An empirical comparison of algorithms to find communities in directed graphs and their application in Web Data Analytics

Santa Agreste, Pasquale De Meo, Giacomo Fiumara, Giuseppe Piccione, Sebastiano Piccolo, Domenico Rosaci, Giuseppe M. L.Sarné, and Athanasios Vasilakos,

Abstract—Detecting communities in graphs is a fundamental data analysis tool to understand the structure of Web-based systems and predict their evolution. Many community detection algorithms handle *undirected graphs* (i.e., graphs with bidirectional edges) but many graphs on the Web – e.g. microblogging Web sites, trust networks or the Web graph itself – are often *directed*. Few community detection algorithms deal with directed graphs but we lack their experimental comparison. In this paper we evaluated some community detection algorithms across accuracy and scalability. A first group of algorithms (Label Propagation and Infomap) are explicitly designed to manage directed graphs while a second group (e.g., WalkTrap) simply ignores edge directionality; finally, a third group of algorithms (e.g., Eigenvector) maps input graphs onto undirected ones and extracts communities from the symmetrized version of the input graph. We ran our tests on both artificial and real graphs and, on artificial graphs, WalkTrap achieved the highest accuracy, closely followed by other algorithms; Label Propagation has outstanding performance in scalability on both artificial and real graphs. The Infomap algorithm showcased the best trade-off between accuracy and computational performance and, therefore, it has to be considered as a promising tool for Web Data Analytics purposes.

Index Terms—Web Data Analytics, Graph Analytics, Community Detection and Clustering, Directed Graphs.

1 Introduction

1.1 Motivations

Web Data Analytics greatly takes advantage of unprecedented scientific and technological progresses in the management of large graphs [1], [2]. Many Web-based platforms, in fact, consist of a multitude of independent but interacting functional units; despite their complexity, Web-based platforms can be modelled as graphs in which vertices correspond to single units and edges model a relationship between the vertices they connect.

Relevant examples of Web-based platforms which can be modelled through graphs are collections of Web pages and their hyperlinks, Facebook users and their friendship relationships, members of a product review Web site and their trust relationships.

The graph-based representation of Web-based platforms is conducive to classify Web Data Analytic tools in two broad categories, namely: *(i) microscopic-level* and *(ii) macroscopic-level*.

Microscopic-level tools include analytical procedures which focus on the role of vertices and edges: a significant example of microscopic-level analysis consists of spotting *influencers* in a Social Web platform, i.e., those users who start new trends and shape opinions [3].

Macroscopic-level tools are based on the fact that vertices create links with other vertices on the basis, for instance, of their distinctive features. Such an aggregation process leads to the creation of *communities* (or *clusters*), i.e., groups of vertices displaying a high internal edge density and which appear well separated each other. Detecting communities is a indispensable tool to understand what are the main constituents of a Web-based platform, how these constituents interact and their role on the evolution and failure of the whole system [4], [5], [6]. As an example, communities extracted from a corpus of Web pages correspond to groups of pages that share a common topic and, then, finding out communities in the corpus is effective to improve precision and recall of a Web search engines [6]. Similarly, communities extracted from members of a Social Web platform identify close-knit groups of like-minded individuals [7]: a business application could leverage the knowledge of communities to disseminate commercial advertisements only to those individuals who are likely to be interested in that advertisement, thus increasing its visibility. Other fields taking advantage of community-level analysis are, for instance, collaborative tagging [8], social recommender systems [9]

- *S. Agreste, G. Fiumara G. Piccione are with the Department of Mathematics and Computer Science, University of Messina, Italy. E-mail: {sagreste, gfiumara}@unime.it piccione.giu@gmail.com*
- *P. De Meo is with the Department of Ancient and Modern Civilizations, University of Messina, Italy. E-mail: pdemeo@unime.it*
- *S. Piccolo is with the Department of Department of Management Engineering, Production and Service Management - Engineering Systems Group at the Danish Technical University (DTU), Lingby, Denmark E-mail: sebpi@dtu.dk*
- *D. Rosaci is with the DIIES Department, University of Reggio Calabria Via Graziella, Loc. Feo di Vito, 89122, Reggio Calabria, Italy. E-mail: domenico.rosaci@unirc.it*
- *G.M.L. Sarné is with the DICEAM Department, University of Reggio Calabria Via Graziella, Loc. Feo di Vito, 89122, Reggio Calabria, Italy. E-mail: sarne@unirc.it*
- *A. Vasilakos is with Department of Computer Science, Electrical and Space Engineering, Lulea University of Technology, SE-931 87 Skelleftea, Sweden E-mail: athanasios.vasilakos@ltu.se*

Manuscript received XX; revised October YY.

and query processing [10].

Microscopic- and macroscopic-level analysis methods are frequently in symbiosis, thus enabling big opportunities for cross-fertilization: for instance, [11] applied community detection algorithms as a preliminary step to accurately predict the formation of new links in social networks; analogously, the identification of the most important vertices in a graph – according to some definition of importance – has been employed to partition the graph itself [12].

Most of the approaches to discovering communities focus on *undirected graphs*, i.e., graphs whose edges model *symmetric relationships* between the vertices they bind [13], [14]. However, many of the graphs available on the Web are *directed* because some edges go from an entity to another but not vice versa. Relevant examples of directed graphs on the Web are snapshots of the Web graph, trust networks and *who-follows-whom* relationships in microblogging platforms like Twitter. Few algorithms have been explicitly designed to find communities in directed graphs; the reader may refer to the excellent survey by [15] to get a taxonomy of existing methods. Unfortunately, we still lack of an objective comparison of the existing methods and it is unclear under what experimental setting an algorithm performs better than others. The goal of this paper is to provide an experimental comparison of some algorithms to discover communities in directed graphs; algorithms under inquiry are evaluated across *accuracy* (i.e., their ability to truly find out communities) and *efficiency* (i.e., their ability of handling massive graphs).

1.2 Main Contributions, Results and Implications

Algorithms evaluated in this paper can be roughly split into three categories: a first category of approaches consists of algorithms like *Infomap* [16] and *Label Propagation* [17], [18] which are designed to manage directed graphs. A second category of approaches (*WalkTrap* [19]) ignores edge directionality in finding out communities. Finally, a third category of algorithms suggest to map the input directed graph G^d onto an undirected one G^u and to apply any off-the-shelf method to detect communities in G^u: an exemplary of this category is the *Eigenvector* algorithm [20].

We used a popular benchmark, known as the *Lancichinetti-Fortunato-Radicchi* model [21], to generate artificial graphs whose communities were known in advance. We created more than 900 graphs with different sizes and levels of connectivity and compared the ability of the algorithms above to truly guess communities. To measure algorithm's accuracy we used the *Normalized Mutual Information – NMI* score [22]: it varies from 0 to 1 and the higher the NMI, the better the algorithm to test is able to correctly find out communities.

We then concentrated on 5 real Web datasets covering a vast array of domains like trust relationships in product review Web sites, P2P microlending systems, a relatively large portion of the Web graph and e-commerce platforms.

We also considered two popular procedures (called hereafter *symmetrization procedures*) to transform a directed graph into an undirected one and we performed experiments on both artificial and real datasets to assess pros and cons of symmetrization procedures.

We compared the running times of algorithms under evaluation on both artificial and real dataset to determine if they were appropriate for large data analytic tasks.

The main outcomes of our analysis as well as their implications are as follows:

1) On artificial datasets, WalkTrap achieved the highest NMI but other algorithms scored an NMI comparable to that of WalkTrap, especially on those graphs clearly showcasing a community structure. As such, all the algorithms to compare succeeded in the objective of finding groups of densely connected vertices which were also well separated each other.
2) Edge directionality plays an important role in the process of discovering communities. Our experiments on both real and artificial graphs showed that symmetrization may deeply alter the structure of the input graph and, then, the accuracy of a community detection algorithm may suffer of modifications in the input graph.
3) LPA is easy to implement and it scales well on very large datasets. Therefore, it is an efficient means to detect communities for Web Analytic purposes. The Infomap algorithm provides the best trade-off between accuracy and computational costs. of community which incorporates both topological and non-topological features.

1.3 Relations with prior literature

Some papers provide an excellent overview of algorithms to find out communities in graphs: we refer the reader to [23] for a general introduction and [15] for an overview of community detection algorithms for directed graphs.

Our paper differs from [23] and [15] because we aim at empirically comparing the performance of some popular algorithms rather than providing a new classification scheme.

[24] carried out an empirical comparison of community detection algorithms for undirected graphs while [25] compared algorithms to discover *overlapping communities*, i.e., communities possibly sharing some vertices.

[26] provides a comparison of community detection algorithms in undirected graphs with the goal of quantifying any biases in the structures and sizes of communities discovered.

Quality of communities is mainly measured in terms of *conductance* [27]: the conductance of a set of vertices S in a graph $G = \langle V, E \rangle$ is the ratio of the number of edges connecting vertices from S with vertices in $V - S$ to the total number of edges in G.

The concept of conductance is hard to define in case of directed graphs and in our study we must resort to other methods to assess the quality of communities discovered by a particular algorithm. As a further difference, the approach of [26] focuses on social networks: it introduces the notion of *network community profile*, i.e. a diagram that quantifies how well a community detection algorithm performs when it attempts at finding the optimal community of size k for $k \in [1, \frac{|V|}{2}]$. An interesting result is that NCP plots show a *V-shape*, meaning that it decreases steadily up to a particular k and then it increases again.

The scope of this paper is slightly broader and we considered datasets covering a wider range of domains of interest to Web Data Analytics.
Finally, [21] considered also directed graphs but experiments were carried out only on artificial datasets while we deal with real graphs too.

1.4 Plan of the Paper

This paper is organized as follows: Section 2 outlines the benefits deriving from the usage of Big Data in community detection. Section 3 provides some background material on community detection in graphs and illustrates two algorithms, namely Label Propagation and Infomap which are able to manage directed graphs. Section 4 illustrates symmetrization procedures, i.e., procedures to map a directed graph onto an undirected one as well as two algorithms – WalkTrap and Eigenvector – which take a symmetrized version of the input graphs. Sections 5 and 6 illustrate experimental tests we carried out on artificial and real graphs, respectively. Section 7 is about the running times of the algorithms discussed on this paper on both artificial and real datasets. In Section 8 we explain how the assumptions underlying the algorithms under investigation are linked to the experimental results we observed and, then, the conclusions we have drawn may serve as a guidance to researchers and practitioners to selects the algorithm best fitting their needs. Finally, Section 9 discusses potential applications of community detection algorithms.

2 The benefits we get from Big Web Data to spot communities

The availability of large amounts of data extracted from multiple Web sites in conjunction with powerful data analytic tools allows us to get relevant answers to questions arising in a broad range of domains like Sociology, Business or Security. Typical questions are: *(i)* what are the most important customers of an e-commerce provider? What are the most important products a company delivers? *(ii)* In a Social Web platform, who are the people having the most social power to influence the perspectives of their peers? *(iii)* Can we detect frauds or a threat to national defence by analysing pattern of communications among people on the Web?

A natural way to answer questions *(i)-(iii)* requires to map data collected from a Web platform onto an undirected/directed graph and apply the graph analytic tool(s) best fitting our questions.

We showcase the potential of community detection algorithms in handling the identification of *strong and weak social ties* [28], [29], [30], [31].

Strong ties are relationships between trusted (or at least known) persons (e.g., close friendships) while weak ties refer to social relationships between people who rarely interact due, for instance, to linguistic or age reasons [28]. In human societies, weak ties play a role to transmit information and spread new ideas to wide segments of user population: as such, their identification is a fundamental research topic in disciplines like Social Sciences and Marketing.

Traditional methods to detect weak ties usually require to interview a (generally small) sample of subjects and ask people to rate the strength of their relationships [32]. Such a procedure was applied in case of Facebook [33] and some variables (like the number of shared friends) were considered as predictors of tie strength; least square regression was finally applied to calculate the tie strength for pairs of unknown users.

Approaches above rely on theoretical models which assume that the intensity of human relationships depends on some behavioural variables and psychological traits like the sociability of an individual or the tendency to create reciprocal relationships. Unfortunately, these model can be experimentally validated only on small samples and they lack of interpretability because they contain many – often related – variables.

We can use software programs to automatically track and collect data from multiple Web platforms and we are no longer restricted to manage small user samples. Moreover, tracking user activities over a long time frame enables to detect the events triggering the creation of new social relationships, their strengthening and their dissolving.

The availability of large, high-quality datasets opens the door to data-driven approaches to calculating tie strength rather than relying on theoretical models [30], [31], [34].

