Unfolding network communities by combining defensive and offensive label propagation

Lovro Šubelj* and Marko Bajec

Laboratory for Data Technologies, University of Ljubljana, Ljubljana, Slovenia

Abstract. Label propagation has proven to be a fast method for detecting communities in complex networks. Recent work has also improved the accuracy and stability of the basic algorithm, however, a general approach is still an open issue. We propose different label propagation algorithms that convey two unique strategies of community formation, namely, defensive preservation and offensive expansion of communities. Furthermore, the strategies are combined in an advanced label propagation algorithm that retains the advantages of both approaches; and are enhanced with hierarchical community extraction, prominent for the use on larger networks. The proposed algorithms were empirically evaluated on different benchmarks networks with planted partition and on over 30 real-world networks of various types and sizes. The results confirm the adequacy of the propositions and give promising grounds for future analysis of (large) complex networks. Nevertheless, the main contribution of this work is in showing that different types of networks (with different topological properties) favor different strategies of community formation.

Keywords: Network communities, label propagation, defensive preservation, offensive expansion.

1 Introduction

Complex networks commonly comprise of local structural *modules* or *communities* that are groups of nodes strongly connected within and only weakly connected with the rest of the network. These modules play crucial roles in many real-world systems [15, 37], moreover, they provide an important insight into structure and function of (large) complex networks [37, 45, 27].

Over the last decade the research community has shown a considerable interest in detecting communities in real-world networks. Thus, a number of approaches has been presented in the literature. In particular, approaches optimizing *modularity*[1] Q [7, 6, 5], graph partitioning [14, 39, 38] and spectral [9, 33] algorithms, statistical methods [36], algorithms based on dynamic processes [40, 43, 38, 42], overlapping, hierarchical and multiresolution methods [37, 16, 42], and other [29, 30] (for a thorough review see [11]).

* Corresponding author: lovro.subelj@fri.uni-lj.si.

[1] Significance of communities due to a selected *null model* [35].

Due to the size of large real-world networks recent research has focused on developing scalable algorithms that can be applied to networks with several millions of nodes and billions of edges. Raghavan et al. [40] proposed using simple *label propagation*, where labels are propagated among nodes until an equilibrium is reached. The main advantage of the label propagation is its near linear time complexity (in the number of edges of the network); however, due to the algorithm's simplicity, the accuracy of revealed community structure is often not state-of-the-art.

The basic algorithm was further analyzed in [47] and refined into a modularity optimization algorithm in [5, 29]. Extension to directed networks was considered in [28]. Furthermore, Leung et al. [28] improved the basic label propagation by applying label *hop attenuation* and *node preference* (i.e. node propagation strength). We proceed their work in developing two unique strategies of community formation, namely, *defensive preservation* of communities, where preference is applied to the core of each community, and *offensive expansion* of communities, where preference is applied to the border of each community. Moreover, the two strategies are combined into an advanced label propagation algorithm (denoted *K-Cores*) that preserves the advantages of both approaches. For the use with larger networks, we also present two different manners of *hierarchical* community extraction.[2]

Proposed algorithms were rigorously analyzed on different benchmark networks with planted partition and on a large number of real-world networks of various types and sizes. The results justify the adequacy of the propositions and give promising grounds for future analysis of (large) complex networks. Furthermore, the analysis also shows that the appropriateness of the strategies of community formation strongly correlates with the type of the network (i.e. with its topological properties).

The rest of the article is structured as follows. Section 2 gives a formal presentation of label propagation and briefly surveys relevant subsequent refinements of the basic algorithm. Defensive and offensive strategies of community formation, and corresponding algorithms, are presented and discussed in section 3. Empirical evaluation with discussion is done in section 4 and conclusion in section 5.

2 Label propagation and advances

Let the network be represented by an undirected (multi-)graph $G(N, E)$, where N is the set of nodes and E is the set of edges. Furthermore, let w_{nm} be the weight of the edge between nodes n and m, $n, m \in N$. Next, denote c_n to be the community (label) of node n and $\mathcal{N}_n$ the set of its neighbors. Moreover, denote $\mathcal{N}_n^l$ to be the set of neighbors of n that share label l.

Label propagation algorithm (*LPA*) [40] reveals network communities by employing the following procedure. At first each node $n \in N$ is labeled with an

[2] The work presented in this article was already (partially) presented in [46].

unique label, $c_n = l_n$. Next, at each iteration, each node adopts the label shared by most of its neighbors. Hence,

$$c_n = \underset{l}{\operatorname{argmax}} |\mathcal{N}_n^l|, \tag{1}$$

where in the case of ties one of the labels is selected at random (node n retains its current label, when it is among most frequent in $\mathcal{N}_n$). The process continues until none of the labels change anymore, i.e. an equilibrium is reached. During the course of the algorithm, densely connected sets of nodes form a consensus on some particular label; thus, at the end, nodes sharing the same label are classified into the same community.

Leung et al. [28] have observed that basic label propagation applied to large (web) graphs commonly produces one *major community* that occupies most of the nodes. However, they have shown that the emergence of a major community can be eliminated by using label *hop attenuation* technique. Each label l_n has associated an additional score s_n (initially set to 1) that decreases by δ after each propagation (δ is an *attenuation ratio*). When s_n reaches 0, the label l_n no longer propagates onward (see Eq. (4)), which successfully eliminates the emergence of a major community.

