Unfolding communities in large complex networks: Combining defensive and offensive label propagation for core extraction

Lovro Šubelj[*] and Marko Bajec[†]
University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia

(Received 1 June 2010; revised manuscript received 6 November 2010; published 8 March 2011)

Label propagation has proven to be a fast method for detecting communities in large complex networks. Recent developments have also improved the accuracy of the approach; however, a general algorithm is still an open issue. We present an advanced label propagation algorithm that combines two unique strategies of community formation, namely, defensive preservation and offensive expansion of communities. The two strategies are combined in a hierarchical manner to recursively extract the core of the network and to identify whisker communities. The algorithm was evaluated on two classes of benchmark networks with planted partition and on 23 real-world networks ranging from networks with tens of nodes to networks with several tens of millions of edges. It is shown to be comparable to the current state-of-the-art community detection algorithms and superior to all previous label propagation algorithms, with comparable time complexity. In particular, analysis on real-world networks has proven that the algorithm has almost linear complexity, $O(m^{1.19})$, and scales even better than the basic label propagation algorithm (m is the number of edges in the network).

DOI: 10.1103/PhysRevE.83.036103 PACS number(s): 89.75.Fb, 89.75.Hc, 87.23.Ge, 89.20.Hh

I. INTRODUCTION

Large real-world networks can comprise local structural modules (*communities*) that are groups of nodes, densely connected within and only loosely connected with the rest of the network. Communities are believed to play important roles in different real-world systems (e.g., they may correspond to functional modules in metabolic networks [1]); moreover, they also provide a valuable insight into the structure and function of large complex networks [1–3]. Nevertheless, real-world networks can reveal even more complex modules than communities [4,5].

Over the last decade the research community has shown a considerable interest in detecting communities in real-world networks. Since the seminal paper of Girvan and Newman [6] a vast number of approaches have been presented in the literature—in particular, approaches optimizing modularity Q (the significance of communities due to a selected null model [7]) [8–12], graph partitioning [13,14] and spectral algorithms [9,15], statistical methods [4], algorithms based on dynamic processes [16–20], overlapping, hierarchical, and multiresolution methods [1,6,20], and others [21] (for an excellent survey see [22]).

The size of large real-world networks has forced the research community to develop scalable approaches that can be applied to networks with several millions of nodes and billions of edges. A promising effort was made by Raghavan *et al.* [18], who employed a simple label propagation to find significant communities in large real-world networks. Tibély and Kertész [23] have shown that label propagation is in fact equivalent to a large zero-temperature kinetic Potts model, while Barber and Clark [11] have further refined the approach into a modularity optimization algorithm. Just recently, Liu and Murata [12] have combined the modularity optimization version of the algorithm with a multistep greedy agglomeration [24] and derived an extremely accurate community detection algorithm.

Leung *et al.* [19] have investigated label propagation on large web networks, mainly focusing on scalability issues, and have shown that the performance can be significantly improved with label hop attenuation and by applying node preference (i.e., node propagation strength). We carry forward their work in developing two unique strategies of community formation, namely, defensive preservation of communities, where preference is given to the nodes in the core of each community, and offensive expansion of communities, where preference is given to the border nodes of each community. Cores and borders are estimated using random walks, formulating the diffusion over the network.

Furthermore, we propose an advanced label propagation algorithm—the diffusion and propagation algorithm—that combines the two strategies in a hierarchical manner: The algorithm first extracts the *core* of the network and identifies *whisker* communities [26] (Appendix A), and then recurses on the network core (Fig. 1). The performance of the algorithm has been analyzed on two classes of benchmark networks with planted partition and on 23 real-world networks ranging from networks with tens of nodes to networks with several tens of millions of edges. The algorithm is shown to be comparable to the current state-of-the-art community detection algorithms and superior to all previous label propagation algorithms, with comparable time complexity. In particular, the algorithm exhibits almost linear time complexity (in the number of edges of the network).

The rest of the paper is structured as follows. Section II gives a formal introduction to label propagation and reviews subsequent advances, relevant for this research. Section III presents the diffusion and propagation algorithm and discusses the main rationale behind it. Empirical evaluation with discussion is done in Secs. IV, and V is our conclusion.

[*]lovro.subelj@fri.uni-lj.si
[†]marko.bajec@fri.uni-lj.si

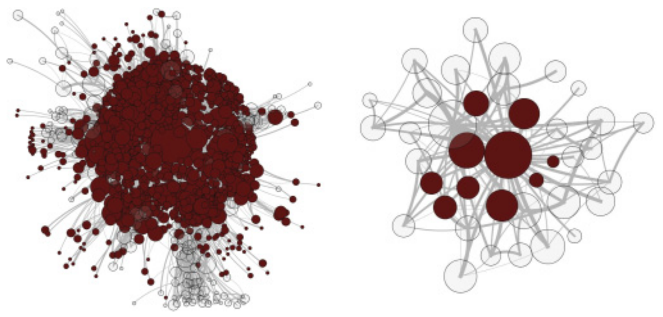

FIG. 1. (Color online) Results of diffusion and propagation algorithm applied to the network of autonomous systems of the Internet [25]. The figure shows two community networks, where the largest nodes correspond to densely connected modules of almost 10^4 nodes in the original network. Network cores, extracted by the algorithm, are colored red (dark gray) and whisker communities are represented with transparent nodes. The results show that the algorithm can detect communities on various levels of resolution—average community sizes are 16.38 and 588.79 nodes (with Q equal to 0.475 and 0.582, respectively).

