborhood. In fact, Epasto et al. [17] analyze the ego-nets community structure of several graphs and show that it is possible to detect high quality communities in them using a simple out-of-the-shelf partitioning algorithms.

Inspired by this encouraging observation, we design a novel framework which we call *ego-splitting*. The main idea behind the framework is to use the guidance of local clustering structure to detect overlapping communities. The idea of using local clustering structure to obtain a global clustering is not new, for instance, Coscia et al. [13] recently designed an algorithm based on this approach. In this paper, however, we leverage for the first time this idea to design a highly scalable and flexible framework with provable theoretical guarantees, that reduces the complex overlapping clustering problem to a simpler and more amenable non-overlapping (partitioning) problem. This is particularly interesting because it allows us to use the large literature on non-overlapping clustering to approach the more complex overlapping clustering problem on large-scale graphs.

More formally, our *ego-splitting* framework works in two steps: a local ego-net analysis and a global graph partitioning. In those steps we use two partitioning algorithms $\mathcal{A}^\ell$ and $\mathcal{A}^g$ as a black box.

The first step of our framework is the ego-nets clustering. In this step for every node u, the framework constructs the ego-nets of u and then uses algorithm $\mathcal{A}^\ell$ to partition the neighborhood of u. For each community in the partition *ego-splitting* creates a new replica of u (which we call *persona*) that is associated uniquely with a cluster in the partition. Then we map each edge between nodes in the original graph to an edge between personas. The output of this step is a new graph which we refer to as the *persona* graph where

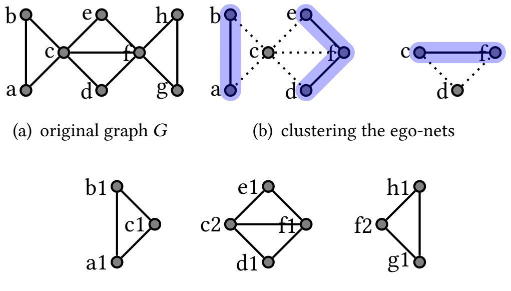
[1] Note that in the paper we use the terms cluster and community interchangeably.

KDD'17, August 13–17, 2017, Halifax, NS, Canada.
© 2017 Copyright held by the owner/author(s). 978-1-4503-4887-4/17/08.
DOI: http://dx.doi.org/10.1145/3097983.3098054

each node u is replaced by a series of copies called the *personas* of u.

Then, in the second step of our framework, the global partition step, *ego-splitting* runs a partitioning algorithm $\mathcal{A}^g$ (potentially the same as $\mathcal{A}^\ell$) on the resulting persona graph and returns the clustering detected by $\mathcal{A}^g$.

To clarify the main intuition behind our framework we now present a visual example in Figure 1 where we show the execution of our method using as clustering algorithms $\mathcal{A}^\ell$ and $\mathcal{A}^g$ the simple connected component algorithm.



(a) original graph G (b) clustering the ego-nets

(c) splitting the ego we obtain the persona graph

Figure 1: The ego-splitting framework applied to a simple graph to transform an overlapping clustering problem into a partitioning problem.

First, note that in the graph in Figure 1 there are 3 overlapping communities: $\{a,b,c\},\{c,d,e,f\},\{f,g,h\}$. In particular nodes c and f are part of two communities. Although, when restricting the attention to the the ego-net of any specific node (which recall, does not includes the node itself), the communities are naturally de-coupled. For instance, consider the ego-net of c in Figure 1(b), when we consider the graph induced on c's neighborhood, the two communities of c are easy to identify–they correspond exactly to the two connected components $\{a,b\}$ and $\{d,e,f\}$. So by using connected components our framework would be able to detect that even if there is one single node c in the graph, c has in reality two *personas* and so it would split c into two different nodes: c_1 and c_2. Note that besides nodes c and f the other nodes participate in a single community. In this case our algorithm keeps these nodes in a unique persona, for instance, d_1 in this case of d.

After the first local step, in the second global step the connected component algorithm can easily detect the overlapping community structure of the graph by partitioning the persona graph in the clusters $\{a_1,b_1,c_1\}$, $\{c_2,d_1,e_1,f_1\}$ and $\{f_2,g_1,h_1\}$ which corresponds to overlapping clusters in the original graph.

Even if this anecdotal example may look artificial at first sight, it captures well the complexity of real world graphs where people or entities are part of a multitude of communities. For example we can imagine node c in Figure 1 as a college student that participates in two clusters representing her college friends and her friends from a sports club. Clearly Figure 1 is an oversimplified scenario but, interestingly we observe empirically that our *ego-splitting* framework works well also in more complex realistic settings. While for the toy example in Figure 1 a simple articulation or bridge detection would work, the ego-splitting framework is able to dis-entangle communities even for highly connected graphs without bridges or articulations as we show in various examples later in the paper.

Notice also that more sophisticated algorithms than simple connected components can be used both at the ego-net clustering and at the persona clustering step. In fact, thanks to its flexibility, our framework can be used to transform any partitioning algorithm in an overlapping clustering algorithm.

Our Contribution. We introduced the new *ego-splitting* framework to reduce the overlapping clustering problem to a non-overlapping partition problem. Our methods scale easily in distributed settings, enabling the analysis of the overlapping community structure in graphs with tens of billions of edges using standard non-overlapping clustering algorithms.

We analyze the performance of the method both experimentally and theoretically. Experimentally we compare the performance of our algorithm against state of the art ego-net based clustering algorithms and measure their performance in terms of standard metrics for clustering detection: F_1-score and Normalized Mutual Information (NMI). We show that ego-splitting outperforms other algorithms in both metrics and both in real-world graphs with labeled communities (amazon, dblp, livejournal, orkut and friendster) and in synthetic benchmark graphs constructed by Lanchinetti et al. [26].

We also analyze ego-splitting theoretically in a random overlapping clusters model, where clusters are chosen to be random subsets of the vertex set and for each cluster we overlay a random Erdös-Renyi graph. We bound the Jaccard similarity between the clusters produced by the algorithm and the original clusters used by the model and show that, for natural ranges of parameters, the algorithm is able to perfectly reconstruct the clusters in the limit.

2 RELATED WORK

The related works span large research areas such ego-net analysis and community detection. In this section we restrict ourselves to reviewing only the most closely related papers in these areas.