Label hop attenuation can be rewritten into an equivalent form that allows altering δ during the course of the algorithm [28]. One keeps the label distance from the origin d_n (initially set to 0) that is updated after each propagation. Hence,

$$d_n = \left(\min_{m \in \mathcal{N}_n^{c_n}} d_m \right) + 1, \tag{2}$$

when the score s_n is then

$$s_n = 1 - \delta d_n. \tag{3}$$

Further analysis in [28] has revealed that label hop attenuation has to be coupled with *node preference* f_m (i.e. node propagation strength), in order for the algorithm to improve on the basic label propagation. Thus, the label propagation updating rule in Eq. (1) is transformed into

$$c_n = \underset{l}{\operatorname{argmax}} \sum_{m \in \mathcal{N}_n^l} f_m^\alpha s_m w_{nm}, \tag{4}$$

where α is a parameter of the algorithm. Leung et al. [28] have experimented with node preference equal to the degree of the node (i.e. $f_m = deg_m$ and $\alpha = 0.1$), however, no general analysis was conducted.

The updating rule of label propagation (Eq. (1)), or its refinements (Eq. (4)), might prevent the algorithm from converging [40]. Imagine a *bipartite network* with two sets of nodes, i.e. red and blue nodes. Let, at some iteration of the algorithm, all red nodes share label l_r and all blue nodes share label l_b. Due to the bipartite structure of the network, at the next iteration, all red, blue nodes

will adopt label l_b, l_r respectively. Furthermore, after the next iteration, all nodes will recover their original labels, failing the algorithm to converge.

The problem can be avoided by using *asynchronous* updating [40]. Nodes are no longer updated all together, but sequentially, in a random order. Thus, when node's label is updated, (possibly) already updated labels of its neighbors are considered (in contrast to *synchronous* updating that considers only labels from the previous iteration). All of the algorithms, presented in the following section, use such asynchronous updating of nodes.

3 Defensive and offensive label propagation

In this section we present different algorithms that employ two unique strategies of community formation, namely, *defensive preservation* and *offensive expansion* of communities. First, we briefly present a *dynamic hop attenuation* technique in section 3.1. Next, section 3.2 introduces and formally discusses the two strategies and associated algorithms (denoted *dDaLPA* and *oDaLPA* respectively). Last, section 3.3 presents an advanced label propagation algorithm (denoted *K-Cores*) that combines the two strategies in an iterative manner, thus retaining the advantages of both approaches.

3.1 Dynamic hop attenuation

Label hop attenuation has proven to be a reliable technique for prevention of emergence of a major community (section 2). Still, it is not immediately evident what should the value of attenuation ratio δ be. In [28] authors have obtained good results with values around 0.1, however, only a limited set of networks was considered.

We propose a *dynamic hop attenuation* technique based on the hypothesis[3] that hop attenuation should only be employed when a label, or a set of labels, rapidly occupies a large portion of the network (which could potentially result in a formation of a major community). Otherwise, the restriction should be (almost) completely relaxed to allow label propagation to reach the equilibrium unrestrained. The technique would thus retain the dynamics of label propagation, but still successfully prevent the emergence of a major community.

We employ the following hop attenuation strategy. After each iteration (i.e. sweep through all the nodes) δ is set to the proportion of nodes that changed their labels[4] (on the first two iterations δ is set to 0.5 and 0.1 respectively). In practice, this results in higher values of δ in the early iterations of the algorithm, which enables the occurrence of a larger number of (smaller) well defined communities, when in the later stages δ gradually converges to 0, which refines the communities and preserves only those strongly depicted in the network topology. Moreover, empirical analysis on real-world networks shows that such dynamic strategy

[3] Similar idea was already discussed in [28].

successfully eliminates the emergence of a major community (the exact results are omitted).

Note that an additional constraint should be imposed to prevent extremely large values of δ in the late stages of the algorithm (which, due to the above discussion, indicates some spurious behavior). Thus, to ensure convergence, when δ is greater than δ_{max} (e.g. $\delta_{max} = 0.5$), we set it to 0.

3.2 Defensive preservation and offensive expansion of communities

Leung et al. [28] have shown that applying node preference (section 2), to alter propagation strength or spread from certain nodes, can greatly improve the performance of the basic label propagation. Nevertheless, our empirical analysis has revealed that different networks favor different strategies for node preference [46].

On small social networks, where high degree nodes reside in the core of each community (e.g. Zachary's karate club network [50]), good performance can be obtained by using *degree* or *eigenvector centrality* [12, 13] for node preference. However, on Girvan and Newman [14] benchmark networks with planted partition, where all nodes have equal degree (on average), the measures render useless and are outperformed by *clustering coefficient* [48]. Furthermore, on Lancichinetti et al. [22] benchmark networks superior performance is obtained by using inverted degree or inverted eigenvector centrality. Interestingly, the measures thus complement each node's degree, decreasing the propagation strength from high degree nodes (and vice-versa). In summary, the analysis has revealed that none of the considered measures is appropriate for general networks (all different kinds of networks).

We have observed that, during the course of the algorithm, applying node preference to the core of each current community (i.e. to its most central nodes) can significantly increase the performance on a wide range of real-world networks. Furthermore, the strategy results in a great ability of detecting communities, even when they are only weakly defined. On the other hand, applying node preference to the border of each current community (i.e. to its edge nodes) results in an extremely accurate detection, expanding communities that are strongly depicted in the network topology.

Based on above observations we propose two algorithms that convey two unique strategies of community formation. The algorithms estimate the core (and border) of each (current) community by means of the *diffusion* over the network; and are denoted *defensive* and *offensive diffusion label propagation algorithm* (*dDaLPA* and *oDaLPA* respectively). Let $p_n \in (0, 1)$ be a value for node $n \in N$ thus that nodes in the core of the community have higher values of p_n than border nodes. The defensive algorithm *dDaLPA* applies preference (i.e. propagation strength) to the core of each community, i.e. $f_n^{\alpha} = p_n$, and the

[4] The proportion of nodes that change their labels on the first five iterations roughly follows the sequence 90%, 30%, 10%, 5%, 3% [46] (on networks of moderate size).