II. LABEL PROPAGATION AND ADVANCES

Let the network be represented by an undirected graph $G(N,E)$, with N being the set of nodes of the graph and E being the set of edges. Furthermore, let c_n be a community (label) of node n, $n \in N$, and $\mathcal{N}(n)$ the set of its neighbors.

The basic label propagation algorithm (LPA) [18] exploits the following simple procedure. At first, each node is labeled with a unique label, $c_n = l_n$. Then, at each iteration, the node is assigned the label shared by most of its neighbors (i.e., the maximal label),

$$c_n = \operatorname*{argmax}_l |\mathcal{N}^l(n)|, \tag{1}$$

where $\mathcal{N}^l(n)$ is the set of neighbors of n that share label l (in the case of ties, one maximal label is chosen at random). Due to the existence of multiple edges within the communities, relative to the number of edges between the communities, nodes in a community will adopt the same label after a few iterations. The algorithm converges when none of the labels changes anymore (i.e., equilibrium is reached) and nodes sharing the same label are classified into the same community.

The main advantage of label propagation is its nearly linear time complexity—the algorithm commonly converges in less then 10 iterations (on networks of moderate size). Raghavan *et al.* [18] observed that after 5 iterations 95% of the nodes already obtained their “right” label. Their observation can be further generalized: The number of nodes that change their label on the first 4 iterations roughly follow the sequence 90%, 30%, 10%, and 5%. However, because of the algorithm’s simplicity, the accuracy of identified communities is often not state of the art (Sec. IV).

Leung *et al.* [19] have noticed that the algorithm, applied to large web networks, often produces a single large community, occupying more than a half of the nodes of the network. Thus, they have proposed a label hop attenuation technique, to prevent the label from spreading too far from its origin. Each label l_n has associated an additional score s_n (initially set to 1) that decreases after each propagation [Eq. (1)]. Hence,

$$s_n = \left(\max_{i \in \mathcal{N}^{c_n}(n)} s_i\right) - \delta, \tag{2}$$

with δ being the attenuation ratio. When s_n reaches 0, the label can no longer propagate onward [Eq. (3)], which successfully eliminates the formation of a single major community [19].

Leung *et al.* [19] have also shown that hop attenuation has to be coupled with node preference f_n (i.e., node propagation strength) in order to achieve superior performance. The label propagation updating rule [Eq. (1)] is thus reformulated into

$$c_n = \operatorname*{argmax}_l \sum_{i \in \mathcal{N}^l(n)} f_i^\alpha s_i w_{ni}, \tag{3}$$

where w_{ni} is the edge weight (equal to 1 for unweighted graphs) and α is a parameter of the algorithm. They have experimented with preference equal to the degree of the node, $f_i = k_i$ and $\alpha = 0.1$; however, no general approach was reported.

Label hop attenuation in Eq. (2) can be rewritten into an equivalent form that allows altering δ during the course of the algorithm [19]. One keeps the label distance from the origin d_n (initially set to 0) that is updated after each propagation. Hence,

$$d_n = (\min_{i \in \mathcal{N}^{c_n}(n)} d_i) + 1, \tag{4}$$

when the score s_n is

$$s_n = 1 - \delta d_n. \tag{5}$$

Raghavan *et al.* [18] have already shown that the updating rule of label propagation [Eq. (1)], or its refinements [Eq. (3)], might prevent the algorithm from converging. 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. Because of the bipartite structure of the network, at the next iteration, all red and blue nodes will adopt the label l_b and l_r, respectively. Furthermore, at the next iteration all nodes will recover their original labels, preventing the algorithm from converging.

The problem can be avoided with asynchronous updating [18]. Nodes are no longer updated all together, but sequentially, in random order. Thus, when a node’s label is updated, (possibly) already updated labels of its neighbors are considered (in contrast to synchronous updating, which considers only labels from the previous iteration). It should be noted that asynchronous updating can even increase the performance of the algorithm [19].

Furthermore, when a node has equally strong connections with two or more communities, its label will, in general, constantly change [18,19]. The problem is particularly apparent in author collaboration (co-authorship) networks, where a single author often collaborates with different research communities. On the collaboration network of network scientists [9] the basic label propagation algorithm fails to converge, as up to 10% of the nodes would change their label even after 10 000 iterations; the results suggest that the network contains at least 20% of such nodes, i.e., over 300 scientists collaborating with different research communities [28].

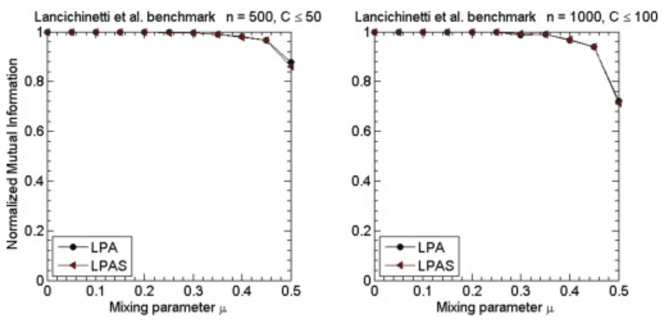

FIG. 2. Comparison of node access strategies for label propagation on two sets of benchmark networks with planted partition [27] (the results are averages over 100 realizations). Network sizes equal 500 and 1000 nodes, and communities comprise up to 50 and 100 nodes, respectively. LPA denotes the basic label propagation algorithm and LPAS denotes LPA without (subsequent) reshuffling of nodes.