For instance, [34] showed that parameters like number of times two Facebook users jointly appear in a photo or the number of times they jointly commented a third-party post are reliable indicators of strong social ties. Analogously, [31] provided an approach to finding out weak ties in Facebook which first applies a community detection algorithm on the Facebook social graph and, then, classifies inter-cluster edges as weak ties. Such an approach ignores data describing user activities (which are not known due to privacy restrictions). Community detection algorithms have also been applied to detect weak and strong ties in Twitter [30].

3 Algorithms to find Community Detection in Directed Graphs

In this section we describe some algorithms which are designed to explicitly find communities in directed graphs. We first introduce some basic definitions about graphs and communities in graphs.

3.1 Basic Definitions

A *graph* G is a pair $G = \langle V, E \rangle$ where V is the set of vertices and E is the set of edges. We indicate as i an arbitrary vertex of G; we use also the symbol $e_{ij} \in E$ to denote the edge connecting vertices i and j.
We say that G is *undirected* if $e_{ij} \in E$ implies that $e_{ji} \in E$, *directed* otherwise.
The *adjacency matrix* $\mathbf{A}$ of G is an $|V| \times |V|$ matrix such that $\mathbf{A}_{ij} = 1$ if and only if there is an edge from i to j, 0 otherwise. The adjacency matrix of a graph is *symmetric* if and only if the graph G is undirected.
Given a vertex i the *indegree* k_i^+ and the *outdegree* k_i of i are defined as:

$$k_i^+ = \sum_{j \in V} \mathbf{A}_{ji} \qquad k_i^- = \sum_{j \in V} \mathbf{A}_{ij}$$

Many graphs describing biological or socio-technical systems divide into group of vertices called *communities* (or *clusters*) such that the number of edges connecting vertices within a community is much larger than the number of edges linking vertices in different communities [23]. Two communities are *overlapping* if they share at least one vertex [25], [35], [36]. In the following, we will consider only non-overlapping communities but we refer the interested reader to [25] for an excellent review on algorithms to find overlapping communities in graphs. We define the *community structure* of a graph as the collections of communities extracted from it.

3.2 What Algorithms to choose?

The task of finding communities in directed graphs received less consideration than in undirected ones and this mainly depends on the fact that the notion of community in directed graphs is harder to define than in undirected ones [37] (see Section 8 for more details).
Recently, [15] provided an in-depth comparative review of algorithms to find communities in directed graphs and, in the light of the work of [15], we classify existing algorithms in the following categories:

1) *Methods based on a null model*. In these methods, communities are interpreted as subgraphs whose connectivity deviates the most from the connectivity we would expect in a given random graph called *null model* [38], [39], [40]. A bright representative of this category is the Eigenvector algorithm [20]), which preliminarily transforms the input directed graph into an undirected one. Due to these reasons, we describe the Eigenvector algorithm in Section 4.2.
2) *Methods based on dynamic processes on graphs*. In these approaches [17], [18], [41], vertices are initially equipped with distinct labels encoding the communities they belong to: in this way, there are as many communities as vertices. A dynamic process is simulated: in it, labels propagate through the graph and each vertex adopts the label that the maximum number of its neighbours has (ties are randomly broken). As labels travel through the graph, densely connected group of vertices form a consensus on their labels. At the end of this process, vertices sharing the same label end up in the same community. We choose the Label Propagation Algorithm (LPA in short) [17] as an exemplary of this category of methods and describe it in Section 3.4.
3) *Methods based on a flow model*. The rationale underlying flow-based methods is that real-life systems consist of some main functional units which communicate and interact to pursue a pre-defined goal. Therefore, if we inject some amount of information in a real system, it will flow along the edges in the graph associated with the system itself; communities are identified as group of vertices among which information persistently circulates once entered in the group itself. Among methods based on flow we considered the Infomap algorithm [16], [42] (see Section 3.3).

3.3 Infomap

Infomap is based on the duality of compressing a dataset and the task of detecting significant patterns in the dataset. Such a duality is a widely explored topic in a field of Statistics known as Maximum Description Length (MDL) [43], [44].
To describe Infomap, let us consider a graph G and the process in which a sender makes use of G to send information (or other kind of goods) to a receiver. We wish to detect what substructures of G are significant to the flow of information/resources.
In few cases, trajectories followed by messages/goods going from the sender to the receiver are known and we directly work on them. More often, we are only aware about network structure along which information/resources can flow. The best we can do in this configuration is to approximate flow trajectories as *random walks* on G (eventually guided by edge directions and/or weights). This motivates a further question: *how can we efficiently record the sequence of vertices visited by a random walker on G?*
The simplest solution consists of assigning a Huffman codeword to each vertex [45]; codeword length is proportional to the average frequency a vertex would be visited by a random walker performing an infinite-length random walk on G. In this way, frequently visited vertices would be associated with short codewords and rarely visited vertices would be equipped with longer ones.
If we target at recording the list of locations the random walker visits at arbitrary and sufficiently distant time steps, then the Huffmann encoding is space-optimal. However, real graphs are structured in *regions* (that coincide with the communities we introduced before) and, once a walker enters in a region, it tends to stay there for a long time; in an equivalent fashion, movements across regions are rather unusual.
The regional structure of G is effective to design a more efficient strategy to trace the path followed by the random walker. In fact, we could think of a two-level encoding scheme in which the codeword defining the position of a walker consists of two components: the former specifies the region in which the walker is in, while the latter denotes the vertex (inside that region) it is currently occupying. Once a message enters in a region, the sender and the receiver must switch to the corresponding codebook; the walk is specified through a set of codebooks denoting the vertices visited by the walker. Region codebooks are stored in an *index codebook*; codes associated with vertices can be reused across different regions.
From the previous reasoning, for each division $\mathbb{M}$ of G in modules, we can calculate how space efficient the encoding associated with that division is and the best partitioning corresponds to the most efficient encoding scheme.
Finding an upper bound $L(\mathbb{M})$ on the efficiency of an encoding is, fortunately, a relatively trivial task. In fact, from the Shannon's source coding theorem [46], [47], we know that any discrete random variable X which can take n possible values with probabilities p_i ($i = 1, \ldots, n$), the average length of a codeword to describe X can not be less than the entropy $H(X) = -\sum_{i=1}^{n} p_i \log p_i$ of X.
The average length of the code describing a step performed

by the walker is encoded in the following equation (known as *map equation*):

$$L(\mathbb{M}) = q_{\curvearrowright} H(Q) + \sum_{i=1}^{m} p^i_{\circlearrowright} H(\mathcal{C}_i) \quad (1)$$

Here $q_{\curvearrowright}$ is the the probability that the random walker switches modules on any given step, $p^i_{\circlearrowright}$ is the fraction of time the random walk spends in module $\mathcal{C}_i$, $H(Q)$ is the average length of codewords in the index codebook (weighted by their frequency of usage) and $H(\mathcal{C}_i)$ is the average length of codewords in module codebook $\mathcal{C}_i$ (weighted by their rate of use).

To find the optimal partition of G, it is sufficient to calculate $L(\mathbb{M})$ for different divisions of G and pick the one that gives the shortest description length. In a nutshell, Infomap consists of the following steps (see Algorithm 1). First, each vertex is assigned to its own community and, in a random ordering, we pick a vertex i and calculate the largest decrease $\Delta L(i,j)^\star$ in L we would achieve if we would move i from its current community to the community $\mathcal{C}^\star_j$ of one of its neighbours. If no move results in a decrease of the map equation, then i stays in its original community. This procedure is repeated until no move generates a decrease of the map equation. Then the graph is rebuilt, with the community of the last level will form the vertices at the next level.

Algorithm 1 Infomap's algorithm

procedure INFOMAP
Input: G: a graph
Output: A set of communities
  for each vertex $i \in V$ **do**
    $\mathcal{C}_i \leftarrow \{i\}$
  end for
  $\overline{\mathcal{C}} \leftarrow \{\mathcal{C}_1, \ldots, \mathcal{C}_{|V|}\}$
  repeat
    Let X be a random ordering of vertices in V
    for each $i \in X$ **do**
      Let $N(i)$ be the neighbours of i
      $j^\star, \Delta L(i,j)^\star \leftarrow \arg\max_{j \in N(i)} \Delta L(i,j)$
      if $\Delta L(i,j)^\star < 0$ **then**
        $\mathcal{C}^\star_j \leftarrow \mathcal{C}^\star_j \cup \{i\}$
      end if
      Update $\overline{\mathcal{C}}$
    end for
  until No further decrease of L is achieved
return $\overline{\mathcal{C}}$
end procedure

3.4 Label Propagation Algorithm

The Label Propagation Algorithm (*LPA* in short) [17] is an iterative algorithm which initializes each vertex i with a unique label $C_i(0)$.

At every step, each vertex updates its label according to the labels of its neighbours. More specifically, let $i_1, i_2, \ldots, i_k$ be the neighbours of i and let $C_t(i_1), C_t(i_2), \ldots, C_t(i_k)$ the corresponding labels at the t-th iteration of the LPA algorithm.

The label $C_{t+1}(v)$ at the $t+1$-th iteration is determined as follows:

$$C_{t+1}(i) = \mathrm{f}\left(C_{t+1}(i_1), \ldots, C_{t+1}(i_k)\right) \quad (2)$$

Here f returns the label occurring at the largest frequency among the neighbours of v; ties are broken uniformly at random. The process expands outwards until every vertex in G has a label to which the largest number of its neighbours belong to. The pseudo-code describing LPA is reported in Algorithm 2. Each iteration of LPA costs $O(|E|)$;

Algorithm 2 The Label Propagation Algorithm (LPA)

procedure LPA
Input: G: a graph
Output: A mapping from vertices to communities
  for each vertex $v \in V$ **do**
    $C_0(x) \leftarrow x$
  end for
  $t \leftarrow 1$
  while True **do**
    Let X be a random ordering of vertices in V
    for each v in X **do**
      Calculate $C_t(v)$ according to Equation 2
    end for
    if no vertex changes its label **then**
      halt the algorithm
    else $t \leftarrow t + 1$
    end if
  end while
  Let $t^\star$ be the latest value of t
return $\mathrm{f} : \mathrm{f}(v) = C_{t^\star}(v)$
end procedure

however, experimental results indicate that few iterations are sufficient to create dense regions of vertices which form a consensus, i.e., share the same label. Consensus groups attempt at recruiting more vertices; if a consensus group touches the border of another consensus group, they start to compete for acquiring new members. A consensus group is able to counteract the pressure of another consensus group if its internal density (i.e., the fraction of edges joining vertices inside the same group) outweighs the external one (i.e., the fraction of incoming/outgoing edges).

The SLPA (Speaker-Listener Label Propagation Algorithm) extends LPA to manage overlapping communities [25]. In SLPA, each vertex can play the role of a listener or a speaker; it can retain as many labels it wants and these labels are stored in an ad-hoc memory [41].

Initially, the memory of each vertex is initialized with the vertex identifier. SPLA performs an iterative process: at every step, one vertex i_0 is selected as a listener. Each neighbour of i_0 sends out a single label which must respect some rules (e.g., it may randomly select a label from its memory with probability proportional to the number of times that label was recorded in its memory). Symmetrically, the listener accepts one label from the collection of labels received from neighbours following some rules (e.g., it may select the most popular label from what it observed in the current step). The loop stops after T iterations have been carried out, being T a pre-defined threshold.

Finally, SLPA calculates the label probability distribution. If the probability that a label occurs is less than a threshold $r \in [0, 1]$, then the label is deleted from the vertex's memory. After this task, connected vertices endowed with a particular label are grouped together to form a community. Observe that a vertex may contain multiple labels and, then it belongs to more than one community. This makes SLPA capable of detecting overlapping communities.

3.5 Other Approaches

In our experimental comparison we considered two further algorithms to detect communities in directed graphs, namely the *Edge Betweenness-EB* [48] algorithm and *Spinglass* [49]. The EB algorithm ranks graph edges on the basis of their *betweenness centrality*, defined as the ratio of the number of shortest paths in the graph crossing an edge to the overall number of shortest paths in the graph [50]. The rationale behind the EB algorithm is that if we would progressively remove edges in order of decreasing betweenness centrality, we would disconnect the graph G, thus spotting its communities. Because of graph topology may change after the deletion of one edge, we are forced to recalculate betweenness centralities at each iteration of the EB algorithm. As such, the overall worst complexity of the EB algorithm is in the order of $O(|V|^4)$ and, then, it is applicable only on graphs containing up to few hundreds of vertices.

[51] described an algorithm to speed up the computation of betweenness centrality up to $O(|V| \cdot |E|)$ which, in case of sparse graphs, amounts to $O(|V|^2)$

. Despite such an improvement, the EB algorithm still works well only on small graphs and, due to these reasons, we did not include it in our experimental comparison.