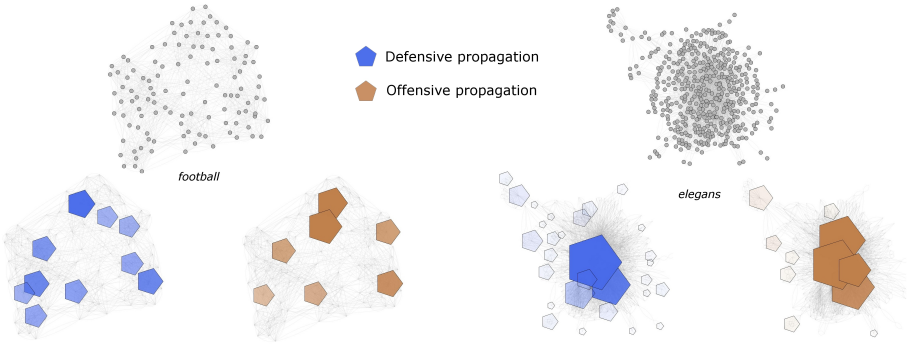

Fig. 1. Comparison of defensive and offensive label propagation on two real-world networks (see Table 1). Revealed communities are shown with pentagonal nodes, when the sizes (and colors) of nodes are proportional to the sizes of communities. Defensive propagation produces a larger set of communities that are (on average) considerably smaller than those revealed by the offensive propagation.

updating rule in Eq. (4) rewrites to

$$c_n = \underset{l}{\operatorname{argmax}} \sum_{m \in \mathcal{N}_n^l} p_m s_m w_{nm}. \tag{5}$$

On the other hand, the offensive version *oDaLPA* applies preference to the border of each community, i.e. $f_n^\alpha = 1 - p_n$, and the updating rule becomes

$$c_n = \underset{l}{\operatorname{argmax}} \sum_{m \in \mathcal{N}_n^l} (1 - p_m) s_m w_{nm}. \tag{6}$$

During the course of the algorithm, values p_n are estimated using *random walks* within each current community. Let p_n be the probability that a random walker, utilized on the community labeled with c_n, visits node n (due to simplicity, we assume that community features *connectedness*). p_n can then be computed as

$$p_n = \sum_{m \in \mathcal{N}_n^{c_n}} p_m / deg_m^{c_n}, \tag{7}$$

where $deg_m^{c_n}$ is the intra-community degree of node m ($c_m = c_n$). Besides deriving an estimate of the core and border of each community, the rationale here is to formulate label propagation (i.e. diffusion) within each of the current communities. Thus, opposed to the algorithm in [28], the main novelty is in considering (current) communities, found by the algorithm, to estimate the (current) state of the label propagation process and then to adequately alter the dynamics of the process.

Defensive and offensive label propagation algorithms result in two unique strategies of community formation, namely, *defensive preservation* and *offensive*

expansion of communities. The defensive algorithm quickly establishes a larger number of strong community cores (in the sense of Eq. (5)) and is able to defensibly preserve them during the course of the algorithm. This results in an immense ability of detecting communities, even when they are only weakly defined in the network topology. On the other hand, the offensive approach produces a much smaller set of communities (of various sizes). Laying the pressure on the edge of each community expands (i.e. enlarges) those that are strongly depicted in the network topology. This constitutes a more natural (offensive) struggle among communities and results in a great accuracy of the communities revealed.

Comparison of the approaches on two real-world networks is shown in Fig. 1; and for pseudo-code of the algorithms see Alg. 1.

Algorithm 1 Defensive label propagation algorithm (*dDaLPA*).

Input: Undirected graph $G(N, E)$ with weights W
Output: Communities C (i.e. node labels)

```
δ ← 0.5
for n ∈ N do
  c_n ← l_n {Unique label.}
  p_n ← 1/|N|
  d_n ← 0
end for
while not converged do
  shuffle(N)
  for n ∈ N do
    c_n ← argmax_l Σ_{m∈N_n^l} p_m(1 − δd_m)w_nm {1 − p_m instead of p_m for oDaLPA.}
    p_n ← Σ_{m∈N_n^{c_n}} p_m/deg_m^{c_n} {deg_m instead of deg_m^{c_n} for oDaLPA.}
    if c_n has changed then
      d_n ← (min_{m∈N_n^{c_n}} d_m) + 1
    end if
  end for
  δ ← proportion of labels changed {δ ← 0.1 at the first iteration.}
  if δ ≥ δ_max then
    δ ← 0 {Ensuring convergence (δ_max is fixed to 0.5).}
  end if
end while
return C {Returns (relabeled) communities that feature connectedness.}
```

3.3 Combining defensive and offensive propagation

Defensive and offensive label propagation (section 3.2) convey two unique strategies of community formation. An obvious improvement would be to combine the strategies, thus retaining the strong detection ability of the defensive approach and high accuracy of the offensive strategy. However, simply using the algorithms one after another does not attain the desired properties – any label propagation

