DEMON: a Local-First Discovery Method for Overlapping Communities

Michele Coscia
CID - Harvard Kennedy School
79 JFK Street, Cambridge, MA, US
michele_coscia@hks.harvard.edu

Giulio Rossetti
KDDLab University of Pisa
Largo B. Pontecorvo 3, Pisa, Italy
giulio.rossetti@isti.cnr.it

Fosca Giannotti
KDDLab ISTI-CNR
Via G. Moruzzi 1, Pisa, Italy
fosca.giannotti@isti.cnr.it

Dino Pedreschi
KDDLab University of Pisa
Largo B. Pontecorvo 3, Pisa, Italy
pedre@di.unipi.it

ABSTRACT

Community discovery in complex networks is an interesting problem with a number of applications, especially in the knowledge extraction task in social and information networks. However, many large networks often lack a particular community organization at a global level. In these cases, traditional graph partitioning algorithms fail to let the latent knowledge embedded in modular structure emerge, because they impose a top-down global view of a network. We propose here a simple local-first approach to community discovery, able to unveil the modular organization of real complex networks. This is achieved by democratically letting each node vote for the communities it sees surrounding it in its limited view of the global system, i.e. its ego neighborhood, using a label propagation algorithm; finally, the local communities are merged into a global collection. We tested this intuition against the state-of-the-art overlapping and non-overlapping community discovery methods, and found that our new method clearly outperforms the others in the quality of the obtained communities, evaluated by using the extracted communities to predict the metadata about the nodes of several real world networks. We also show how our method is deterministic, fully incremental, and has a limited time complexity, so that it can be used on web-scale real networks.

Categories and Subject Descriptors

I.5.3 [**Clustering**]: Algorithms

Keywords

complex networks, data mining, community discovery

1. INTRODUCTION

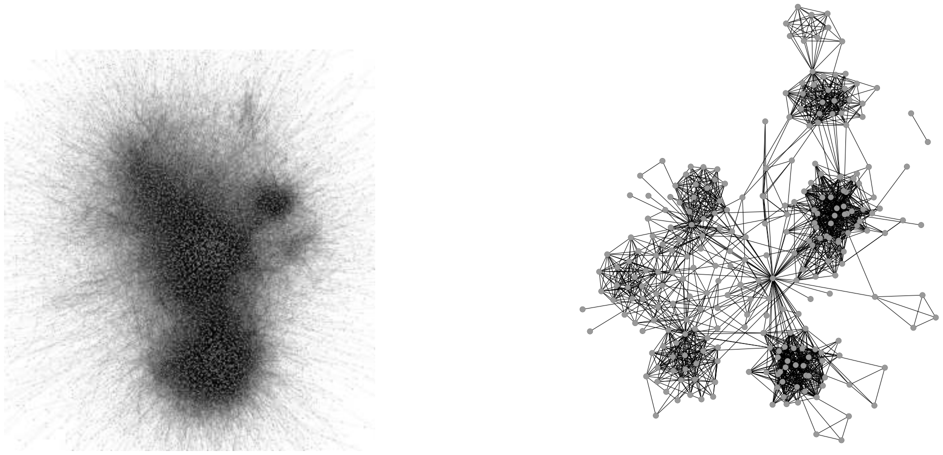
Complex network analysis has emerged as one of the most exciting domains of data analysis and mining over the last decade. One of the most prolific sub field is community discovery in complex network, or *CD* in short. The concept of a "community" in a (web, social, or informational) network is intuitively understood as a set of individuals that are very similar, or close, to each other, more than to anybody else outside the community [6]. This has often been translated in network terms into finding sets of nodes densely connected to each other and sparsely connected with the rest of the network. Community discovery can be seen as a network variant of traditional data clustering. To efficiently detect these structures is very useful for a number of applications, ranging from targeted vaccinations and outbreak prevention [23], to viral marketing [16] and to many web data analysis tasks such as finding tribes in online information exchanges [12, 25], data compressing, clustering [4] and sampling [14].

The classical problem definition of community discovery finds a very intuitive counterpart for small networks, where the denser areas are easily identifiable by visual inspection, while the problem becomes much harder for medium and large scale networks. At the global level, very little can be said about the modular structure of the network, because on larger scales the organization of the system becomes simply too complex. The friendship graph of Facebook includes more than 845 millions nodes as of February 2012[1], but the difficulty of the CD task can be appreciated even considering a tiny fragment of the Facebook friendship graph, illustrated in Figure 1(a). We depicted the connections among 15,000 nodes, i.e., less than 0.002% of the total network. Even in this small subset of the network, no evident organization can be identified easily. Big networks are not analyzable with the naked eye. Very often, a visualization of ten thousands nodes results in a *structureless* hairball. In cases like this, also generic community discovery algorithms tend to return not meaningful communities, as they typically try to cluster the whole structure and return some huge communities and a long list of small branches (see [6]). Often, superimposing an order with a top-down approach leads to failure.

On the contrary, human eyes are good in finding denser areas in simple networks, i.e., the structure of cohesive groups of nodes that emerge considering a *local* fragment of an otherwise big network. But what does *local* mean? Commonsense goes that people are good at identifying the reasons why they know the people they know; therefore, each node

KDD'12, August 12–16, 2012, Beijing, China.
Copyright 2012 ACM 978-1-4503-1462-6 /12/08 ...$15.00.

[1] http://newsroom.fb.com/content/default.aspx?NewsAreaId=22



(a) A global view of the Facebook graph from 15k users.

(b) The "ego minus ego" network of one Facebook user among the 15k.

