Uncovering Hierarchical and Overlapping Communities with a Local-First Approach

Michele Coscia, CID - Harvard Kennedy School, Cambridge, MA, US
Giulio Rossetti, KDDLab, ISTI-CNR, Pisa, Italy
Fosca Giannotti, KDDLab, ISTI-CNR, Pisa, Italy
Dino Pedreschi, Computer Science Department, University of Pisa, Pisa, Italy

Community discovery in complex networks is the task of organizing a network's structure by grouping together nodes related to each other. Traditional approaches are based under the assumption that there is a global level organization in the network. However, in many scenarios, each node is the bearer of complex information, and cannot be classified in disjoint clusters. The top-down global view of the partition approach is not designed for this. Here, we represent this complex information as multiple latent labels, and we postulate that edges in the networks are created among nodes carrying similar labels. The latent labels are the communities a node belongs to and we discover them with a simple local-first approach to community discovery. This is achieved by democratically letting each node vote for the communities it sees surrounding it in its limited view of the global system, its ego neighbourhood, using a label propagation algorithm, assuming that each node is aware of the label it shares with each of its connections. The local communities are merged hierarchically, unveiling the modular organization of the network at the global level and identifying overlapping groups and groups of groups. We tested this intuition against the state-of-the-art overlapping community discovery, and found that our new method advances in the chosen scenarios in the quality of the obtained communities. We perform a test on benchmark and on real-world networks, evaluating the quality of the community coverage by using the extracted communities to predict the metadata attached to the nodes, that we consider an external information about the latent labels. We also provide an explanation about why real-world networks contain overlapping communities and how our logic is able to capture them. Finally, we show how our method is deterministic, incremental, and has a limited time complexity, so that it can be used on real-world scale networks.

Categories and Subject Descriptors: I.5.3 [**Clustering**]: Algorithms

General Terms: Design, Algorithms, Performance

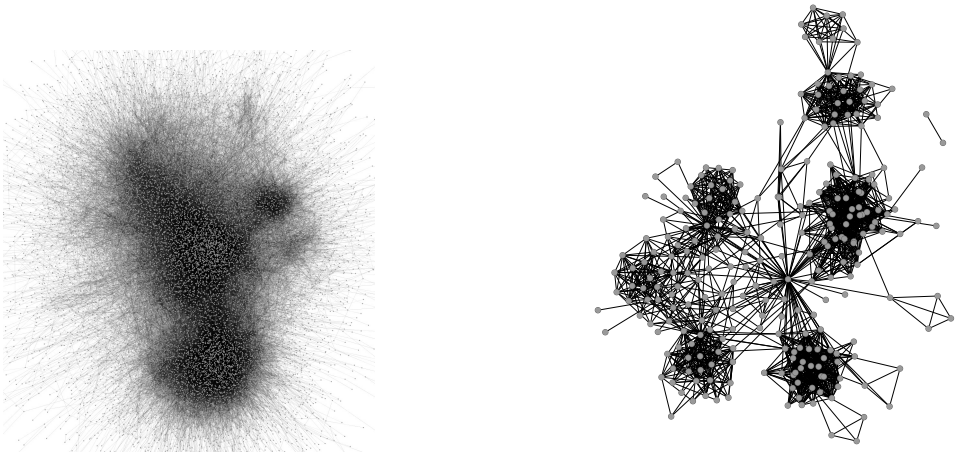
Additional Key Words and Phrases: complex networks, data mining, community discovery

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 fields 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 entities that have some latent factors in common with each other, and thus play a specific role in the overall function of the complex system [Coscia et al. 2011]. The traditional approach assumes that latent factors drive network connectivity, thus finding sets of

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/. This material is based upon work supported by the National Science Foundation under Grant No. 1216028.

DOI:http://dx.doi.org/10.1145/0000000.0000000



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

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

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

nodes with a high edge density among each other and low edge density with the rest of the network effectively detects the functional modules of the network. Community discovery is then a network variant of data clustering, where proximity is replaced with edge connectivity. To efficiently detect the latent modules of the network is intuitively useful as they give insights about how the network works. In fact, one can find a number of applications in literature, ranging from targeted vaccinations and outbreak prevention [Ruan and Zhang 2007], to viral marketing [Leskovec et al. 2007] and to many web data analysis tasks such as finding tribes in online information exchanges [Goyal et al. 2009], [Wang et al. 2011], data compressing, clustering [Boldi et al. 2011] and sampling [Katzir et al. 2011].

The classical problem definition of community discovery assumes that each node plays a single role in the network. This is easily understood by looking at examples of limited size, where it is likely that this assumption is true, as the phenomenon represented is likely to be properly isolated. In this case, the denser areas are easily identifiable by visual inspection. The problem becomes much harder for medium and large scale networks, where many different phenomena are at play at the same time, tangled the one with another. At the global level, very little can be said about the modular structure of most networks: on larger scales the organization of the system becomes simply too complex. The friendship graph of Facebook includes more than 1.11 billion nodes as of March 2013[1]. But even on a tiny fragment of the Facebook friendship graph to assume that there is only a handful of disjoint latent factors at play is naive. In Figure 1(a), we depicted the connections among 15,000 nodes, i.e., less than 0.00002% of the total network. Even in this small subset of the network, the friendship dynamics are too interconnected and there is simply no global level organization. In cases like this, the traditional community discovery assumption of a global level disjoint partition tends to return not meaningful communities. The typical aim is to cluster the whole structure and return some huge communities and a long list of small branches (see [Coscia et al. 2011]).

[1] http://investor.fb.com/releasedetail.cfm?ReleaseID=761090

However, as we noted, the structure of cohesive groups of nodes emerge easily considering a *local* fragment of an otherwise big network. The key lies in the fact that in a local view there are few latent factors included and they are usually disjoint one from the other. To use a social metaphor, common sense goes that people are good at identifying the reasons why they know the people they know. In network terms, each node has presumably an incomplete, yet clear, vision of the communities it is part of. Being a bearer of a collection of latent factors, that we represent as labels, he connects to its neighbourhood to the nodes bearing the same labels. Then, we can exploit this idea for the CD problem, as 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. Here, it is clear which nodes share which factor, or label, around the ego. Some of these factors are the high school and university friends of the ego, 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 carries all these labels and knows which subsets of its neighbourhood carry one, or more, of these labels too.

Different egos will detect different labels over the same neighbours. The union of all these perspectives, or a hierarchical view in which communities with common labels can be merged together at different aggregation levels, creates an optimal detection of the latent factors 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, because all nodes agree that A and B share the same factors, or labels. If they are considered in the same communities by most or many nodes connected to them, then they are probably part of a higher level super community. This is achieved using a *democratic* bottom-up mining approach: in turn, each node detects the labels attached to the neighbours surrounding it and then all the different perspectives are merged together in an overlapping structure. This overlapping structure can be view as just a flat community coverage, or it can be merged together in a hierarchical fashion.

In the vast CD literature, the general approach does not consider nodes as bearers of different label. The modular structure of a network is usually detected with a (greedy) algorithm, optimizing different quality functions and then returning a set of communities extracted from the global structure (we discuss some of these methods in Section 2). This approach generally ignores the networks latent factors and just operates on the edges of the network without any assumption on why they are distributed as they are. Instead, we propose a change of mentality, in which we make assumptions about the edge distribution and we use these assumptions as drivers of the community discovery process. We propose a simple local-first approach to community discovery in complex networks by letting the latent factors of the 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 identify the labels it shares with different groups of nodes 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 [Raghavan et al. 2007] on this structure, ignoring the presence of the ego itself, whose labels will be evaluated by its peers neighbours. We then combine, with equity, the vote of everyone in the network. The result of this combination is a set of (overlapping) modules, our latent factors, detected not by a top-down approach, but by the actors of the network itself. We then either stop the process here, or we consider again a community as a collection of labels, the ones carried by the nodes composing it,

connected to other communities by the nodes shared with them, that are the common labels between them. In this way, there is no logical distinction between a community and a node, and therefore we can then reapply the same process and obtain an additional level of the community hierarchy. We repeat the process for each hierarchy level until we collapse the entire network in a set of disconnected communities.