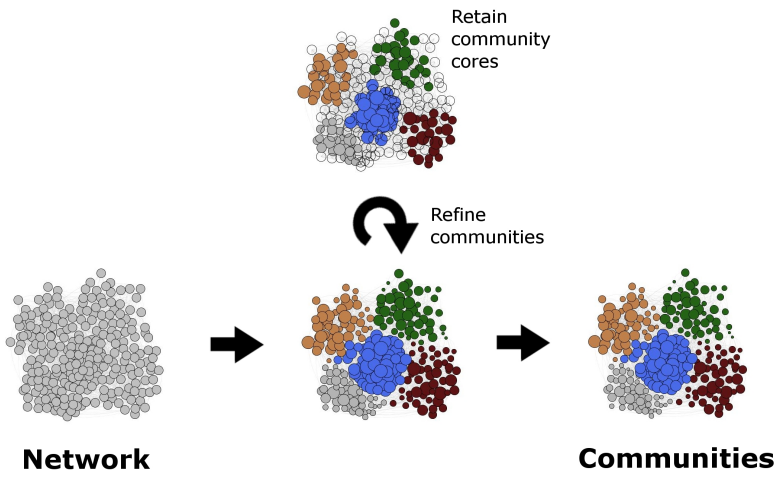

Fig. 2. Schematic representation of *K-Cores* algorithm.

algorithm, being run until convergence, finds a local optimum (i.e. local equilibrium) that is hard to escape from.

Raghavan et al. [40] have already discussed the idea (however, in different context) that label propagation could be improved, if one had *a priori* knowledge about community cores. Core nodes could then be labeled with the same label, leaving all the other nodes labeled with an unique label. During the course of the algorithm, the (uniquely labeled) nodes would tend to adopt the label of their nearest *attractor* (i.e. community core) and thus join its community. This would improve the algorithm's stability [40] and also the accuracy of the identified communities (section 4).

The defensive and offensive label propagation algorithms are thus combined in the following manner (Fig. 2). First, the defensive strategy is applied, to produce initial estimates of the communities and to accurately detect their cores. All border nodes of each community are then relabeled (labeled with unique labels), so that (approximately) one half of the nodes retain their original label. Next, the offensive strategy is applied, which refines the community cores and accurately detects also their borders. Relabeling and offensive refinement are then repeated until the number of communities decreases. Such combined strategy preserves advantages of both, defensive and offensive, label propagation (section 4) and is denoted *K-Cores*[5] algorithm (due to its resemblance to a well-known *K-Means* algorithm [31]).

The core (and border) of each community is again estimated by means of (diffusion) values p_n (section 3.2). Thus, within the algorithm, the node n is

[5] The term should not be confused with *k-core* [44] that denotes the maximal subgraph in which each node has degree at least k.

relabeled due to the following rule,

$$c_n = \begin{cases} c_n & \text{for } p_n > m_{c_n} \quad (8a) \\ l_n & \text{for } p_n \leq m_{c_n}, \quad (8b) \end{cases}$$

where m_{c_n} is the *median* of values p_n, for nodes in the community c_n, and l_n is an unique label. Hence, the core nodes retain their original labels, when all border nodes are relabeled.

Schematic representation of the algorithm is depicted in Fig. 2; and for the pseudo-code of the algorithm see Alg. 2. For a further discussion on all presented algorithms see [46].

Algorithm 2 *K-Cores* algorithm.

Input: Undirected graph $G(N, E)$ with weights W
Output: Communities C (i.e. node labels)

```
C ← dDaLPA(G, W) {Defensive label propagation.}
while |C| decreases do
  for c ∈ C do
    m_c ←median({p_n| n ∈ N ∧ c_n = c}) {Retain community cores.}
    for n ∈ N and c_n = c do
      if p_n ≤ m_c then
        c_n ← l_n {Unique label.}
        p_n ← 1/|N|
      end if
      d_n ← 0
    end for
  end for
  C ← oDaLPA(G, W) {Offensive label propagation.}
end while
return C {Returns best communities found.}
```

4 Empirical evaluation and discussion

In this section we present and discuss results of the empirical evaluation of the proposed algorithms. Section 4.1 gives results of the analysis on benchmark networks with planted partition, when the results on real-world network are reported in section 4.2. For the use with larger networks, we also briefly present and empirically compare two manners of hierarchical community detection in section 4.3.

The results are assessed using two measures of community structure, namely, *Normalized Mutual Information NMI* [8] and *modularity Q* [35]. The latter measures the relative significance of the communities due to a selected *null model*. Let A_{nm} denote the number of edges incident to nodes $n, m \in N$ and let P_{nm} be the expected number of incident edges in the null model. The modularity then

reads

$$Q = \frac{1}{2|E|} \sum_{n,m \in N} (A_{nm} - P_{nm})\, \delta(c_n, c_m), \tag{9}$$

where c_n is the identified community (label) for node $n \in N$ and δ is the Kronecker delta. The modularity thus measures the fraction of the difference between the number intra-community edges and the expected number of edges in the null model ($Q \in [-1, 1]$). Commonly a random graph with the same degree distribution as the original is selected for the null model. Hence, $P_{nm} = \frac{deg_n deg_m}{2|E|}$.

Furthermore, the analysis on networks with planted partition is conducted using *Normalized Mutual Information NMI* [8]. Let $\mathcal{C}$ be the partition (i.e. communities) extracted by some algorithm and let $\mathcal{P}$ be the planted partition of the network (corresponding random variables are C and P respectively). The *NMI* of $\mathcal{C}$ and $\mathcal{P}$ is then

$$NMI = \frac{2I(C, P)}{H(C) + H(P)}, \tag{10}$$

where $I(C, P)$ is the *mutual information* of the partitions, $I(C, P) = H(C) - H(C|P)$, and $H(C)$, $H(P)$ and $H(C|P)$ are standard and conditional entropies. *NMI* of identical partitions equals 1, and is 0 for independent partitions.

4.1 Networks with planted partition

The proposed algorithms were first analyzed on four different types of Lancichinetti et al. [22] benchmark networks with planted partition. The results are shown in Fig. 3.