The concept of ego-net (or ego-network) was first introduced in the seminal work of Freeman [20]. Then the study of ego-nets established itself at the basis of social network analysis [11, 15, 18, 38]. Recently, a widespread attention has been devoted in the computer science community on mining ego-nets. In their pioneering work, Rees and Gallagher [36] proposed to the use of ego-nets to find a global clustering of the graph. The core idea of their algorithm is to find basic communities by computing the weakly connected components for each ego-net after removing the ego from it. Then to obtain a global clustering they merge communities that overlap significantly. Coscia et al. [13] built on this work employing label propagation algorithms to cluster the ego-nets and analyze different merging strategies. The merging procedures applied are not scalable as they require $O(n^2)$ computation in the worst case. More recently Buzun et al. [12] and Liakos et al. [14] introduced distributed algorithms for the variation of this problem problem. We observe that unfortunately both solutions are more complex and less flexible (they are tailored to the use a specific underlying clustering algorithm) than ours. In addition the authors do not show any theoretical guarantees of their algorithms. The work [14] also assumes to know a set of seeds in each community which the algorithm expands locally. This is a different problem from the one we address of producing an overlapping clustering of the entire graph. To the best of our knowledge we are the first to introduce the concept of *persona graph* and to leverage it for obtaining a scalable and distributed ego-net based overlapping community detection method with provable guarantees.

In a different line of work, McAuley and Leskovec [33] provided a machine learning approach to cluster ego-nets. Subsequently Yang et al. [43] proposed an extension of this model for directed and undirected graphs. Finally, Li et al. [30] extended their learning model to also capture hidden attributes that are not explicitly present in the input.

The literature on community detection is very rich, for a good survey on the topic refer to [19]. Our paper is particularly related to the overlapping community literature. Whang et al. [40] develop an overlapping community detection algorithm based on seed set expansion. The algorithm optimizes for conductances and requires a set of input seeds. Unfortunately, the algorithm does not have any theoretical guarantees. Furthermore, no distributed implementation of the algorithm is presented in the paper. In another paper Amoretti et al. [4] propose a parallel (not distributed) implementation of Demon [13] on mid-size graphs. Other relevant overlapping community algorithms have been presented in [7] and in [23], unfortunately no distributed implementation of those algorithms is known and so they cannot scale to very large graphs. Finally, the overlapping community problem has been analyzed theoretically in several papers [5, 6, 24], although the proposed algorithms are mostly theoretical and have not been evaluated in practice.

Finally, our work is also related to the edge partitioning problem [2, 32]. In the edge partition problem the objective is to find an algorithm to partition the edges in different communities. This problem received significant attention because it allows us to compute an overlapping clustering of the graph by simply computing a partitioning of the edges of the graph. Interestingly, we note here that our framework can be used to obtain scalable algorithms for this problem as well.

3 CLUSTER DETECTION PROBLEM

In the overlapping community detection problem the goal is to design an algorithm $\mathcal{R}$ which consumes an undirected graph $G = (V, E)$ and outputs a collection $\mathcal{S}' = \mathcal{R}(G)$ of (possibly overlapping) subsets of the node set V which we call *clusters*, i.e. each $C \in \mathcal{S}'$ is a subset $C \subseteq V$ and two sets $C, C' \in \mathcal{S}'$ can overlap.

In this paper we try, as much as possible, to be agnostic to a precise definition of what a cluster is. For this reason instead of specific quality function to optimize we use a cluster reconstruction approach to evaluate our method. In a cluster reconstruction formulation we assume that there is a set $\mathcal{S}$ of subsets of the nodes which we call *ground-truth clusters* and the algorithm is tasked with recovering such clusters. Of course, for the detection problem to be meaningful, $\mathcal{S}$ and G must be related in a way that it is possible to extract information about $\mathcal{S}$ from G.

In order for the problem to be more concrete, we define two different scenarios where we want to evaluate our algorithms. The first is that of *labeled datasets*, where graphs come with metadata identifying subsets of nodes that are known to be communities and that we want to retrieve. This is particularly common for social networks (many examples are shown in the experimental section). A second scenario is that of *generative models*, where a random process generates a graph from a set of clusters and the algorithm needs to recover those clusters having only access to the graph. We evaluate our methods in this context in our theoretical analysis.

3.1 Evaluating a detection algorithm

In most cases, exact reconstructions of the communities is unrealistic and we will settle for approximate reconstructions. In the rest of the subsection we define several notions of approximation in comparing two clusterings which are standard in the literature.

Given a ground truth cluster $C \subseteq V$ and reconstructed cluster $C' \subseteq V$ we define the precision $P(C', C) = |C \cap C'|/|C'|$ as the fraction of the reconstruction that is in the ground truth and the recall $R(C', C) = |C \cap C'|/|C|$ as the fraction of the ground truth that is in the reconstruction. The notions of precision and recall are often combined in a single number between 0 and 1 called F_1-score, defined as:

$$F_1(C', C) = 2 \cdot \frac{P(C', C) \cdot R(C', C)}{P(C', C) + R(C', C)}$$

The notion of F_1 has the additional advantage of being symmetric, i.e., $F_1(C', C) = F_1(C, C')$ and of being such that $F_1(C, C') = 1$ iff the sets C and C' are equal.

Now that we can compare two clusters, we define a metric for evaluating the set of clusters detected:

F_1 score. Given a collection of ground truth clusters $\mathcal{S}$ and a collection of detected clusters $\mathcal{S}'$, a widely used [13] measure of accuracy is the F_1 score of the reconstruction with respect to the ground truth as follows:

$$F_1(\mathcal{S}', \mathcal{S}) = \frac{1}{|\mathcal{S}'|} \sum_{C' \in \mathcal{S}'} \max_{C \in \mathcal{S}} F_1(C', C)$$

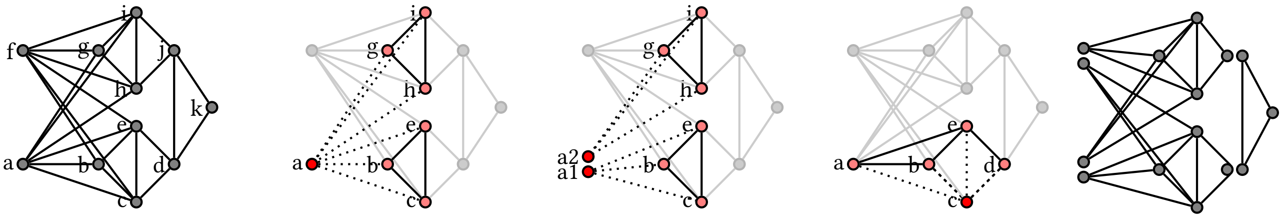
which corresponds to the average F_1 score of a reconstructed cluster with respect to the best match in the ground truth.