To better visualize how our algorithm works, we can imagine to analyse a product network: nodes representing products from a supermarket and edges connecting nodes who share product categories. In the first step, *DEMON* identifies the micro communities to which a specific product belongs: those are sets of other products sharing, for instance, one or more specific types/categories (fruits, meats, vegetables, gloves, shirts. . .), while, in the second step, such sets are merged to identify higher-level category definitions (i.e. food, clothing. . .).

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 [Dean and Ghemawat 2004]; although the post-process FlatOverlap 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 interested in establishing a link between the communities found by *DEMON* and the real world knowledge about the labels attached to the nodes. Intuitively, the two should correspond. This is the primary focused objective of *DEMON*, and we leave other problem definitions to other CD algorithms.

First, we test the performance of the algorithms in an established benchmark setting [Lancichinetti and Fortunato 2009a], able to generate directed and weighted networks with overlapping communities. Second, we confront the performance with real world networks. In this setting, 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. Finally, we provide our latent factor based explanation about why social networks include overlapping communities and why *DEMON* is good in finding them.

This paper is an extension of our previous work [Coscia et al. 2012]. The extension that we present here provides the following additional contributions:

— We already provided in this introduction the theoretical ground on which the algorithm has been built, concerning why networks contain overlapping community, based on the idea of multiple latent factors (i.e. labels) attached to nodes that drive their connectivity;
— We extend the *DEMON* algorithm by introducing the possibility of returning the hierarchical organization of the communities;
— We introduce the experimental evaluation using benchmark networks;
— We extend the experimental section by introducing a new dataset including multidimensional networks [Berlingerio et al. 2012] and dealing with the problem of explaining the overlap in social networks;
— We translate the formulation of the *DEMON* assumption about the latent factors role in the generation of communities, paving the road for the development of better network benchmarks and generators;

— We extend the related work section by placing *DEMON* in a community discovery algorithm classification.

The rest of the paper is organized as follows: in Section 2 we present related works in community discovery literature, placing *DEMON* in the community discovery classification. 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 with benchmark and real world networks are presented in Section 5. In Section 6 we address the problem of explaining the overlap in social network and we lay down the foundations of the *DEMON*-based community logic explanation. Section 7 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 [Fortunato 2010] 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 [Yang and Leskovec 2012], and in the years many different solutions started to explore alternative definitions of communities in complex networks [Coscia et al. 2011].

In [Coscia et al. 2011] the main categories of community discovery are: Feature Distance, Internal Density, Bridge Detection, Diffusion, Closeness, Structure Definition, Link Clustering and Meta Clustering.

The algorithms in the Feature Distance category usually apply some information theory principles by considering the network as a matrix in which each node is represented as a vector of attributes. Then the matrix is clustered according to these features. One example is Cross Associations [Papadimitriou et al. 2008].

In the Internal Density category the idea is to partition the network by maximizing the edge density inside the communities. The majority of methods in this category are based on the *modularity* concept, a quality function of a partition proposed by Newman [Clauset et al. 2004], [Newman 2006]. 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. For instance, two of the issues affecting modularity approaches are the resolution problem and the degeneracy of good solutions [Fortunato and Barthélemy 2007]. One of the most advanced examples of modularity maximization CD is [Mucha et al. 2010], 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 [Blondel et al. 2008].

The Bridge Detection algorithms aim to find similar structure, but with the opposite point of view: they want to separate the communities by identifying the sparser areas of the network. Examples in this category are [Girvan and Newman 2002], [Bagrow and Bollt 2005].

In the Closeness category, Infomap has been proven to be one among the best performing non overlapping algorithms [Lancichinetti and Fortunato 2009b]. In the same category of Infomap there is also the Walktrap algorithm [Pons and Latapy 2006].

A very important property for community discovery is the ability to return an overlapping coverage, 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. The above categories do not consider the possibility of having overlapping communities, and the extensions proposed to do so are usually not universally accepted. The Link Clustering category has been created with this specific focus in mind: the community partition is applied on the edges and then the nodes are part of all the communities of their edges (see [Ahn et al. 2010] and [Evans and Lambiotte 2009]).

An overlapping coverage can be achieved also in the Structure Definition category, that groups together those algorithms that aim to find a given community structure in the network. An example is the k-clique percolation algorithm [Derényi et al. 2005].

Also the Meta Clustering category may provide a way to obtain an overlapping coverage, as it is designed to add community features to community discovery algorithms that are part of different categories. An example of such algorithm is HCDF [Henderson et al. 2010]. *DEMON* places itself in this category, as it merges local communities calculated for every ego network. The algorithm to extract the local communities in the ego networks can be part of any other category. A similar approach that uses ego networks for community discovery can be found in [McAuley and Leskovec 2012].

We consider as last category the Diffusion category, as it is very relevant here. In the present article we used a Diffusion category algorithm as the method to extract the communities from the ego networks. We used the most known approach in this category: Label Propagation [Raghavan et al. 2007]. In [Raghavan et al. 2007] 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 neighbours, 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 coverage.

In Section 6 we show how to apply *DEMON* to understand the origins of the overlap in social communities. This line of research is connected with the analysis of what we call "multidimensional communities": communities that are formed in networks with multiple different relations types (known as multidimensional networks [Berlingerio et al. 2012]). This line of research is connected with the problem of multi-view clustering, explored in [Bickel and Scheffer 2004], [Kumar and III 2011], [Long et al. 2008], [Zhou and Burges 2007]. In complex network, the problem of the definition of network density in multidimensional communities has been addressed in [Berlingerio et al. 2011]. *DEMON* cannot be currently classified as a proper multidimensional community discovery algorithm, as it does not address the problem of multidimensional density and we apply it separately to the different dimensions of the network we study in 6, but its multidimensional development is left as a future work.

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 [Goyal et al. 2009]. Many other works investigate the possibility of applying network analysis for studying, for instance, the dynamics of viral marketing [Leskovec et al. 2007].

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 unweighed 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 and directed

graphs. In our representation of the problem, each node is considered carrying a set of latent labels.

In general terms, our problem definition is to find communities in complex networks. Usually, there is some ambiguity connected to the concept of "community" in a complex network [Coscia et al. 2011]. To solve it, we use the latent labels of the nodes as drivers of the community discovery process. In complex and semantically rich settings (such as the modern Web, social networks or other kinds of complex networks), nodes are complex entities carrying multiple attributes that can make them part of different communities for different reasons. In our problem representation, these reasons are represented by the latent labels. We then need to extend the community discovery problem definition to be able to detect them.

Firstly, 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 neighbours 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 $EgoMinusEgo(v, \mathcal{G})$. Each set $C \in \mathcal{C}(v)$ is grouped according to common latent labels. Each node in C shares more common latent labels with other nodes in C, more than with any other node in $C' \in \mathcal{C}(v)$, with $C \neq C'$.

There are two different ways to go from local communities to *global communities*, bringing together an overlapping community coverage with a hierarchical community structure. These two properties have for long been thought as mutually exclusive, but recent approaches proved this assumption wrong [Ahn et al. 2010].

The first is merging the overlapping communities according to the amount of common latent labels they contain, represented by the fact that they share many nodes. In this scenario, we define the set of communities of a graph $\mathcal{G}$ as:

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

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.

