Exorcising the Demon: Angel, Efficient Node-centric Community Discovery

Giulio Rossetti

KDD Lab, ISTI-CNR, Pisa, Italy
giulio.rossetti@isti.cnr.it

Abstract. Community discovery is one of the most challenging tasks in social network analysis. During the last decades, several algorithms have been proposed with the aim of identifying communities in complex networks, each one searching for mesoscale topologies having different and peculiar characteristics. Among such vast literature, an interesting family of Community Discovery algorithms, designed for the analysis of social network data, is represented by overlapping, node-centric approaches. In this work, following such line of research, we propose Angel, an algorithm that aims to lower the computational complexity of previous solutions while ensuring the identification of high-quality overlapping partitions. We compare Angel, both on synthetic and real-world datasets, against state of the art community discovery algorithms designed for the same community definition. Our experiments underline the effectiveness and efficiency of the proposed methodology, confirmed by its ability to constantly outperform the identified competitors.

Keywords: Complex Network Analysis, Community Discovery

1 Introduction

Community discovery (henceforth CD), the task of decomposing a complex network topology into meaningful node clusters, is allegedly oldest and most discussed problem in complex network analysis [3, 6]. One of the main reasons behind the attention such task has received during the last decades lies in its intrinsic complexity, strongly tied to its overall ill-posedness. Indeed, one the few universally accepted axioms characterizing this research field regards the impossibility of providing a single shared definition of what community should look like. Usually, every CD approach is designed to provide a different point of view on how to partition a graph: in this scenario, the solutions proposed by different authors were often proven to perform well when specific assumptions can be made on the analyzed topology. Nonetheless, decomposing a complex structure in a set of meaningful components represents *per se* a step required by several analytical tasks – a need that has transformed what usually is considered a problem definition weakness, the existence of multiple partition criteria, into one of

its major strength. Such peculiarity has lead to the definition of several "meta" community definitions, often tied to specific analytical needs. Classic works intuitively describe communities as sets of nodes closer among them than with the rest of the network, while others, only define such topologies as dense network subgraphs. A general, high-level, formulation of the Community Discovery problem definition is following:

Definition 1 (Community Discovery (CD)). *Given a network G, a community C is a set of distinct nodes: $C = \{v_1, v_2, \ldots, v_n\}$. The community discovery problem aims to identify the set $\mathcal{C}$ of all the communities in G.*

In this work, we introduce a CD algorithm, ANGEL, tailored to extract overlapping communities from a complex network. Our approach is primarily designed for social networks analysis and belongs to a well-known sub family of Community Discovery approaches often identified by the keywords *bottom-up* and *node-centric* [18]. ANGEL aims to provide a fast way to compute reliable overlapping network partitions. The proposed approach focuses on lowering the computational complexity of existing methods proposing scalable sequential – although, easily parallelizable – solutions to a very demanding task: overlapping network decomposition.

The paper is organized as follows. In Section 2 we introduce ANGEL. There we discuss its rationale, the properties it holds as well as its computational complexity. In Section 3 we evaluate the proposed method on both synthetic and real-world datasets for which ground truth communities are known in advance. To better discuss the resemblance of ANGEL partitions to ground truth ones as well as its execution times, we compare the proposed method with state-of-art competitors sharing the same rationale. Finally, in Section 4 the literature relevant to our work is discussed and Section 5 concludes the paper.

2 Angel

In this section, we present our bottom-up solution to the community discovery problem: ANGEL[1]. Our approach, as we will discuss, follows a well-known pattern composed by two phases: i) construction of local communities moving from ego-network structures and, ii) definition of mesoscale topologies by aggregating the identified local-scale ones. Since ANGEL main goal is reducing the computational complexity of previous node-centric approaches, we will detail the merging strategy it implements to build up the final community partition and, finally, we will discuss its properties and study its complexity.

Algorithm Rationale. The algorithmic schema of ANGEL is borrowed from the DEMON [4] one, an approach whose main goal was to identify local communities capturing individual nodes perspectives on their neighborhoods and to use them to build mesoscale ones.