Normalized Mutual Information (NMI). We will also use another standard measure of detection quality based on information theory developed by Lancichinetti and Fortunato [26] and later refined by McDaid et al [34]. The measure was carefully crafted to avoid various pitfalls of previous measures and it is quite involved. We refer to the cited papers for an exact description and a comprehensive discussion of its merits.

4 EGO-SPLITTING FRAMEWORK

The main algorithmic idea in the paper is that each node in the graph is a blend of different *personas*. Instead of seeking to solve the clustering problem directly, we first split each node in different personas. This disentangles the different clusters and makes the graph simpler to cluster.

Before we can describe the procedure in detail we establish notation and define some standard notions in graph theory:



(a) original graph G (b) ego-net of a (c) splitting a in two personas (d) ego-net of c (one persona) (e) persona graph

Figure 2: Clustering the ego-nets, splitting the ego and building the persona graph

4.1 Graph Theory Notation

An (undirected, unweighted) graph $G = (V, E)$ consists of a finite set V of nodes and an edge set $E \subset V \times V$ such that $u \neq v$ for each $(u,v) \in E$ and $(u,v) \in E$ iff $(v,u) \in E$.[2]

A *non-overlapping clustering algorithm* $\mathcal{A}$ produces for each graph $G = (V, E)$ a partition $\mathcal{A}(G) = (V_1, \ldots, V_t)$ of variable size. We use the notation $\text{np}_{\mathcal{A}}(G) = t$ to denote the number of sets in the partition. The fact that $\mathcal{A}(G)$ is a partition means that $V_i \cap V_j = \emptyset$ for $i \neq j$ and that $V_1 \cup \ldots V_t = V$. An example of a non-overlapping clustering algorithm is the connected components algorithm, which simply splits the graph into connected components.

Given a graph $G = (V, E)$ and a subset of nodes $U \subset V$, we define the induced graph $G[U] = (U, E \cap U \times U)$. For a node $u \in V$, the neighborhood of u consists of the set of nodes connected to it $N_u = \{v; (u,v) \in E\}$ and the ego-net of u consists of the graph induced on the neighborhood $G[N_u]$. The ego-net represents the local view from node u on the graph connections (notice that the ego-net of u does not include the node u).

4.2 The Framework

The ego-splitting framework provides a general methodology for constructing an overlapping clustering algorithm starting from two (possibly equal) non-overlapping clustering algorithms: the local clustering algorithm $\mathcal{A}^{\ell}$ and the global clustering algorithm $\mathcal{A}^{g}$. The ego-splitting algorithm processes a graph G and outputs a set of clusters $\mathcal{S}'$ as follows:

- *Step 1:* For each node u we use the local clustering algorithm to partition the ego-net of u. Let $\mathcal{A}^{\ell}(G[N_u]) = \{N_u^1, N_u^2, \ldots, N_u^{t_u}\}$ where $t_u = \text{np}(\mathcal{A}^{\ell}, G[N_u])$.
- *Step 2:* Create a set V' of personas. Each node u in V will correspond to t_u personas in V' denoted by u_i for $i = 1, \ldots, t_u$.
- *Step 3:* Add edges between personas. If $(u,v) \in E$, $v \in N_u^i$ and $u \in N_v^j$ then add an edge (u_i, v_j) to E'.
- *Step 4:* Apply the global clustering algorithm $\mathcal{A}^{g}$ to $G' = (V', E')$ and obtain a partition $\mathcal{S}''$ of V'.
- *Step 5:* For set $C' \in \mathcal{S}''$ in the partition of V' associate a cluster $C(C') \subseteq V$ formed by the corresponding nodes of V, i.e., $C(C') = \{u \in V | \exists i \text{ s.t. } u_i \in C'\}$. Output $\mathcal{S}' = \{C(C') | C' \in \mathcal{S}''\}$.

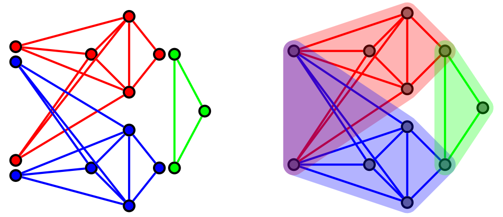
In Figure 2 we show an example execution using connected components as clustering method. In Figure 2(b) we depict Step 1 for node a: we look at the ego-net $G[N_a]$ and partition it. In Figure 2(c), we add for each partition of the ego-net a persona of a associated with that partition (Step 2). For example, persona a_1 is associated with nodes b, c, e. In Figure 2(d) we do the same process for node c, but since his ego-net has only one partition, we add single persona of c associated with a, b, e, d. After this is done for all nodes (in parallel), we build the persona graph, depicted in Figure 2(e). In this graph there is an edge for each edge in the original graph, for example, for edge (a, c) in the original graph, we add an edge between the persona of a associated with c (i.e. the persona of a associated with the cluster to which c belongs in the ego-net of a) and the persona of c associated with a (Step 3). Figure 3(a) depicts Step 4 where we apply the global non-overlapping to the persona graph and obtain clusters. Step 5 is finally in Figure 3(b) where we map the clusters defined on personas to overlapping clusters defined on original nodes.

Clustering edges. The transformation from the original graph to the graph of personas can increase the number of nodes in the graph, but keeps the number of edges constant, so in terms of memory, the persona graph consumes the same space as the original one. Also, since there is a one-to-one mapping of the edges in both graphs, the non-overlapping partitions of the persona graph also imply a clustering of the egdes. We are able to say for each edge, which cluster it belongs to. In that sense, our methodology can be also viewed as a edge-disjoint clustering approach.