Figure 1: The real world example of the "local vs global" structure intuition.

has presumably an incomplete, yet clear, vision of the social communities it is part of, and that surrounds it. The consequences of exploiting this idea for the CD problem is effectively illustrated by Figure 1(b). Here, we chose one of the 15k nodes from the previous example and extracted what we call its "ego minus ego" network, i.e. its ego network in which the ego node has been removed, together with all its attached edges. Suddenly, everything around the ego makes sense and some groups can be easily spotted. These groups correspond to the high school and university friends, mates from different workplaces and the members of an online community (we know all these details because the chosen ego is one of the authors of this paper). The ego is part of all these communities and knows that particular subsets of its neighborhood are part of these communities too. Probably, different egos have different perspectives over the same neighbors and it is the union of all these perspectives that creates an optimal partition of the network. In other words: if node A and node B are considered in the same communities by all the nodes connected to both A and B, then they should be grouped in the same community. This is achieved by a *democratic* bottom-up mining approach: in turn, each node gives the perspective of the communities surrounding it and then all the different perspectives are merged together in an overlapping structure.

In the vast CD literature, the general approach for the detection of the modular structure of a network is usually to develop a particular (greedy) algorithm, testing a general quality function with a particular heuristic and then return a set of communities extracted from the global structure (we discuss some of these methods in Section 2). This approach generally fails for large networks due to the difference in structural organization at global and local scale. To cope with this difficulty, we propose a change of mentality. Since our community definition works perfectly in the small scale, then it should be applied only at this small scale. We propose a simple local-first approach to community discovery in complex networks by letting the hidden modular organization of a network emerge from local patterns.

Essentially, we adopt a *democratic* approach to the discovery of communities in complex networks. We ask each node to vote for the communities present in its local view of the network. For this reason, we chose to name our algorithm **D**emocratic **E**stimate of the **M**odular **O**rganization of a **N**etwork, or *DEMON* in short. In practice, we extract the ego network of each node and apply a Label Propagation CD algorithm [21] on this structure, ignoring the presence of the ego itself, that will be judged by its peers neighbors. We then combine, with equity, the vote of everyone in the network. The result of this combination is a set of (overlapping) modules, the guess of the real communities in the global system, made not by an external observer, but by the actors of the network itself. Our democratic algorithm is *incremental*, allowing to recompute the communities only for the newly incoming nodes and edges in an evolving network. Nevertheless, *DEMON* has also a low theoretical linear time complexity. The main core of our method has also the interesting property of being easily *parallelizable*, since only the ego network information is needed to perform independent computations, and it can be easily combined in a MapReduce framework [7]; although the post-process Merge procedure is not trivially solvable in a MapReduce framework (and for this reason we leave a discussion about the parallel implementation as future work). The properties of *DEMON* support its use in massive real world scenarios.

We provide an extensive empirical validation of *DEMON*. In our experimental setting, we are particularly interested in investigating what useful knowledge we can discover. We test the results obtained with our method against selected state-of-the-art algorithms, both overlapping and not overlapping, since we believe that the possibility to cluster the same nodes in different communities is one of the crucial properties that a community discoverer should allow: online social networks have proved that individuals are part of many different communities and groups of interest. To evaluate this knowledge, we make use of a multilabel predictor fed with the extracted communities as input, with the aim of correctly classifying the metadata attached to the nodes in real life. Our datasets include the international store Amazon, the database of collaborations in movie industry IMDb, and the register of the activities of the US Congress GovTrack.us.

The rest of the paper is organized as follows: in Section 2 we present related works in community discovery literature. Section 3 is dedicated to the problem representation and definition. Section 4 describe the *DEMON* algorithm structure, with algorithmic details and an account of the formal properties of the method. Our experiments are presented in Section 5, and finally Section 6 concludes the paper.

2. RELATED WORK

The problem of finding communities in complex networks is very popular among network scientists, as witnessed by an impressive number of valid works in this field. A huge survey by Fortunato [9] explores all the most popular techniques to find communities in complex networks. Traditionally, a community is defined as a dense subgraph, in which the number of edges among the members of the community is significantly higher than the outgoing edges. However, this definition does not cover many real world scenarios, and in the years many different solutions started to explore alternative definitions of communities in complex networks [6].

A variety of CD methods are based on the *modularity* concept, a quality function of a partition proposed by Newman [5, 18]. Modularity scores high values for partitions in which the internal cluster density is higher than the external density. Hundreds of papers have been written about modularity, either using it as a quality function to be optimized, or studying its properties and deficiencies. One of the most advanced examples of modularity maximization CD is [17], where the authors use an extension of the modularity formula to cluster multiplex (evolving and/or multirelational) networks. A fast and efficient greedy algorithm, Modularity Unfolding, has been successfully applied to the analysis of huge web graphs of millions of nodes and billions of edges, representing the structure in a subset of the WWW [3].

Many algorithms have been proposed that are unrelated to modularity. Among them, a particular important field is the application of information theory techniques, as for example in Infomap [22] or Cross Associations [19]. In particular, Infomap has been proven to be one among the best performing non overlapping algorithms [15]. For this reason we chose Infomap as alternative to modularity approaches as a baseline method. Further, modularity approaches are affected by known issues, namely the resolution problem and the degeneracy of good solutions [10]. Similarly to Infomap, Walktrap [20] is based on flow methods and random walks.

A very important property for community discovery is the ability to return overlapping partitions, i.e., the possibility of a node to be part of more than one community. This property reflects the common sense intuition that each of us is part of many different communities, including family, work, and probably many hobby-related communities. Specific algorithms developed over this property are Hierarchical Link Clustering [1], HCDF [13] and k-clique percolation [8].

Finally, an important approach is known as Label Propagation [21]: in this work authors detect communities by spreading labels through the edges of the graph and then labeling nodes according to the majority of the labels attached to their neighbors, iterating until a general consensus is reached. With a reasonable good quality on the partition, this algorithm is extremely fast and known to be one of the very few quasi-linear solutions to the community discovery problem, even if its plain application leads to worse results than Infomap and it does not return an overlapping partition. A related work is also [2], whose aim is also to discover local communities. However, authors are only interested in those local communities and they do not return any global structure modular organization.