[1] Code available at: https://github.com/GiulioRossetti/ANGEL

ALGORITHM 1: ANGEL

Input: $\mathcal{G} : (V, E)$, the graph; ϕ, the merging threshold.
Output: $\mathcal{C}$ a set of overlapping communities.

```
1  for v ∈ V do                                                    // Step #1
2      e ← EgoMinusEgo(v, G) ;                                     // Step #2
3      C(v) ← LabelPropagation(e) ;                                // Step #3
4      C ← C ∪ C(v)
5  ncoms = |C|
6  acoms = 0
7  while ncoms != acoms do                                         // Step #4
8      acoms = ncoms
9      C ← DecreasingSizeSorting(C) ;                              // Step #5
10     for c ∈ C do
11         C ← PrecisionMerge(c, C, φ) ;                           // Step #6
12     ncoms = |C|
13 return C
```

ANGEL takes as input a graph G, a merging threshold ϕ and an empty set of communities $\mathcal{C}$. The main loop of the algorithm cycles over each node, so to generate all the possible points of view of the network structure (Step #1 in Algorithm 1). To do so, for each node v, it applies the $EgoMinusEgo(v, \mathcal{G})$ (Step #2 in Algorithm 1) operation as defined in [4]. Such function extracts the ego-network centered in the node v – e.g., the graph induced on $\mathcal{G}$ and built upon v and its first order neighbors – then removes v from it, obtaining a novel, filtered, graph substructure. ANGEL removes v since, by definition, it is directly linked to all nodes in its ego-network, connections that would lead to noise in the identification of local communities. Obviously, a single node connecting the entire sub-graph will make all nodes very close, even if they are not in the same local community. Once obtained the ego-minus-ego graph, ANGEL computes the local communities it contains (Step #3 in Algorithm 1). The algorithm performs this step by using a community discovery algorithm borrowed from the literature: Label Propagation (LP)[13]. This choice, as in [4], is justified by the fact that: (i) LP has low algorithmic complexity ($\sim O(N)$, with N number of nodes), and, (ii) it returns results of a quality comparable to more complex algorithms[3].

Reason (i) is particularly important since Step #3 of ANGEL needs to be performed once for every node of the network, thus making unacceptable to spend a super-linear time for each node. Notice that instead of LP any other community discovery algorithm (both overlapping or not) can be used (impacting both on the algorithmic complexity and partition quality). Given the linear complexity (in the number of nodes of the extracted ego-minus-ego graph) of Step #3, we refer to this as the inner loop for finding the local communities.

Due to the importance of LP for our approach and to shed lights on how it works we briefly describe its classical formulation [13]. Suppose that a node v has neighbors $v_1, v_2, ..., v_k$ and that each one of them carries a label denoting the community that it belongs to: then, at each iteration the label of v is updated to the majority label of its neighbors. As the labels propagate, densely connected groups of nodes quickly reach a consensus on a unique label. At the end of the propagation process, nodes with the same labels are grouped as one

ALGORITHM 2: PrecisionMerge

Input: x, a community; $\mathcal{C}$, a set of overlapping communities; ϕ, the merging threshold.
Output: $\mathcal{C}$, a set of overlapping communities.

```
1 com_to_freq ← community_frequency(x) ;                    // Step #A
2 for com, freq ∈ com_to_freq do
3     if freq/|x| ≥ φ then                                  // Step #B
4         C = C − {x, com}
5         C = C ∪ {x ∪ com}
6 return C
```

community. In case of bow-tie situations – e.g., a node having an equal maximum number of neighbors in two or more communities – the classic definition of the LP algorithm randomly selects a single label for the contended node. Angel, conversely, handle this situation allowing soft community memberships, thus producing deterministic local partitions.

The result of Steps #1-3 of Algorithm 1 is a set of local communities $\mathcal{C}(v)$, according to the perspective of a specific node, v, of the network. Conversely, from what done in Demon, Angel does not reintroduce the ego in each local community to reduce the noisy effects hubs play during the merging step. Local communities are likely to be an incomplete view of the real community structure of $\mathcal{G}$. Thus, the result of Angel needs further processing: namely, to merge each local community with the ones already present in $\mathcal{C}$.

Once the outer loop on the network nodes is completed, Angel leverage the PrecisionMerge function to compact the community set $\mathcal{C}$ so to avoid the presence of fully contained communities in it. Such function (Step #6, detailed in Algorithm 2) implements a deterministic merging strategy and is applied iteratively until reaching convergence (Step #4) – e.g., until the communities in $\mathcal{C}$ cannot be merged further. To assure that all the possible community merges are performed at each iteration $\mathcal{C}$ is ordered from the smallest community to the biggest (Algorithm 1, #Step 6).