Ego-splitting at scale. A naïve way to bound all ego-nets could be prohibitively expensive, at the order of $O(nm)$ for $n = |V|$ and $m = |E|$. Epasto et al [17] use a combinatorial bound on the number of triangles to show all ego-nets can be constructed in time $O(m^{3/2})$, which is a considerable gain for sparse graphs. Indeed, the bound in practice can be much better than $O(m^{3/2})$ and depends directly on the number of triangles in the graph. They also show that if T is the time to cluster a graph with m edges, the total work to build all ego-nets and cluster them is $O(\sqrt{m}T + m^{3/2})$. Furthermore they show that this step can be performed in two rounds of MapReduce. One more step is required to build the persona graph and one to associate the clusters of the persona graph to the original nodes. Taking those results together we have:

Theorem 4.1. *If $T_\ell(m)$ and $T_g(m)$ are the total work of the local and global algorithms on a graph of size m (respectively), R_g is the number of rounds of MapReduce required by the global partition*

[2] Notice that our algorithm could be conceivably adapted to directed and weighted graphs but we do not pursue this direction in this paper.



(a) non-overlapping partition of the persona graph

(b) overlapping clusters in original graph

Figure 3: Clustering the persona graph

algorithm and if the local partition can be run in memory for each ego-net, the ego-splitting framework can be implemented in $4 + R_g$ *rounds of MapReduce with total work* $O(m^{3/2} + \sqrt{m}T_\ell(m) + T_g(m))$.

Notice for instance that if the both the local and global partitioning algorithms are $O(m)$, and if the global partitioning operates in $O(1)$ rounds we can implement the entire approach using $O(m^{3/2})$ total work and $O(1)$ rounds.

5 PROVABLE RECONSTRUCTION GUARANTEES

We present a generative model with overlapping clusters in which we are able to reconstruct most of the clusters exactly by using the *connected component* algorithm as a non-overlapping partition algorithm both in the ego-net clustering phase and in the persona graph clustering phase. Connected component is arguably the simplest (and most rudimentary) non-overlapping clustering algorithm possible, but its analysis sheds light on the ability of our framework to achieve provable guarantees even using a simple partitioning method as a building block. It is conceivable that using more sophisticated algorithms allows to prove guarantees in other more difficult models.

In our model there are k clusters and each node $u \in V$ is assigned to a cluster $C \in \mathcal{S}$ iid with probability q. Now, between two nodes of each cluster we add an edge with probability p. A different way to describe our model is that we pick k random subsets and overlay an Erdös-Renyi graph on each of those subsets. We are presented with the graph obtained by the union of such random graphs and are asked to reconstruct what are the original subsets chosen.

We will show that for a range of parameters, it is possible to reconstruct most of the random sets with high probability. We don't make any claim that the model corresponds to any real-world network. Our goal here is simply to show that even in highly-connected graphs with heavily overlapping clusters, it is possible to disentangle them by analyzing the local ego-net structures using a simple clustering algorithm in our framework.

Random Overlapping Clusters. We define the generative process $\mathcal{P}(n,k,q,p)$ as the random process that starts with a set of n nodes $V = [n]$ and first sample a collection of k subsets $\mathcal{S}$ as follows: from 1 to k we sample $C \in \mathcal{S}$ by adding each node $u \in V$ to C with probability q, independently.

Now, we sample G as follows: we visit each $C \in \mathcal{S}$ and for each we sample the edge sets E_C as follows: for each $u, v \in C$ with $u \neq v$ we add edge (u,v) to E_C with probability p. In other words, E_C is the edge set of an Erdös-Renyi graph with vertex set C. Now the edge set of the G is simply the union $E = \cup_{C \in \mathcal{S}} E_C$.

We note that the same edge (u,v) can be present in more than one set E_C. In such case we still have only a single edge added to E.

Jaccard similarity of the reconstruction. The ego-splitting algorithm will be given access to G and will process it and produce a set of clusters $\mathcal{S}'$. For sake of the theoretical analysis we will measure the quality of the reconstruction using the simple Jaccard similarity over the set of clusters:

$$J(\mathcal{S},\mathcal{S}') = \frac{|\mathcal{S} \cap \mathcal{S}'|}{|\mathcal{S} \cup \mathcal{S}'|}$$

The Jaccard similarity is such that $0 \leq J(\mathcal{S},\mathcal{S}') \leq 1$ and $J(\mathcal{S},\mathcal{S}') = 1$ iff the reconstruction is exact, i.e., $\mathcal{S} = \mathcal{S}'$. In our experimental evaluation of the algorithm we show results for other more nuanced and widely used quality measures like F_1 score and NMI, notice that Jaccard similarity as defined is a very demanding measure (it consider a cluster with a single error as entirely wrong) and implies also bounds on F_1, since:

$$F_1 = \frac{\sum_{C' \in \mathcal{S}'} \max_{C \in \mathcal{S}} F_1(C',C)}{|\mathcal{S}'|} \geq \frac{|\mathcal{S} \cap \mathcal{S}'|}{|\mathcal{S}'|} \geq \frac{|\mathcal{S} \cap \mathcal{S}'|}{|\mathcal{S} \cup \mathcal{S}'|} = J$$

Our main result is as follows:

Theorem 5.1. *If* $\mathcal{S}$ *and* G *are sampled from a* $\mathcal{P}(n,k,q,p)$ *with* $kq \geq 1$ *and* $p \geq c' \log(npq/2)/(npq/2)$, *then:*

$$\mathbb{E}[J(\mathcal{S},\mathcal{S}')] \geq 1 - nk\exp(-\Omega(np^2q)) - O\left(n^3k^2p^2q^6\right)$$

Notice that the assumption $kq \geq 1$ is natural as otherwise nodes have fewer than one cluster in expectation.

To make our model more concrete we first consider the following example:

Example 5.2. In the random overlapping clusters model, for $0 < \epsilon < \frac{1}{6}$ constant, let $k = n$, $q = n^{\epsilon}/n$ and $p = 1/n^{\epsilon/4}$. Under those parameters each node is on average on $kq = n^{\epsilon}$ clusters. Each cluster has average size $nq = n^{\epsilon}$. So a back-of-envelope calculation will get us that the degree of each node is roughly: $n^{\epsilon} \cdot n^{\epsilon} \cdot p = n^{1.75\epsilon}$. Since the theorem conditions holds, the Jaccard coefficient is $\mathbb{E}[J(\mathcal{S},\mathcal{S}')] \geq 1 - O(n^{5.5\epsilon}/n)$.

Example 5.3. Set $k = n$, $q = c\log(n)/n$ (for a large enough constant c) and $p = O(1)$. Each node is on average on $O(\log n)$ clusters and each cluster has average size $O(\log n)$. The degree of each node is $O(\log^2 n)$. Since the theorem conditions hold, the Jaccard coefficient is $\mathbb{E}[J(\mathcal{S},\mathcal{S}')] \geq 1 - O((\log^6 n)/n)$.