To to extract useful knowledge from the modular structure of networked data is also a prolific track of research. We recall the GuruMine framework, whose aim is to identify leaders in information spread and to detect groups of users that are usually influenced by the same leaders [12]. Many other works investigate the possibility of applying network analysis for studying, for instance, the dynamics of viral marketing [16].

3. NETWORKS AND COMMUNITIES

We model networks and their properties in terms of simple graphs. For the sake of simplicity, a network is represented as an undirected, unlabeled and unweighted simple graph, denoted by $\mathcal{G} = (V, E)$ where V is a set of nodes and E is a set of edges, i.e., pairs (u, v) representing the fact that there is a link in the network connecting nodes u and v. It should be noted, however, that our method can handle weighted, directed and labeled multi-graphs.

In general terms, our problem definition is to find communities in complex networks. However, this is an ambiguous goal, as the definition itself of "community" in a complex network, similarly to the notion of clustering in statistics and data mining, is not unique [6]. Furthermore, in a complex and semantically rich setting as the modern Web, one may want to cluster many different kinds of objects for many different reasons. Therefore, we need to narrow down our problem definition as follows.

We define two basic graph operations. The first one is the Ego Network extraction EN. Given a graph $\mathcal{G}$ and a node $v \in V$, $EN(v, \mathcal{G})$ is the subgraph $\mathcal{G}'(V', E')$, where V' is the set containing v and all its neighbors in E, and E' is the subset of E containing all the edges (u, v) where $u \in V' \wedge v \in V'$. The second operation is the Graph-Vertex Difference $-g$: $-g(v, \mathcal{G})$ will result in a copy of $\mathcal{G}$ without the vertex v and all edges attached to v. The combination of these two functions yields the $EgoMinusEgo$ function: $EgoMinusEgo(v, \mathcal{G}) = -g(v, EN(v, \mathcal{G}))$. Given a graph $\mathcal{G}$ and a node $v \in V$, the set of *local communities* $\mathcal{C}(v)$ of node v is a set of (possibly overlapping) sets of nodes in $EgoMinusEgo(v, \mathcal{G})$, where each set $C \in \mathcal{C}(v)$ is a community according to node similarity: each node in C is more similar to any other node in C than to any other node in $C' \in \mathcal{C}(v)$, with $C \neq C'$. Finally, we define the set of *global communities*, or simply communities, of a graph $\mathcal{G}$ as:

$$\mathcal{C} = Max(\bigcup_{v \in V} \mathcal{C}(v)) \qquad (1)$$

where, given a set of sets $\mathcal{S}$, $Max(\mathcal{S})$ denotes the subset of $\mathcal{S}$ formed by its maximal sets only; namely, every set $S \in \mathcal{S}$ such that there is no other set $S' \in \mathcal{S}$ with $S \subset S'$. In other words, by equation (1) we generalize from local to global communities by selecting the maximal local communities that cover the entire collection of local communities, each found in the $EgoMinusEgo$ network of each individual node.

Algorithm 1 The pseudo-code of *DEMON* algorithm.

```
Require: $\mathcal{G} : (V, E)$; $\mathcal{C} = \emptyset; \epsilon \in [0..1]$
Ensure: set of overlapping communities $\mathcal{C}$
1: for all $v \in V$ do
2:     $e \leftarrow EgoMinusEgo(v, \mathcal{G})$
3:     $\mathcal{C}(v) \leftarrow LabelPropagation(e)$
4:     for all $C \in \mathcal{C}(v)$ do
5:         $C \leftarrow C \cup v$
6:         $\mathcal{C} \leftarrow Merge(\mathcal{C}, C, \epsilon)$
7:     end for
8: end for
9: return $\mathcal{C}$
```

4. THE ALGORITHM

In this section we present our solution to the community discovery problem. The pseudo code of *DEMON* is specified in Algorithm 1.

4.1 The Core of the Algorithm

The set of discovered communities $\mathcal{C}$ is initially empty. The external (explicit) loop of *DEMON* cycles over each individual node, and it is necessary to generate all the possible points of view of the structure and get a complete coverage of the network itself. For each node v, we apply the $EgoMinusEgo(v, \mathcal{G})$ operation defined in Section 3, obtaining a graph e. We cannot apply simply the ego network extraction $EN(v, \mathcal{G})$ because the ego node v is directly linked to all nodes $\in EN(v, \mathcal{G})$. This would lead to noise in the subsequent steps of *DEMON*, since by our definition of local community the nodes would be put in the same community if they are close to each other. Obviously a single node connecting the entire sub-graph will make all nodes very close, even if they are not in the same community. For this reason, we remove the ego from its own ego network.

Once we have the e graph, the next step is to compute the communities contained in e. We chose to perform this step by using a community discovery algorithm borrowed from the literature. Our choice fell on the Label Propagation (LP) algorithm [21]. This choice has been made for the following reasons:

1. LP shares with this work the definition of what is a community.

2. LP is known as the least complex algorithm in the literature, reaching a quasi-linear time complexity in terms of nodes. However,

3. LP will return results of a quality comparable to more complex algorithms [6].

Reason #2 is particularly important, since Step #3 of our pseudo code needs to be performed once for every node of the network. It is unacceptable to spend a superlinear time for each node at this stage, if we want to scale up to millions of nodes and hundreds of millions edges. Given the linear complexity of Step #3, we refer to this as the internal (implicit) loop for finding the local communities.