In Spinglass, each vertex i is viewed as a particle of a physical system and it is labelled with a spin variable σ_i (called *spin state*): vertices in the same community (i.e., particles in the same spin state) should be connected whereas vertices in different communities (i.e., particles in different spin states) should be disconnected.

The task of finding communities in a graph is therefore equivalent to spot the spin configuration yielding the lowest level of energy of a physical system. More formally, we define the *Hamiltonian* of G as follows:

$$\mathcal{H}(\overrightarrow{\sigma}) = -\sum_{i=1}^{n} \sum_{j=i+1}^{n} (\mathbf{A}_{ij} - \gamma \cdot p_{ij})\, \delta(\sigma_i, \sigma_j) \qquad (3)$$

Here p_{ij} is the probability of an edge between i and j and $\delta(\sigma_i, \sigma_j) = 1$ if and only if i and j share the same spin state (0 otherwise). The parameter γ is a parameter functional to explore the community structure of G at different granularity levels: it varies from 0 (in such a case we get a single community containing all vertices) to $+\infty$ (in this case we manage n communities, each one consisting of one vertex). Simulated annealing has been applied to minimizing Hamiltonian, thus making Spinglass a non-deterministic algorithm.

Unfortunately, the number of spin states crucially affects the running time and accuracy of Spinglass. For relatively large dataset, we acknowledged that the largest number of admissible spin states was 500, which is also the largest number of communities Spinglass can detect. In general we have no supportive arguments to decide how large the number of spin states should be; in addition, it is hard to find a right trade-off between the number of spin states and the running time of Spinglass. Therefore, we did not report the results of our tests about Spinglass.

4 Mapping Directed Graphs onto Undirected Ones

Many authors suggest to transform a directed graph $G^d = \langle V, E \rangle$ onto an undirected one – say G^u – and apply on G^u any of the available algorithms to find communities in undirected graphs. We call these procedures as *symmetrization procedures*.

The easiest symmetrization procedure consists of ignoring edge directionality. If we let $\mathbf{A}$ denote the adjacency matrix of G^d, then ignoring edge directions is equivalent to consider a new graph G^u whose adjacency matrix is $\mathbf{B} = \max\{\mathbf{A}, \mathbf{A}^T\}$ where $\mathbf{A}^T$ denotes the transpose of $\mathbf{A}^T$.

A further symmetrization procedure to consider is *cocitation*; in it, the adjacency matrix $\mathbf{B}$ associated with G^u is defined as $\mathbf{B} = \mathbf{A} + \mathbf{A}^T$.

Finally, in our study we will also consider the *bibliographic coupling* procedure, in which $\mathbf{B}$ is defined as $\mathbf{B} = \mathbf{A}\mathbf{A}^t$.

In both cocitation and bibliographic coupling procedures, G^u is a *weighted graph* which will exactly contain the same vertices of G^d. The weight w_{ij} of the edge e_{ij} in G^u has to be intended as the strength of similarity of vertices i and j.

In case of cocitation, if $e_{ij} \in E$ or $e_{ji} \in E$ (but not both), then $w_{ij} = 1$. If both $e_{ij} \in E$ and $e_{ji} \in E$, then $w_{ij} = 2$. In case of bibliographic coupling, observe that:

$$\mathbf{B}_{ij} = \left(\mathbf{A}\mathbf{A}^T\right)_{ij} = \sum_{k=1}^{|V|} \mathbf{A}_{ik}\mathbf{A}^T_{kj}$$

The term $\mathbf{A}_{ik}\mathbf{A}^T_{kj}$ differs from 0 if and only if there is an intermediate vertex k such that $e_{ik} \in E$ and $e_{kj} \in E$. Then, $\mathbf{B}_{ij}$ counts the number of vertices which are out-neighbours of i and in-neighbours of j.

After performing one of the mappings above, we can apply on G^u any of the community detection algorithms designed for undirected graphs.

In this way, vertices which will form a community $\mathcal{C}'_a$ in G^u will also form a community $\mathcal{C}_a$ in G^d; the whole procedure is equivalent to say that communities in G^d consists of groups of highly similar vertices, being vertex similarity computed by taking the topological structure of G^u into account.

In this paper we will combine the Infomap and LPA algorithms described in Sections 3.3 and 3.4 with the cocitation and bibliographic coupling procedures. In this way we get 4 new algorithms that we called LPA-C (LPA plus cocitation), LPA-Bib (LPA plus bibliographic coupling), Infomap-C (LPA plus cocitation) and Infomap-Bib (Infomap plus bibliographic coupling).

We will also consider the WalkTrap algorithm (Section 4.1) which ignores edge directionality and the Eigenvector algorithm (see Section 4.2) which performs a symmetrization procedure which closely mimics the cocitation procedure.

4.1 WalkTrap

To study the effects associated with ignoring edge directionality we focused on *WalkTrap* [19], a popular algorithm to detect communities in undirected graphs. The rationale behind WalkTrap is that a random walker starting from an arbitrary vertex i will get trapped in a dense region of G which corresponds to the community of i.
More formally, let $t > 0$ be a positive and fixed integer and let p_{ij}^t be the probability that a random walker moves from the vertex i to the vertex j in t steps. We define the distance from i to j in t steps as[1]:

$$r_{ij}^t = \sqrt{\sum_{l \in V} \frac{(p_{il}^t - p_{il}^t)^2}{k_l}} \tag{4}$$

Given a community $\mathcal{C}$, the probability of moving from $\mathcal{C}$ to i in t steps is defined as follows:

$$p_{\mathcal{C}i}^t = \frac{1}{|\mathcal{C}|} \sum_{j \in \mathcal{C}} p_{ji}^t \tag{5}$$

Finally, the distance of two communities $\mathcal{C}_1$ and $\mathcal{C}_2$ is defined as:

$$r_{\mathcal{C}_1\mathcal{C}_2}^t = \sqrt{\sum_{l \in V} \frac{(p_{\mathcal{C}_1 l}^t - p_{\mathcal{C}_2 l}^t)^2}{k_l}} \tag{6}$$

Bearing these definitions in mind, we are now able to sketch the WalkTrap algorithm (see Algorithm 3).

Algorithm 3 The WalkTrap Algorithm

```
procedure WALKTRAP
Input: G: a graph, t: an integer
Output: A set of communities
    for each vertex i ∈ V do
        C_i ← {i}
    end for
    P_0 ← {C_1, ..., C_|V|}
    for C_a, C_b ∈ P_0 do
        if C_a and C_b are adjacent then
            Calculate r^t_{C_a C_b} (see Equation 6)
        end if
    end for
    for k = 1 ... |V| − 1 do
        Select C_a and C_b according to the Ward's method
        Merge C_a and C_b in C_ab
        P_k ← (P_{k−1} − {C_a, C_b}) ∪ C_ab
        Update distances among adjacent communities
    end for
return P_{|V|−1}
end procedure
```

WalkTrap initially builds $|V|$ communities, each consisting of one vertex. Let $\mathcal{P}_0$ be the partitioning of G into communities; WalkTrap calculates the distance of each pair of *adjacent* communities in $\mathcal{P}_0$ (here, two communities are adjacent if they share at least one edge). The partitioning is refined by performing a loop which consists of $|V| - 1$ iterations. At each iteration, WalkTrap selects two adjacent partitions, say $\mathcal{C}_a$ and $\mathcal{C}_b$, to merge. Communities to merge are selected according to the *Ward's method*, i.e., selected communities are those ones that minimize the mean of the squared distances between each vertex and its community. WalkTrap merges $\mathcal{C}_a$ and $\mathcal{C}_b$ into $\mathcal{C}_{ab}$ and it updates $\mathcal{P}$ by deleting $\mathcal{C}_a$ and $\mathcal{C}_b$ and inserting $\mathcal{C}_{ab}$ in; because of communities have changed, WalkTrap recalculate distances among adjacent communities. After $|V| - 1$ steps, the algorithm halts. Each instance $\mathcal{P}_k$ identifies a partitioning of the graph into communities, which, at the last iteration, collapses onto a single community containing all vertices. In this way we manage a hierarchical organization of communities *dendrogram*: it can be graphically represented as a tree in which the leaves correspond to the vertices and each internal node is associated to a merging of communities in the algorithm: it corresponds to a community composed of the union of the communities corresponding to its children. The parameter t crucially influences the performance of WalkTrap: low values of t allow WalkTrap to explore only a small portion of G; in contrast, large values of t imply that the probability of visiting a vertex depends only on its degree, and, then, high degree vertices may turn out to be "close" to all other vertices. Experimentally, we found that the best compromise occurs at $t = 5$. For a fixed value of t, the worst-case time complexity of WalkTrap is $O(|V|\,|E|\,H)$, where H is the height of the dendrogram. Because of most real graphs are sparse and H is generally small, we have that the time complexity of WalkTrap is reasonably around $O(|V|^2 \log |V|)$.

1. Here k_l is the degree of vertex l, which, in case of undirected graphs, coincides with the in-degree and the out-degree of l.

4.2 The Eigenvector Algorithm

The Eigenvector Algorithm attempts at optimizing a suitable function called *modularity* [23], [38], [48]. Modularity was first defined in case of undirected graphs as follows:

$$Q^u = \frac{1}{2|E|} \sum_{i,j} \left(\mathbf{A}_{ij} - \frac{k_i \cdot k_j}{2|E|} \right) \delta(i,j) \tag{7}$$

Here, k_i and k_j are the degrees of i and j and, finally, $\delta(i,j) = 1$ if and only if i and j are assigned to the same community (0 otherwise). The function Q^u calculates the difference between the number of edges between communities in G and the expected number of edges between these same communities in an equivalent random graph $G^{\texttt{null}}$ – called *null model* – displaying the same edge density of G and in which the probability of having an edge between vertices i and j is $\frac{k_i \cdot k_j}{2|E|}$.
We expect that the modularity associated with any possible partitioning of $G^{\texttt{null}}$ is 0 because the probability of an edge is proportional to the degrees of its endpoints; therefore, the larger Q^u the better the community structure associated with G. The optimization of Q^u is a widely accepted criterion to find communities in graphs. Unfortunately, two major drawbacks makes the optimization of Q^u hard.
Firstly, Q^u is defined for undirected graphs and it should be properly extended to manage directed graphs; secondly, in case of large graphs, exploring all possible divisions to find out the one achieving the largest modularity is computationally infeasible and, consequently, we must resort to proper approximate methods.

To address the first drawback, [20], [52], [53] provided a new definition of modularity Q^d for directed graphs:

$$Q^d = \frac{1}{m}\sum_{i,j}\left(A_{ij} - \frac{k_i^+ \cdot k_j^-}{m}\right)\delta(i,j) \tag{8}$$

Reference [20] provides an efficient procedure to optimize Q^d: let us define $\mathbf{B}_{ij} = \mathbf{A}_{ij} - \frac{k_i^+ \cdot k_j^-}{m}$ and $\mathbf{R} = \mathbf{B} + \mathbf{B}^T$. Let us consider a simplified version of the community detection problem, known as *bisection*. In bisection, we target at dividing the input graph G into only two communities $\mathcal{C}_a$ and $\mathcal{C}_b$. Let us consider a vector $\mathbf{s}$ such that $\mathbf{s}_i = +1$ if the vertex i is assigned to $\mathcal{C}_a$ and $\mathbf{s}_i = -1$ if i is assigned to $\mathcal{C}_b$. In view of such a definition, it is possible to show [20], [53], [54] that Q^d can be written as follow

$$Q^d = \frac{1}{m}\mathbf{s}^T\mathbf{R}\mathbf{s} \tag{9}$$

the entries of the vector $\mathbf{s}$ must satisfy the following constraint: $\sum_{i=1}^{|V|}\mathbf{s}_i^2 = |V|$. To optimize Q^d we must choose $\mathbf{s}$ parallel to the leading eigenvector $\mathbf{e}^1$ of $\mathbf{R}$ [54]; such a configuration is however forbidden because the entries of $\mathbf{s}$ are constrained to be equal to $+1$ or -1. We therefore require that $\mathbf{s}$ is as close as possible to $\mathbf{e}^1$ and this leads to find $\mathbf{s}$ in such a way as to the inner product $\mathbf{e}^1 \cdot \mathbf{s}$ is as large as possible. Such a problem, fortunately, admits a straightforward solution: in fact, we must choose $\mathbf{s}_i = +1$ if $\mathbf{e}_i^1 > 0$ and $\mathbf{s}_i = -1$ if $\mathbf{e}_i^1 \leq 0$.
Observe that if the largest eigenvalue of $\mathbf{R}$ is 0, then the leading eigenvector $\mathbf{e}$ is proportional to $\mathbf{1}_{|V|}$, i.e., a $|V|$-dimensional array whose entries are all equal to 1. In this case, the bisection task is trivial because all vertices are arranged in a group. Algorithm 4 details all steps required to bisect G.
Multiple strategies can be adopted to generalize the bisection approach to more than two communities. The simplest one consists of repeatedly applying bisection: we first divide the network into two groups by means of Algorithm 4 and, then, we divide those groups. At any intermediate stage we get a collection of communities and the process stops when we reach a point at which further division does not increase the total modularity of our input graph.