In either example as $n \to \infty$ the similarity $\mathbb{E}[J(\mathcal{S},\mathcal{S}')] \to 1$ so we achieve perfect reconstruction in the limit.

5.1 Proof of Main Theorem

Our main tools will be Chernoff bounds and the connectivity threshold in the Erdös-Renyi model. We will use the following version of Chernoff bounds: if $X_i \in [0,1]$ are independent random variables and $\mu = \mathbb{E}[\sum_i X_i]$, then:

$$\mathbb{P}\left(|\textstyle\sum_i X_i - \mu| \leq \epsilon\mu\right) \geq 1 - 2\exp(-\epsilon^2\mu/4)$$

The other probability statement we will require is the following. Let $G(n,p)$ be the Erdös-Renyi random graph on n nodes where each edge is added with probability p. The following classical lemma bounds the probability of the graph being connected:

Lemma 5.4. *In the Erdös-Renyi random graph $G(n,p)$ if $p \geq 6\log(n)/n$, then*

$$\mathbb{P}[G(n,p) \text{ is connected}] \geq 1 - \exp(-\Omega(np))$$

The proof of this lemma is standard and thus omitted.

Our first step in proving the theorem will be to analyze the connectivity of the ego-net of u. For each $C \in \mathcal{S}$, let $G_C = (V, E_C)$ and define N_u^C to be the neighborhood of u in graph G_C. Since the final graph G is the union of the edges of the G_C graphs, $N_u = \cup_{C \ni u} N_u^C$.

Ideally we would like to look at the induced graph $G[N_u]$ and from it identify the sets N_u^C and split u into k_u personas, where $k_u = |\{C \in S; u \in C\}|$. If all N_u^C are disjoint and each is a connected component of $G[N_u]$, then we are done.

Our first statement is that for each cluster C and each $u \in C$, the graphs $G_C[C]$ and $G_C[N_u^C]$ are connected with high probability. The proof follows from concentration arguments. For each C we bound the probability that $|C|$ is at least $\frac{1}{2}\mathbb{E}|C|$ using Chernoff bounds and condition on that event, we use Lemma 5.4 to bound the probability that $G_C[C]$ is connected. For $G_C[N_u^C]$ the same argument can be done being more careful regarding which events to condition on. Due to space limitations we omit the proofs of the following two lemmas:

Lemma 5.5. *If $p \geq 6\log(npq/2)/(npq/2)$, then with at least*

$$1 - nk\exp\left(-\Omega(np^2q)\right)$$

probability, the graph $G[C]$ is connected for all $C \in \mathcal{S}$ and $G[N_u^C]$ is connected for all $u \in C \in \mathcal{S}$.

The previous lemma shows that with high probability, node u won't split N_u^C when performing the ego-splitting operation using connected components. However, it is also possible for two clusters to be wrongly merged. The next lemma describes the necessary conditions so that a cluster can be exactly reconstructed:

Lemma 5.6. *Fix a cluster C, if for for all $u \in C$ the following conditions hold:*

(1) the induced graph $G[C]$ is connected.
(2) the induced graph $G[N_u^C]$ is connected.
(3) there are no edges between N_u^C and $N_u - N_u^C$.

then ego-splitting with connected component reconstructs cluster C exactly.

Proof. The three conditions guarantee that for each $u \in C$, when we analyze the graph $G[N_u]$, the set N_u^C will be a connected component, so the local step of ego-splitting will create a persona of u associated with N_u^C. We name this persona u_C.

For each $v \in N_u^C$, the personas u_C and v_C are connected in the personas graph. This component can't contain any other personas than u_C for $u \in C$, otherwise it would imply an edge from N_u^C to $N_u - N_u^C$. □

The following corollary follows directly from the proof of the previous lemma:

Corollary 5.7. *If the induced graph $G[N_u^C]$ and the induced graph $G[C]$ is connected for all C and all $u \in C$, then each connected component of the personas graph corresponds to a cluster $C \in \mathcal{S}$ or to an union of clusters in $\mathcal{S}$. Moreover, under this condition, ego-splitting using connected components outputs at most k clusters.*

Proof. In the conditions of the corollary, each node u will have at most $k_u = |\{C; C \ni u\}|$ personas. We say that a persona of u is compatible with cluster C if the connected component of $G[N_u]$ corresponding to it contains N_u^C. Each persona is compatible with at least one cluster (but can be compatible with more than one). Now, by the same argument as in Lemma 5.6, for each fixed $C \in \mathcal{S}$, all personas compatible with cluster C are connected. Therefore, the personas graph has at most k connected components. □

Lemma 5.6 tells us that if all induced graphs $G[C]$ and $G[N_u^C]$ are connected, there is only one source of errors: edges from N_u^C to $N_u - N_u^C$. We say that an edge (v, w) is bad for u, C if $u, v \in C$, $v \in N_u^C$, $w \in N_u - N_u^C$ and (v, w) is in G.

Lemma 5.8. *Assuming $kq \geq 1$, given $u, v, w \in V$ and C, an edge (v, w) is bad for u, C with probability $O(k^2q^6p^3)$.*

Proof. We need $u, v \in C$, $v \in N_u^C$ and $w \notin C$ which happens with probability $pq^2(1-q)$. Now, to bound the probability that edges (v, w) and (u, w) are in the graph we take the union bound over the following event that there are clusters $C', C'' \neq C$ such that $u, v \in C'$, $v, w \in C''$, (u, w) is added to $G_{C'}$ and (v, w) is added to $G_{C''}$.

- if $C' = C''$ we have that the probability that $u, v, w \in C'$ and edges (u, v) and (v, w) are added is: q^3p^2.
- if $C' \neq C''$ we have that the probability that $u, v \in C'$, $v, w \in C''$ and edges (u, v) and (v, w) are added are: q^4p^2.

So taking the union bound we get that the probability of a bad event is $O(pq^2(1-q)[kq^3p^2 + k^2q^4p^2]) = O(k^2q^6p^3)$. □

Proof of Theorem 5.1. Let D denote the event that $G[C]$ and $G[N_u^C]$ are connected for every $C \in \mathcal{S}$ and $u \in C$. The probability of $\mathbb{P}[D]$ is bounded by Lemma 5.5.