Leung *et al.* [19] suggested including the concerned label itself in the maximal label consideration (and not merely neighbors' labels); however, we use a slightly modified version [18]. When there are multiple maximal labels among neighbors and one of them equals the concerned label, the node retains its label. The main difference here is that the modified version considers the concerned label only when there exist multiple maximal labels among neighbors. On the discussed collaboration network, such an algorithm converges in around 4 iterations.

Never-converging nodes can also be regarded as a clear signature of overlapping communities [1], where nodes can belong to multiple communities. Extension of label propagation to detect overlapping communities was recently proposed by Gregory [29] (and previously discussed in [18,19]). However, due to simplicity, we investigate only basic (no-overlap) versions of the label propagation algorithm.

Another important issue of label propagation is the stability of identified community structure [18], especially in large networks. For more detailed discussion see [12,18,23].

Label propagation with asynchronous updating accesses the nodes in a random order. Nodes are then shuffled after each iteration, mainly to address the problems discussed above. Although this subsequent reshuffling does not increase the algorithm's complexity, it does indeed increase its computational time. Nevertheless, the results in Fig. 2 show that LPA without subsequent reshuffling of nodes (LPAS) only slightly decreases the performance of the basic LPA. Thus, all the approaches presented in the following section use asynchronous updating with a single (initial) shuffling of nodes.

III. DIFFUSION AND PROPAGATION ALGORITHM

The section presents the diffusion and propagation algorithm that combines several approaches, also introduced in this section. We thus give here a brief review of these.

First, we further analyze label hop attenuation for LPA (Sec. II) and propose different dynamic hop attenuation strategies in Sec. III A. Next, we consider various approaches for node propagation preference (Sec. II). By estimating node preference by means of the diffusion over the network, we derive two algorithms that result in two unique strategies of community formation, namely, *defensive preservation* and *offensive expansion* of communities. The algorithms are denoted *defensive* and *offensive* diffusion and attenuation LPA (DDALPA and ODALPA) and are presented in Sec. III B.

The DALPA algorithms are combined into the basic diffusion and propagation algorithm (BDPA), preserving the advantages of both defensive and offensive approaches (Sec. III C). BDPA already achieves superior results on networks of moderate size (Sec. IV); for use with larger networks, the algorithm is further enhanced with core extraction and whiskers identification. The improved algorithm is denoted the (general) diffusion and propagation algorithm (DPA) and is presented in Sec. III C.

A. Dynamic hop attenuation

Hop attenuation has proven to be a reliable technique for preventing the emergence of a major community occupying most of the nodes of the network [19]. It is, however, not evident what the value of attenuation ratio δ should be [Eq. (2)]. Leung *et al.* [19] have experimented with values around 0.10 and obtained good results; still, their experimental setting was rather limited. Furthermore, our preliminary empirical analysis suggests that there is no (simple) universal value for δ applicable for *all* different types of networks (the results are omitted here).

Leung *et al.* [19] have also observed that large values of δ may prevent the natural growth of communities and have proposed a dynamic strategy that decreases δ from 0.50 toward 0. In the early iterations of the algorithm, large values of δ prevent a single label from rapidly occupying a large set of nodes and ensure the emergence of a number of strong community cores. The value of δ is then decreased, to gradually relax the restriction and to allow formation of the actual communities depicted in the network topology. The results on real-world networks show that such a strategy has very good performance on larger networks (Sec. IV); still, the results can be further improved. The empirical evaluation in Sec. IV also proves that the strategy is too aggressive for smaller networks, where it is commonly outperformed even by basic LPA.

We propose different dynamic hop attenuation strategies, based on the hypothesis that hop attenuation should only be employed when a community, or a set of communities, is rapidly occupying a large portion of the network. Otherwise, the restriction should be (almost) completely relaxed, to allow label propagation to reach the equilibrium unrestrained. Thus, the approach would retain the dynamics of label propagation and still prevent the emergence of a major community.

We have considered several strategies for detecting the emergence of a large community or a set of large communities. Because of limited space, we limit the discussion to two. After each iteration, the value of δ (initially set to 0) is updated according to the following rule:

nodes: δ is set to the proportion of nodes that changed their label;

communities: δ is set to the proportion of communities (i.e., labels) that disappeared.

Both strategies successfully address the problem of major community formation; however, a detailed comparison is omitted here. The algorithms proposed here all use the *nodes* strategy, because of its much finer granularity, as opposed to the *communities* approach—after 4 iterations the number of communities is, in general, already 20 times smaller than the number of nodes (Sec. II); thus, the estimate of δ is rather rough for the *communities* strategy. For the empirical evaluation see Sec. IV.

B. Defensive and offensive propagation

Leung *et al.* [19] have proved that using node preference, to increase the propagation strength (i.e., label spread) from certain nodes, can improve the performance of basic LPA. We conducted several experiments by using variations of different measures of node centrality for node propagation preference (i.e., degree and eigenvector centrality [30,31] and node clustering coefficient [32]). The results are omitted here; however, they clearly indicate that none of these static measures applies for *all* different types of networks (i.e., general networks).

We have also observed that good performance can be obtained by putting higher preference to the core of each community (i.e., to its most central nodes). For instance, on the Zachary's karate club network [33], where three high-degree nodes reside in the core of the two (natural) communities, degree and eigenvector centralities are superior. However, on Girvan and Newman [6] benchmark networks, where all the nodes have equal degree (on average), the measures are rendered useless and are outperformed by the node clustering coefficient. On the Lancichinetti *et al.* [27] benchmark networks, the best performance is, interestingly, obtained by inverted degree or eigenvector centrality. The measures seem to counter each node's degree (low-degree nodes have high propagation strength, and vice versa); thus, the propagation utilizes merely the connectedness among nodes, disregarding its strength.