Algorithm 4 The Bisect Routine

```
1: procedure BISECT
2: Input: G: a graph
3: Output: A bisection of G
4:     R ← B + B^T.
5:     C_a ← ∅
6:     C_b ← ∅
7:     Let e_M be the leading eigenvector associated with R
8:     for each vertex i ∈ V do
9:         if e_i = −1 then
10:            C_a ← C_a ∪ {i}
11:        else C_b ← C_b ∪ {i}
12:        end if
13:    end for
14: return {C_a, C_b}
15: end procedure
```

Symbol	Description	Value
$\|V\|$	Number of Vertices	$100 - 10\,000$
$\bar{k}$	Average Degree	$2\% - 5\%$ of $\|V\|$
$k_{\max}$	Maximum Degree	$\frac{\|V\|}{20}$
μ	Topological Mixing	$0.1 - 0.9$
τ_1	Vertex Degree Distribution	-2
τ_2	Community Size Distribution	-1
m_C	Community Size (min)	2
M_C	Community Size (max)	$\frac{\|V\|}{10}$

TABLE 1: Parameters adopted in the LFR benchmark and their reference values.

The worst-case time complexity of the Eigenvector algorithm depends on: *(a)* Calculation of the largest eigenvalue/eigenvector of $\mathbf{R}$ and *(b)* the number d of divisions to produce. If we assume that $\mathbf{R}$ is sparse, we could apply Lanczos algorithm to solve Stage *(a)* in $O(|V|^2)$ steps [54]; as for Stage *(b)*, observe that we perform at most $O(\log|V|)$ divisions of the input graph. By combining these two facts we get that overall time complexity of the Eigenvector algorithm is $O(|V|^2 \log|V|)$.

5 Experiments on Artificial Datasets

We start our experimental analysis by comparing the accuracy of the algorithms described in Sections 3 and 4 on artificially generated graphs with a known community structure.

5.1 Benchmarks

To study the effectiveness of community detection algorithms, it is fundamental to have graphs in which the true community structure (*ground truth*) is known. To this end, many procedures to generate synthetic graphs with known community structure have been developed [21], [48].

A realistic framework to generate synthetic networks with known community structure is due to Lancichinetti, Fortunato and Radicchi and it is known as the *LFR benchmark* [21]. Graphs generated with the LFR benchmark display heterogeneities in degree and community size distribution and, therefore, they mimic well real graphs. To apply LFR benchmark we need to specify 7 parameters, which are listed in Table 1 along with their reference values.

We generated graphs with a number $|V|$ of vertices from 100 to 100 000. The average vertex degree varied from the 2% to the 5% of $|V|$, with the maximum degree equal to $\frac{|V|}{20}$. A relevant parameter to specify is the *topological mixing* μ, defined as the ratio of the number of neighbours of each vertex i which do not belong to its community to the total degree of i. The topological mixing μ ranges in $[0,1]$ and indicates how well separated the communities are: values of μ near 0 are associated with graphs in which communities are clearly separated, while values of μ close to 1 define a fuzzier, hard to detect, community structure.
Vertex degree distribution and community size distribution are regulated by power laws with exponents τ_1 and τ_2, respectively. We adopted default values for both τ_1 and τ_2. Community sizes varied from 2 to $\frac{|V|}{10}$. The combination of the parameters above produce more than 900 graphs.

5.2 Quality Metrics for artificial graphs

To measure the agreement between the community structure detected by an algorithm and the ground truth given by LFR, we rely on the *Normalized Mutual Information* (NMI) [22].

To introduce NMI, let us consider a directed graph G created by means of LFR and let A be its true community structure, consisting of c_A communities. Let us apply a community detection algorithm $\mathcal{B}$ on G and let B the set of c_B communities it returns.

We define the NMI as follows:

$$NMI(A,B) = \frac{-2\sum_{i=1}^{c_A}\sum_{j=1}^{c_B} N_{ij}\log\left(\frac{N_{ij}N}{N_{i\cdot}N_{\cdot j}}\right)}{\sum_{i=1}^{c_A} N_{i\cdot}\log\left(\frac{N_{i\cdot}}{N}\right)+\sum_{j=1}^{c_B} N_{\cdot j}\log\left(\frac{N_{\cdot j}}{N}\right)} \quad (10)$$

where N_{ij} is the number of vertices shared by the i-th community of A and the j-th community of B and $N_{i\cdot}$ (resp., $N_{\cdot j}$) is the number of vertices in the i-th community of A (resp., j-th community of B).

The NMI ranges from 0 to 1 and the larger NMI, the better the community detection algorithm $\mathcal{B}$ works: if $\mathcal{B}$ would work perfectly, then $NMI(A,B)$ would be 1 [22]. In contrast, NMI equals 0 if the communities detected by $\mathcal{B}$ are totally independent of the true communities.

5.3 Discussion

Due to space limitations, we report the results of our experiments on the largest artificial graphs we generated. In Figure 1 we plot the NMI as function of μ for datasets consisting of $5\,000$ and $10\,000$ vertices.

We start discussing the behaviour of LPA, WalkTrap and Infomap: Figures 1a-1b suggest to define a threshold value $\mu^\star$ such that if $\mu \leq \mu^\star$ the NMI is equal to 1 and it starts decreasing if $\mu > \mu^\star$.

The threshold $\mu^\star$ depends on the algorithm we used as well as on the graph size $|V|$: for instance, if $|V| = 5\,000$, the value of $\mu^\star$ for WalkTrap occurs at 0.8 whereas it occurs at 0.6 for Infomap (see Figure 1a).

In general, as $|V|$ gets larger and larger, $\mu^\star$ increases too: for instance, in case of Infomap, $\mu^\star$ occurs at 0.6 for graphs with $5\,000$ vertices and at 0.7 for graphs with $10\,000$ vertices. This is a positive effect because $\mu^\star$ has to be considered as an upper limit beyond which an algorithm starts losing its ability of correctly guessing true communities. The higher $\mu^\star$, the better the algorithm works if communities are hard to detect. However, it is worth observing that the higher $\mu^\star$, the faster the decay of NMI. So, for instance, if $|V| = 10\,000$, $\mu^\star$ occurs at 0.7 in case of Infomap but it suffices to set $\mu = 0.9$ to almost nullify NMI (see Figure 1b).

From Figures 1a-1b we observe that WalkTrap performs the best because it achieves the largest values of $\mu^\star$

Our results confirm and extend the findings of [21], who studied the NMI of Infomap and a community detection method based on simulated annealing on LFR graphs of $1\,000$ and $5\,000$ vertices. [21] reports that the NMI of Infomap was equal to 1 for graphs with $1\,000$ vertices with $\mu \leq 0.7$ and for graphs with $5\,000$ vertices with $\mu \leq 0.8$. In addition to [21], our study highlights the good performance of WalkTrap and points out that $|V|$ positively affects on the NMI.

Algorithms which "symmetrize" the input graph generally display a lower NMI: in fact, the NMI of LPA-C and LPA-Bib are in line with (or slightly better than) that of LPA on graphs with $5\,000$ vertices but they achieve a lower NMI than that of LPA in case of graphs with $10\,000$ vertices. As for Eigenvector, the NMI steadily decreases from 0.91 – if $|V| = 5\,000$ – to 0.86 – if $|V| = 10\,000$. In contrast, the NMI of Infomap-C and Infomap-Bib is relatively stable and independent of μ. Such a behaviour is due to the symmetrization process: in fact, symmetrization procedures may introduce artificial edges between pair of vertices which were disconnected in the input graph. After symmetrization, some dense substructures emerge and they are wrongly classified as communities thus explaining the decrease of NMI.

Dataset	# Vertices	# Edges
`Epinions`	131 828	841 372
`Prosper I`	89 270	1 839 576
`Prosper II`	72 335	1 492 483
`Amazon`	403 394	3 387 388
`Web Graph`	875 713	5 105 039

TABLE 2: Number of vertices and number of edges of real-life graphs employed in our experimental comparison.

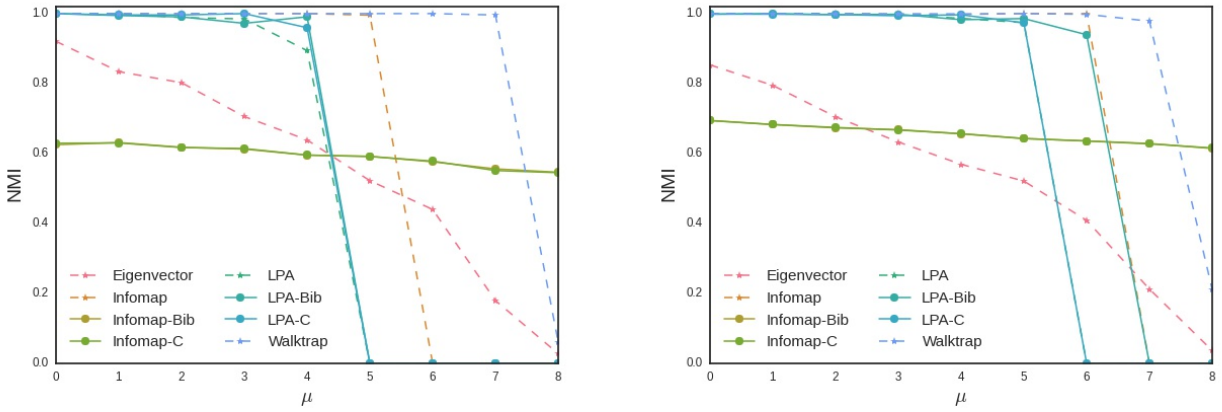
6 Experiments on Real Datasets

We take a further step in our study by analysing the performance of the algorithms described in Section 3 and Section 4 on real graphs.

6.1 Datasets

Datasets adopted in our analysis are reported in Table 2 and we briefly overview them.

(i) `Epinions` [55]. Epinions was a consumer review website launched in 1999 and subsequently bought by eBay. It had a large catalogue of goods/services that consumers could review. Users were allowed to create trust relationships in favour of other users. In our analysis we considered a *who-trust-whom* graph whose vertices are Epinions users and edges capture trust relationships. *(ii)-(iii)* `Prosper I and II` [56], [57]. Prosper is an online peer-to-peer lending platform and it is based on a bidding mechanism in which borrowers (resp., lenders) set the amount of money they want to borrow (resp., lend) as well as the maximum (resp., minimum) interest rate they are willing to pay (resp., offer). The bidding mechanism allows innovative lending practices such as splitting a large loan among several lenders. Prosper was offline from September 2008 to July 2009, for regulatory reasons. In our experiments we were able to manage two instances of Prosper: the former (called `Prosper I`) captures money flowing before August 31st, 2008 while the latter (`Prosper II`) spans a time period from August 2009 to July 2011. `Prosper I` and `II` represent the entire Prosper platform, not just a sample. *(iv)* `Amazon` [58]. This is the graph of frequently co-purchased items in Amazon. Each item is a vertex and a directed edge from i to j specifies that people who bought the item i generally bought also j. *(v)*



(a) LFR benchmark (with $|V| = 5\,000$)

(b) LFR benchmark (with $|V| = 10\,000$)

Fig. 1: Normalized Mutual Information on artificial datasets as function of the topological mixing parameter μ and for $|V| \in \{500, 1\,000\}$.

`Web Graph` [59]. This dataset contains a collection of Web pages and their hyperlinks.

6.2 Quality Metrics for real graphs

The lack of a ground truth makes hard to evaluate the quality of a partitioning discovered by a community detection algorithm [15] and, then, we must resort to alternative metrics to assess the quality of a partitioning [23], [26], [52]. We observe that the size of the communities discovered by a target algorithm plays a relevant role to decide on the appropriateness of that algorithm: in fact, a statistical study on social networks presented in [60] proved that real-world communities with high quality are quite small and, in general, they consist of no more than 100 vertices. Due to these reasons, the first aspect we investigated was the community-size distribution associated with algorithms under inquiry.