If edge (v, w) is bad for u, C we say that a bad edge event occurred for (v, w, u, C). Lemma 5.8 bounds the probability of each bad edge event. So if B is a random variable measuring the total number of bad edge events, then: $\mathbb{E}[B] = O(n^3k^3q^6p^3)$ by Corollary 5.7, conditioned on D, $|\mathcal{S}'| \leq |\mathcal{S}|$. Also, each bad edge can cause at most two clusters to be merged, therefore: $|S'| \geq |S| - 2B$. This implies a bound on the size of the union $|S \cup \mathcal{S}'| = |S| + |S'| - |S \cap S'| \leq 2k - (k - 2B) = k + 2B$. Now, we can bound the Jaccard similarity as:

$$J(\mathcal{S}, \mathcal{S}') = \frac{|\mathcal{S} \cap \mathcal{S}'|}{|\mathcal{S} \cup \mathcal{S}'|} \geq \frac{k - 2B}{k + 2B} = 1 - \frac{4B}{k + 2B} \geq 1 - \frac{4B}{k}$$

Computing expectations:

$$\mathbb{E}[J(\mathcal{S}, \mathcal{S}')] \geq \mathbb{E}[J(\mathcal{S}, \mathcal{S}')|D]\mathbb{P}[D] = \left(1 - \frac{4\mathbb{E}[B|D]}{k}\right)\mathbb{P}[D]$$

Using that $\mathbb{E}[B|D]\mathbb{P}[D] \leq \mathbb{E}[B]$ and substituting the bounds for $\mathbb{E}[B]$ and $\mathbb{P}[D]$ we get the desired result. □

Remarks about the model. We can also extend our model to consider edges from two nodes that don't share a cluster. The usual way to incorporate this to the model is to add an edge between any two nodes with some probability r. We defer this discussion to the full version of the paper.

6 EXPERIMENTS

6.1 Experimental setup

We implemented our framework using a large-scale distributed computing infrastructure based on MapReduce.

Clustering algorithms. Our framework can use any non-overlapping clustering algorithm to partition the ego-nets and the persona graph (the two algorithms need not to be the same). For our experimental evaluation we used iterative label propagation clustering algorithms as non-overlapping partitioners in both phases. This choice is motivated by two reasons. First, label propagation algorithms are highly scalable and can be easily implemented in distributed settings. Second, previous ego-net based works have used successfully label propagation algorithms [13, 17].

We use a standard non-overlapping label propagation method based on the Absolute Potts Model technique [37]. This is the same algorithm that was used in [17] to cluster ego-nets so we omit its description here for lack of space. In our experiments in this paper we set the parameter $\alpha = 0.1$ (the penalty for missing a neighbor with a certain label). During the ego-net clustering phase we apply an in-memory version of this algorithm to cluster the ego-net. For the larger datasets, during the persona graph clustering step, the graph may not fit in memory, so we use a distributed variant of the algorithm. In this variant each label update iteration is carried out in parallel and we set $\alpha = 0$.

Pre-processing and post-processing. We use the following two heuristics that improves both the scalability and the accuracy of our methods.

First, we preprocess the graphs to restrict our analysis to at most 2000 neighbors of each node. If a neighbor v of u is not processed in the ego-net of u we discard the edge in the persona graph that corresponds to (u, v). This, besides increasing scalability, also improves the accuracy of the algorithms as high degree nodes are usually hubs connecting multiple communities so using them in the ego-nets can confuse the community structure.

Second, we post-process the overlapping communities produced by the algorithm discarding communities of size at most 4. This is because small communities are less informative. Notice that previous work [13] used a more complex post-processing of the output communities which was $O(n^2)$ while our post-processing is straightforward and fast.

6.2 Comparison with other algorithms

We compare our method with a state of the art ego-net based overlapping clustering algorithm **DEMON** [13]. For this method we use code provided by the authors and we set the parameter $\epsilon = 1$ in the post-processing as suggested (i.e. overlapping clusters are merged if and only if one is subset of the other). Consistently with our method we discard communities of size at most 4.

Table 1: Statistics on the real-world graphs used

Graph	Nodes	Edges	Persona nodes
amazon [42]	334,863	1,851,744	799,080
dblp [42]	317,080	2,099,732	611,068
livejournal [42]	3,997,962	69,362,378	22,970,759
orkut [42]	3,072,441	234,370,166	69,030,8073
friendster [42]	65,608,366	3,612,134,270	1,492,532,217
clueweb12 [9, 10]	955,207,488	67,823,249,559	55,433,497,920

As a baseline, we also used an off-the-shelf distributed overlapping label propagation algorithm (henceforth **OLP**). In this method nodes are allowed to retain up to k most frequent labels. Then a node is assigned to all the at most k clusters defined by the nodes that have retained a label. We fix $k = 3$ in our experiments and keep communities of size at least 5.

6.3 Datasets

Synthetic benchmarks. In the previous section we have shown how our method can provably reconstruct highly overlapping communities in a simple stylized generative model. For our experimental evaluation we employ instead random graphs with planted overlapping clusters produced by the more widely used and sophisticated model of Lanchinetti et al. [26]. We chose this model for consistency with previous ego-net based works [13] and because the model replicates several properties of real-world graphs, such as power law distribution of degrees, varying community sizes and membership of nodes in varying community number.

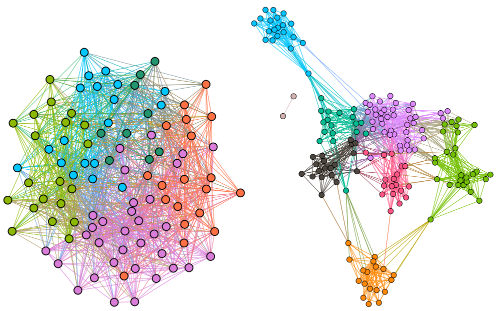
We refer to [26] for a detailed description of the model. With the code provided online by the authors we generated 3 set of graphs referred as Benchmark-0.01, Benchmark-0.1 and Benchmark-0.3, respectively. Each set of graphs contains 10 random instantiation of the model with the same settings, we report averages over those 10 graphs for all algorithms. The Benchmark-0.01 consist of the same settings of [13] paper which we report here for convenience (`N=1000, k=25, maxk=50, mu=0.01, minc=20, maxc=50, on=500, om=3`) for Benchmark-0.1 and Benchmark-0.3 instead have the following settings (`N=1000, k=10, maxk=50, minc=5, maxc=50, on=100, om=2`) and `mu` set to 0.1 and 0.3, respectively.