Based (also) on the above observations, we have developed two algorithms that estimate node preference by means of the diffusion over the network. During the course of the algorithms, the diffusion is formulated using a random walker within each of the (current) communities of the network. The rationale here is twofold: (1) to estimate the (label) propagation within each of the (current) communities [34], and (2) to derive an estimation of the core and border of each (current) community (with the core being the most central nodes of the community and the border being its edge nodes).

Let p_n be the probability that a random walker, utilized on the community labeled with c_n, visits node n. p_n can be computed as

$$p_n = \sum_{i \in \mathcal{N}^{c_n}(n)} p_i / k_i^{c_n}, \tag{6}$$

where the sum goes over all the neighbors of n within the community c_n, and $k_i^{c_n}$ is the intracommunity degree of node i. The employed formulation is similar to algorithms such as PageRank [35] and HITS [36], and also to the basic eigenvector centrality measure.

Finally, we present the two algorithms mentioned above, namely, defensive and offensive diffusion and attenuation LPA (DDALPA and ODALPA). The defensive algorithm applies preference (i.e., propagation strength) to the core of each community, i.e., $f_n^\alpha = p_n$, and the updating rule in (3) is rewritten as

$$c_n = \underset{l}{\operatorname{argmax}} \sum_{i \in \mathcal{N}^l(n)} p_i s_i w_{ni}. \tag{7}$$

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

$$c_n = \underset{l}{\operatorname{argmax}} \sum_{i \in \mathcal{N}^l(n)} (1 - p_i) s_i w_{ni}. \tag{8}$$

As opposed to the algorithm of Leung *et al.* [19], the main novelty here 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.

To better estimate the border of each community, the offensive algorithm uses degrees k_i (instead of intracommunity degrees $k_i^{c_n}$) for the estimation of diffusion values p_n [see Eq. (6)]. The modification results in higher values of $1 - p_n$ for nodes with large intercommunity degrees (i.e., nodes that reside in the borders of communities) and thus provides more adequate formulation of the node propagation strength for the offensive version (the results are omitted here).

When a node's label changes, the values p_n should be reestimated for each node in the concerned node's previous or current community. However, this would likely render the algorithm inapplicable on larger networks. Thus, we only update the value p_n [according to Eq. (6)] when the node n changes its label (initially all p_n are set to $1/|N|$). Although the approach is only a rough approximation of an exact version, preliminary empirical experiments reveal no significant gain by using the exact values for p_n.

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. (7)] 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 range of communities of various sizes, as commonly observed in the real-world networks [3,18]. Laying the pressure on the border of each community expands those that are strongly defined in the network topology. This constitutes a more natural (offensive) struggle among the communities and results in a great accuracy of the communities revealed.

Comparison of the algorithms on two real-world networks is depicted in Fig. 3. The examples show that defensive propagation prefers networks with rather homogeneous distribution of the sizes of the communities, and that offensive propagation favors networks with more heterogeneous (e.g., power law) distribution. It should, however, be noted that both approaches can achieve superior performance on both of the networks. Still, on average, the defensive approach performs better on the social network *football* [6], while offensive outperforms defensive on the metabolic network *elegans* [37].

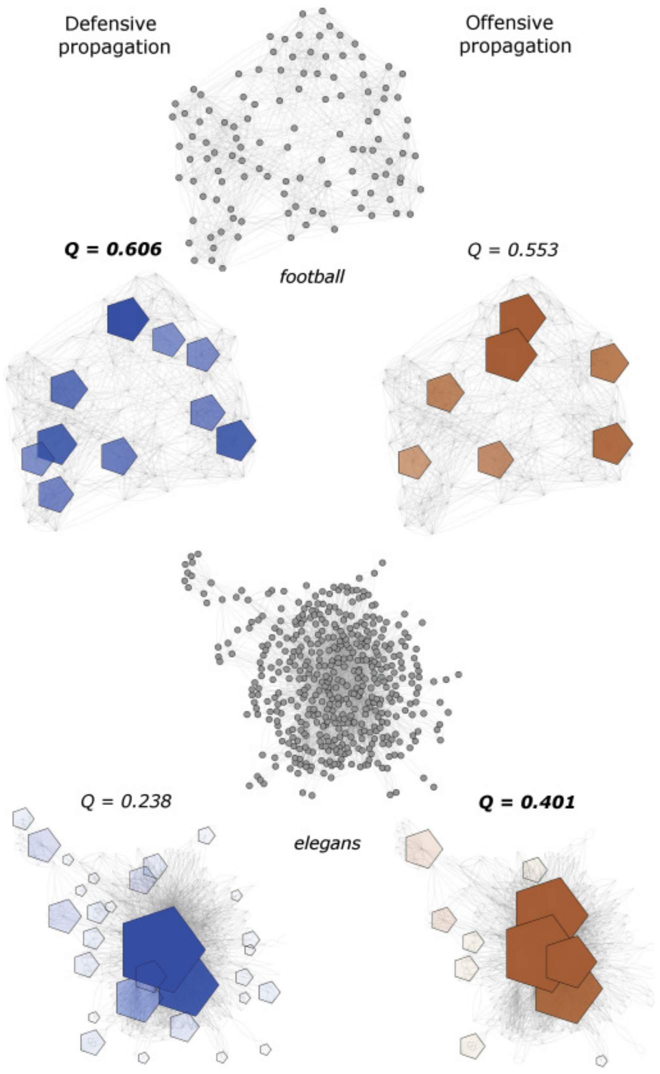

FIG. 3. (Color online) Comparison of defensive and offensive label propagation on two real-world networks, i.e., a social network of American football matches at a US college [6] and a metabolic network of the nematode *Caenorhabditis elegans* [37]. The revealed communities are shown with pentagonal nodes, and the sizes and intensities of colors (shadings) of the nodes are proportional to the sizes of communities. The networks comprise two relatively different community structures, considering the distribution of sizes of the communities. This is rather homogeneous in the case of *football* and (presumably) a power law in the case of *elegans*.