Because of we applied a range of community detection algorithms, a second question comes: what are the commonalities and differences between the communities discovered by different algorithms? In agreement with [15], we think that understanding differences in communities returned by different algorithms is of great interest for analytical purposes: in fact, if we would know that two community detection algorithms $\mathcal{A}_i$ and $\mathcal{A}_j$ agree on putting the same vertices on the same communities, we would conclude that $\mathcal{A}_i$ and $\mathcal{A}_j$ would provide a similar perspective on the data to analyse and, therefore, it may be necessary to consult a third algorithm $\mathcal{A}_k$ to get a more comprehensive view of data to explore. To do such an analysis, for each input graph G and for each pair of algorithms $\mathcal{A}_i$ and $\mathcal{A}_j$, we applied $\mathcal{A}_i$ and $\mathcal{A}_j$ on G and we calculated the NMI associated with the community structures discovered by $\mathcal{A}_i$ and $\mathcal{A}_j$.

In the following we comment on the results of our analysis.

6.3 Discussion

We start discussing the results achieved by LPA, Infomap, WalkTrap and Eigenvector on real graphs introduced before. We found that all these algorithms tend to identify many small communities and few large ones.

Due to space limitations, we report in Figure **??** the community-size distribution observed in `Prosper I`; for instance, we observe that the size of the largest community detected by WalkTrap is about 7 times bigger than the size of the second largest one.

The result of our experiments are in line with theoretical findings of [60] and underline the ability of algorithms under comparison to truly portray the structure of large-scale real graphs.

As proved in [60], real-life graphs consists of: *(i)* a large number of relatively small and well-connected *whisker-like* communities (a whisker is a component connected to the rest of the graph via a single edge) and *(ii)* a large core which is formed by multiple intermingled communities, whose boundaries get less and less well-defined as the community sizes increase and as they gradually blend in with rest of the network.

Differences in the shape of community-size distributions are directly tied to the theoretical features of algorithms under inquiry: for instance, we detected important differences in the size of communities discovered by Infomap and WalkTrap. Infomap, in fact (see Section 3.3), attempts at quantifying how information flows in a graph and it classifies a group of vertices as a community if such a group ensures a quick flow of information among its vertices. WalkTrap is blind to the notion of flow but it seeks at optimizing the modularity. From the theoretical features of the algorithms under inquiry, we have a propensity to believe that Infomap should work well on graphs where edges denote pattern of movements between vertices (think of `Prosper I` and `Prosper II` in which edges specify money flow between pairs of users).

Algorithms based on the optimization of modularity should yield accurate results on graphs in which the notion of information flow is missing, e.g., on graphs in which edges model pairwise relationships between vertices (like the Web graph).

To make our analysis more precise, we used the methodology described in [61] and the toolbox illustrated in [62] to

check whether the community size distribution was shaped as a power law. We found that a power law emerged for LPA ($\alpha = 2.6$, p-value less 0.01) and for Infomap ($\alpha = 2.65$, p-value less than 0.01).

The next step of our analysis focused on the effects of symmetrization procedures on community size distribution. We observed that community size distribution after in case of symmetrized graphs decayed slower than in no-symmetrized ones.

Such a behaviour is due to the transformation the input graph undergoes: in fact, operations like cocitation and bibliographic coupling applied on a directed graph G^d and have the effect of creating edges between pair of vertices in G^d that were not connected. As an extreme case, think of two groups of disjoint vertices $\mathcal{S}_a$ and $\mathcal{S}_b$. If we would transform G^d into an undirected graph by applying cocitation/bibliographic coupling, we would create artificial links between $\mathcal{S}_a$ and $\mathcal{S}_b$; due to these artificial links, a community detection algorithm may wrongly decide to merge $\mathcal{S}_a$ and $\mathcal{S}_b$ into a unique community. The final result is that community size distribution is less sharply sloped than in the case in which no symmetrization procedure was applied.

The second stage of our analysis is concerned with the comparison of community structures produced by algorithms under inquiry. In Figures 2 - 6 we used NMI grids to display pairwise NMI scores on each input dataset. An NMI grid is an all-to-all matrix where each entry (i, j) holds the NMI score calculated on the communities produced by the algorithms $\mathcal{A}_i$ and $\mathcal{A}_j$; here $\mathcal{A}_i$ (resp., $\mathcal{A}_j$) is one of LPA, WalkTrap, Infomap and Eigenvector. We used colors to encode the NMI values: the darker a cell, the higher the NMI score it stores; diagonal elements are, of course, equal to 1.

If no symmetrization procedure is applied, we observed that NMI scores varied from 0.03 to 0.84; this suggests that graph topology has a major impact on the output of each algorithm. In some cases (think of `Web Graph` – Figure 6, left panel), two algorithms may find pairs of highly similar communities despite they adopt deeply different strategies. Our analysis suggests to run multiple community detection algorithms on the same graph and to assess the amount of overlap between the community structures identified by each algorithm. Such a procedure helps us to more effectively understand the structure of complex networked-datasets available on the Web.

We then compared the community structures identified by LPA, LPA-C, LPA-Bib, Infomap, Infomap-C and Infomap-Bib algorithms (see Figures 2 - 6, right panel). Once again, symmetrization procedure brings deep repercussions on community structures: for instance, think of the `Prosper II` dataset and the community structures found by LPA and LPA-Bib. We record an NMI equal to 0.064, which means that a large percentage of vertices which are grouped together by LPA to form a community will spread across multiple (and diverse) communities detected by LPA-Bib. This means that the introduction of artificial edges due to cocitation and bibliographic coupling deeply alter graph topology and modifies the communities an algorithm discovers.

Dataset/Algorithm	Eigenvector	LPA	Infomap	WalkTrap
`Epinions`	1754.342	0.992	312.024	312.494
`Prosper I`	2339.86	0.882	4294.376	1437.989
`Prosper II`	1002.324	0.478	185.856	491.452
`Amazon`	5266.21	64.556	75	196.223
`Web Graph`	1999.39	14.782	45	196.223

TABLE 3: We report the running times (in seconds) of Eigenvector, LPA, Infomap and WalkTrap algorithms on the 5 datasets adopted in our tests.

Dataset/Procedure	Cocitation	Bibliographic Coupling
`Epinions`	0.031	1.985
`Prosper I`	0.162	12.502
`Prosper II`	0.162	10.664
`Amazon`	0.349	3.526
`Web Graph`	0.448	127.189

TABLE 4: We report the running times (in seconds) of cocitation and bibliographic coupling procedures on the 5 real datasets adopted in our tests

7 Scalability Analysis

Scalability analysis was performed on both real and artificial graphs. We first calculated the running time of LPA, Infomap, WalkTrap and Eigenvector on real graphs introduced in Section 6.1 and the results we came up are reported in Table 3.

From Table 3 we observe that LPA achieves the best computational performance and, then, it should be taken into consideration when processing large Web graphs. Such a result is not surprising and it agrees fairly well with theoretical results that indicate that the computational complexity of LPA is $O(|E|)$.

On the basis of the results we observed in artificial datasets (see Section 5), we conclude that Infomap is able to combine precision with computational efficiency: for instance, it takes only 45 seconds to manage large Web graphs.

Finally, Table 3 indicates that the graph size is not crucial in determining the computational performance of a community detection algorithm. For instance, `Prosper I` and `Prosper II` datasets are smaller than `Amazon` or `Web Graph` datasets but some algorithms require more time to process `Prosper I`/`Prosper II` than `Amazon`/`Web Graph`. This depends on the internal structure of input graph: if input graph does not display a clear community structure, the time an algorithm needs to halt is bigger than if the graph would clearly fragmented into communities and this result holds independently of graph size.

We then calculated the computational costs associated with the execution of cocitation and bibliographic coupling procedures over real graphs (see Table 4).

Even on large datasets, both cocitation and bibliographic coupling procedures are quite fast and this mostly depend on the fact that all the input graphs are *sparse*; because of efficient data structures are now available to represent and manipulate sparse matrices, we have that cocitation and bibliographic coupling procedures are fast also on relatively large graphs (and with modest hardware architectures). As expected, from Table 4, we observe that the bibliographic coupling procedure is much slower than the cocitation one (e.g., in case of `Web Graph` the ratio between the running time of bibliographic coupling and cocitation is

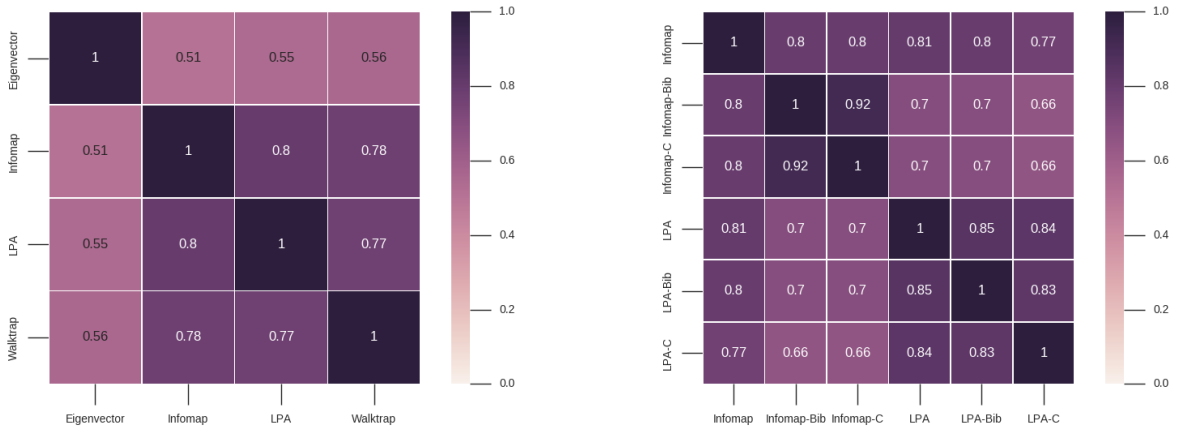


Fig. 2: **Left**: The NMI grid associated with WalkTrap, LPA, Infomap and Eigenvector on `Epinions` datasets. **Right**: The NMI grid associated with LPA, Infomap,LPA-C, LPA-Bib, Infomap-C and Infomap-Bib on `Epinions`.

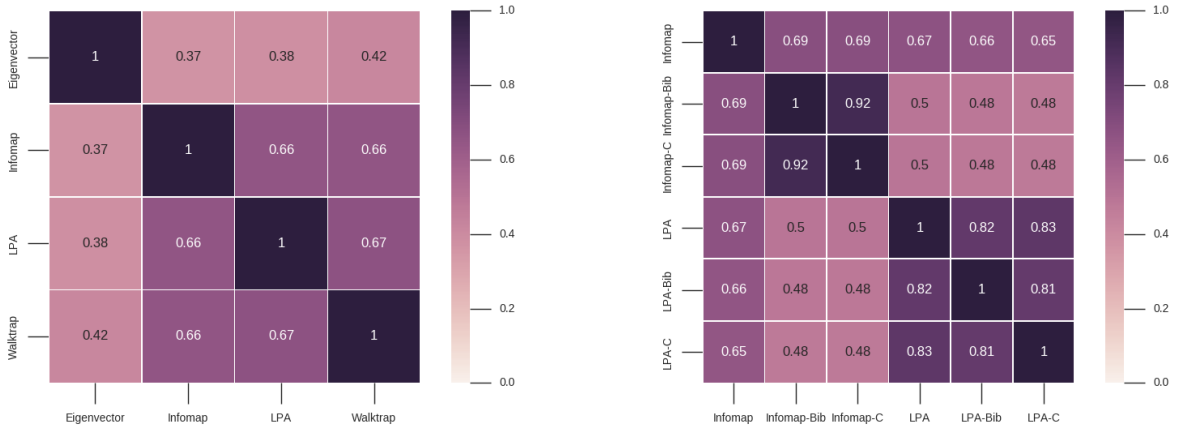


Fig. 3: **Left**: The NMI grid associated with WalkTrap, LPA, Infomap and Eigenvector on `Prosper I`. **Right**: The NMI grid associated with LPA, Infomap,LPA-C, LPA-Bib, Infomap-C and Infomap-Bib on `Prosper I`.