In the second approach we recursively apply our logic by seeing the communities as "super nodes" in the network. Just like the nodes, also the communities are collections of latent labels. Therefore, they can be clustered together according to the labels they share. In this approach, the first level of the hierarchy is the set of all $\mathcal{C}(v)$. Then, all communities in $\mathcal{C}(v)$ are collapsed in a single node. Edges are set between the collapsed communities if the two original communities shared at least one node, weighted proportionally to the number of shared nodes. On this new graph structure, the community discovery is applied again. The procedure is repeated recursively until we find a set of disconnected communities.

4. THE ALGORITHM

In this section we present our solution to the community discovery problem. We first present the core of ***DEMON*** algorithm in Section 4.1, with its corresponding pseudo

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

Require: $\mathcal{G} : (V, E)$; $\mathcal{C} = \emptyset$
Ensure: set of overlapping communities $\mathcal{C}$

```
for all v ∈ V do
    e ← EgoMinusEgo(v, G)
    C(v) ← LabelPropagation(e)
    for all C ∈ C(v) do
        C ← C ∪ v
    end for
end for
return C
```

code in Algorithm 1. Then, we present the two alternative approaches: the Merge function for the flat overlap in Section 4.2 and the Merge function for the hierarchical overlap in Section 4.3. Finally, we present the properties of *DEMON* and its time complexity in Sections 4.4 and 4.5.

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 [Raghavan et al. 2007]. 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 [Coscia et al. 2011].

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 [Raghavan et al. 2007]. Suppose that a node v has neighbours $v_1, v_2, ..., v_k$ and that each neighbour carries a label denoting the community that it belongs to. Then v determines its community based on the labels of its neighbours. 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 neighbours belong. As the labels propagate, densely connected groups of nodes quickly reach a consensus on a unique label. At the end of

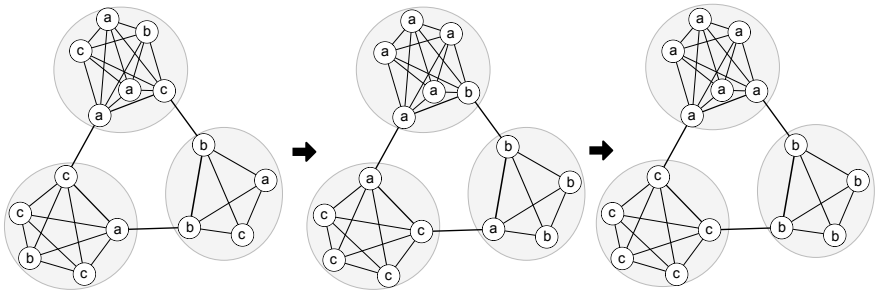


Fig. 2: A simple simulation of the Label Propagation process for community discovery.

the propagation process, nodes with the same labels are grouped together as one community. Clearly, a node with an equal maximum number of neighbours 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:

(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).

At the end of the LP algorithm we reintroduce, in each local community, the ego node v.

The result of Steps #1-7 of Algorithm 1 is a set of local communities $\mathcal{C}$, according to the perspective of all nodes of the network. Please note that there are not repeated communities or communities contained in other communities, as each community has a hash of nodes representing its content. This ensures the first property of *DEMON* that we will see in Section 4.4. However, these communities are likely to be an incomplete view of the real community coverage of $\mathcal{G}$. Thus, the result of *DEMON* needs further processing: to merge each local community of $\mathcal{C}$ in order to obtain a proper community coverage. There are two different versions of the function that carries out this task, called Merge function. The two versions are presented in the following two sections.

4.2. The Flat Overlap Merge

In $FlatOverlap$, two communities C and I are merged if and only if a fraction at most equal to 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 fraction of common elements provided from each couple of communities: $\epsilon = 0$ ensures 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. This procedure is repeated for each community discovered. Returning to the product network example previously introduced, $FlatOverlap$ tries to identify higher level communities by merging two sets of nodes

Algorithm 2 The pseudo-code of FlatOverlap function.

Require: $\mathcal{C} = Local\ community\ set; \epsilon \in [0..1]$
Ensure: set of global overlapping communities $\mathcal{C}$

for all $C \in \mathcal{C}$ **do**
  for all $I \in \mathcal{C}$ **do**
    if $C \subseteq_\epsilon I$ **then**
      $u = C \cup I$;
      $\mathcal{C} - C; \mathcal{C} - I$;
      $\mathcal{C} = \mathcal{C} \cup u$;
    end if
  end for
end for
return $\mathcal{C}$

Algorithm 3 The pseudo-code of HDemon function.

Require: $\mathcal{G} = (V, E), \mathcal{C} = Local\ community\ set$
Ensure: set of global overlapping communities $\mathcal{C}$

$l \leftarrow 0$
$\mathcal{C}_0 \leftarrow \mathcal{C}$
while $|\mathcal{C}_l| > |CC(\mathcal{G})|$ **do**
  for all $C_1 \in \mathcal{C}_l$ **do**
    for all $C_2 \in \mathcal{C}_l$ **do**
      $E \leftarrow E \cup (C_1, C_2, |C_1 \cup C_2|)$
    end for
  end for
  $l \leftarrow l + 1$
  $\mathcal{G}_l \leftarrow (V \leftarrow \mathcal{C}_{l-1}, E)$
  $\mathcal{C}_l \leftarrow DEMON(\mathcal{G}_l, \mathcal{C}_l)$
end while
return $\{\mathcal{C}_0, \mathcal{C}_1, ..., \mathcal{C}_l\}$

if the majority of the products in smallest one also belongs to the bigger one. This iterative procedure aims at identifying groups of products which can describe broader semantic product categories.

4.3. The Hierarchical Extension

The $HDemon$ function starts by obtaining the initial set of local communities $\mathcal{C}$. It puts all these communities in a subset of $\mathcal{C}$ and we refer to it as $\mathcal{C}_0$.

For each pair of community discovered $HDemon$ calculates the amount of shared nodes between the two. It connects the two communities, collapsed in a single node, with an edge whose weight is proportional to this amount. At the end of this procedure, we obtain a higher hierarchical view of the original graph $\mathcal{G}$ that we call $\mathcal{G}_1$.

At this point, $HDemon$ calls again the main core of *DEMON*, this time providing as input not $\mathcal{G}$, but $\mathcal{G}_1$. The resulting local communities of $\mathcal{G}_1$ are put in a separate subset of $\mathcal{C}$ that we call $\mathcal{C}_1$. The procedure is repeated until we find a number of communities in the network lower or equal than the number of connected components of the network ($|CC(\mathcal{G})|$). At this point, we return $\mathcal{C}$, containing all the $\mathcal{C}_0, \mathcal{C}_1, ...\mathcal{C}_i$, representing the hierarchical view of the overlapping communities of $\mathcal{G}$.

4.4. DEMON Properties

To prove the correctness of the *DEMON* algorithm w.r.t. the problem definition in Section 3, we prove by induction some of its properties. It is worthwhile to note that these

properties assume that the results of step #3 are constant. The LP algorithm does not satisfy this requirement, i.e. with different random seeds it will return different results. However, here we just want to prove that *DEMON* holds these properties assuming a constant random seed, because this is the crucial feature that enables the incrementality and ability to be parallelized of *DEMON*.