The analysis clearly depicts the difference between defensive and offensive label propagation. The offensive approach (*oDaLPA*) performs considerably better than the basic label propagation (*LPA*) and can still accurately detect communities, when *LPA* already fails. On the other hand, the defensive propagation (*dDaLPA*) does not detect communities as accurately as the offensive approach, and *LPA* on larger networks, but still reveals communities, even when they are only weakly defined. Furthermore, *K-Cores* algorithm outperforms all other approaches in all but one case. Note that the algorithm retains the advantages of both defensive and offensive approach, still, the performance does not simply equal to the *upper-hull* of those for *dDaLPA* and *oDaLPA*.

The algorithms were also applied to a random graph à la Erdös-Rényi [10] that (presumably) has no community structure (Fig. 4). However, the defensive label propagation still reports communities, when the average degree is small enough. Nevertheless, further analysis reveals that defensive label propagation is still a preferred approach on a wide range of real-world networks (section 4.2).

As *K-Cores* algorithm is initialized using the defensive propagation, and best communities are reported at the end, the performance for *K-Cores* on a random graph is similar to that for *dDaLPA*. However, if we discard the initial communities obtained by *dDaLPA* (i.e. *K-Cores*[1] algorithm), the results correspond to those for *LPA* and *oDaLPA* that reveal no community structure.

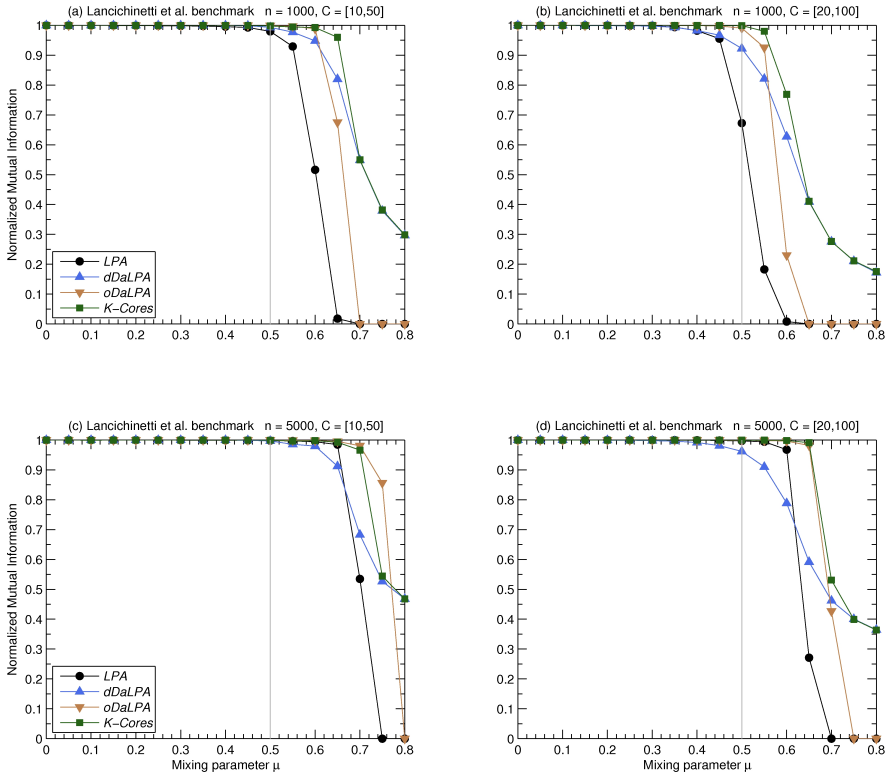

Fig. 3. Comparison of the proposed algorithms on Lancichinetti et al. [22] benchmarks networks with planted partition. The network sizes equal 1000 and 5000 nodes respectively; and communities comprise of up to 50 and 100 nodes respectively. The results were averaged over 100 realizations of the benchmarks networks.

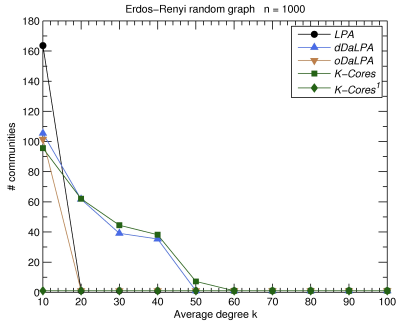

Fig. 4. Comparison of the proposed algorithms on a random graph à la Erdös-Rényi [10] with 1000 nodes (the results were averaged over 10 runs).

4.2 Real-world networks

The algorithms were further analyzed on over 20 real-world networks of moderate size (Table 2). Due to a large number of networks considered, the detailed description is omitted (see Table 1). However, the set includes different communication, social, biological, web, (author) collaboration, Internet and other networks. Due to simplicity, all networks are considered as unweighted and undirected, i.e. all weighted or directed edges are treated as simple undirected edges (same holds for networks in section 4.3).

We also introduce a new type of networks denoted *software networks* (sort of *component dependency networks* [49]). Here nodes represent a set of classes of some software system, written in an object-oriented programing language, and edges represent relations among them. Two classes A and B are defined as related when B extends or implements A, when B contains a field of type A or when A, B contains a method that returns, requires an object of type B, A respectively. The hypothesis here is that network communities would correspond to software packages, which could result in numerous applications in software engineering domain (Fig. 5). In this article we consider the ground case, where networks are represented with simple undirected and unweighted graphs[6].