This merging step is a crucial since it needs to be repeated for each of the local communities. In Demon such operation requires the computation for each pair of communities (x, y), $x \in \mathcal{C}(v)$ and $y \in \mathcal{C}$, of an overlap measure (i.e. Jaccard index) and to evaluate if its value overcome a user defined threshold. This approach, although valid, has a major drawback: given a community $x \in \mathcal{C}(v)$ it requires $O(|\mathcal{C}|)$ evaluations to identify its best match among its peers. Indeed, such kind of strategy represents a costly bottleneck requiring an overall $O(|\mathcal{C}|^2)$ complexity while applied to all the identified local communities.

Angel aims to drastically reduce such computational complexity by performing the matches leveraging a greedy strategy.

To do so, it proceeds in the following way:

i) Angel assumes that each node carries, as additional information, the identifiers of all the communities in $\mathcal{C}$ it already belongs to;
ii) in Step #A (Algorithm 2) for each local community x is computed the frequency of the community identifiers associated with its nodes;

iii) in Step #B, for each pair (*community_id*, $frequency$) is computed its Precision w.r.t. x, namely the percentage of nodes in x that also belong to *community_id*;
iv) iff the precision ratio is greater (or equal) than a given threshold ϕ the local community x is merged with *community_id*: their union is added to $\mathcal{C}$ and the original communities are removed from the same set.

Operating in this way it is avoided the time expensive computation of community intersections required by Jaccard-like measures since all the containment testing can be done in place.

Angel Properties. The proposed approach posses two nice properties: it produces a deterministic output (once fixed the network G and threshold ϕ), and it allows for a parallel implementation.

Property 1 (Determinism) *There exists a unique* $\mathcal{C}$=Angel(G, ϕ) *for any given* G *and* ϕ*, disregarding the order of visit of the nodes in* G*.*

To prove the determinism of Angel it is mandatory to break its execution in two well-defined steps: (i) local community extraction and (ii) merging of local communities.

i) Local communities: Label Propagation identifies communities by applying a greedy strategy. In its classical formulation [13] it does not assure convergence to a stable partition due to the so-called "label *ping-pong* problem" (i.e., instability scenario primarily due to bow-tie configurations. However, as already discussed, Angel addresses such problem by relaxing the node single label constraint thus allowing for the identification of a stable configuration of overlapping local communities.
ii) Merging: this step operates on a well-determined set of local communities on which the PrecisionMerge procedure is applied iteratively. Since we explicitly impose the community visit ordering the determinism of the solution is given by construction.

Property 2 (Compositionality) Angel *is easily parallelizable since the local community extraction can be applied locally on well defined subgraphs (i.e., ego-minus-ego networks).*

Given a graph $G = (V, E)$ it is possible to instantiate Angel local community extraction simultaneously on all the nodes $u \in V$ and then apply the PrecisionMerge recursively in order to reduce and compact the final overlapping partition:

$$Angel(G, \phi) = PMerge(\bigcup_{u \in V} LP(EME(u))) \quad (1)$$

The underlying idea is to operate community merging only when all the local communities are already identified (i.e., LabelPropagation is applied to all the ego-minus-ego of the nodes $u \in V$ – LP(EME(u)) in Equation 1 – as shown in Figure 1). Moreover, this parallelization schema is assured to produce the

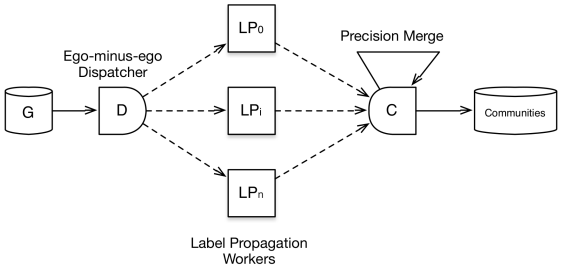


Fig. 1: ANGEL parallelization schema. The graph G is decomposed in $|V|$ ego-minus-ego network by a dispatcher D and distributed to n workers $\{LP_0, \ldots, LP_n\}$ that extract local communities from them. At the end of such parallel process, a collector C iteratively apply PRECISIONMERGE till obtaining the final overlapping partition.

same network partition obtained by the original sequential approach due to the determinism property.