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

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

where $v_1, \ldots, v_k$ 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 $v_1, \ldots, v_{k-1}$. By always merging a local community C of node v_k into $\mathcal{C}$ if we find a superset of it in $\mathcal{C}$, we guarantee that C is added to the result only if it is not covered by any pre-existing community, and, if added, any pre-existing 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 $v_1, \ldots, v_k$. 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 pre-existing 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 $\mathcal{C}$:*

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

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 pre-existing nodes reached by new links, then

$$\textbf{\textit{DEMON}}(\mathcal{G} \cup \Delta\mathcal{G}, \mathcal{C}) = \textbf{\textit{DEMON}}(\Delta\mathcal{G}, \textbf{\textit{DEMON}}(\mathcal{G}, \mathcal{C})) \quad (5)$$

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.

4.5. 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 $\mathcal{O}(n + m)$ [Raghavan et al. 2007], 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 $k = 1$, thus contributing $\mathcal{O}(0)$ to the complexity; or $k = 2$, thus contributing $\mathcal{O}(1)$. 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 higher 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 $\alpha = 2$, $K << n$).

It has to be noted that this complexity evaluation holds only for the core of the *DEMON* algorithm. The $FlatOverlap$ function is more complex, as it has to merge usually thousands of communities. Currently, we have not developed an efficient solution to this problem, that is then quadratic in the number of nodes and dependent on the ϵ parameter.

5. EXPERIMENTS

In this section we investigate the performance improvement that *DEMON* provides over the state of the art of community discovery. First, in Section 5.1 we generate synthetic networks with an established network benchmark generator [Lancichinetti and

Parameter	Description	Value
N	Number of nodes	1,000
k	Average degree	25
Max k	Maxiumum degree	50
μ	Mixing	0.01
Min c	Minimum community size	20
Max c	Maximum community size	50
On	Number of overlapping nodes	500
Om	Number of communities of overlapping nodes	3

Table I: Parameter choice for the benchmark analysis.

Fortunato 2009a] to evaluate the quality of the community coverage with a known artificial community structure, as a standard robustness check. Then, we switch to three real world networks in Section 5.2, namely networks extracted from bill co-sponsorship in the US Congress, the Internet Movie Database and Amazon. We also provide some examples of the insights that is possible to extract from the flat overlap communities extracted with *DEMON* (Section 5.3) as well as from the hierarchical version of the algorithm (Section 5.4).

The selected competitors for our assessment are: Hierarchical Link Clustering (HLC) [Ahn et al. 2010], that has been proven able to outperform all the overlapping algorithms, including the k-clique Propagation algorithm by Palla et al [Derényi et al. 2005]; and two overlapping algorithms, the first based on Label Propagation (SLPA [Xie and Szymanski 2012], [Xie et al. 2011]) and the second on non-negative matrix factorization (BigClam [Jaewon and Leskovec 2013]).

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 decreasing ϵ values can take from one minute to one hour.

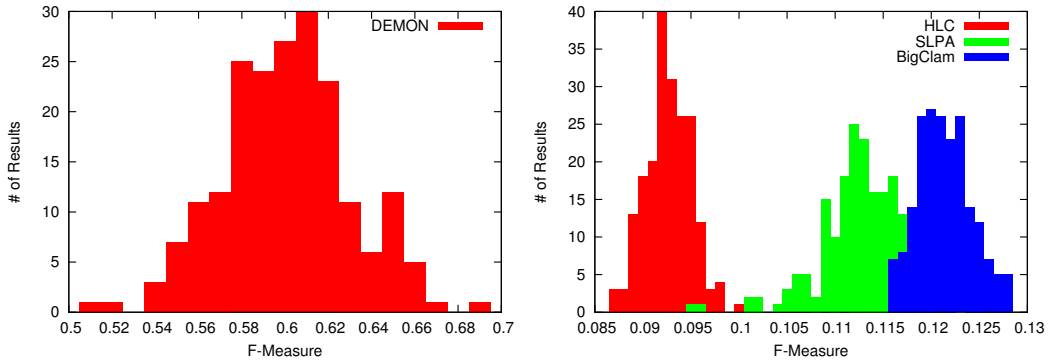
5.1. Performances on Benchmark Networks

In this section we assess the quality of the community coverage extracted with *DEMON* using synthetic benchmark networks. The usage of the benchmark networks is useful as we can plug a known community structure and evaluate how well the algorithm is able to uncover it. Of course there are several limitations: as we saw in Section 2 there are many different community definitions and benchmark networks can only cover a few. This problem is solved by checking the community discovery quality also in real world networks, and we do it in Section 5.2.

Another problem is that usually benchmark networks are generated in very simple scenarios, i.e. assuming that the network is unweighed, undirected and with non-overlapping community. For this reason, we adopt a benchmark network generator that is able to provide directed, weighted and overlapping networks, that has been introduced in [Lancichinetti and Fortunato 2009a].

For each community discovery algorithm we want to test, we generate 200 benchmark networks with a known community structure. The generator developed in [Lancichinetti and Fortunato 2009a] requires to specify how many overlapping nodes are in the networks and to how many communities they belong. The parameter choice for the benchmark has been reported in Table I for experiment repeatability purposes.

[2]http://www.michelecoscia.com/?page_id=42



(a) DEMON (b) Other Methods

Fig. 3: The f-measure distribution for the 200 tested benchmark networks.

The generator outputs the network and the communities each node belongs to. Given this information, we can calculate the f-measure between the coverage returned by the community discovery and the real communities of the benchmark network.

Since this is a many-to-many matching problem we adopt a simple strategy of matching the discovered and the original communities. For each discovered community we calculate the f-measure with all the real communities. The community that maximizes the f-measure is the corresponding community and we use that to calculate the average f-measure of the community coverage.

We report in Figure 3 the results of our evaluation. Figure 3 reports for how many benchmark networks (y axis) we obtained a given value of f-measue (x-axis). Figure 3(a) reports the distribution for *DEMON*, while Figure 3(b) reports for all the other methods. We can see from the x axis of the two figures that *DEMON* clearly performs on average significantly better than the other tested algorithms, although it is less stable as its performance is 0.6 ± 0.1 while the other methods' deviation spans from 0.01 to 0.03. In any case, from Figure 3 we can conclude that *DEMON* is able to create a much clearer one to one correspondence with the communities in the benchmark networks.

5.2. Performances on Real-World Networks

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. This annotated information is an external explicit information about the latent labels attached to the nodes which drive their connectivity, as explained in the introduction and in Section 3.

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.

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 SLPA was not able to conclude in reasonable times.

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 II: Basic statistics of the studied 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 II, 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 [Ahn et al. 2010], 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:

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.

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 we are dealing with 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 multi-label classifier, i.e. a learner able to predict multiple target attributes [Tsoumakas and Katakis 2007]. 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, 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

[3]http://www.govtrack.us/developers/data.xpd
[4]http://www.imdb.com/interfaces
[5]http://snap.stanford.edu/data/index.html

Measure	Network	*DEMON*	HLC	BigClam	SLPA
F-Measure	Congress	**0.21275**	0.14740	0.08987	0.03461
	IMDb	**0.44252**	0.43078	0.38520	0.35431
Accuracy	Congress	**0.10351**	0.08038	0.04420	0.01670
	IMDb	0.34106	**0.38113**	0.33373	0.32011

Table III: The F-Measure scores for Congress and IMDb dataset and each community coverage.