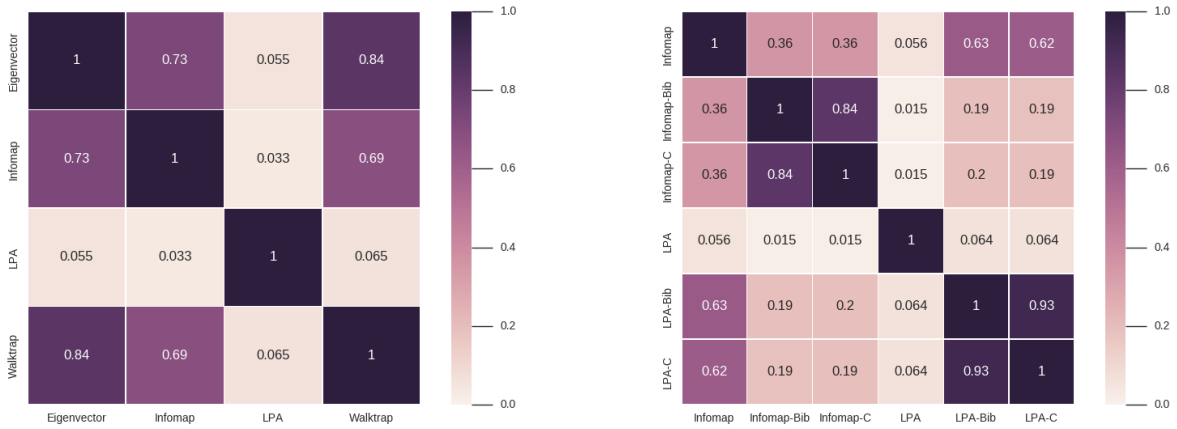


Fig. 4: **Left**: The NMI grid associated with WalkTrap, LPA, Infomap and Eigenvector on `Prosper II`. **Right**: The NMI grid associated with LPA, Infomap,LPA-C, LPA-Bib, Infomap-C and Infomap-Bib on `Prosper II`.

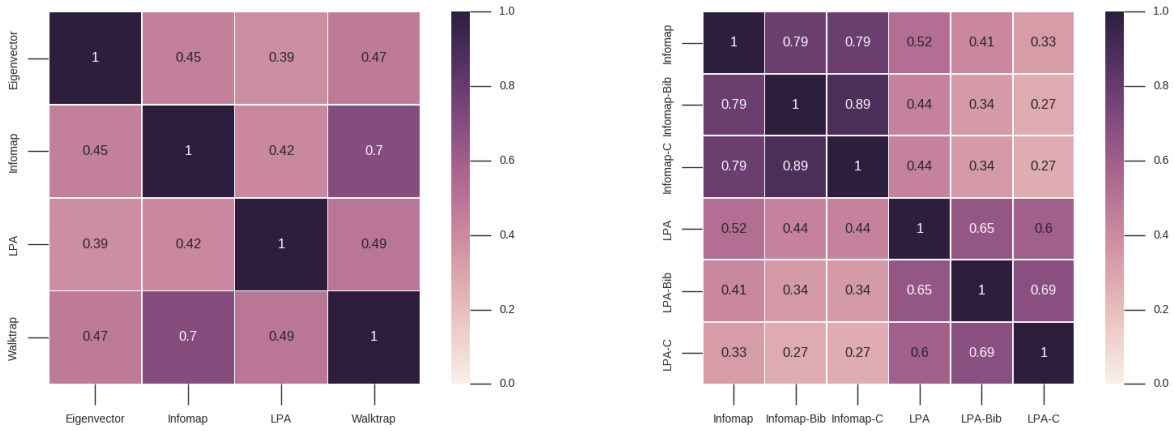


Fig. 5: **Left**: The NMI grid associated with WalkTrap, LPA, Infomap and Eigenvector on `Amazon` datasets. **Right**: The NMI grid associated with LPA, Infomap,LPA-C, LPA-Bib, Infomap-C and Infomap-Bib on `Amazon`.

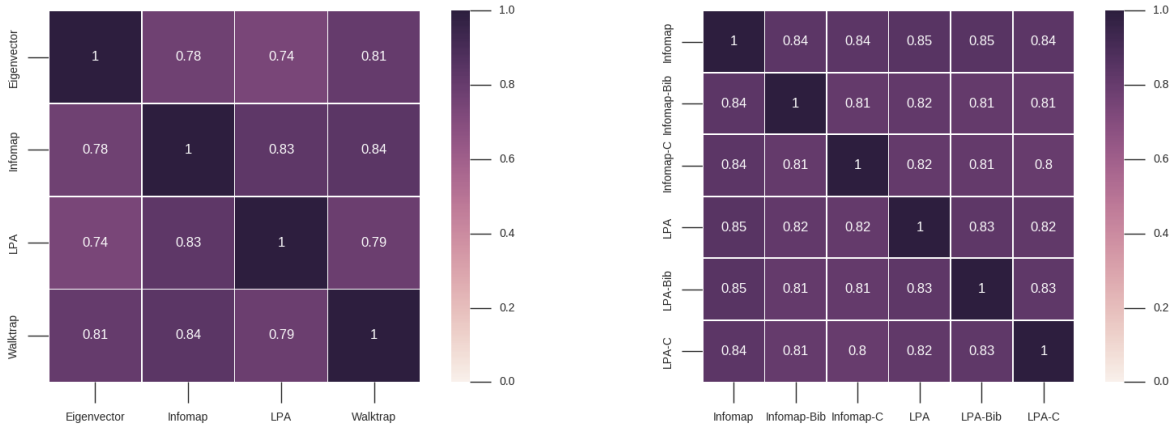


Fig. 6: **Left**: The NMI grid associated with WalkTrap, LPA, Infomap and Eigenvector on `Web Graph` datasets. **Right**: The NMI grid associated with LPA, Infomap,LPA-C, LPA-Bib, Infomap-C and Infomap-Bib on `Web Graph`.

about 277.78); this clearly depends on the fact that the time required to calculate the product of two matrices is much bigger than the time required for summing them.

We ran a further experiment to check if the edge density somewhat influences the execution time. To do so, we artificially generated some Barabasi-Albert graphs [40] with a number of vertices $|V|$ ranging from 20 000 to 300 000. Barabasi-Albert graphs appear to have power-law degree distribution as well as small-world property and, therefore, they are an appropriate choice to model many socio-technical systems of interest in the context of Web Data Analitycs. A Barabasi-Albert graph with $|V|$ vertices is grown by progressively adding new vertices; each new vertex v_i will display degree m and it will decide to connect to a vertex v_j with probability proportional to the degree of v_j.

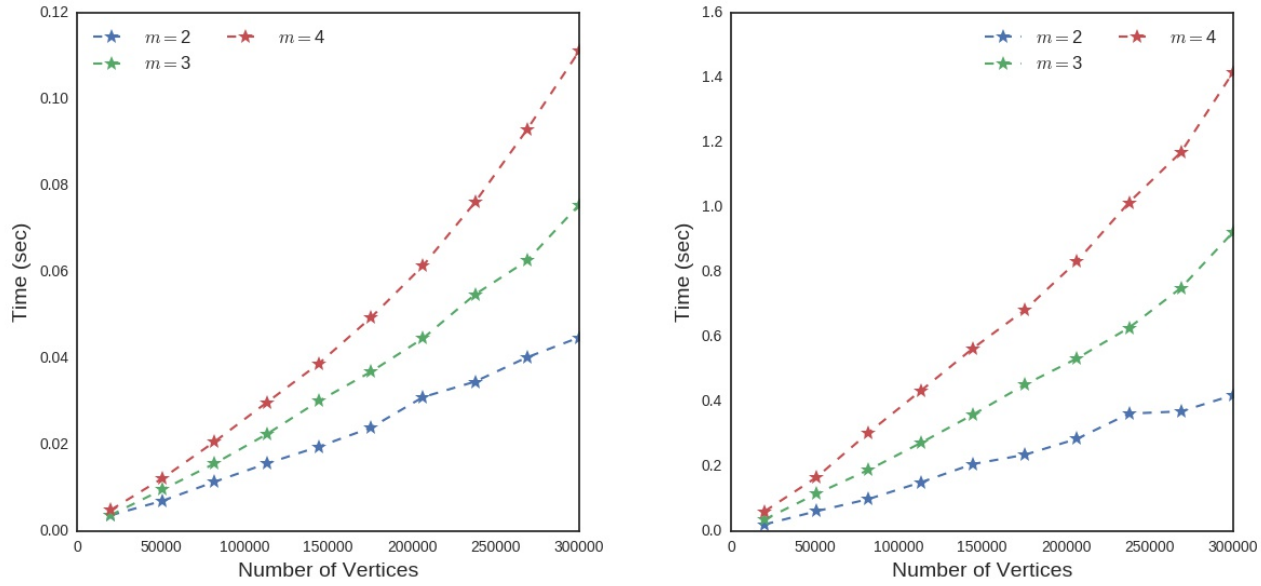
We considered three different values of m, namely $m = 2, 3, 4$. For a fixed value of $|V|$ and m we artificially generated a BA graph with $|V|$ vertices and an average vertex degree equal to m and calculated the time required to perform Cocitation and Bibliographic Coupling tasks on that graph. We repeated the above procedure ten times to avoid statistical fluctuations and calculated the average running time (which are graphically in Figures 7a and 7b.

Figures 7a and 7b outline a superlinear increase of the running time in a narrow range of variation of $|V|$. As such, we may conclude that graph density (i.e., the ratio of edges in a graph to the overall number of pairs of vertices) plays a crucial role in the execution of cocitation and bibliographic coupling procedures and this explains great differences in execution times we observed in graphs roughly having the same number of vertices.

8 Linking Algorithms to Experimental Performances

As the range of methods to detect communities in graphs is rapidly expanding, researchers and practitioners may benefit from our discussion, which will provide some guidelines about which of the available algorithms is the most appropriate for answering the research or business questions they are asking.

Our discussion will also advise some fundamental questions related to the problem of detecting communities in directed graphs that, despite the large body of work, have yet to be fully addressed.



(a) Cocitation

(b) Bibliographic Coupling

Fig. 7: **Left**: Running time (in sec.) to perform Cocitation task on Barabasi-Albert Graphs with size ranging from 20 000 to 300 000 and average vertex degree in $\{2, 3, 4\}$. **Right**: Running time (in sec.) to perform Bibliographic Coupling task on Barabasi-Albert Graphs with size ranging from 20 000 to 300 000 and average vertex degree in $\{2, 3, 4\}$.

Our first remark is about the importance of testing community detection algorithms on artificially generated graphs. Accuracy is a major concern in choosing a community detection algorithm. From our previous discussion, the most reliable way to evaluate the accuracy of a community detection algorithm requires that real graphs – with known community structure (ground truth) – are available. In this case, accuracy assessment can be done by comparing the output of the algorithm to test with the *a-priori* assignment of vertices to communities. Unfortunately, in case of directed networks, we lack of sufficiently large graphs with known community structure.

In the light of these observations, we recommend practitioners and researchers to evaluate the accuracy of a community detection algorithm on benchmark graphs with an inherent community structure. In this paper, we used the popular LFR benchmark to generate benchmark graphs; LFR benchmark was initially designed to handle undirected graphs and it was later extended to consider directed graphs. Our belief is that further research is needed to produce more realistic benchmark graphs which fully replicate the structure of true graphs.

Finally, we noticed that WalkTrap achieved the highest NMI on artificial graphs even if the gap separating the NMI of WalkTrap, LPA and Infomap was not particularly big, especially if we concentrate on graphs showcasing a clear community structure. As such, all algorithms examined in this paper are good candidates to extract communities from graphs describing Web-based system.

Secondly, edge directionality plays an important role in the process of discovering communities. Our experiments on both real and artificial graphs showed that operations like cocitation and bibliographic coupling may deeply alter the structure of the input graph. The accuracy of a community detection algorithm may be negatively influenced by symmetrization procedures because, after mapping a directed graph onto an undirected one, some graph substructures in the symmetrized graph my be wrongly classified as communities of the directed graph. A further drawback is that we have no guidance in deciding what is the best symmetrization procedures to adopt for answering a specific research question. *In general, we do not recommend to symmetrize the input graph.*

On one hand, further research is required to design and test algorithms explicitly defined for operating on directed graphs. On the other hand, we should revisit our notion of community to take link directionality into account.

Most of the proposed approaches formalize the concept of community as a group of vertices featuring a larger intra-group than inter-group connectivity. However, in large graphs over the Web, a vertex may belong to a small community and, then, it is likely it has more links pointing outside its community than inside.

This raises some relevant questions: is it sufficient to consider only within- and intra-connectivity as leading criteria to define communities?

Some researchers [37] introduced the concepts of *in-communities* (i.e., subgraphs with few inbound edges and some outbound edges), *out-communities* (i.e., subgraphs with some inbound edges and few outbound edges) and finally *in/out-communities* (i.e., dense subgraphs showing both few inbound and outbound edges).

These definitions have been later generalized to the case of directed weighted graphs to include new peculiar substructures known as *pseudo-communities*.