We briefly describe in more detail the LP algorithm, given its importance in the *DEMON* algorithm, following the original article [21]. Suppose that a node v has neighbors $v_1, v_2, ..., v_k$ and that each neighbor carries a label denoting the community that it belongs to. Then v determines its community based on the labels of its neighbors. A three-step example of this principle is shown in Figure 2. The authors assume that each node in the network chooses to join the community to which the maximum number of its neighbors belong. 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 together as one community. Clearly, a node with an equal maximum number of neighbors in two or more communities can belong to both communities, thus identifying possible overlapping communities. The original algorithm does not handle this situation. For clarity, we report here the procedure of the LP algorithm, that is the expansion of Step #3 of Algorithm 1 and represents our inner loop:

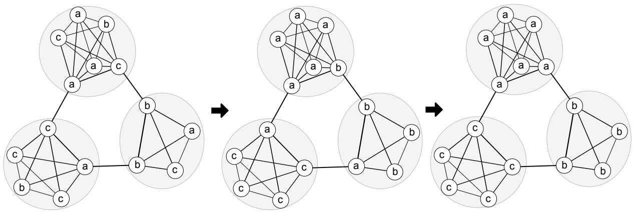


Figure 2: A simple simulation of the Label Propagation process for community discovery.

1. Initialize the labels at all nodes in the network. For any given node v, $C_v(0) = v$.

2. Set $t = 1$.

3. Arrange the nodes in the network in a random order and set it to V.

4. For each $v_i \in V$, in the specific order, let $C_{v_i}(t) = f(C_{v_{i1}}(t-1), \ldots, C_{v_{ik}}(t-1))$. f here returns the label occurring with the highest frequency among neighbors and ties are broken uniformly randomly.

5. If every node has a label that the maximum number of their neighbors have, or t hits a maximum number of iterations t_{max} then stop the algorithm. Else, set $t = t + 1$ and go to (3).

4.2 The Merge Function

The result of Step #3 of Algorithm 1 is a set of local communities, according to the perspective of node v: at the end of the LP algorithm we reintroduce, in each local community, the node v. These communities are likely to be incomplete and should be used to enrich what *DEMON* already discovered so far. Thus, the next step is to merge each local community of $\mathcal{C}$ in order to obtain the result set. The $Merge$ operation is defined as follows.

Two communities C and I are merged if and only if at most the ϵ% of the smaller one is not included in the bigger one; in this case, C and I are removed from $\mathcal{C}$ and their union is added to the result set. The ϵ factor is introduced to vary the percentage of common elements provided from each couple of communities: $\epsilon = 0$ ensure that two communities are merged only if one of them is a proper subset of the other, on the other hand with a value of $\epsilon = 1$ even communities that do not share a single node are merged together.

Algorithm 2 The pseudo-code of Merge function.

```
Require: $\mathcal{C} = Community\ set; C = Community; \epsilon \in [0..1]$
Ensure: set of overlapping communities $\mathcal{C}$
1: for all $I \in \mathcal{C}$ do
2:     if $C.size \leq I.sizeandC \subseteq_\epsilon I$ then
3:         $u = C \cup I$;
4:         $\mathcal{C} - C; \mathcal{C} - I$;
5:         $\mathcal{C} = \mathcal{C} \cup u$;
6:     end if
7: end for
8: return $\mathcal{C}$
```

4.3 DEMON Properties

To prove the correctness of the *DEMON* algorithm w.r.t. the problem definition in Section 3, we prove by induction that the following holds:

Property 1. *At the k-th iteration of the outer loop of DEMON, for all $k \geq 0$:*

$$\mathcal{C} = Max(\bigcup_{v=v_1,\dots,v_k} \mathcal{C}(v)) \tag{2}$$

where $v_1, \dots, v_k$ are the nodes visited after k iterations.

Property (1) trivially holds for $k = 0$, i.e., at initialization stage. For $k > 0$, assume that the property holds up to $k - 1$. Then $\mathcal{C}$ contains the maximal local communities of the subgraph with nodes $v_1, \dots, v_{k-1}$. By merging every local community C of node v_k into $\mathcal{C}$, we guarantee that C is added to the result only if it is not covered by any preexisting community, and, if added, any preexisting community covered by C is removed from $\mathcal{C}$. As a result, after merging all communities in $\mathcal{C}(v_k)$ into $\mathcal{C}$ in Steps #4-6, the latter is the set of maximal communities covering all local communities discovered in $v_1, \dots, v_k$. Therefore, we can conclude that *DEMON* is a correct and complete implementation of the CD problem stated by equation (1). More generally, denoting by *DEMON*$(\mathcal{G}, \mathcal{C})$ the set of communities $\mathcal{C}'$ obtained by running the *DEMON* algorithm on graph $\mathcal{G}$ starting with the (possibly non-empty) set of communities $\mathcal{C}$, the following properties hold.

Property 2. **Correctness** *and* **Completeness**. *If DEMON$(\mathcal{G}, \mathcal{C}) = \mathcal{C}'$, where $\mathcal{G} = (V, E)$, then*

$$\mathcal{C}' = Max(\mathcal{C} \cup \bigcup_{v \in V} \mathcal{C}(v)) \tag{3}$$

In other words, given a preexisting set of communities $\mathcal{C}$ and a graph $\mathcal{G}$, *DEMON* returns all and only the communities obtained extending $\mathcal{C}$ with the communities found in $\mathcal{G}$, coherently with the definition of communities given in equation (1).

Property 3. **Determinacy** *and* **Order insensitivity**. *There exists a* unique *$\mathcal{C}' = DEMON(\mathcal{G}, \mathcal{C})$ for any given $\mathcal{G}$ and $\mathcal{C}$, disregarding the order of visit of the nodes in $\mathcal{G}$.*

This is a direct corollary of property (2) and of the uniqueness of the set $Max(\mathcal{S})$ for any set of sets $\mathcal{S}$, under the assumption that the set of local communities $\mathcal{C}(v)$ is also uniquely assigned, for any node v. Therefore, the order in which the nodes in $\mathcal{G}$ are visited by *DEMON* is irrelevant.