Angel Complexity. To evaluate the time complexity we proceed by decomposing ANGEL in its main components. Given the pseudocode description provided in Algorithm 1 we can divide our approach into the following sub-procedures:

i) Outer loop (lines 3-6): the algorithm cycles over the network nodes to extract the ego-minus-ego networks and identify local communities. This main loop has thus complexity $\mathcal{O}(|V|)$.
ii) Local Communities extraction: the Label Propagation algorithm has complexity $\mathcal{O}(n+m)$ [13], where n is the number of nodes and m is the number of edges of the ego-minus-ego network. Let us assume that we are working with a scale free network, whose degree distribution is $p_k = k^{-\alpha}$: in this scenario the majority of the identified ego-minus-ego networks are composed by $n << |V|$ nodes and $m << |E|$ edges, thus the average complexity of each iteration will be $\mathcal{O}(n+m) << \mathcal{O}(|V|+|E|)$.
iii) PRECISIONMERGE final cycle (lines 9-14): for each local community ANGEL evaluate if it can be merged with one or more previously identified substructures. In order efficiently implement this task we assume that once identified a community a new identifier is generated and assigned to all the nodes within it. All the nodes will then have attached multiple labels (one representing an identifier of a community the node belongs to). Given a community x the PRECISIONMERGE function (Algorithm 2) leverage such information to compute – for each community identifier y attached to the nodes in x – the ratio of nodes in it that already belongs to y w.r.t. the size of x. If the ratio is greater than (or equal to) a given threshold, the merge is applied and the node label updated. This step can be performed with constant complexity, $\mathcal{O}(1)$, employing an hash-map. Considering the complete loop the overall cost is thus given by the initial sorting of the communities

by decreasing size, $\mathcal{O}(|C|log|C|)$ (where C is the community set), and the evaluation of PrecisionMerge on each community in C, $\mathcal{O}(|C|)$. Moreover, we can assume the number of iteration $k << |C|$ since at each step the number of communities decreases: thus we can consider k as a constant factor giving as final complexity, $\mathcal{O}(|C|log|C|) + \mathcal{O}(|C|) = \mathcal{O}(|C|log|C|)$.

Such sub procedures gives us a final complexity of $\mathcal{O}(|V|(n+m)) + \mathcal{O}(|C|log|C|)$: considering a scale free network, for which we can reasonably expect $|V| >> (n+m)$ and $|V| > |C|$, the final complexity can be approximated as $\mathcal{O}(|V|)$.

3 Evaluation

Evaluating a community discovery approach is not an easy task due to the overall ill-posedness of the problem. In this section we propose a two-stage evaluation, focusing both on underlining Angel efficiency – in terms of scalability and running time – as well as on its ability to retrieve ground truth communities. As a first step, we identify the competitors of our algorithm, approaches that share with it the same rationale. After that, we briefly describe the quality function we adopt to compare the partition produced by the selected algorithms and to assess their resemblance w.r.t. ground truth communities. Finally, we evaluate Angel and its competitors on two different community resemblance tasks: (i) identification of planted ground truth partition in synthetically generated networks, and (ii) identification of annotated communities in real-world datasets.

Competitors. To evaluate Angel performances, we compare it with state-of-art competitors having a similar rationale[2][3].
(i) Demon [4] is an incremental and limited time complexity algorithm for community discovery. It extracts ego networks, i.e., the set of nodes connected to an ego node u, and identifies the real communities by adopting a democratic, bottom-up merging approach of such structures.
(ii) PanDemon [1] is a parallel implementation of Demon designed to increase its scalability and to reduce the computational its complexity.
(iii) NodePerception. In [19] the authors propose a generalization of the Demon approach: NodePerception instantiate the local two-phase schema leveraging different a CD approaches in the local community extraction phase, allowing for the identification of partitions that optimize specific quality functions.
(iv) SLPA. In [21] is introduced SLPA, an overlapping hierarchical community discovery algorithm designed for large-scale networks. SLPA leverages a label propagation strategy built upon dynamic interaction rules.