Network	Demon		HLC		BigClam		SLPA	
	$\lvert\mathcal{C}\rvert$	$\lvert\bar{c}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\bar{c}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\bar{c}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\bar{c}\rvert$
Congress	425	63.3671	1,476	4.5867	99	19.4545	2	263
IMDb	14,004	12.6824	88,119	8.3426	16,411	7.66462	6,717	8.7688

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

with a single-label multi-class problem using a binary classifier. 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:

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

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

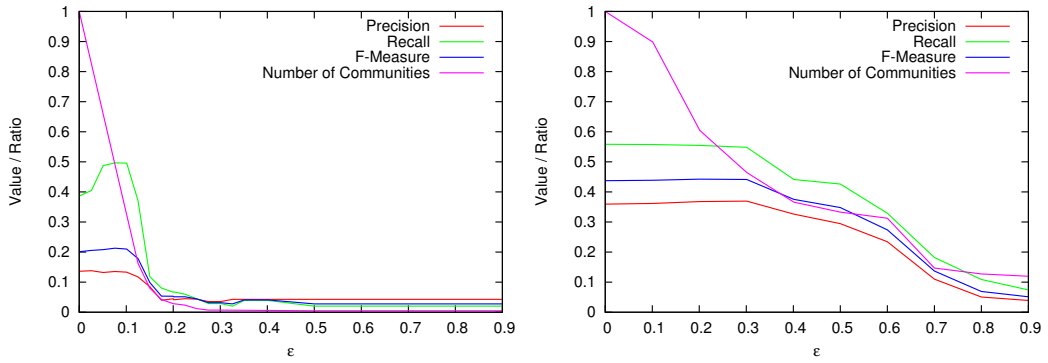
We then derive the F-measure from Precision and Recall. We also calculate the multi-label equivalent of Accuracy, that is:

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

These multi-label evaluations are described in [Godbole and Sarawagi 2004]. The results are reported in Table III and show that *DEMON* comes in first for most tests, and in second just in one case. 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 outscore *DEMON* in Accuracy. However, there is an important distinction to be made about the quantity of the results: if the com-

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



(a) Congress (b) IMDb

Fig. 4: Precision, Recall, F-Measure and number of communities for different ϵ values.

munity discovery returns too many communities, then it is difficult to actually extract useful knowledge from them. We reported in Table IV the basic statistics about the community coverages 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, *DEMON* scores are achieved returning 70-80% less communities than HLC.

We report in Table IV 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 report the results in Figure 4. We can see that for both Congress and IMDb the Precision, Recall and F-Measure scores remain 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 the F-Measure 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 final consideration is needed about the size distribution of the communities detected by *DEMON* and the other community discovery algorithms. In Figure 5 we depicted the community size distribution for *DEMON* and BigClam for the IMDb network. While Bigclam returned more or less the same number of communities of *DEMON*, we can see that these communities are concentrated in the head of the distribution, i.e. they are on average very small. This is especially true if we consider that these small communities mostly disappear for increasing ϵ thresholds. Communities smaller than a handful nodes are usually less significant and they are often the results of an artefact of the algorithm.

We can conclude that *DEMON* with a manageable number of medium-sized 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.

As presented at the beginning of this section, the networks studied here possess *qualitative attributes*, i.e. a set of annotations attached to each node. Assuming that these qualitative attributes correspond to the node's latent factors, we assume 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

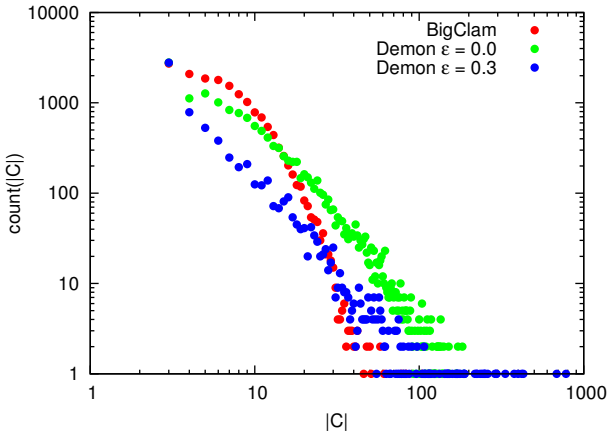


Fig. 5: The distribution of the community sizes for DEMON and BigClam in the Amazon network.

exclusively) dependent on the graph structure. What we want evaluate is not how a graph measure is maximized, but how good is our community coverage in describing real world knowledge about the clustered entities.

We quantify the matching between a community coverage and the metadata by evaluating how much higher are on average the Jaccard coefficients 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 V the average value of the CQ obtained. Also in this case, *DEMON* was able to outperform all the other algorithms in three out of four cases, ending up second in one case, this time to BigClam instead of HLC.

We also calculated the Overlapping Mutual Information between the set of communities selected as described above and the collection of labels attached to the nodes. Traditional Mutual Information is not defined for a multi-label setting. Some researchers defined a Normalized Mutual Information for this purpose [Lancichinetti et al. 2009]. However, further research [McDaid et al. 2011] pointed out that this version of the Normalized Mutual Information has some drawbacks, namely its unintu-

Measure	Network	*DEMON*	HLC	BigClam	SLPA
CQ	Congress	1.1792	1.1539	**1.2737**	0.9508
	IMDb	**5.6158**	5.1589	0.5954	0.2020
ONMI	Congress	**0.0172**	0.0083	0.0051	0.0127
	IMDb	**0.0536**	0.0429	0.0280	0.0234

Table V: The Community Quality scores for Congress and IMDb dataset and each community coverage.

Fig. 6: A representation of parts of the two communities surrounding our case study in the amazon network.

itive behaviour. For this reason, we used the measure defined in [McDaid et al. 2011] and we refer to it as "ONMI". The results of our experiments are again reported in Table V. This time, we can observe that *DEMON* is able to outperform all the considered alternatives.

5.3. Overlapping Communities

In this section we present a brief case study using the communities extracted for the previously exposed evaluation of *DEMON*, that uses the *FlatOverlap* function to merge the communities. We focus on the Amazon network. Aim of the section is to demonstrate that the overlap between the extracted communities carries meaningful information. By analysing the overlap, we can have practical applications in the extraction of knowledge from real world scenarios. In the next section we focus instead on the hierarchy of the communities, rather than their overlap.

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.

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 6). 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

Rank	Level 0	Level 1	Level 2
1	Sen. Com. on Commerce, Science, and Transp.	Sen. Com. on Foreign Relations	Sen. Com. on Commerce, Science, and Transp.
2	Sen. Com. on Foreign Relations	Sen. Com. on Commerce, Science, and Transp.	Sen. Com. on Indian Affairs
3	International scientific cooperation	Sen. Spec. Com. on Aging	Sen. Spec. Com. on Aging
4	Office of Science and Technology Policy	Sen. Com. on Environment and Public Works	Sen. Com. on Health, Education, Labor, and Pensions

Table VI: The top four topics of one community in the Congress network across the hierarchy.

Twenty-First Century America". This is clearly a community composed mainly by purchases made by the people more interested in the socio-economic 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 ones 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").

5.4. Hierarchical Communities

The aim of this section is to demonstrate that, besides the overlap, also the hierarchy of the extracted communities carries meaningful information. In this case we focus on the Congress network, as the US Congress has a particular structure that is easy to confront with. In the US, the Congress is divided in two parts: the House and the Senate. Members of the House are not members of the Senate and vice versa. Then, inside both the House and the Senate, there are several subcommittees, each with a different focus. We expect to find members of similar subcommittees in the communities at the lower level of the hierarchy, and just two large communities at the top of the hierarchy: the community of the House and the community of the Senate.

We report in Table VI the first four topics of a particular community across the entire hierarchy. As we can see, at the bottom level this community is clearly a senate community composed by a small group of senators very focused on science and technology. In the intermediate and top level, the community merges with more and more general communities from the senate. At level 1 is still focused on broader social issues, while at level 2 it is basically composed by almost any senate committee.

At level 2, we expected to find only two communities. We found, instead, 28 of them. However, these 28 communities are easily split into the two expected groups with little overlap. Since *DEMON* does not return any community with less than three nodes, it does not create the additional hierarchy level with the two nodes representing the House and the Senate communities.