Property 4. **Compositionality**. *Consider any partition of a graph $\mathcal{G}$ into two subgraphs $\mathcal{G}_1$, $\mathcal{G}_2$ such that, for any node v of $\mathcal{G}$, the entire ego network of v in $\mathcal{G}$ is fully contained either in $\mathcal{G}_1$ or $\mathcal{G}_2$. Then, given an initial set of communities $\mathcal{C}$:*

$$DEMON(\mathcal{G}_1 \cup \mathcal{G}_2, \mathcal{C}) = Max(DEMON(\mathcal{G}_1, \mathcal{C}) \cup DEMON(\mathcal{G}_2, \mathcal{C})) \tag{4}$$

This is a consequence of two facts: *i*) each local community $\mathcal{C}(v)$ is correctly computed under the assumption that the subgraphs do not split any ego network, and *ii*) for any two sets of sets $\mathcal{S}_1, \mathcal{S}_2$, $Max(\mathcal{S}_1 \cup \mathcal{S}_2) = Max(Max(\mathcal{S}_1) \cup Max(\mathcal{S}_2))$.

Property 5. **Incrementality**. *Given a graph $\mathcal{G}$, an initial set of communities $\mathcal{C}$ and an incremental update $\Delta\mathcal{G}$ consisting of new nodes and new edges added to $\mathcal{G}$, where $\Delta\mathcal{G}$ contains the entire ego networks of all new nodes and of all the preexisting nodes reached by new links, then*

$$DEMON(\mathcal{G} \cup \Delta\mathcal{G}, \mathcal{C}) = DEMON(\Delta\mathcal{G}, DEMON(\mathcal{G}, \mathcal{C})) \tag{5}$$

This is a consequence of the fact that only the local communities of nodes in $\mathcal{G}$ affected by new links need to be reexamined, so we can run *DEMON* on $\Delta\mathcal{G}$ only, avoiding to run it from scratch on $\mathcal{G} \cup \Delta\mathcal{G}$.

Properties (4) and (5) have important computational repercussions. The compositionality property entails that the core of *DEMON* algorithm as described in subsection 4.1 is highly parallelizable, because it can run independently on different fragments of the overall network with a relatively small combination work. Each node of the computer cluster needs to obtain a small fragment of the network, as small as the ego network of one or a few nodes. The Map function is simply the LP algorithm. The incrementality property entails that *DEMON* can efficiently run in a streamed fashion, considering incremental updates of the graph as they arrive in subsequent batches; essentially, incrementality means that it is not necessary to run *DEMON* from scratch as batches of new nodes and new links arrive: the new communities can be found by considering only the ego networks of the nodes affected by the updates (both new nodes and old nodes reached by new links). This does not trivially hold for the Merge function presented in subsection 4.2, therefore the actual parallel implementation of *DEMON* is left as future work. However, different and simpler Merge functions can be define to combine the results provided by the core of the algorithm, thus preserving its possibility to scale up in a parallel framework.

4.4 Complexity

We now evaluate the time complexity of our approach. *DEMON* core (Section 4.1) is based on the Label Propagation algorithm, whose complexity is $\mathcal{O}(n + m)$ [21], where n is the number of nodes and m is the number of edges. LP is performed once for each node. Let us assume that we are working with a scale free network, whose degree distribution is $p_k = k^{-\alpha}$. This means that there are $\frac{n}{k^\alpha}$ nodes with degree k. If K is the maximum degree, the complexity would be $\sum_{k=1}^{K}(\frac{n}{k^\alpha} \times (k + \frac{k(k-1)}{2}))$ because for each node of degree k we have an ego network of k nodes and at worst $\frac{k(k-1)}{2}$ edges. This number is very small for the vast majority of nodes, being the degree distribution right skewed, thus many nodes have degree 1 or 2. We omit the solution of the sum with the integral and we report that the complexity is then dominated by a single term, ending up to be $\mathcal{O}(nK^{3-\alpha})$. This means that the stronger is the α exponent, the faster is *DEMON*: with $\alpha = 3$ we have few super-hubs for which we basically check the entire network few times and the rest of nodes add nothing to the complexity; with $\alpha = 2$ we have many high degree nodes and we end up with higher complexity, but still subquadratic in term of nodes (as, with $\alpha = 2$, $K << n$).

5. EXPERIMENTS

We now present our experimental findings. We make use of three networked datasets, representing very different phenomena. We first concentrate on evaluating the quality of a set of communities discovered in these datasets, comparing the results with those of other competing methods in terms of the predictive power of the discovered communities. Since real world data are enriched with annotated information, we measure the ability of each community to predict the semantic information attached with the metadata of the nodes within the community itself.

Next, we assess the community quality using a global measure of community cohesion, based on the intuition that nodes into the same community should possess similar semantic properties in terms of attached metadata.

The selected competitors for our assessment are: Hierarchical Link Clustering (HLC) [1], that has been proven able to outperform all the overlapping algorithms, including the k-clique Propagation algorithm by Palla et al [8]; two random walks based methods, one focusing on minimizing random walk entropy (Infomap [22]) and the other relying on a general flow method (Walktrap [20]); a leading eigenvector-based community discovery, namely Modularity maximization in the fast greedy implementation introduced in [5]. Finally, we present some examples of knowledge that we are able to extract from the communities found by the *DEMON* algorithm.

Note that we are not able to provide the analytic evaluation for Amazon dataset: for that network HLC algorithm was not able to provide results due to memory consumption problems, while the other community discovery algorithms usually returned some huge communities that was not possible to analyze (see Section 5.2 and particularly Figure 4 for more information).

The experiments were performed on a Dual Core Intel i7 64 bits @ 2.8 GHz, equipped with 8 GB of RAM and with a kernel Linux 3.0.0-12-generic (Ubuntu 11.10). The code was developed in Java and it is available for download with the network datasets used[2]. For performances purposes, we mainly refer to the biggest dataset, i.e. Amazon: the core of the algorithm (Section 4.1) took less than a minute, while the Merge function (Section 4.2) with increasing thresholds can take from one minute to one hour.