The former three approaches move from the same algorithmic schema of Angel. They all are *node-centric* algorithms [18] that, moving from the analysis of

[2] All the algorithms were executed on a Linux 4.4.0 machine with an Intel Core i7-5820 CPU @3.3GHzx16 and 32GB of RAM.
[3] All algorithms have been integrated within the CDlib python library [15] https://github.com/GiulioRossetti/cdlib

	$\|V\|$	$\|E\|$	$\|C\|$	CC	d
emailEu	1,005	25,571	42	0.3994	7
Amazon	334,863	925,872	75,149	0.3967	44
dblp	317,080	1,049,866	13,477	0.6324	21
Youtube	1,134,890	2,987,624	8,385	0.0808	20

Table 1: **Datasets Statistics.** Number of nodes, edges, ground truth communities, average clustering coefficient and diameter for the analyzed datasets.

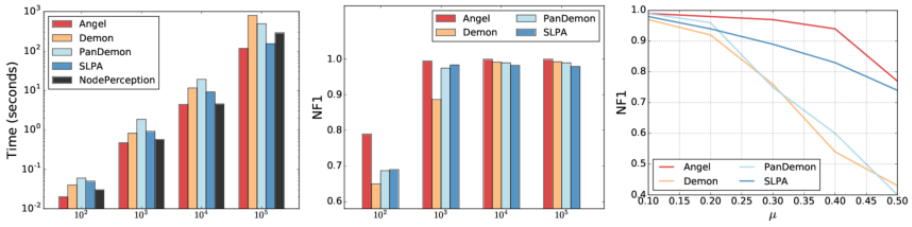
ego-networks, generate overlapping partitions while following a non-deterministic approach. Conversely, the latter competitor, SLPA, represents a fast implementation of the label propagation algorithm used by ANGEL to identify ego-network local communities.

Synthetic benchmarks. To evaluate how ANGEL behave under specific, controlled, settings we tested it, along with its competitors, against synthetic networks having planted ground truth communities generated through the LFR benchmark[4] [9]. The networks described by LFR have well-known characteristics: among the others, both their node degrees and community sizes follow a power law distributions. Moreover, similar to the *planted l-partition* model[2], LFR network vertices share a predefined fraction of their links with other vertices of their cluster. We generated multiple networks varying the following LFR parameters: (i) N, the network size (from 100 to 100k nodes);(ii) C, the network density (from 0.1 to 0.4, steps of 0.1); (iii) μ, the mixing coefficient describing the average per-node ratio between the number of edges to its communities and the number of edges with the rest of the network (from 0.1 to 0.5, steps of 0.1).

Real world data. To understand how the ANGEL and its competitors behave on real-world data, we tested them against four network datasets having annotated ground truth community structure[5]. We analyzed the following datasets (whose synthetic statistics are reported in Table 1):
(i) *emailEU.* Email exchange network among members of a large European research institution. The ground truth communities are members' departments.
(ii) *Amazon.* Network built using the *Customers Who Bought This Item Also Bought* Amazon feature. Product categories define ground-truth communities.
(iii) *dblp.* Co-authorship network where two authors are connected if they publish at least one paper together. Publication venue define ground-truth communities.
(iv) *Youtube.* Subgraph of the Youtube social network. User-defined groups identify ground-truth communities.

Conversely, from synthetic benchmarks, where the planted communities respect specific topological characteristics, real data annotation provides a semantic partition of network nodes. Since none of the considered algorithms is pa-

[4] Code available at https://sites.google.com/site/santofortunato/inthepress2
[5] Datasets available at https://snap.stanford.edu/data/.



(a) (b) (c)

Fig. 2: **Synthetic Benchmarks.** (a) Running time of the compared algorithms w.r.t. network size (number of nodes); (b) Avg. community resemblance score per network size; (c) Community resemblance varying LFR mixing coefficient (number of nodes 10^6). In (b-c) the NF1 scores for NodePerception are omitted due to their low values (NF1$\leq$0.2).

rameter free, in our analysis we instantiate each one of them multiple times performing a grid-search estimation of the optimal parameter values for each target network, thus ensuring that we compare their best performances. One way to asses the effectiveness of a CD algorithm is to compare how much the communities it identifies can provide a good approximation of a given ground truth partition. To do so we apply a community resemblance score proposed in [17], $NF1 = \frac{F1 * Coverage}{Redundancy} \in (0, 1]$. NF1 can be applied to both crisp/overlapping partitions. It is maximized when: (i) the average F1 (harmonic mean of precision and recall) is maximal (perfect match), (ii) the computed partition provide a complete coverage for the ground truth one (iii) the redundancy is minimized (i.e., each identified community is matched with a distinct ground truth one). As shown in [17] it is possible to compute F1 (and thus NF1) paying a linear complexity in the size of the community set X.