6. THE OVERLAP IN SOCIAL NETWORKS

6.1. Explaining the Overlap

As we saw in the previous sections and in countless other examples in literature, overlapping communities are ubiquitous in many social and complex networks. We chose as our explanation of the overlap the fact that many different latent factors drive the nodes' connectivity. Nodes are collections of latent labels and they tend to connect with nodes with similar labels. This is the assumption we share with the creators of the BigClam algorithm [Jaewon and Leskovec 2013].

Network	Nodes	Edges	Facebook	Twitter	Foursquare
Facebook	2,081	5,618	1	0.57	0.94
Twitter	3,745	31,638	0.32	1	0.85
FourSquare	5,783	42,691	0.34	0.55	1
Total	7,461	79,947	-	-	-

Table VII: The statistics of our multidimensional network per dimension.

This corresponds to the most successful explanation used for overlapping communities: in a network there are different types of relations whose interplay brings together people from different communities (the starting assumption of the HLC algorithm [Ahn et al. 2010]). For example, one person is part of the community of her college mates and also of the sport team she practices. Several of her team-mates may be also college mates, generating an overlap between the two communities. In this section we do not focus on a systematic proof of this theory, that goes beyond the scope of this paper. We provide, instead, empirical evidences of this theory, along with the proof that *DEMON* is able to correctly detect actual overlap.

To do so, we need to add context information to the relationships connecting two people. One of the most important approach to this problem in literature involves using multidimensional networks. Multidimensional networks are networks in which we have multiple different relations [Berlingerio et al. 2012]. Community discovery in multidimensional networks is a problem studied in [Berlingerio et al. 2011] and it requires specialized multidimensional community discovery algorithms. *DEMON* is not a multidimensional community discovery algorithm, so we need to create a special analytic setting.

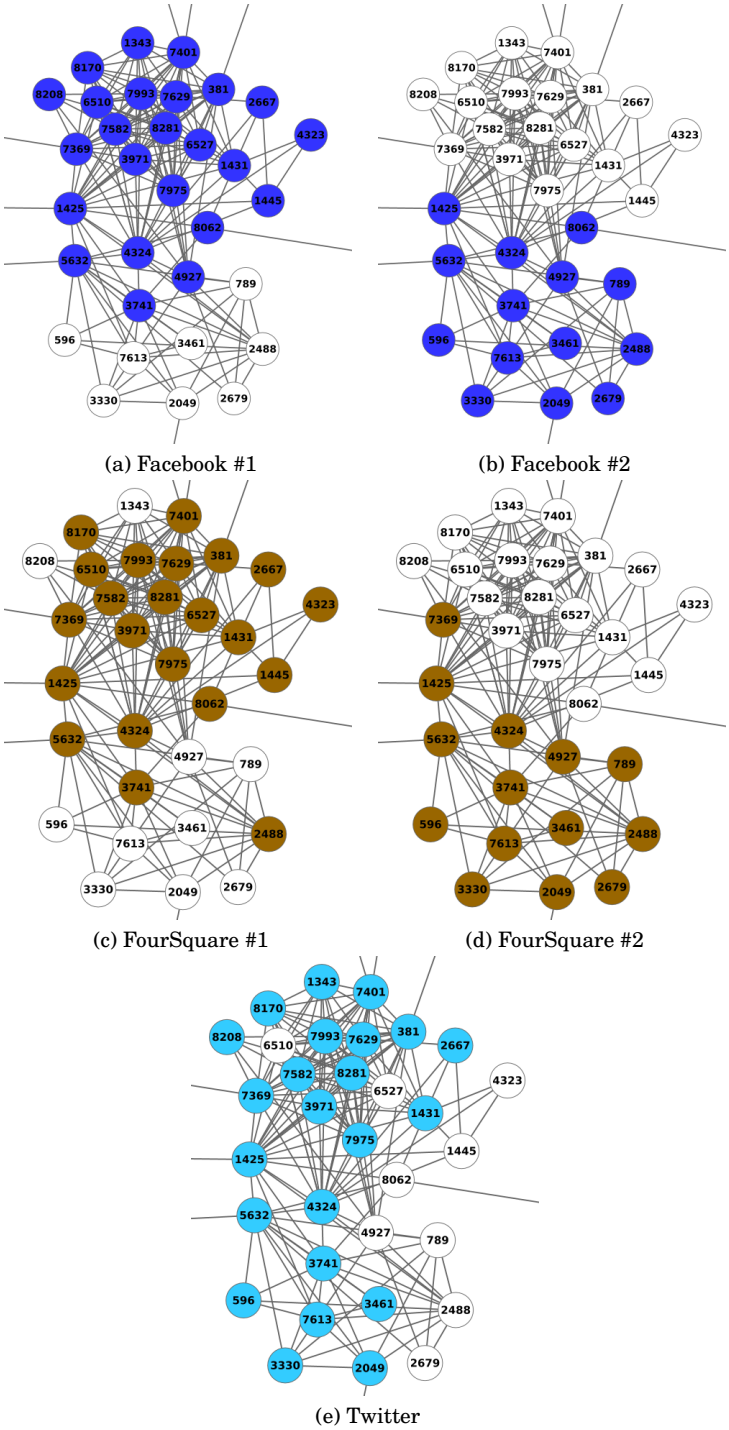
We create a multidimensional network by joining three different social networks: Facebook, Twitter and Foursquare. We were able to crawl the relationships of the same set of users in these three social media websites. Table VII records some statistics about the topology of the three social networks and their aggregate multidimensional network. For each dimension of the network we report the number of nodes and edges appearing in it and the node overlap with the other network dimensions.

We then applied *DEMON* to each dimension of the network separately. The algorithm found overlapping communities for each dimension of the network without information about the edges from the other dimensions. The aim of our analysis is to examine communities with a large overlap in one dimension and verify the community affiliations in the other dimensions of the network of the nodes belonging to these two communities.

An example of this analysis is depicted in Figure 7. In Figures 7(a-e) we depicted the same set of 32 nodes with the edges connecting them in the Facebook dimension. Figures 7(a-b) depicts two communities extracted by *DEMON* in the Facebook dimension by colouring in blue the nodes belonging to them, leaving in white the remaining nodes. We can see that the two communities share an overlap of six nodes.

In Figures 7(c-d) we depict two overlapping communities in the FourSquare dimension (highlighted with a brown color). Please note that the actual edges depicted are from the Facebook network, making the comparison of the nodes' community affiliations more clear. We can see that the two FourSquare communities are still overlapping, but with different common elements: by interacting with their Facebook friends that are part of the overlap in the Facebook direction, some users visit the same places of other users that are not directly their friend, creating a new community.

Finally, in Figure 7(e) we have the community extracted from Twitter network (light blue color). We can see that, in Twitter, the two overlapping communities are actually one single community, with the exception of some nodes that do not use the Twitter



(a) Facebook #1

(b) Facebook #2

(c) FourSquare #1

(d) FourSquare #2

(e) Twitter

Fig. 7: The overlapping communities in the three dimensions of the network.