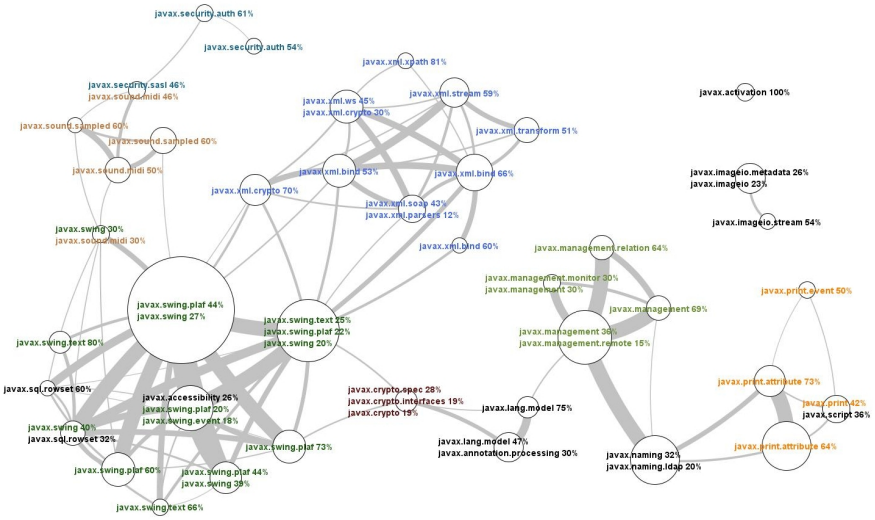

Fig. 5. Communities revealed in *javax* software network by applying *K-Cores* algorithm. The sizes of nodes correspond to the sizes of communities; and the widths of the edges correspond to the number of inter-community edges (due to clarity, weakly represented nodes and edges were discarded). Text shows the distribution of *javax* packages within the communities, where all weakly represented packages were omitted.

[6] The networks were obtained by parsing the documentation of the corresponding software. Thus, due to various reasons, some false relations might have been introduced.

Table 1. Networks used for the analysis of community detection algorithms.

Network	Description	Reference
uni	Emails within an university.	[17]
enron	Emails within *Enron*.	[25]
football	American college football league.	[14]
jazz	Network of jazz musicians.	[15]
wiki	Voting network of *Wikipedia*.	[24]
epinions	*Epinions* web of trust.	[41]
yeast	Yeast protein interactions.	[20]
elegans	Nematode *Caenorhabditis elegans*.	[21]
gnutella	*Gnutella peer-to-peer* network.	[26]
blogs	Weblogs on U.S. politics.	[2]
genrelat	*General Relativity* archive 2003.	[26]
$codmat^3$	*Condensed Matter* archive 2003.	[32]
$codmat^5$	*Condensed Matter* archive 2005.	[32]
hep	*High Energy Physics* archive 2003.	[26]
astro	*Astro Physics* archive 2003.	[26]
engine	*Google App Engine* library.	
jung	*JUNG* graph and network library.	
javax	*Java 6 javax* namespace.	
power	Western U.S. power grid.	[48]
$oregon^3$	Aut. syst. of Internet 2003 (*Oregon*).	[25]
$oregon^6$	Aut. syst. of Internet 2006 (*Oregon*).	[34]
nec	*nec* web overlay map.	[19]
amazon	*Amazon* co-purchasing network.	[23]
ndedu	Web graph of *nd.edu* domain.	[3]
road	Roads in Pennsylvania.	[27]
google	Web graph of *Google*.	[27]
skitter	Aut. syst. of Internet 2005 (*Skitter*).	[25]
movie	Movie actors collaborations.	[4]
nber	*NBER* patents citations.	[18]
live	*Live Journal* friendships.	[27]
webbase	Web graph from *WebBase*.	[1]

Table 2. Mean modularities Q for different label propagation algorithms (averaged over 100 to 100000 runs). The highest values of Q are shown with solid font; and underlined values correspond to the highest values among only *dDaLPA* and *oDaLPA*.

Type	Network	Nodes	Edges	*LPA*	*dDaLPA*	*oDaLPA*	*K-Cores*
Communication	*uni*	1133	5451	0.364	0.481	0.389	**0.518**
	enron	36692	367662	0.355	0.514	0.380	**0.516**
Social	*football*	115	616	0.592	0.593	0.595	**0.600**
	jazz	198	2742	0.346	**0.418**	0.377	**0.418**
	wiki	7115	103689	0.056	0.195	0.046	**0.202**
	epinions	75879	508837	0.106	0.288	0.111	**0.291**
Protein	*yeast*	2114	4480	0.665	0.733	0.720	**0.793**
Metabolic	*elegans*	453	2025	0.122	0.172	0.131	**0.173**
Peer-to-peer	*gnutella*	62586	147892	0.338	0.412	0.387	**0.447**
Web	*blogs*	1490	16718	0.400	0.424	0.424	**0.426**
Collaboration	*genrelat*	5242	28980	0.737	0.769	0.779	**0.820**
	codmat^3	27519	116181	0.596	0.611	0.627	**0.687**
	codmat^5	36458	171736	0.548	0.575	0.590	**0.648**
	hep	12008	237010	0.484	**0.585**	0.518	**0.585**
	astro	18772	396160	0.326	**0.538**	0.337	**0.538**
Software	*engine*	139	243	0.689	0.724	0.726	**0.747**
	jung	436	1303	0.611	0.587	0.623	**0.631**
	javax	2089	7934	0.723	0.687	0.725	**0.768**
Power	*power*	4941	6594	0.595	0.690	0.698	**0.820**
Internet	*oregon*^{3}	767	3591	0.302	0.210	**0.354**	0.210
	oregon^6	22963	48436	0.498	0.347	**0.541**	0.347
	nec	75885	357317	0.683	0.628	0.688	**0.736**

Comparison of the algorithms on real-world networks (Table 2) firstly confirms the adequacy of the *K-Cores* algorithm that obtains highest modularity on all but two Internet networks. Note that both *dDaLPA* and *oDaLPA* also outperform the basic *LPA* in most cases. Moreover, the analysis clearly separates different types of networks due to the preferred strategy of community formation. For instance, social and communication networks clearly favor defensive preservation of communities, due to high density of such networks (and rather weakly defined communities). On the other hand, sparse software or Internet networks, with longer paths among nodes, obviously prefer the offensive expansion of communities. The middle case is represented by the considered collaboration networks. On smaller networks that are relatively sparse (*genrelat*, *codmat*^{3} and *codmat*^{5} network) the offensive approach prevails. However, on larger networks (*hep* and *astro* network), with significantly higher degrees then the former, the defensive algorithm is superior. In summary, denser networks (with higher av-

erage degrees) prefer the defensive preservation, whereas sparser networks (with lower average degrees) favor the offensive expansion of communities.