For an empirical analysis and further discussion of the algorithms, see Sec. IV; and for the pseudocode of the algorithms and discussion on some of the implementation issues, see Appendix B.

C. Diffusion and propagation algorithm

Defensive and offensive label propagation (Sec. III B) 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. The reason is that any label propagation algorithm, being run until convergence, finds local optimum (i.e., local equilibrium) that is hard to escape from.

Raghavan *et al.* [18] have already discussed the idea (however, in a 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 a unique label. During the course of the algorithm, the (uniquely labeled) nodes would tend to adopt the label of their nearest attractor (i.e., the community core) and thus join its community. This would improve the algorithm's stability [18] and also the accuracy of the identified communities (Sec. IV).

The defensive and offensive label propagation algorithms are combined in the following manner. 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 a unique label), so that approximately one-half of the nodes retain their original label. Last, the offensive strategy is applied, which refines the community cores and accurately detects also their borders. Such combined strategy preserves advantages of both defensive and offensive label propagation algorithms and is denoted the basic diffusion and propagation algorithm (BDPA). Schematic representation of the algorithm is depicted in Fig. 4 (steps 3 and 4).

The core (and border) of each community is estimated by means of diffusion p_n (Sec. III B). As core nodes possess more intracommunity edges then border nodes, this results in higher values of p_n for core nodes. Thus, within the algorithm, the node n is relabeled due to the following rule:

$$c_n = \begin{cases} c_n & \text{for } p_n > m_{c_n}, \\ l_n & \text{for } p_n \leqslant m_{c_n} \end{cases}, \tag{9}$$

where m_{c_n} is the median of values p_n for nodes in community c_n, and l_n is a unique label. Thus, the core nodes retain their original labels, while all border nodes are relabeled. Note that all nodes with p_n equal to the median are also relabeled, to adequately treat smaller communities, where most of the nodes share the same value of p_n.

Empirical evaluation shows that BDPA significantly outperforms basic LPA and also the algorithm of Leung *et al.* [19] on smaller networks. However, when networks become larger, the hop attenuation strategy of Leung *et al.* [19] produces much larger communities, with higher values of modularity (on average).

Different authors have proposed approaches that detect communities in a hierarchical manner (e.g., [10]). The algorithm is first applied to the original network and initial communities are obtained. One then constructs the community network, where nodes represent communities and edges are added between them, while their nodes are connected in the original network. The algorithm is then recursively applied to the community network and the process repeats. At the end, the best communities found by the algorithm are reported (according to some measure).

The idea was also proposed in the context of label propagation [19]; however, the authors did not report any empirical results. We have analyzed the behavior of

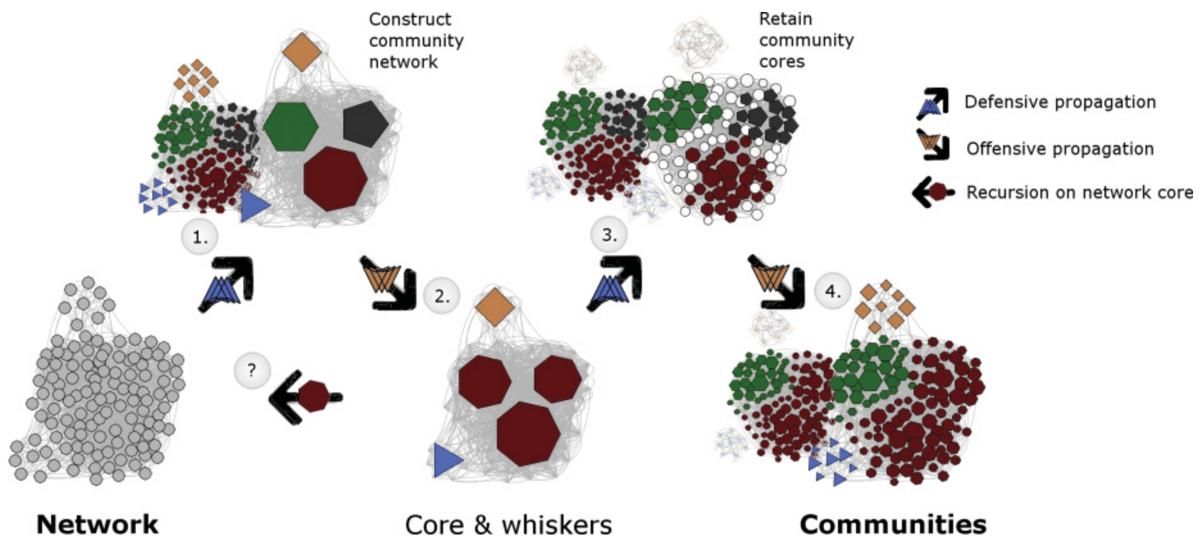

FIG. 4. (Color online) Diagram of (general) diffusion and propagation algorithm (DPA; the figure is merely a schematic representation of the algorithm and does not correspond to the actual result of the given network). The algorithm combines defensive and offensive label propagation in a hierarchical manner (steps 1 and 2) to extract the core of the network (red heptagon communities) and to identify whisker communities (blue triangle and orange square communities). Whiskers are retained as identified communities, while the algorithm is recursively applied to the core of the (community) network. The recursion continues until all of the nodes of the (current) network are classified into the same community (i.e., offensive propagation in step 2 flood-fills), when the basic diffusion and propagation algorithm (BDPA) is applied (steps 3 and 4). For more detailed discussion on the algorithms, see text.