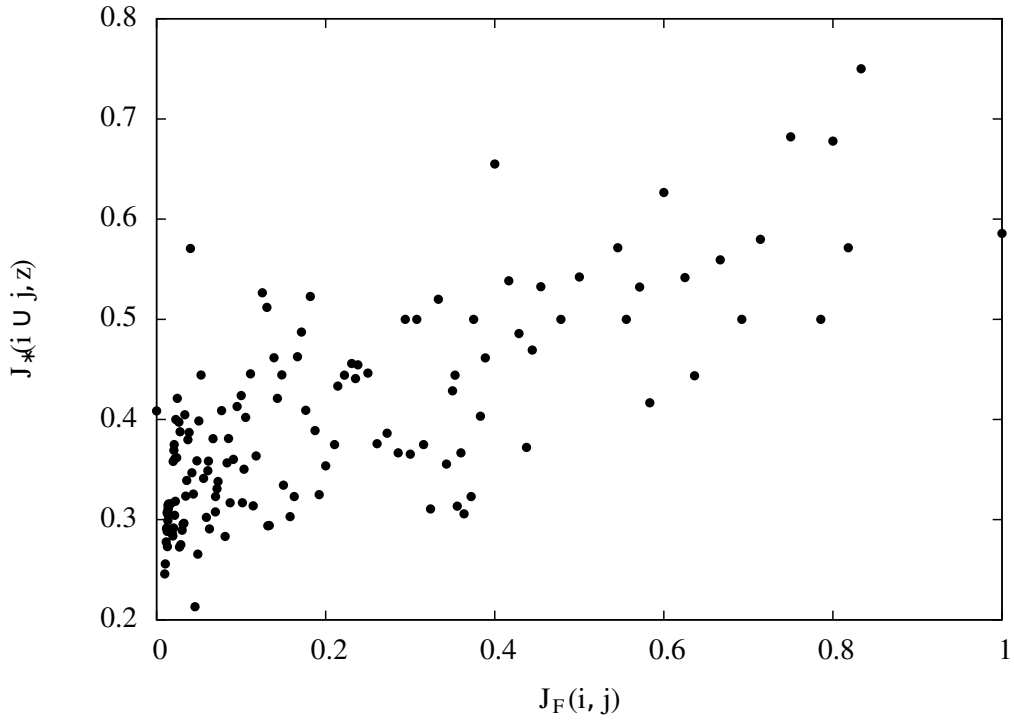


Fig. 8: The relationship between $J_F(i,j)$ and $J_*(i \cup j, z)$ for the communities in the Facebook dimension.

service. The overlap in the Facebook dimension is likely to be playing a role here: even if user a is not friend of user b, he still may be interested in user b's tweets, as many of user a's friends are friend with user b.

This is only one example out of many that can be found. To prove this, we took each community couple (i,j) out of the 367 communities we found in the Facebook dimension. We calculated the Jaccard index $J_F(i,j)$ of the two communities, i.e. their degree of overlap. Then, we joined the two communities and we calculated the Jaccard index between the community union $i \cup j$ and each community in the Twitter and Foursquare dimension, i.e. how much the two overlapping communities are part of a single community in another dimension. We refer to this quantity as $J_*(i \cup j, z)$. We plot the relationship between $J_F(i,j)$ and $J_*(i \cup j, z)$ in Figure 8. Since many couples of communities may have the same $J_F(i,j)$ value, we took the average of all $J_*(i \cup j, z)$ for each $J_F(i,j)$ value. As we can see, the larger the overlap in the Facebook dimension, the more the two communities are included in a single community in another dimension.

6.2. The DEMON-based Explanation of Communities

In the previous sections, we saw that the approach implemented by *DEMON* is able to better uncover the community structure implied in real world networks. We can conclude that there is a correlation between how the overlap in communities forms in the real world and how *DEMON* is able to detect the overlap. As a consequence, by explaining how *DEMON* detects the overlap we may have an insight about how the overlap works in the real world.

Algorithm 4 The pseudo-code of the NeGen DEMON.

Require: α, β, γ, $|V|$
Ensure: set of nodes and edges $\mathcal{G} = (V, E)$
1: $V \leftarrow POWERLAW(\alpha)$
2: $SORTDEGDESC(V)$
3: **for all** $v \in V$ **do**
4: **if** $deg'(v) > 1$ **then**
5: $n \leftarrow deg'(v)$
6: $deg'(v) \leftarrow 0$
7: $N \leftarrow RANDOMSAMPLE(V, n)$
8: $E \leftarrow CONNECTNEIGHBORS(v, N)$
9: $E \leftarrow E \cup COMMUNITIES(v, N, \beta, \gamma)$
10: $UPDATE(deg'(N))$
11: **end if**
12: **end for**
13: **return** $\mathcal{G}$

We decided to translate this idea into a generative model. In other words, we can use the principle of *DEMON* to generate synthetic networks. As a by-product, we will have benchmarks for other overlapping community discovery algorithms that reflect better the overlap mechanics of social networks, when these mechanics matches with our theory of connectivity driven by latent factors.

This is a useful track of research as most of the benchmark networks for community discovery do not generate overlapping communities. The benchmark network presented in [Lancichinetti and Fortunato 2009a] does, and we used it in the previous section. However, there are some shortcomings included in that method. First, it is mandatory to specify in how many communities the overlapping nodes lie, and each node will belong to exactly the same number of communities, which is unrealistic. Second, it is mandatory to specify how many nodes are part of more than one community and how many are not, an information that is difficult to understand if we want to model real world networks.

DEMON starts from the assumption that communities are generated locally around each node. Therefore, it expects to find in the ego network of each node a set of well-separated semi-cliques. We describe the Network Generator based on *DEMON* in Algorithm 4. First, for each $v \in V$ we extract its number of neighbours, by generating a power law degree distribution with exponent α (step #1). So, for each node v we keep in $deg'(v)$ the number of connections that v is accepting.

Then we cycle over the nodes, starting from the ones with higher degree (steps #2-3). We first check that the node v is still accepting connections (step #4). In steps #5-7 we randomly select from V $n = deg'(v)$ nodes that will be neighbours of v, and we set $deg'(v) = 0$. We connect these n nodes with v (step #8) and then we generate the communities around v using the function $COMMUNITIES$ (step #9). Of course this will modify the number of connections that can be still attached to the extracted nodes, thus we update their deg' values in step #10.

How we create the local communities in the ego network is the task of the $COMMUNITIES$ function, and it pseudocode is reported in Algorithm 5. First we define the distribution of the community size around a node. We assume this distribution to be normal with an average equal to β. Then, for each community we extract randomly a number of nodes equal to its size and we connect them with probability γ (that should be larger than 0.5).

Algorithm 5 The COMMUNITIES routine of NeGen DEMON.

Require: v, N, β, γ
Ensure: set of edges E'
1: $C \leftarrow NORMAL(v, \beta)$
2: **for all** $c \in C$ **do**
3: $m \leftarrow RANDOMSAMPLE(N, size(c))$
4: $E' \leftarrow RANDOMCONNECT(m, \gamma)$
5: **end for**
6: **return** E'

Using this algorithm, it is possible to generate well-separated local communities in the ego networks of each node. These communities will eventually overlap when each neighbour of a previously considered node will generate its own local communities.

7. CONCLUSION AND FUTURE WORK

In this paper we proposed to see the emergence of overlapping communities in complex networks as the effect of latent factors driving nodes' connectivity. Based on this assumption, we created a new method for solving the problem of detecting this latent knowledge from significant communities in complex networks. We propose a democratic approach, where the peer nodes judge if their neighbours should be clustered together. We extended previous work by creating a consistent theoretical ground for our method. We extended the algorithm to find hierarchical communities and we have provided evidences that the underlying assumption of the work could be correct.

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 coverage, a community organization that allows nodes to be in different communities at the same time, is improved w.r.t state-of-the-art algorithms, evaluated using both a standard synthetic network generator and real world networks, in which we use the communities to predict the metadata attached to the nodes. We also show that the performances of the algorithm are useful to shed some light about how and why social communities are overlapping, by analysing a multidimensional network and providing the intuition that *DEMON* can give us about how overlapping communities form.

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, or just have an improved time efficiency; 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.