4.3 Analyzing large networks

Last, we also briefly present and empirically evaluate two different manners of *hierarchical* community investigation (prominent for the use with larger networks). Besides *LPA* and *K-Cores* algorithm, we consider the following approaches.

Basic diffusion and propagation algorithm (*DPA*) [46] is an optimized version of *K-Cores* that scales significantly better then the basic algorithm. Furthermore, *hierarchical diffusion and propagation algorithm* (*DPA*$^+$) represents a simple hierarchical detection, where the algorithm is recursively applied to the previously constructed *community network*[7] (the algorithm employs defensive label propagation, and *DPA* on the last step). Moreover, (general) *diffusion and propagation algorithm* (*DPA**) [46] represents a hierarchical *core extraction* technique, where the algorithm recursively extracts the *core* [27] of the network and identifies *whisker* communities. For a further discussion on the algorithms see [46].

Table 3. Peak modularities Q and average number of iterations for different label propagation algorithms (obtained over 1 to 10 runs). Solid values correspond to the largest values of Q, where missing values could not be obtained due to limited time resources.

Network	Nodes	Edges	*LPA*	*K-Cores*	*DPA*	*DPA*$^+$	*DPA**
amazon	0.3M	1.2M	0.681/15	0.783/273	0.700/34	**0.883/65**	0.856/78
ndedu	0.3M	1.5M	0.838/53	0.891/471	0.860/50	0.897/37	**0.901/58**
road	1.1M	3.1M	0.552/10	0.847/895	0.626/82	**0.985/136**	0.883/142
google	0.9M	4.3M	0.801/15	0.889/444	0.820/59	0.962/45	**0.967/48**
skitter	1.7M	11.1M	0.746/25	-	0.755/126	0.680/52	**0.801/76**
movie	0.4M	15.0M	0.524/21	-	0.533/147	0.474/39	**0.606/71**
nber	3.8M	16.5M	0.576/109	-	0.582/336	0.707/112	**0.739/308**
live	4.8M	69.0M	0.673/100	-	0.548/206	0.683/73	**0.688/125**
webbase	14.5M	101.0M	0.894/38	-	0.923/114	0.942/43	**0.954/39**

Algorithms were applied to a set of large real-world networks (Table 3) of various types (see Table 1), when the analysis on (even) larger networks was limited due to limited memory resources (i.e. 4 GB of memory). On average, all of the considered algorithms again perform better then the basic label propagation (*LPA*). Furthermore, the hierarchical algorithms (*DPA*$^+$ and *DPA**) obtain the highest values of modularity on all of the networks considered, whereas the core extraction technique (*DPA**) seems more prominent.

[7] A network whose nodes represent communities and edges represent edges between nodes in the original network.

The average number of iterations made by the algorithms[8] (Table 3) shows that "theoretical" approach *K-Cores* does not scale to larger networks, where the optimized version *DPA* is preferred. Note also that hierarchical detection even decreases the total number of iterations on larger networks, which gives promising grounds for future analysis of large complex networks.

For a further empirical evaluation and comparison with other label propagation algorithms reported in the literature see [46].

5 Conclusion

In the article we present different label propagation algorithms that employ two unique strategies of community formation, namely, defensive preservation and offensive expansion of communities. The strategies are combined in an advanced label propagation algorithm that retains the advantages of both approaches. Furthermore, we also show how the algorithm can be extended to larger networks using (hierarchical) core extraction. Nevertheless, the main contribution of this work is in showing that different types of networks (with different topological properties) favor different strategies of community formation.

Future work will focus mainly on further analyses of defensive and offensive label propagation, in order to develop an enhanced algorithm that would *decide* between defensive and offensive strategy (an intermediate approaches) during the course of the algorithm. This could result in higher accuracy of the revealed communities and also in better scalability of the algorithm.

References

1. Web graph from the Stanford WebBase Project (crawl from January 2010). http://diglib.stanford.edu:8091/~testbed/doc2/WebBase/ (2010)
2. Adamic, L.A., Glance, N.: The political blogosphere and the 2004 U.S. election. In: Proceedings of the International Workshop on Link Discovery. pp. 36–43 (2005)
3. Albert, R., Jeong, H., Barabási, A.: The diameter of the world wide web. Nature 401, 130–131 (1999)
4. Barabási, A., Albert, R.: Emergence of scaling in random networks. Science 286(5439), 509–512 (1999)
5. Barber, M.J., Clark, J.W.: Detecting network communities by propagating labels under constraints. Phys. Rev. E 80(2), 026129 (2009)
6. Blondel, V.D., Guillaume, J., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. P10008 (2008)
7. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E 70(6), 066111 (2004)
8. Danon, L., Díaz-Guilera, A., Duch, J., Arenas, A.: Comparing community structure identification. J. Stat. Mech. P09008 (2005)
9. Donetti, L., Munoz, M.A.: Detecting network communities. J. Stat. Mech. P10012 (2004)

[8] As the employed implementation was optimized for memory, not time, resources, we report the number of iterations instead of the exact running times.