Pseudo-communities are subgraphs with a "star-like" topology in which most of the vertices points toward vertices inside the pseudo-community but few vertices emanate edges directed outside the pseudo-community itself.
An example of pseudo-community can be found in the flight transportation network where regional "hubs" collect most of the traffic generated by domestic airports and forward it to the rest of the world.
[37] describes also an algorithm to discover in-, out, in/out-communities as well as pseudo communities and experiments carried out on real-life graphs prove that these substructures play a crucial role to correctly interpret the behaviour of many real-life systems. Unlike all the algorithms presented in this paper, the approach of [37] does not generate a partitioning of the input graph, i.e., a vertex may not belong to any of the graph substructures the algorithm is able to detect. Such a choice appears to be fully reasonable because if we would force an algorithm to assign each vertex to a community, we may incur in low quality communities.

As a further remark, Time complexity is a crucial factor in the selection of a community detection algorithm. Processing large-scale Web datasets poses significant computational challenges: in fact, if we are in charge of detecting communities in graphs consisting of millions of vertices and edges, only algorithms with $O(|V|)$ time and space complexity are a valid option.
From our experiments, we found that the most efficient algorithm was the Label Propagation Algorithm; we also found that Infomap was able to process large amount of data in a relatively small time-frame. *Therefore, we recommend the usage of LPA and Infomap when the graphs to process are large.*
A major research avenue in the efficient processing of big graphs by means of distributed or parallel computational models like message passing interface (MPI) [63], bulk synchronous parallel (BSP) [64], MapReduce [65] and Apache Spark [66].
A parallel implementation of the algorithms analysed in this paper is quite promising; parallel algorithms that run on clusters of computers have two advantages over serialized algorithms: the running time meaningfully is reduced by processing the data in parallel and the large availability of memory/disk space beyond that provided by a single machine ensures that large amount of data can be processed. The first research results in this direction are discussed in [67] which describes two implementation based on Hadoop and Spark of a community detection algorithms which is based on the propagation of local and global information.
A further approach to cite is [68], which describes an extensible community detection framework with shared-memory parallelism. The described platform implements a parallel version of the LPA algorithms, a parallel version of the Louvain method [14] (a popular method to find communities in undirected graphs) and an ensemble scheme combining the methods cited above.

9 Community Detection, when is useful?

In the following we describe some domains which may benefit of community detection tools in directed graphs.

9.1 Criminal Networks

Law enforcement agencies often complement traditional investigation methods with procedures which make extensive use of new technologies.

A powerful and well-known investigation method is *tapping*, i.e., the procedure of recording information flow among suspected criminals which has been sent using any type of electronic media, like phone calls (from both fixed lines and mobiles), emails, SMS messages and private communications over Social Media platforms. Tapping has proven to be effective for preventing and solving many crimes like terrorism, drugs, kidnapping and political corruption and, among the possible sources of intelligence, Web 2.0 platforms occupy a prominent position: in fact, analysts can monitor Social Web platforms to get the first tip-off that breaking events like terrorist attacks or incidents have occurred as well as to geo-locate emerging threats [69].

Some studies agree on representing criminal organizations as sparse and clustered networks, often showcasing scale-free and small world properties [70], [71], [72].

Calderoni and Piccardi [73] observed that the larger a criminal organization, the most likely is the emergence of subgroups; the identification of subgroups in criminal networks is a key tool to unveiling the internal structure of large organized crime groups and designing effective strategies against them.

Community detection algorithms have been applied on the graph encoding relationships among criminals to identify and visualize groups of individuals who frequently interact and quantify information flows within a criminal organization [74].

To the best of our knowledge, most studies about the structure and communities within a criminal organization depict the organization itself as an undirected graph, thus assuming that relations among individuals are symmetric.

In a recent paper, Zheng *et al.* [75] criticized such an assumption; they suggested that members of a gang can have different power within an organization and, as such, relations should be asymmetric. [75] introduced a new *directed-graph embedding techniques* that is useful to visualize and analyse criminal networks. The proposed embedding enables the definition of a new centrality metrics which assesses the importance of individuals on the basis of their tendency to spread information to other gang members. Community detection algorithms for directed graphs has not yet been applied to the study of criminal network and we see in these techniques a high potential. In fact, we could run community detection algorithms to check if criminal organizations gravitate around few but powerful groups of criminal or by contrast, if an horizontal organization (in which many gangs equally control illicit activities like gambling or money laundering) emerges. Such a knowledge would represent the first step to point out vulnerabilities in a gang and leverage them to arrest the criminals who reduce most the level of criminality of the gang.

9.2 Analysis of Political Debates

Many political institutions (like national or transnational parliaments) freely distribute verbatim reports of their ple-

nary meetings[2]. Events in parliaments present a hierarchical structure: monthly sessions occupy the top of the hierarchy, followed by daily sessions, agenda items and speeches. Information about speakers (like their parties or membership to a commission) is usually available.

Recently, [76] showcased how the proceedings of European Parliament can be published as Linked Open Data. Such a procedure gave rise to a dataset in which heterogeneous entities (e.g., countries and speakers) and their relationships were recorded. Entities were identified by a unique code, thus allowing to link each entity to its contraparty in other Semantic Web datasets: for instance, entities referring to countries were connected to entities in GeoNames[3], a geographical database.

Members of Parliament were linked to their entries in DBpedia and, if possible, to their homepage in their home parliament. A powerful query interface was available, enabling lay users to extract data about a speech, biographical information of the speaker and information about the speaker's country.

This wealth of information generates various kinds of directed graphs. For instance, we could see the speakers (members of parliament) as vertices, and create an edge if they mention (or react to) each other in a debate.

Alternatively, we could see debates as vertices and, in that case, edges could have different meanings: an edge may connect two debates if they share the same speaker, if they deal with the same topics, if the speakers are members of the same party or they are from the same country and so on.

From our discussion, we see a huge potential from analysing community structure in directed graphs associated with political debates. Such a procedure will allow, in fact, to identify what are the interests shared by parliamentarians who sit in different parties or are from different countries. Community detection algorithms allows for measuring the *degree of cohesion* of political groups: parliamentarians often vote against their group line and, sometimes, this leads to the creation of new groups. If we would periodically run community detection algorithms on graphs describing political debates, we would be able to identify what are the *most destabilizing policy areas*: for instance, we could understand if discussion on budgetary control affects the cohesion of a political group more than discussion on foreign and security policy.

9.3 Analysis of Media Coverage

Events like US political elections, London riots in 2012 or the Fukushima nuclear disaster in 2011 are able to generate high volumes of traffic on the Web and capture the interest of large segments of public opinion [77].

Community detection algorithms are a powerful tool to analyse how media cover a particular event, what are the key actors associated with that event and their relationships.

Political elections and their media coverage are an interesting but challenging case study. In advanced democracies, in fact, elections-related data are quite large and usually come from a multitude of sources (e.g., traditional/online newspapers, blogs, Social Media and so on). Even assuming the full coverage of available sources, the core algorithmic methods used till now mainly rely on content analysis [78].

Recently, [79] suggested to apply Natural Language Processing along with graph theory tools (in particular, community detection algorithms) to better define the ideological positions of candidates. [79] extracted $130,213$ news articles related to the US presidential elections from 719 news outlets. Candidates expressed their political orientation through claims by which they either endorse or oppose to other candidates. These claims were converted into subject-verb-object triplets and were subsequently mapped onto a directed graph G, whose edges retained the semantics of original triplet. The graph G was partitioned into two groups and, in this way, for each candidate it was possible to infer her/his degree of membership to each group.

The discussion above illuminates on the great promises of techniques developed in areas like Web Scraping, Natural Language Processing, Community Detection and Big Data. Taken together, these technologies enable to collect data generated from independent sources and cast them into graphs. Community Detection algorithms intervene in the last stage of this pipeline and they enable to monitor how the population reacts to political facts and act as sensors to early detect large-scale social phenomena.

9.4 Device-to-Device Communications, Proximity-Aware Applications and Public Safety

Device-to-Device (in short, D2D) communications is a novel transmission paradigm that enables mobile users to send/receive data from one of their peers without the need of routing traffic through base-stations.

D2D offers key technology to enabling powerful *proximity-aware applications* as well as for *public safety*.

Proximity-aware applications search their local physical environment to find out those people, products and services that, in a specific time and in a specific place, can be useful to the end users. For example, a Facebook user could be alerted to the presence of a friend in a cinema or in a coffee shop.

Traditional architectures to implement proximity-aware applications are *centralized* and inefficient in managing resources like radio spectrum and device batteries.

A decentralized approach to handling communication among devices is essential to appraise the market impact of proximity-aware applications but it has relevant consequences in case of public safety services: in fact, organizations like police, ambulances or fire services should support direct communication between mobile devices to effectively deal with emergencies, even without the coverage of a mobile network or when the network is not operational.

Some communication standards like WiFi and Bluetooth already support short-range wireless communication but they require manual pairing of devices. Additionally, the security features of WiFi and Bluetooth are much less robust than those existing in cellular systems and this is unacceptable for public safety applications. Finally, WiFi and Bluetooth run in parallel with cellular communication system, thus yielding a relevant drain in device batteries.

D2D overcomes these drawbacks: users can form clusters to communicate more efficiently and this reduces battery

2. http://www.europarl.europa.eu/plenary/en/home.html
3. See http://www.geonames.org/

consumption as well as latency in sharing files [80], [81], [82], [83]. Usually, user clusters are found by taking only physical proximity into account but, more recently, social interactions and user proximities have been considered [80]. The proposed approach relies on the *Chinese Restaurant Process-CRP*, a popular stochastic process [84] to perform clustering.

Research papers discussed in this section show that the formation of user clusters is a cost-effective solution to data sharing in a mobile environment. Efficient community detection algorithms capable of handling a broad range of user-related data are therefore a great asset to design new mass market mobile applications.

ACKNOWLEDGMENTS

We are grateful to Associate Editor and anonymous reviewers for their valuable and constructive comments which helped us to improve the quality of our manuscript. The authors would like to thank Dr. Ursula Redmond for kindly providing the Prosper dataset and the Department of Computer Science, Mathematics, Physics and Earth Sciences at the University of Messina for kindly providing computing facilities.