3.1 Experimental Results

Synthetic Benchmarks. In Fig. 2 we report the execution time and NF1 score for the compared CD approaches. Our experiments show that Angel is able

	Angel	Demon	PanDemon	NodePerception	SLPA
emailEu	3.53	4.72	2.34	9.91	2.42
Amazon	88.49	16862.61	3032.63	256.09	504.61
dblp	115.44	24273.36	1059.54	382.43	321.46
Youtube	2209.20	8362.28	4076.98	11533.74	2860.01

Table 2: **Running Times.** The execution times reported are expressed in seconds and do not include network loading and results serialization on file. PanDemon has been executed on 16 cores.

sensibly to improve the running times of its competitors while increasing the network size. In particular, it is worth noticing that Fig. 2(a) reports execution times on a log scale: considering the average runtime of ANGEL on the generated 100k nodes graphs it registers a speedup of an order of magnitude w.r.t. its competitors. In Fig.2(b-c) the NF1 score is used to compare the adherence of the partitions identified by the selected CD algorithms to the ground truth ones: we omitted NODEPERCEPTION's results since their overall NF1 were always lower than 0.4. In particular Fig. 2(b) compare the average NF1 scores obtained by each algorithm on different sized LFR graphs. To compute the NF1 mean value for the pair $< algorithm, network\_size >$ we considered the results provided by the optimal parameter configuration w.r.t. each network size instantiation (e.g., varying graph density and mixing coefficient). Among the compared methods ANGEL is always able to reach the highest scores, often producing the perfect match for the planted communities. Fig. 2(c) underline the impact of the LFR mixing coefficient on the quality of extracted communities once fixed the network size. We can observe that ANGEL and SLPA can assure relatively stable performances while varying μ.

Evaluation on Real World Data. Table 2 shows the running times – expressed in seconds – of the compared CD approaches when applied to the selected networks. Similarly to what observed in the synthetic scenario, ANGEL is constantly able to outperform its competitors, often reducing the execution times of one or more orders of magnitude. Differently from the synthetic scenario, when it comes to assessing community resemblance – quantitative values in Table 3 – we observe a relatively low quality for all the partitions produced by the compared algorithms. Indeed, such results are somehow expected. Conversely, from the synthetic benchmark where the planted communities were designed to follow specific topological characteristics, the semantic annotation provided for the analyzed real-world network do not necessarily reflect structural properties [8]. Such decoupling makes difficult, if not impossible, for CD algorithms that do not leverage semantic information to capture the same partition identified by the ground truth. However, even in this more complex scenario, ANGEL communities are the ones able to better approximate the provided ground truths. To provide a statistical significance bound to our experiments on real data we also applied

	Angel	**Demon**	**PanDemon**	**NodePerception**	**SLPA**
emailEu	0.51	0.20	0.04	0.12	0.23
Amazon	0.17	0.10	0.12	0.05	0.09
dblp	0.52	0.32	0.52	0.02	0.27
Youtube	0.19	0.08	0.08	0.04	0.18

Table 3: **Community Resemblance.** NF1 scores achieved by the compared algorithms on the real world datasets. For each model, we report the score achieved by its optimal parameter settings.

a Friedman test [7] with Li post-hoc [10] on the evaluation proposed in Table 3. The test was rejected for the NF1 scores with a p-value of 0.05, thus implying that the compared methods do actually behave differently when tested on multiple datasets. Moreover, the post-hoc underlined that ANGEL significantly outperforms NODEPERCEPTION under the same confidence interval, and all the others when p-value is imposed equal to 0.1.

4 Related Works

Community discovery is a widely discussed and studied problem able to attract the attention of heterogeneous communities. As already discussed, researchers continuously propose novel approaches with the aim of solving specific instantiation of this complex, and ill-posed, problem. Due to the massive literature available in this field, during the years several attempts were made to organize and cluster methods identifying some common grounds. Among the others, the surveys of Fortunato [6] and Coscia [3] propose complete, detailed and extensive taxonomies for classic algorithms. However, due to the peculiar problem definition, also more thematic surveys emerged, focusing for instance on overlapping [20], directed [11], node-centric [18] as well as dynamic community discovery [14].