10. Erdős, P., Rényi, A.: On random graphs 1. Publ. Math. Debrecen 6, 290–297 (1959)
11. Fortunato, S.: Community detection in graphs. Phys. Rep. 486(3-5), 75–174 (2010)
12. Freeman, L.: A set of measures of centrality based on betweenness. Sociometry 40(1), 35–41 (1977)
13. Freeman, L.C.: Centrality in social networks: Conceptual clarification. Soc. Networks 1(3), 215–239 (1979)
14. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. In: Proceedings of the National Academy of Sciences of United States of America. pp. 7821–7826 (2002)
15. Gleiser, P., Danon, L.: Community structure in jazz. Adv. Complex Syst. 6(4), 565 (2003)
16. Gregory, S.: Finding overlapping communities in networks by label propagation (2009)
17. Guimerà, R., Danon, L., Díaz-Guilera, A., Giralt, F., Arenas, A.: Self-similar community structure in a network of human interactions. Phys. Rev. E 68(6), 065103 (2003)
18. Hall, B.H., Jaffe, A.B., Tratjenberg, M.: The NBER patent citation data file: Lessons, insights and methodological tools. Tech. rep., National Bureau of Economic Research (2001)
19. Hoerdt, M., Jaeger, M., James, A., Magoni, D., Maillard, J., Malka, D., Merindol, P.: Internet IPv4 overlay map produced by network cartographer (nec). http://www.labri.fr/perso/magoni/nec/ (2003)
20. Jeong, H., Mason, S.P., Barabási, A., Oltvai, Z.N.: Lethality and centrality of protein networks. Nature 411, 41–42 (2001)
21. Jeong, H., Tombor, B., Albert, R., Oltvai, Z.N., Barabási, A.: The large-scale organization of metabolic networks. Nature 407, 651–654 (2000)
22. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Phys. Rev. E 78(4), 046110 (2008)
23. Leskovec, J., Adamic, L.A., Huberman, B.A.: The dynamics of viral marketing. ACM Trans. Web 1(1) (2007)
24. Leskovec, J., Huttenlocher, D., Kleinberg, J.: Predicting positive and negative links in online social networks. In: Proceedings of the ACM International Conference on World Wide Web (2010)
25. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graphs over time: Densification laws, shrinking diameters and possible explanations. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2005)
26. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: Densification and shrinking diameters. ACM Transactions on Knowledge Discovery from Data 1(1) (2007)
27. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. e-print arXiv:0810.1355 (1) (2008)
28. Leung, I.X.Y., Hui, P., Liò, P., Crowcroft, J.: Towards real-time community detection in large networks. Phys. Rev. E 79(6), 066107 (2009)
29. Liu, X., Murata, T.: Advanced modularity-specialized label propagation algorithm for detecting communities in networks. Physica A 389(7), 1493 (2009)
30. Liu, X., Murata, T.: Community detection in large-scale bipartite networks. In: Proceedings of the International Conference on Web Intelligence and Intelligent Agent Technology. vol. 1, pp. 50–57 (2009)

31. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of Berkeley Symposium on Mathematical Statistics and Probability. pp. 281–297 (1967)
32. Newman, M.E.J.: The structure of scientific collaboration networks. In: Proceedings of the National Academy of Sciences of the United States of America. vol. 98, pp. 404–409 (2001)
33. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E 74(3), 036104 (2006)
34. Newman, M.E.J.: Symmetrized snapshot of the structure of the internet at the level of autonomous systems. http://www-personal.umich.edu/~mejn/netdata/ (2006)
35. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E 69(2), 026113 (2004)
36. Newman, M.E.J., Leicht, E.A.: Mixture models and exploratory analysis in networks. In: Proceedings of the National Academy of Sciences of the United States of America. vol. 104, pp. 9564–9569 (2007)
37. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435, 814 (2005)
38. Pang, C., Shao, F., Sun, R., Li, S.: Detecting community structure in networks by propagating labels of nodes. In: Proceedings of the International Symposium on Neural Networks. pp. 839–846 (2009)
39. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. In: Proceedings of the National Academy of Sciences of the United States of America. vol. 101, pp. 2658–2663 (2004)
40. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E 76(3), 036106 (2007)
41. Richardson, M., Agrawal, R., Domingos, P.: Trust management for the semantic web. In: Proceedings of the International Semantic Web Conference. vol. 2, pp. 351–368 (2003)
42. Ronhovde, P., Nussinov, Z.: Local resolution-limit-free potts model for community detection. Phys. Rev. E 81(4), 046114 (2010)
43. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. In: Proceedings of the National Academy of Sciences of United States of America. vol. 105, pp. 1118–1123 (2008)
44. Seidman, S.B.: Network structure and minimum degree. Soc. Networks 5(3), 269–287 (1983)
45. Strogatz, S.H.: Exploring complex networks. Nature 410, 268 (2001)
46. Subelj, L., Bajec, M.: Unfolding communities in large complex networks: Combining defensive and offensive label propagation for core extraction. Submitted to Phys. Rev. E
47. Tibély, G., Kertész, J.: On the equivalence of the label propagation method of community detection and a potts model approach. Physica A 387(19-20), 4982–4984 (2008)
48. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature 393(6684), 440–442 (1998)
49. Wen, L., Kirk, D., Dromey, R.G.: Software systems as complex networks. In: Proceedings of the IEEE International Conference on Cognitive Informatics. pp. 106–115 (2007)
50. Zachary, W.W.: An information flow model for conflict and fission in small groups. J. Anthropol. Res. 33(4), 452–473 (1977)