hierarchical label propagation on real-world networks and also on benchmark networks with planted partition. The analysis has shown that on the second iteration (when the algorithm is first run on the community network), the label propagation (already) produces one major community or even *flood-fills* (all nodes are classified into the same community).

Although the analysis revealed undesirable behavior, we have observed that the major community commonly coincides with the core of the network, while other communities correspond to whisker communities. Leskovec *et al.* [3] have extensively analyzed large social and information networks and observed that (these) networks reveal clear core-periphery structure—most of the nodes are in the central core of the network, which does not have a clear community structure, whereas the best communities reside in the periphery (i.e., whiskers), which is only weakly connected with the core. For further discussion see Appendix A.

Based on the above observations, we propose the following algorithm denoted the (general) diffusion and propagation algorithm (DPA); a schematic representation of the algorithm is depicted in Fig. 4. First, defensive label propagation is applied to the original network (step 1), which produces a larger number of smaller communities that are used to construct the corresponding community network. Second, offensive label propagation is used on the constructed community network (step 2), to extract the core of the network (i.e., its major community) and to identify whisker communities (i.e., all other communities). The above procedure is then recursively applied only to the core of the (community) network, while the whisker communities are retained as identified communities. The recursion continues until the offensive propagation in step 2 flood-fills (i.e., the extracted core contains all of the nodes of the network analyzed), when the basic BDPA is applied (steps 3 and 4).

Empirical analysis on real-world networks shows that DPA outperforms all other label propagation algorithms (with comparable time complexity) and is comparable to current state-of-the-art community detection algorithms. Furthermore, the algorithm exhibits almost linear complexity (in the number of edges of the network) and scales even better than the basic LPA. It should also be noted that the application of the algorithm is not limited to networks that exhibit core-periphery structure.

For a thorough empirical analysis and further discussion on both presented algorithms, see Sec. IV; and for the pseudocode of the algorithms and discussion on some of the implementation issues, see Appendix B.

IV. EVALUATION AND DISCUSSION

The section presents the results of the empirical evaluation of the proposed algorithms.

The algorithms were first compared on two classes of benchmark networks with planted partition, namely, Girvan and Newman [6] and Lancichinetti *et al.* [27] benchmark networks. For the latter, we also varied the size of the networks (1000 and 5000 nodes) and the size of the communities (from 10 to 50 and from 20 to 100 nodes). The results are assessed in terms of normalized mutual information (NMI) [52] and are shown in Fig. 5.

Analysis clearly shows the difference between defensive and offensive propagation, especially on larger networks [Figs. 5(d) and 5(e)]. The offensive propagation (ODALPA) performs slightly better than the basic LPA and can still relatively accurately detect communities, while LPA already performs rather poorly [Fig. 5(d)]. On the other hand, the