Our algorithmic solutions has a very specific goal: efficiently identify overlapping network partitions following a bottom-up, node-centric, strategy. In literature, such strategy is often adopted while analyzing social network contexts [16], scenarios in which it is important to take into account the individual perspective on their local communities. ANGEL implements such model leveraging individual ego-networks to access the node-centric perspective of the analyzed social graph. Such strategy, originally proposed by [4] were also extended to parallel implementations, as in [1], and generalized in a high-level framework [19]. Finally, several approaches leverage the concept of ego-network as the basilar brick to propose heterogeneous community definitions [5]. Other common strategies to design node-centric approaches are the seed set expansion [12], and community diffusion ones [13].

5 Conclusion

In this paper, we introduced ANGEL, a node-centric approach to overlapping community discovery. ANGEL was designed with the aim of lowering the computational complexity of existing approaches while ensuring the identification of high-quality partitions. Experimental results, both on synthetic and real-world networks, highlight the efficiency and effectiveness of the proposed approach, underlying its ability to outperform its direct competitors. As future works, we plan to extend ANGEL to the analysis of dynamic network topologies.

Acknowledgment

This work is partially supported by the European Community's H2020 Program under the funding scheme "INFRAIA-1-2014-2015: Research Infrastructures" grant agreement 654024, http://www.sobigdata.eu, "SoBigData".

References

1. M. Amoretti, A. Ferrari, P. Fornacciari, M. Mordonini, F. Rosi, and M. Tomaiuolo. Local-first algorithms for community detection. In *KDWeb*, 2016.
2. A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116–140, 2001.
3. M. Coscia, F. Giannotti, and D. Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining*, 2011.
4. M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi. Demon: a local-first discovery method for overlapping communities. In *International conference on Knowledge discovery and data mining*, pages 615–623. ACM, 2012.
5. A. Epasto, S. Lattanzi, and R. Paes Leme. Ego-splitting framework: from non-overlapping to overlapping clusters. In *SIGKDD*, pages 145–154. ACM, 2017.
6. S. Fortunato. Community detection in graphs. *Physics reports*, 486(3), 2010.
7. M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32, 1937.
8. D. Hric, R. K. Darst, and S. Fortunato. Community detection in networks: Structural communities versus ground truth. *Physical Review E*, 90(6):062805, 2014.
9. A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78(4):046110, Oct. 2008.
10. J. D. Li. A two-step rejection procedure for testing multiple hypotheses. *Journal of Statistical Planning and Inference*, 138(6):1521–1527, 2008.
11. F. D. Malliaros and M. Vazirgiannis. Clustering and community detection in directed networks: A survey. *Physics Reports*, 533(4):95–142, 2013.
12. F. Moradi, T. Olovsson, and P. Tsigas. A local seed selection algorithm for overlapping community detection. In *ASONAM*. IEEE, 2014.
13. U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3), 2007.
14. G. Rossetti and R. Cazabet. Community discovery in dynamic networks: A survey. *ACM Computing Surveys (CSUR)*, 51(2):35, 2018.
15. G. Rossetti, L. Milli, and R. Cazabet. Cdlib: a python library to extract, compare and evaluate communities from complex networks. *Applied Network Science*, 2019.
16. G. Rossetti, L. Pappalardo, R. Kikas, D. Pedreschi, F. Giannotti, and M. Dumas. Community-centric analysis of user engagement in skype social network. In *ASONAM*. ACM, 2015.
17. G. Rossetti, L. Pappalardo, and S. Rinzivillo. A novel approach to evaluate community detection algorithms on ground truth. In *Complex Networks*, 2016.
18. G. Rossetti, D. Pedreschi, and F. Giannotti. Node-centric community discovery: From static to dynamic social network analysis. *OSNEM*, 3:32–48, 2017.
19. S. Soundarajan and J. E. Hopcroft. Use of local group information to identify communities in networks. *Transactions on Knowledge Discovery from Data*, 2015.
20. J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *Computing Surveys*, 2013.
21. J. Xie and B. K. Szymanski. Towards linear time overlapping community detection in social networks. In *PAKDD*, 2012.