Real-world graphs. We analyzed a set of widely used graphs (amazon, dblp, livejournal, orkut, friendster) that have ground truth clusters and are available from the *SNAP* collection. For these graphs, we use the dataset with top quality communities as ground-truth (more details in [42]). The details of the graphs are reported in Table 1. In this table we also report the number of persona nodes identified by our framework using label propagation as clustering algorithm.

We also run our algorithms on the clueweb12 [9, 10] graph which is web graph with tens of billions of edges but for which have no ground truth clusters. All graphs are made undirected ignoring the direction of the edges if present.

6.4 Results on synthetic benchmarks

Example persona graph. To gain an insight on how our framework operates we first provide a visualization of a smaller synthetic graph produced with Lanchinetti et al. model. For sake of visualization we used a graph with only 100 nodes and 9 highly overlapping



(a) Original Graph (b) Persona Graph

Figure 4: Color visualization of a small synthetic graph and its persona graph.

Table 2: Accuracy in synthetic benchmarks

	Ego-splitting		DEMON		OLP	
Graph	F1	NMI	F1	NMI	F1	NMI
Benchmark-0.01	**0.9368**	**0.9403**	0.4765	0.1670	0.6254	0.3149
Benchmark-0.1	**0.7878**	**0.7100**	0.1200	0.0000	0.7723	0.5571
Benchmark-0.3	**0.6714**	**0.5076**	0.1216	0.0000	0.6151	0.4405

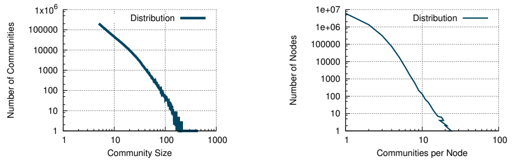
ground-truth communities. Figure 4 shows the both original graph and the persona graph output by our method, as plotted by the standard Gephi [8] tool with the same visualization setting. Both graphs have 1269 edges while the persona graph has 164 nodes (from the 100 nodes in the original graph). The colors represent the communities identified by Gephi on both graphs based the standard non-overlapping modularity-based algorithm of the tool. It is possible to observe that while the community structure in the first graph is not immediately clear, in the second the clusters are visibly more separated. In fact in the first graph Gephi found only 5 communities with a modularity of 0.25 while in the second it found 8 communities with a higher modularity of 0.60. This visually shows how our framework is able to disentangle the communities.

Quantitative analysis. We now provide a more quantitative analysis of the accuracy of our algorithm in reconstruct the ground truth communities in the benchmark graphs. The results are shown in Table 2. The NMI is computed using publicly available implementation of [34], for F_1 we use distributed computing to scale the computation to large community outputs.

It is possible to observe our method using label propagation as clustering algorithm outperforms all other benchmark methods in all the graphs evaluated and in both accuracy measures. Notice how in particular for the benchmark graph Benchmark-0.01 reported in [13] we get close to > 93% F_1 compared to 47% of DEMON [13].

6.5 Results on real graphs

We first report statistics on the communities identified by our algorithm. Figure 5(a) shows the distribution of the size of the overlapping communities identified by our algorithm on the largest social network analyzed Friendster. Notice how the distribution of community sizes follows a clear heavy tailed distribution (the plot is in log scale, the other graphs have similar distributions). This is consistent with previous results in social networks [35]. Consistent with previous studies [28] most communities have < 100 nodes. In total our algorithm outputs the following number of communities in the graphs analyzed: amazon 27,004; dblp 33,626; livejournal 49,954; orkut 131,773; friendster 905,520; clueweb 31,642,414. Figure 5(b) shows for the same graph the distribution of the number of communities to which a node belongs in our output. Notice that a large fraction of nodes belong to more than one community (∼ 30% in Friendster) and that again the distribution of participation of nodes in multiple communities is heavy tailed.



(a) Friendster: Size of the Communities

(b) Friendster: Communities per Node

Figure 5: Statistics on the overlapping communities output.

Table 3: Accuracy in real-world graphs

	Ego-splitting		DEMON		OLP	
Graph	F1	NMI	F1	NMI	F1	NMI
amazon	**0.0374**	**0.0809**	0.0337	0.0310	0.0339	0.0450
dblp	**0.1662**	**0.1041**	0.1539	0.0309	0.1448	0.0645
livejournal	**0.0490**	**0.0394**	-	-	0.0115	0.0148
orkut	**0.0332**	0.0060	-	-	0.0267	**0.0129**
friendster	**0.0051**	**0.0008**	-	-	0.0010	0.0006

Accuracy. Finally, we report the accuracy of our method in real-world graphs with ground-truth communities. The results are in Table 3. Consistently with the results in the random graphs our method outperforms the other two methods in almost all cases. In particular our method has always the highest F_1 score and in all but one case it has the best NMI score. This confirms our theoretical results that shows that splitting the ego's in persona allows the overlapping community structure to be more easily detectable.

Results for DEMON on large graphs are omitted as the algorithm did not finish to run in the allocated time (the method employed a slow post-processing which is not scalable in large graphs).

6.6 Scalability

Finally we evaluate the scalability of our method. In Figure 6 we show the relationship between the total running time our algorithm (total wall-clock time of the distributed execution) and the size of the graph. We report the results as a ratio of the time used to process a graph (resp. number of edges of the graph) and the time used to process our smaller graph amazon (resp. number of edges in amazon). It is possible to observe the high scalability of our method. Even if clueweb has 30000 times more edges the execution

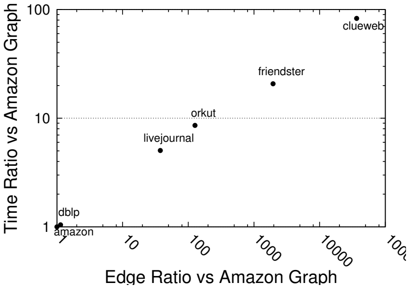


Figure 6: Running time vs size of the graph

time is only 80 times longer that of amazon. This shows that our method can scale to very large graphs with billions of nodes and edges.

7 CONCLUSIONS AND FUTURE WORK

We presented a scalable and flexible framework for finding overlapping clusters. Our empirical and theoretical findings shows that, using the local clustering structure of the ego-nets as guidance, it is possible to disentangle the complex and highly overlapping community structures of real-networks by splitting nodes into their "personas".