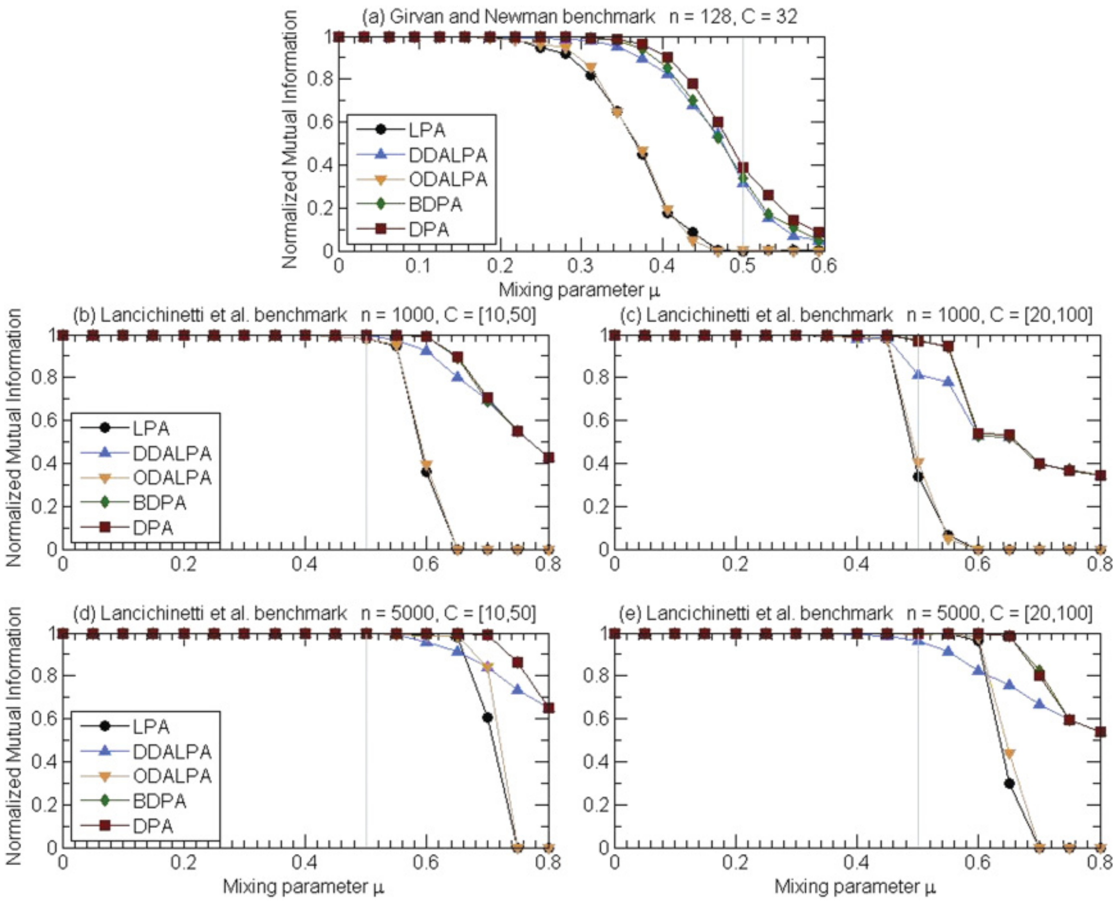

FIG. 5. (Color online) Comparison of the proposed algorithms on two classes of benchmark networks with planted partition, namely, Girvan and Newman [6] networks and four sets of Lancichinetti *et al.* [27] networks (the results are averages over 100 realizations). Network sizes equal 128, 1000, and 5000 nodes, and communities comprise up to 100 nodes. Straight (gray) lines at $\mu = 0.5$ denote the point beyond which the communities are no longer defined in the strong sense [13].

defensive propagation (DDALPA) does not detect communities as accurately as the other two approaches [Figs. 5(d) and 5(e)]; however, the algorithm still reveals the communities even when they are only weakly defined (and the other two approaches clearly fail). In other words, the defensive algorithm has high *recall*, whereas the offensive approach achieves high *precision*.

Furthermore, BDPA (and DPA) outperforms all three aforementioned algorithms. Note that the performance does not simply equal the upper hull of those for DDALPA and ODALPA. The analysis also shows that core extraction (i.e., DPA) does not improve the results on networks with thousands of nodes or less; the slight improvement on Girvan and Newman [6] benchmark networks results only from hierarchical investigation, and not core extraction. Nevertheless, as shown below, the results can be significantly improved on larger networks.

Lancichinetti and Fortunato [53] have conducted a thorough empirical analysis of more then 10 state-of-the-art community detection algorithms. To enable the comparison, the benchmark networks in Fig. 5 were selected so that they exactly coincide with those used in [53]. By comparing the results, we can conclude that DPA does indeed perform at least as good as the best algorithms analyzed in [53], namely, the hierarchical modularity optimization of Blondel *et al.* [10], the model selection approach of Rosvall and Bergstrom [16], the spectral algorithm proposed by Donetti and Munoz [15], and the multiresolution spin model of Ronhovde and Nussinov [20]. Moreover, on larger networks [Figs. 5(d) and 5(e)], DPA obtains even better results than all of the algorithms analyzed in [53]—for $\mu = 0.8$, none of the analyzed algorithms can obtain NMI above $\approx$0.35, while the values for DPA are 0.651 and 0.541, respectively.

DPA (and BDPA) was further analyzed on 23 real-world networks (Table I), ranging from networks with tens of nodes to networks with several tens of millions of edges [54]. To conduct a general analysis, we have considered a wide range of different types of real-world networks, in particular, social, communication, citation, collaboration, web, Internet, biological, and other networks (all networks were treated as unweighted and undirected). Because of the large number of networks considered, detailed description is omitted here.

The DPA algorithm was compared with all other proposed label propagation algorithms (to our knowledge), and with a greedy modularity optimization approach (Table I). The algorithms are as follows: LPA denotes basic label propagation [18], and LPAD denotes LPA with decreasing

TABLE I. Peak (maximal) modularities Q for various label propagation algorithms and a greedy optimization of modularity. The modularity for DPA for *elegans* was obtained with $\delta_{\max} = 1$ and for *asi* with $\delta_{\max} = 0$ (Appendix B); otherwise the values are 0.420 and 0.588, respectively. Values in italics correspond to the approaches that have significant time complexity compared to DPA.