REFERENCES

[1] D.Cook and L. Holder, *Mining graph data*. John Wiley & Sons, 2006.
[2] M. Nisar, A. Fard, and J. Miller, "Techniques for graph analytics on Big Data," in *Proc. of IEEE International Congress on Big Data (BigData 2013)*. Santa Clara, CA , USA: IEEE, 2013, pp. 255–262.
[3] S. Aral and D. Walker, "Identifying influential and susceptible members of social networks," *Science*, vol. 337, no. 6092, pp. 337–341, 2012.
[4] S. Papadopoulos, Y-Kompatsiaris, A. Vakali, and P. Spyridonos, "Community detection in social media," *Data Mining and Knowledge Discovery*, vol. 24, no. 3, pp. 515–554, 2012.
[5] L. Tang and H. Liu, "Community detection and mining in social media," *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 1–137, 2010.
[6] G. Flake, S. Lawrence, C. Giles, and F. Coetzee, "Self-organization and identification of Web communities," *IEEE Computer*, vol. 35, no. 3, pp. 66–70, 2002.
[7] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Mixing local and global information for community detection in large networks," *Journal of Computer and Systems Sciences*, vol. 80, no. 1, pp. 72–87, 2014.
[8] P. De Meo, G. Quattrone, and D. Ursino, "A query expansion and user profile enrichment approach to improve the performance of recommender systems operating on a folksonomy," *User Modeling and User-Adapted Interaction*, vol. 20, no. 1, pp. 41–86, 2010.
[9] A. Milicevic, A. Nanopoulos, and M. Ivanovic, "Social tagging in recommender systems: a survey of the state-of-the-art and possible extensions," *Artificial Intelligence Review*, vol. 33, no. 3, pp. 187–209, 2010.
[10] P. De Meo, G. Quattrone, and D. Ursino, "Integration of the HL7 Standard in a multiagent system to support personalized access to e-health services," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 8, pp. 1244–1260, 2011.
[11] S. Soundarajan and J. Hopcroft, "Using community information to improve the precision of link prediction methods," in *Proc. of the International Conference Companion on World Wide Web (WWW 2012)*. ACM, 2012, pp. 607–608.
[12] R. Andersen, F. Chung, and K. Lan, "Local graph partitioning using pagerank vectors," in *Proc. of the Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*. Berkeley, CA, USA: IEEE, 2006, pp. 475–486.
[13] A. Mahmood and M. Small, "Subspace based network community detection using sparse linear coding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 801–812, 2016.
[14] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.
[15] F. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Physics Reports*, vol. 533, no. 4, pp. 95–142, 2013.
[16] M. Rosvall and C. Bergstrom, "Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems," *PLoS ONE*, vol. 6, no. 4, p. e18209, 2011.
[17] U. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review E*, vol. 76, no. 3, p. 036106, 2007.
[18] J. Xie, B. Szymanski, and X. Liu, "SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process," in *Proc. of the International Conference on Data Mining Workshops*. IEEE, 2011, pp. 344–349.
[19] P. Pons and M. Latapy, "Computing communities in large networks using random walks," *Journal of Graph Algorithms and Applications*, vol. 10, no. 2, pp. 191–218, 2006.
[20] E. Leicht and M. Newman, "Community structure in directed networks," *Physical Review Letters*, vol. 100, no. 11, p. 118703, 2008.
[21] A. Lancichinetti and S. Fortunato, "Community detection algorithms: a comparative analysis," *Physycal Review E*, vol. 80, p. 056117, 2009.
[22] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, p. P09008, 2005.
[23] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.
[24] S. Harenberg, G. Bello, L. Gjeltema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, and N. Samatova, "Community detection in large-scale networks: a survey and empirical evaluation," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 6, no. 6, pp. 426–439, 2014.
[25] J. Xie, S. Kelley, and B. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study," *ACM Computing Surveys*, vol. 45, no. 4, p. 43, 2013.
[26] J. Leskovec, K. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in *Proc. of the International Conference on World Wide Web (WWW 2010)*, Raleigh, North Carolina, USA, 2010, pp. 631–640.
[27] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," *Journal of the ACM*, vol. 51, no. 3, pp. 497–515, 2004.
[28] M. Granovetter, "The strength of weak ties: A network theory revisited," *Sociological theory*, vol. 1, no. 1, pp. 201–233, 1983.
[29] J. Zhao, J. Wu, and K. Xu, "Weak ties: Subtle role of information diffusion in online social networks," *Physical Review E*, vol. 82, no. 1, p. 016105, 2010.
[30] P. Grabowicz, J. Ramasco, E. Moro, J. Pujol, and V. Eguiluz, "Social features of online networks: The strength of intermediary ties in online social media," *PloS one*, vol. 7, no. 1, p. e29358, 2012.
[31] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "On Facebook, most ties are weak," *Communications of the ACM*, vol. 57, no. 11, pp. 78–84, 2014.
[32] P. Marsden and K. Campbell, "Measuring tie strength," *Social forces*, vol. 63, no. 2, pp. 482–501, 1984.
[33] E. Gilbert and K. Karahalios, "Predicting tie strength with social media," in *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009, pp. 211–220.
[34] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic, "The role of social networks in information diffusion," in *Proc. of the World Wide Web Conference (WWW 2012)*, Lyon, France, 2012, pp. 519–528.
[35] J. Whang, D. Gleich, and I. Dhillon, "Overlapping community detection using neighborhood-inflated seed expansion," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 5, pp. 1272–1284, 2016.
[36] Z. Ding, X. Zhang, D. Sun, and B. Luo, "Overlapping community detection based on network decomposition," *Scientific reports*, vol. 6, 2016.
[37] P. Landi and C. Piccardi, "Community analysis in directed networks: In-, out-, and pseudocommunities," *Physical Review E*, vol. 89, no. 1, p. 012814, 2014.
[38] M. Newman, "Detecting community structure in networks," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 38, no. 2, pp. 321–330, 2004.
[39] J. Cao, Z. Bu, G. Gao, and H. Tao, "Weighted modularity optimization for crisp and fuzzy community detection in large-scale

networks," *Physica A: Statistical Mechanics and its Applications*, vol. 462, pp. 386–395, 2016.
[40] A. Barabasi, *Network science*. Cambridge University Press, 2016.
[41] S. Gregory, "Finding overlapping communities in networks by label propagation," *New Journal of Physics*, vol. 12, no. 10, p. 103018, 2010.
[42] M. Rosvall and C. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.
[43] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.
[44] P. Grunwald, I. Myung, and M. Pitt, *Advances in minimum description length: Theory and applications*. MIT press, 2005.
[45] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.
[46] C. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.
[47] D. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.
[48] M. Girvan and M. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.
[49] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Phys. Rev. E*, vol. 74, p. 016110, Jul 2006.
[50] L. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.
[51] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of Mathematical Sociology*, vol. 25, pp. 163–177, 2001.
[52] A. Arenas, J. Duch, A. Fernández, and S. Gómez, "Size reduction of complex networks preserving modularity," *New Journal of Physics*, vol. 9, no. 6, p. 176, 2007.
[53] P. Chen and S. Redner, "Community structure of the physical review citation network," *Journal of Informetrics*, vol. 4, no. 3, pp. 278–290, 2010.
[54] M. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E*, vol. 74, no. 3, p. 036104, 2006.
[55] P. Massa and P. Avesani, "Controversial users demand local trust metrics: An experimental study on `epinions.com` community," in *Proc. of the National Conference on Artificial Intelligence (AAAI 2005)*, Pittsburgh, Pennsylvania, 2005, pp. 121–126.
[56] U. Redmond and P. Cunningham, "A temporal network analysis reveals the unprofitability of arbitrage in the prosper marketplace," *Expert Systems with Applications*, vol. 40, no. 9, pp. 3715–3721, 2013.
[57] S. Agreste, P. De Meo, E. Ferrara, S. Piccolo, and A. Provetti, "Trust networks: Topology, dynamics, and measurements," *IEEE Internet Computing*, vol. 19, no. 6, pp. 26–35, 2015.
[58] J. Leskovec, L. Adamic, and B. Huberman, "The dynamics of viral marketing," *ACM Transactions on the Web*, vol. 1, no. 1, p. 5, 2007.
[59] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney, "Statistical properties of community structure in large social and information networks," in *Proc. of the International Conference on World Wide Web, (WWW 2008)*, Beijing, China, 2008, pp. 695–704.
[60] ——, "Statistical properties of community structure in large social and information networks," in *Proc. of the International Conference on World Wide Web (WWW 2008)*. Beijing, China: ACM Press, 2008, pp. 695–704.
[61] A. Clauset, C. Shalizi, and M. Newman, "Power-law distributions in empirical data," *SIAM review*, vol. 51, no. 4, pp. 661–703, 2009.
[62] J. Alstott, E. Bullmore, and D. Plenz, "powerlaw: a Python package for analysis of heavy-tailed distributions," *PloS one*, vol. 9, no. 1, p. e85777, 2014.
[63] D. Walker and J. Dongarra, "MPI: a standard message passing interface," *Supercomputer*, vol. 12, pp. 56–68, 1996.
[64] L. Valiant, "A bridging model for parallel computation," *Communications of the ACM*, vol. 33, no. 8, pp. 103–111, 1990.
[65] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[66] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012, pp. 2–2.
[67] K. Guo, W. Guo, Y. Chen, Q. Qiu, and Q. Zhang, "Community discovery by propagating local and global information based on the mapreduce model," *Information Sciences*, vol. 323, pp. 73–93, 2015.
[68] C. Staudt and H. Meyerhenke, "Engineering parallel algorithms for community detection in massive networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 1, pp. 171–184, 2016.
[69] O. Oh, M. Agrawal, and R. Rao, "Information control and terrorism: Tracking the mumbai terrorist attack through twitter," *Information Systems Frontiers*, vol. 13, no. 1, pp. 33–43, 2011.
[70] A. Malm and G. Bichler, "Networks of collaborating criminals: Assessing the structural vulnerability of drug markets," *Journal of Research in Crime and Delinquency*, vol. 48, no. 2, pp. 271–297, 2011.
[71] C. Morselli, C. Giguère, and K. Petit, "The efficiency/security trade-off in criminal networks," *Social Networks*, vol. 29, no. 1, pp. 143–153, 2007.
[72] C. Morselli, *Inside criminal networks*. Springer, 2009.
[73] F. Calderoni and C. Piccardi, "Uncovering the structure of criminal organizations by community analysis: The infinito network," in *Proc. of the International Conference on Signal-Image Technology and Internet-Based Systems (SITIS 2014)*. Marrakhech, Morocco: IEEE, 2014, pp. 301–308.
[74] F. Calderoni, D. Brunetto, and C. Piccardi, "Communities in criminal networks: A case study," *Social Networks*, vol. 48, pp. 116–125, 2017.
[75] Q. Zheng, D. Skillicorn, and F. Calderoni, "Analysis of criminal social networks with typed and directed edges," in *Proc. of the International Conference on Intelligence and Security Informatics, (ISI 2015)*. Baltimore, MD, USA: IEEE, 2015, pp. 1–6.
[76] L. Hollink, M. Kleppe, M. Kemman, A. van Aggelen, and W. van Hage, "The possibilities and challenges of using linked data for academic research: the case of the talk of europe project," 2015.
[77] T. Lansdall-Welfare, S. Sudhahar, G. A. Veltri, and N. Cristianini, "On the coverage of science in the media: A big data study on the impact of the Fukushima disaster," in *Proc. of the IEEE International Conference on Big Data (Big Data 2014)*, Washington, DC, USA,, 2014, pp. 60–66.
[78] H. A. Semetko and P. M. Valkenburg, "Framing european politics: A content analysis of press and television news," *Journal of communication*, vol. 50, no. 2, pp. 93–109, 2000.
[79] S. Sudhahar, G. De Fazio, R. Franzosi, and N. Cristianini, "Network analysis of narrative content in large corpora," *Natural Language Engineering*, vol. 21, no. 1, 2015.
[80] G. Araniti, M. De Sanctis, S. Spinella, M. Monti, E. Cianca, A. Molinaro, A. Iera, and M. Ruggieri, "Hybrid system HAP-WiFi for incident area network," in *Personal Satellite Services*. Springer, 2010, pp. 436–450.
[81] L. Wang, L. Yang, V. Schober, and M. Song, "Adaptive cooperation schemes for energy efficient physical layer security," in *Proc. of the Computer Communications Workshops (IEEE INFOCOM WKSHPS 2014)*. IEEE, 2014, pp. 159–160.
[82] B. Zhou, H. Hu, S. Huang, and H. Chen, "Intracluster device-to-device relay algorithm with optimal resource utilization," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 5, pp. 2315–2326, 2013.
[83] L. Militano, M. Condoluci, G. Araniti, A. Molinaro, and A. Iera, "When D2D communication improves group oriented services in beyond 4G networks," *Wireless Networks*, vol. 21, no. 4, pp. 1363–1377, 2014.
[84] J. Pitman, *Combinatorial Stochastic Processes: Ecole D'Eté de Probabilités de Saint-Flour XXXII-2002*. Springer, 2006.

Santa Agreste is assistant professor of Computer Science at the Messina University, Italy. In 2007, she received the Ph.D. degree in computer science from the University of Milan. Her research interests include image processing, social network analysis and parallel computing. Contact her at `sagreste@unime.it`.

Pasquale De Meo is an associate professor of computer science at the Department of Ancient and Modern Civilizations at the University of Messina, Italy. His main research interests are in the area of social networks, recommender systems, and user profiling. De Meo has a PhD in systems engineering and computer science from the University of Calabria. He has been the Marie Curie Fellow at Vrije Universiteit Amsterdam. Contact him at pdemeo@unime.it.

Giacomo Fiumara is Assistant Professor of Computer Science at the University of Messina, Italy since 2009. In 1993 he took his PhD in Physics. His research interests include network science, criminal networks, simulations of model systems. He has published more than 50 papers in international journals and conference proceedings. He is members of various conference PCs. Contact him at gfiumara@unime.it.

Giuseppe Piccione is a BsC student at the University of Messina, Italy. His main research interests are in graph mining and analytics, network science, and Big Data processing.

Sebastiano Piccolo is a PhD student at the Department of Management Engineering, Production and Service Management. His main research interests are in data and process mining, network science, and complex systems. Contact him at sebpi@dtu.dk.

Domenico Rosaci is assistant professor of Computer Science at the University Mediterranea of Reggio Calabria, Italy. In 1999, he took the PhD in Electronic Engineering. His research interests include distributed artificial intelligence, multi-agent systems, trust and reputation in social communities. He has published more than 120 papers in outstanding international journals and conference proceedings. He is a member of a number of conference PCs and he is Associate Editor of Journal of Universal Computer Science (Springer). Contact him at domenico.rosaci@unirc.it.

Giuseppe M. L. Sarné is assistant professor of Computer Science at the University Mediterranea of Reggio Calabria, Italy. His research interests include distributed artificial intelligence, multi-agent systems, trust and reputation in social communities. He has published more than 100 papers in outstanding international journals and conference proceedings. He is a member of a number of conference PCs. Contact him at sarne@unirc.it.

Athanasios V. Vasilakos is Professor with the Lulea University of Technology . He served or is serving as an Editor for many technical journals,such as the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT; IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY,IEEE TRANSACTIONS ON CYBERNETICS; IEEE TRANSACTIONS ON NANOBIOSCIENCE; IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE; ACM Transactions on Autonomous and Adaptive Systems; the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS . He is also General Chair of the European Alliances for Innovation(www.eai.eu).