5.1 Networks

We tested our algorithms on three real world complex networks extracted from available web services of different domains. A general overview about the statistics of these networks can be found in Table 1, where: $|V|$ is the number of nodes, $|E|$ is the number of edges and $\bar{k}$ is the average degree of the network. Congress and IMDb networks are similar to the ones used in [1], generally updating the source dataset with a more recent set of data, and we refer to that paper for a deeper description of them. The networks were generated as follows:

Network	$\|V\|$	$\|E\|$	$\bar{k}$
Congress	526	14,198	53.98
IMDb	56,542	185,347	6.55
Amazon	410,236	2,439,437	11.89

Table 1: Basic statistics of the studied networks.

Congress. The network of legislative collaborations between US representatives of the House and the Senate during the 111st US congress (2009-2011). We downloaded the data about all the bills discussed during the last Congress from GovTrack[3], a web-based service recording the activities of each member of the US Congress. The bills are usually co-sponsored by many politicians. We connect politicians if they have at least 75 co-sponsorships and delete all the connections that are created only by bills with more than 10 co-sponsors. Attached to each bills in the Govtrack data we have also a collection of subjects related to the bill. The set of subjects a politicians frequently worked on is the *qualitative attribute* of this network.

IMDb. We downloaded the entire database of IMDb from their official APIs[4] on August 25th 2011. We focus on actors who star in at least two movies during the years from 2001 to 2010, filtering out television shows, video games, and other performances. We connect actors with at least two movies in which they both appear. This network is weighted according to the number of co-appearances. Our *qualitative attributes* are the user assigned keywords, summarizing the movies each actor has been part of.

Amazon. We downloaded Amazon data from the Stanford Large Network Dataset Collection[5]. In this dataset, frequent co-purchases of products are recorded for the day of May 5th 2003. We transformed the directed network in an undirected version. We also downloaded the metadata information about the products, available in the same repository. Using this metadata, we can define the *qualitative attributes* for each product as its categories.

5.2 Quality Evaluation via Label Prediction

We first assess *DEMON* performances using a classical prediction task. We attach the community memberships of a node as known attributes, then its qualitative attributes (real world labels) as target to be predicted; we then feed these attributes to a state-of-the-art label predictor and record its performance. Of course, a node may have one or more known attributes, as both *DEMON* and HLC are overlapping community discoverers; and it may have also one or more unknown attributes, as it can carry many different labels.

For this reason, we need a multilabel classificator, i.e. a learner able to predict multiple target attributes [24]. We chose to use the Binary Relevance Learner. The BRL learns $|L|$ binary classifiers $H_l : X \rightarrow \{l, \neg l\}$, one for each different label $l \in L$. It transforms the original data set into $|L|$ data sets D_l that contain all examples of the original data set,

[2] http://www.di.unipi.it/∼coscia/demon/

[3] http://www.govtrack.us/developers/data.xpd

[4] http://www.imdb.com/interfaces

[5] http://snap.stanford.edu/data/index.html

Network	*DEMON*	HLC	Infomap	Modularity	Walktrap
Congress	**0.21275**	0.14740	0.00535	0.00099	0.00725
IMDb	**0.44252**	0.43078	0.38470	0.10692	0.17488

Table 2: The F-Measure scores for Congress and IMDb dataset and each community partition.

labeled as l if the labels of the original example contained l and as $\neg l$ otherwise. It is the same solution used in order to deal with a single-label multi-class problem using a binary classifier. Note that this classifier does not penalize per se non-overlapping partitions, as each target label is classified independently, and this property is requested to fairly confront overlapping algorithms such as *DEMON* and HLC, with the other non-overlapping algorithms. We used the Python implementation provided in the Orange software[6]. For time and memory constraints due to the BRL complexity, for IMDb we used as input only the biggest communities (with more than 15 nodes) and eliminating all nodes that are not part of any of the selected communities.

Multi-label classification requires different metrics than those used in traditional single-label classification. Among the measures that have been proposed in the literature, we use the multi-label version of the standard Precision and Recall measures. Let D_l be our multi-label evaluation data set, consisting of $|D_l|$ multi-label examples $(x_i, Y_i), i = 1..|D_l|, Y_i \subseteq L$. Let H be our BRL multi-label classifier and $Z_i = H(x_i)$ be the set of labels predicted by H for x_i. Then, we can evaluate Precision and Recall of H as:

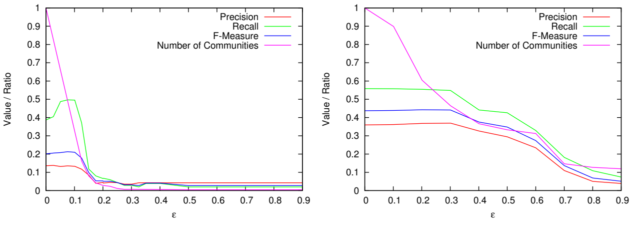
$$Precision(H, D_l) = \frac{1}{|D_l|} \sum_{i=1}^{|D_l|} \frac{|Y_i \cap Z_i|}{|Z_i|},$$

$$Recall(H, D_l) = \frac{1}{|D_l|} \sum_{i=1}^{|D_l|} \frac{|Y_i \cap Z_i|}{|Y_i|}.$$

We then derive the F-measure from Precision and Recall. For alternatives multi-label evaluations, we refer to [11]. The results are reported in Table 2 and show how *DEMON* outperforms its competitors. We did not test Amazon network as HLC was not able to provide results due to its complexity and further the BRL classifier was not able to scale for the overall number of nodes and labels.

For IMDb dataset, HLC was able to score almost like *DEMON*. However, there is an important distinction to be made about the quantity of the results: if the community discovery returns too many communities, then it is difficult to actually extract useful knowledge from them. We reported in Table 3 the basic statistics about the partitions returned by the algorithms: number of communities ($|\mathcal{C}|$) and average community size ($|\bar{c}|$). For *DEMON*, we report the statistics of the communities extracted with $\epsilon = 0$. As we can see, not only *DEMON* scores better results, but it does with 70-80% less communities than HLC and with an average community size more manageable than Infomap.