REFERENCES

Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann. 2010. Link communities reveal multiscale complexity in networks. *Nature* 466, 7307 (June 2010), 761–764. DOI:http://dx.doi.org/10.1038/nature09182

James P. Bagrow and Erik M. Bollt. 2005. Local method for detecting communities. *Physical Review E* 72, 4 (Oct. 2005), 046108+. DOI:http://dx.doi.org/10.1103/PhysRevE.72.046108

Michele Berlingerio, Michele Coscia, and Fosca Giannotti. 2011. Finding and Characterizing Communities in Multidimensional Networks. In *ASONAM*. 490–494.

Michele Berlingerio, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. 2012. Multidimensional networks: foundations of structural analysis. *World Wide Web* (2012), 1–27. DOI:http://dx.doi.org/10.1007/s11280-012-0190-4

Steffen Bickel and Tobias Scheffer. 2004. Multi-View Clustering. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM '04)*. IEEE Computer Society, Washington, DC, USA, 19–26. http://dl.acm.org/citation.cfm?id=1032649.1033432

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *J.STAT.MECH.* (2008), P10008. doi:10.1088/1742-5468/2008/10/P10008

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

Aaron Clauset, M. E. J. Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical Review E* 70 (2004), 066111. doi:10.1103/PhysRevE.70.066111

Michele Coscia, Fosca Giannotti, and Dino Pedreschi. 2011. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining* 4, 5 (2011), 512–546. DOI:http://dx.doi.org/10.1002/sam.10133

Michele Coscia, Giulio Rossetti, Fosca Giannotti, and Dino Pedreschi. 2012. DEMON: a local-first discovery method for overlapping communities. In *KDD*. 615–623.

Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. *OSDI* (2004), 137–150. http://www.usenix.org/events/osdi04/tech/dean.html

Imre Derényi, Gergely Palla, and Tamás Vicsek. 2005. Clique Percolation in Random Networks. *Physical Review Letters* 94, 16 (April 2005), 160202+. http://dx.doi.org/10.1103/PhysRevLett.94.160202

T. S. Evans and R. Lambiotte. 2009. Line graphs, link partitions, and overlapping communities. *Physical Review E* 80, 1 (July 2009), 016105+. DOI:http://dx.doi.org/10.1103/physreve.80.016105

S. Fortunato. 2010. Community detection in graphs. *Physics Reports* 486 (Feb. 2010), 75–174. DOI:http://dx.doi.org/10.1016/j.physrep.2009.11.002

Santo Fortunato and Marc Barthélemy. 2007. Resolution limit in community detection. *PNAS* 104, 1 (Jan. 2007), 36–41. http://dx.doi.org/10.1073/pnas.0605965104

M. Girvan and M. E. J. Newman. 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* 99, 12 (June 2002), 7821–7826. DOI:http://dx.doi.org/10.1073/pnas.122653799

Shantanu Godbole and Sunita Sarawagi. 2004. Discriminative Methods for Multi-labeled Classification. In *PAKDD*. 22–30.

Amit Goyal, Byung-Won On, Francesco Bonchi, and Laks V. S. Lakshmanan. 2009. GuruMine: A Pattern Mining System for Discovering Leaders and Tribes. *ICDE* 0 (2009), 1471–1474. DOI:http://dx.doi.org/10.1109/ICDE.2009.59

Keith Henderson, Tina Eliassi-Rad, Spiros Papadimitriou, and Christos Faloutsos. 2010. HCDF: A Hybrid Community Discovery Framework. In *SDM*. 754–765.

Yang Jaewon and Jure Leskovec. 2013. Overlapping community detection at scale: A nonnegative matrix factorization approach. In *WSDM*.

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

Abhishek Kumar and Hal Daumé III. 2011. A Co-training Approach for Multi-view Spectral Clustering. In *ICML*. 393–400.

Andrea Lancichinetti and Santo Fortunato. 2009a. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E* 80, 1 (July 2009), 016118. DOI:http://dx.doi.org/10.1103/PhysRevE.80.016118

A. Lancichinetti and S. Fortunato. 2009b. Community detection algorithms: A comparative analysis. *Physical Review E* 80, 5 (Nov. 2009), 056117–+. DOI:http://dx.doi.org/10.1103/PhysRevE.80.056117

Andrea Lancichinetti, Santo Fortunato, and Jnos Kertsz. 2009. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics* 11, 3 (2009), 033015. http://stacks.iop.org/1367-2630/11/i=3/a=033015

Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. 2007. The dynamics of viral marketing. *ACM Trans. Web* 1 (May 2007). http://dx.doi.org/10.1145/1232722.1232727

Bo Long, Philip S. Yu, and Zhongfei Zhang. 2008. A General Model for Multiple View Unsupervised Learning. In *Proceedings of the 2008 SIAM International Conference on Data Mining*.

Julian J. McAuley and Jure Leskovec. 2012. Learning to Discover Social Circles in Ego Networks.. In *NIPS*, Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger (Eds.). 548–556. http://dblp.uni-trier.de/db/conf/nips/nips2012.html#McAuleyL12

Aaron F. McDaid, Derek Greene, and Neil Hurley. 2011. Normalized Mutual Information to evaluate overlapping community finding algorithms. (Oct. 2011). http://arxiv.org/abs/1110.2515

Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and J-P Onnela. 2010. Community Structure in Time-Dependent, Multiscale, and Multiplex Networks. *Science* 328, 5980 (2010), 876–878. http://dx.doi.org/10.1126/science.1184819

M. E. J. Newman. 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* 103, 23 (June 2006), 8577–8582. DOI:http://dx.doi.org/10.1073/pnas.0601602103

Spiros Papadimitriou, Jimeng Sun, Christos Faloutsos, and Philip S. Yu. 2008. Hierarchical, Parameter-Free Community Discovery. In *ECML PKDD*. 170–187. DOI:http://dx.doi.org/10.1007/978-3-540-87481-2_12

Pascal Pons and Matthieu Latapy. 2006. Computing Communities in Large Networks Using Random Walks. *J. Graph Algorithms Appl.* 10, 2 (2006), 191–218.

Usha N. Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E* 76, 3 (Sept. 2007), 036106+. DOI:http://dx.doi.org/10.1103/PhysRevE.76.036106

Jianhua Ruan and Weixiong Zhang. 2007. An Efficient Spectral Algorithm for Network Community Discovery and Its Applications to Biological and Social Networks. *Data Mining, IEEE International Conference on* 0 (2007), 643–648. DOI:http://dx.doi.org/10.1109/ICDM.2007.72

G. Tsoumakas and I. Katakis. 2007. Multi Label Classification: An Overview. *International Journal of Data Warehousing and Mining* 3, 3 (2007), 1–13. http://mlkd.csd.auth.gr/publication_details.asp?publicationID=219

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

J. Xie and B. Szymanski. 2012. Towards Linear Time Overlapping Community Detection in Social Networks. *PAKDD* (2012).

Jierui Xie, Boleslaw K. Szymanski, and Xiaoming Liu. 2011. SLPA: Uncovering Overlapping Communities in Social Networks via a Speaker-Listener Interaction Dynamic Process. In *ICDM Workshops*. 344–349.

Jaewon Yang and Jure Leskovec. 2012. Defining and Evaluating Network Communities based on Ground-truth. (Nov. 2012). http://arxiv.org/abs/1205.6233

Dengyong Zhou and Christopher J. C. Burges. 2007. Spectral clustering and transductive learning with multiple views. In *Proceedings of the 24th international conference on Machine learning (ICML '07)*. ACM, New York, NY, USA, 1159–1166. DOI:http://dx.doi.org/10.1145/1273496.1273642