For future work, we would like to establish theoretical guarantees in more nuanced models by using more sophisticated partitioning algorithms. Another important direction is to adapt our methods to incremental models of computation, since real world networks are dynamic. Recent work [41] has observed a rich structure in the overlap of communities which could be analyzed to further improve our method. Finally, we believe further analysis of the structural properties of the persona graphs could yield other insights on the social network besides its clustering such as, for instance, the roles of actors in a social network [16].

ACKNOWLEDGMENTS

We thank Sergei Vassilvitskii and Bryan Perozzi for their comments and Bryan for his help with graph visualization. We thank Michele Coscia, Giulio Rossetti, Dino Pedreschi and Fosca Giannotti for making their code available and for their helpful comments.

REFERENCES

[1] B. D. Abrahao, S. Soundarajan, J. E. Hopcroft, and R. D. Kleinberg. A separability framework for analyzing community structure. *TKDD*, 2014.

[2] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.

[3] L. Akoglu, M. McGlohon, and C. Faloutsos. oddball: Spotting anomalies in weighted graphs. In *PAKDD 2010*, 2010.

[4] M. Amoretti, A. Ferrari, P. Fornacciari, M. Mordonini, F. Rosi, and M. Tomaiuolo. Local-first algorithms for community detection. In *KDWeb 2016*, 2016.

[5] S. Arora, R. Ge, S. Sachdeva, and G. Schoenebeck. Finding overlapping communities in social networks: toward a rigorous approach. In *EC*, 2012.

[6] M. Balcan, C. Borgs, M. Braverman, J. T. Chayes, and S. Teng. Finding endogenously formed communities. In *SODA 2013*.

[7] S. Bandyopadhyay, G. Chowdhary, and D. Sengupta. FOCS: fast overlapped community search. *IEEE Trans. Knowl. Data Eng.*, 27(11):2974–2985, 2015.

[8] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks, 2009.

[9] P. Boldi, M. Rosa, M. Santini, and S. Vigna. Layered label propagation: a multiresolution coordinate-free ordering for compressing social networks. In *WWW*, 2011.

[10] P. Boldi and S. Vigna. The WebGraph framework I: Compression techniques. In *WWW 2004*, pages 595–601, Manhattan, USA, 2004. ACM Press.

[11] R. Burt. *Structural Holes: The Social Structure of Competition.* Harvard Press, 1995.

[12] N. Buzun, A. Korshunov, V. Avanesov, I. Filonenko, I. Kozlov, D. Turdakov, and H. Kim. Egolp: Fast and distributed community detection in billion-node social networks. In *2014 IEEE ICDM Workshops*, pages 533–540, 2014.

[13] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi. Uncovering hierarchical and overlapping communities with a local-first approach. *TKDD*, 2014.

[14] A. Delis, A. Ntoulas, and P. Liakos. Scalable link community detection: A local dispersion-aware approach. In *IEEE BigData 2016*, pages 716–725, 2016.

[15] R. I. M. Dunbar and S. G. B. Roberts. Communication in social networks: Effects of kinship, network size and emotional closeness. *Personal Relationships*, 2010.

[16] D. Easley and J. Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world.* Cambridge University Press, 2010.

[17] A. Epasto, S. Lattanzi, V. Mirrokni, I. O. Sebe, A. Taei, and S. Verma. Ego-net community mining applied to friend suggestion. *VLDB*, 9(4):324–335, 2015.

[18] M. Everett and S. P. Borgatti. Ego network betweenness. *Social Networks*, 2005.

[19] S. Fortunato. Community detection in graphs. *Physics reports*, 2010.

[20] L. C. T. Freeman. Centered graphs and the structure of ego networks. *Mathematical Social Sciences*, 1982.

[21] M. Girvan and E. J. Newman. Community structure in social and biological networks. *PNAS*, 2002.

[22] L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray. From molecular to modular cell biology. *Nature*, 402:C47–C52, 1999.

[23] K. He, Y. Sun, D. Bindel, J. E. Hopcroft, and Y. Li. Detecting overlapping communities from local spectral subspaces. In *IEEE ICDM*, pages 769–774, 2015.

[24] R. Khandekar, G. Kortsarz, and V. S. Mirrokni. On the advantage of overlapping clusters for minimizing conductance. *Algorithmica*, 69(4):844–863, 2014.

[25] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Random graph models for the web graph. In *FOCS*, 2000.

[26] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):016118, 2009.

[27] S. Lattanzi and D. Sivakumar. Affiliation networks. In *STOC 2009*.

[28] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 2009.

[29] J. Leskovec, K. J. Lang, and M. W. Mahoney. Empirical comparison of algorithms for network community detection. In *WWW 2010*.

[30] R. Li, C. Wang, and K. C. Chang. User profiling in an ego network: co-profiling attributes and relationships. In *WWW*, 2014.

[31] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology*, 58(7):1019–1031, 2007.

[32] W. Liu, X. Jiang, M. Pellegrini, and X. Wang. Discovering communities in complex networks by edge label propagation. *Scientific reports*, 6, 2016.

[33] J. J. McAuley and J. Leskovec. Learning to discover social circles in ego networks. In *NIPS*, 2012.

[34] A. F. McDaid, D. Greene, and N. Hurley. Normalized mutual information to evaluate overlapping community finding algorithms. Oct. 2011.

[35] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 2005.

[36] B. S. Rees and K. B. Gallagher. Overlapping community detection by collective friendship group inference. In *ASONAM*, 2010.

[37] P. Ronhovde and Z. Nussinov. Local resolution-limit-free potts model for community detection. *Phys. Rev. E*, 2010.

[38] S. Wasserman and K. Faust. *Social network analysis: methods and applications.* Cambridge University Press, 1994.

[39] J. Weng and B.-S. Lee. Event detection in twitter. *ICWSM*, 11:401–408, 2011.

[40] J. J. Whang, D. F. Gleich, and I. S. Dhillon. Overlapping community detection using seed set expansion. In *ACM CIKM'13*, pages 2099–2108, 2013.

[41] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM, 2013.

[42] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

[43] J. Yang, J. J. McAuley, and J. Leskovec. Detecting cohesive and 2-mode communities indirected and undirected networks. In *WSDM*, 2014.