We report in Table 3 the results for $\epsilon = 0$. However, we vary the ϵ threshold and see what happens to the number of communities and to the quality of the results. We can see that for both Congress and IMDb the Precision, Recall and F-Measure scores remains constant (and actually F-Measure peaks at $\epsilon = 0.076$ and $\epsilon = 0.301$ for Congress and IMDb respectively) before falling for increasing ϵ values, while the relative number of communities dramatically drops. For Congress, we have the maximum F-Measure with only 175 communities; while for IMDb F-Measures peaks with 6,508 communities (in both cases, less than 50% of the communities at $\epsilon = 0$ and than an order of magnitude of HLC).



(a) Congress (b) IMDb

Figure 3: Precision, Recall, F-Measure and number of communities for different ϵ values.

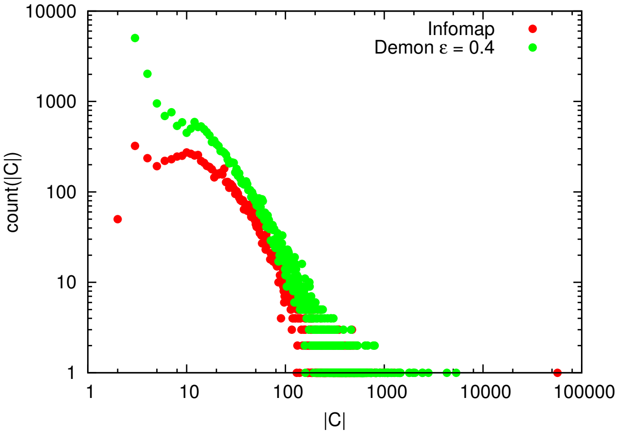


Figure 4: The distribution of the community sizes for DEMON and Infomap in the Amazon network.

A final consideration is needed about the size distribution of the communities detected by *DEMON* and the other community discovery algorithms. In Figure 4 we depicted the community size distribution for *DEMON* and Infomap for the Amazon network. As we can see, Infomap returned, among the others, a giant community, one order of magnitude greater than the biggest one returned by *DEMON*. To analyze such a community has been proved impossible, and this is another reason why we are not able to provide an analitycal evaluation of the results extracted from the Amazon network.

We can conclude that *DEMON* with a manageable number of communities is able to outperform more complex methods and the choice of ϵ can make the difference in obtaining a narrower set of communities with the same (or greater) overall quality.

5.3 Quality Evaluation via Community Cohesion

As presented at the beginning of this section, the networks studied here possess *qualitative attributes* that attaches a

[6] http://orange.biolab.si/

Network	Demon		HLC		Infomap		Modularity		Walktrap	
	$\lvert\mathcal{C}\rvert$	$\lvert\bar{c}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\bar{c}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\bar{c}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\bar{c}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\bar{c}\rvert$
Congress	425	63.3671	1,476	4.5867	6	87.6667	3	175.3333	7	71.8571
IMDb	14,004	12.6824	88,119	8.3426	5,991	27.1574	4,746	11.9157	7,877	7.1781

Table 3: Statistics of the community set returned by the different algorithms.

Network	*DEMON*	HLC	Infomap	Modularity	Walktrap
Congress	**1.1792**	1.1539	1.0229	1.0373	1.0532
IMDb	**5.6158**	5.1589	0.1400	1.4652	0.0211

Table 4: The Community Quality scores for Congress and IMDb dataset and each community partition.

small set of annotations or tags to each node. Assuming that these qualitative attributes form a description of the node, beyond the network itself, we can reasonably state that "similar" nodes share more *qualitative attributes* than dissimilar nodes. This procedure is not standard in community discovery results evaluation. Usually authors prefer to use the established measure of Modularity. However, Modularity is strictly (and exclusively) dependent on the graph structure. What we want evaluate is not how a graph measure is maximized, but how good is our community partition in describing real world knowledge about the clustered entities.

We quantify the matching between a community partition and the metadata by evaluating how much higher are on average the Jaccard coefficient of the set of *qualitative attributes* for pair of nodes inside the communities over the average of the entire network, or:

$$CQ(P) = \frac{\sum_{(n_1,n_2)\in P} \frac{|QA(n_1)\cap QA(n_2)|}{|QA(n_1)\cup QA(n_2)|}}{\sum_{(n_1,n_2)\in E} \frac{|QA(n_1)\cap QA(n_2)|}{|QA(n_1)\cup QA(n_2)|}},$$

where P is the set of node pairs that share at least one community, $QA(n)$ is the set of qualitative attributes of node n and E is the set of all edges. If $CQ(P) = 1$, then there is no difference between P and the average similarity of nodes, i.e. P is practically random. Lower values implies that we are grouping together dissimilar nodes, higher values are expected for an algorithm able to group together similar nodes.

To calculate the Jaccard coefficient for each pair of the network is computationally prohibitive. Therefore, for IMDb we chose a random set of 400k pairs. Moreover, CQ is biased towards algorithms returning more communities. For this reason, we just collected random communities from the community pool, trying to avoid too much overlap as we want also to maximize the number of nodes considered by CQ (i.e. we try not to consider more than one community per node). We apply this procedure for each algorithm and calculated the CQ value. We repeated this process for 100 iterations and we report in Table 4 the average value of the CQ obtained. Also in this case, *DEMON* was able to easily outperform all the other algorithms.

5.4 Interpretation of Discovered Communities

In this Section we present a brief case study using the communities extracted for the previously exposed evaluation of *DEMON*. Aim of the section is to demonstrate that the extracted communities have practical applications in the extraction of knowledge from real world scenarios. In the Amazon network to have different communities for each item is very useful. A recommendation system is able to better discern if a user may be interested in a product or not given that he bought something else; however being part of one community of products does not mean that that particular community describes all aspects of a particular product.

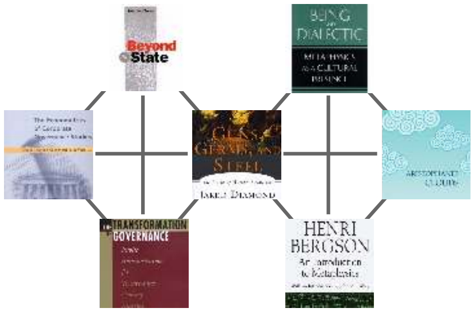


Figure 5: A representation of parts of the two communities surrounding our case study in the amazon network.

Let us consider, as an example, the case of Jared Diamond's best selling book "Guns, Germs, and Steel: The Fates of Human Societies". Clearly, it is difficult to say that the people interested in this book are always interested in the same things. Checking the communities to which it belongs, we find two very different big communities (a depiction of the two communities is provided by Figure 5). These communities have some sort of overlap, however they can be characterized by looking at the products that appear exclusively in one or in the other. In the first one we find books such as: "Beyond the State: An Introductory Critique", "The Econometrics of Corporate Governance Studies" and "The Transformation of Governance: Public Administration for Twenty-First Century America". This is clearly a community composed mainly by purchases made by the people more interested in the socioeconomic aspects of Diamond's book. The second community hosts products such as: "An Introduction to Metaphysics", "Aristophanes' Clouds Translated With Notes and Introduction" and "Being and Dialectic: Metaphysics and Culture". This second communities is apparently composed by the purchases of customers more attracted by the underlying philosophical implications of Diamond's study. Products in one communities may have something in common, but they are part of two distinct and very well characterized groups, and the one in one group are not expected to be found in the other.

This is of course one of the many cases. We report as an additional example the two communities around the historical novel "The Name of the Rose" by Umberto Eco: one community is characterized by history related products (such as "Ancestral Passions : The Leakey Family and the Quest for Humankind's Beginnings"), the other by costume fiction (for example the 1932 Dreyer's movie "Vampyr").

6. CONCLUSION AND FUTURE WORKS

In this paper we proposed a new method for solving the problem of detecting latent knowledge from significant communities in complex networks. We propose a democratic approach, where the peer nodes judge where their neighbors should be clustered together. This approach has robust theoretical properties, including correctness and completeness w.r.t. a precise community definition, determinacy, compositionality and incrementality, that make it amenable to challenge the conceptual and computational challenge to analyze networks with millions of nodes. We have shown in the experimental section that this method allows a discovery of communities in different real world networks collected from information rich datasets. The quality of the overlapping partition, a partition that allows nodes to be in different communities at the same time, is improved w.r.t state-of-the-art algorithms, evaluated using the communities to predict the metadata attached to the nodes, and according to a quantitative quality function, also metadata-based.

Many lines of research remain open for future work, such as an efficient parallel implementation that may make *DEMON* the first community discovery algorithm able to scale to billions of nodes; different merging strategies that may further improve the quality of the results; different hosted algorithms can be used instead of the Label Propagation algorithm in the inner loop of *DEMON*, to extract communities according to different definitions.

Acknowledgments. Michele Coscia is a recipient of the Google Europe Fellowship in Social Computing, and this research is supported in part by this Google Fellowship. This work has been partially supported by the European Commission under the FET-Open Project n. FP7-ICT-270833, DATA SIM – DATa science for SIMulating the era of electric vehicles http://www.datasim-fp7.eu/.

7. REFERENCES

[1] Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, June 2010.

[2] James P. Bagrow and Erik M. Bollt. Local method for detecting communities. *Physical Review E*, 72(4):046108+, October 2005.

[3] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *J.STAT.MECH.*, page P10008, 2008.

[4] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. Layered label propagation: a multiresolution coordinate-free ordering for compressing social networks. In *WWW*, pages 587–596, 2011.

[5] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70:066111, 2004.

[6] Michele Coscia, Fosca Giannotti, and Dino Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining*, 4(5):512–546, 2011.

[7] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *OSDI*, pages 137–150, 2004.

[8] Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique Percolation in Random Networks. *Physical Review Letters*, 94(16):160202+, April 2005.

[9] S. Fortunato. Community detection in graphs. *Physics Reports*, 486:75–174, February 2010.

[10] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *PNAS*, 104(1):36–41, January 2007.

[11] Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *PAKDD*, pages 22–30, 2004.

[12] Amit Goyal, Byung-Won On, Francesco Bonchi, and Laks V. S. Lakshmanan. Gurumine: A pattern mining system for discovering leaders and tribes. *ICDE*, 0:1471–1474, 2009.

[13] Keith Henderson, Tina Eliassi-Rad, Spiros Papadimitriou, and Christos Faloutsos. Hcdf: A hybrid community discovery framework. In *SDM*, pages 754–765, 2010.

[14] Liran Katzir, Edo Liberty, and Oren Somekh. Estimating sizes of social networks via biased sampling. In *WWW*, pages 597–606, 2011.

[15] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80(5):056117–+, November 2009.

[16] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1, May 2007.

[17] Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and J-P Onnela. Community structure in Time-Dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

[18] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June 2006.

[19] Spiros Papadimitriou, Jimeng Sun, Christos Faloutsos, and Philip S. Yu. Hierarchical, parameter-free community discovery. In *ECML PKDD*, pages 170–187, 2008.

[20] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *J. Graph Algorithms Appl.*, 10(2):191–218, 2006.

[21] Usha N. Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106+, September 2007.

[22] Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *PNAS*, 105(4):1118–1123, January 2008.

[23] Jianhua Ruan and Weixiong Zhang. An efficient spectral algorithm for network community discovery and its applications to biological and social networks. *Data Mining, IEEE International Conference on*, 0:643–648, 2007.

[24] G. Tsoumakas and I. Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

[25] Dashun Wang, Zhen Wen, Hanghang Tong, Ching-Yung Lin, Chaoming Song, and Albert-László Barabási. Information spreading in context. In *WWW*, pages 735–744, 2011.