Network	Description	*Nodes*	*Edges*	GMO	LPA	LPAD	LPAQ	LPAM	BDPA	DPA	No. CE[c]	T[c]
karate	Zachary's karate club [33]	34	78	0.381	0.416	0.402	0.399	**0.420**	**0.419**	**0.420**	0.02	
dolphins	Lusseau's bottlenose dolphins [38]	62	159		**0.529**	0.526	0.516	**0.529**	**0.528**	**0.529**	0.59	
books	Co-purchased political books [39]	105	441		**0.526**	0.519	0.522	**0.527**	**0.527**	**0.527**	0.46	
football	American football league [6]	115	616	0.556	**0.606**	**0.606**	0.604	**0.605**	**0.606**	**0.606**	0.37	
elegans	Metabolic network *C. elegans* [37]	453	2025	0.412	0.421	0.413	0.409	*0.452*	0.424	**0.427**[b]	0.17	
jazz	Jazz musicians [40]	198	2742	0.439	0.443	0.443	**0.445**	**0.445**	**0.444**	**0.444**	0.00	
netsci	Network scientists [9]	1589	2742		0.902	0.947			0.907	**0.960**	1.00	
yeast	Yeast protein interactions [41]	2114	4480		0.694	0.799			0.725	**0.824**	1.04	
emails	Emails within a university [42]	1133	5451	0.503	0.557	0.560	0.537	*0.582*	0.555	**0.562**	0.01	
power	Western US power grid [32]	4941	6594		0.612	0.804			0.668	**0.908**	1.14	
blogs	Weblogs on politics [43]	1490	16718		**0.426**	**0.426**			**0.426**	**0.426**	1.00	
pgp	*PGP* web of trust [44]	10680	24340	0.849	0.754	0.844	0.726	*0.884*	0.782	**0.869**	1.08	
asi	Autonomous syst. of Internet [25]	22963	48436		0.511	0.591			0.528	**0.600**[b]	1.02	0 s
codmat^3	*Cond. Matt.* archive 2003[a] [45]	27519	116181	0.661	0.616	0.683	0.582	*0.755*	0.634	**0.735**	1.00	1.5 s
codmat^5	*Cond. Matt.* archive 2005[a] [45]	36458	171736		0.586	0.643			0.608	**0.683**	1.00	
kdd^3	*KDD-Cup* 2003 dataset [46]	27770	352285		0.624	**0.630**			0.619	0.617	1.00	3 s
nec	*nec* web overlay map [47]	75885	357317		0.693	0.738			0.703	**0.767**	1.03	
epinions	*Epinions* web of trust [48]	75879	508837		0.382	0.362			0.399	**0.402**	1.00	4.5 s
amazon^3	*Amazon* co-purchasing 2003 [49]	262111	1.2M		0.682	0.749			0.701	**0.857**	1.01	20 s
ndedu	Webpages in nd.edu domain [50]	325729	1.5M		0.840	0.890			0.863	**0.903**	1.14	
google	Web graph of *Google* [3]	875713	4.3M		0.805	0.923			0.822	**0.968**	1.01	2.5 m
nber	*NBER* patents citations [51]	3.8M	16.5M		0.573	0.624			0.583	**0.759**	1.20	
live	*Live Journal* friendships [3]	4.8M	69.0M		0.538	0.539			0.557	**0.693**	1.00	44 m

[a]Reduced to the largest component of the original network.
[b]Obtained with slightly modified version of DPA (see caption).
[c]Average number of core extractions and computational times for DPA.

hop attenuation and node preference equal to the degree of the node [19] (Sec. II). The modularity optimization version of LPA is denoted LPAQ [11], and its refinement with multistep greedy merging LPAM [12]. Furthermore, GMO denotes greedy modularity optimization, proposed by Clauset *et al.* [8].

For each algorithm, we report peak (maximal) modularities obtained on the networks analyzed. Modularities for LPA, LPAD, BDPA, and DPA were obtained by running the algorithms from 2 to 100 000 times on each network (depending on the size of the network). On the other hand, peak modularities for LPAQ and LPAM (and also GMO) were reported by Liu and Murata [12].

The results show that DPA outperforms all other label propagation algorithms, except LPAM on networks of medium size (i.e., *elegans*, *emails*, *pgp*, and *codmat*^{3}). However, further analysis reveals that on these networks, LPAM already has considerable time complexity compared to DPA. It should also be noted that modularities obtained by LPAM on three of these networks correspond to the highest modularity values ever reported in the literature. Similarly, peak modularities obtained by DPA (and some others) on smaller networks also equal the highest modularities ever published (to our knowledge, the modularity for *football* even slightly exceeds the highest value ever reported, i.e., 0.606, as opposed to 0.605). In summary, DPA obtains significantly higher values of modularity than other comparable label propagation approaches, especially on larger networks (with millions of nodes and edges).

As already discussed in Sec. III C, BDPA achieves superior results on smaller networks, better than LPA, LPAQ, and LPAD (and GMO). However, the algorithm is not appropriate for

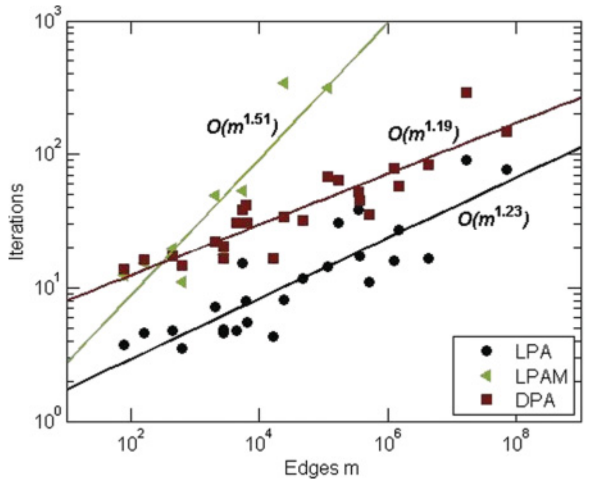

FIG. 6. (Color online) Time complexity of different label propagation algorithms estimated on real-world networks from Table I (results are averages over 100 iterations). From top to bottom, straight lines correspond to $0.83m^{0.51}$, $5.15m^{0.19}$, and $1.03m^{0.23}$, while the text denotes the overall time complexity of the algorithms (LPAM, DPA, and LPA, respectively). On a network with a billion edges, the (projected) number of iterations for DPA and LPA would equal 265 and 113, respectively.

TABLE II. Mean pairwise NMI of distinct community structures identified by different label propagation algorithms in 10 000 iterations (on selected set of networks from Table I).

Network	Nodes	Edges	LPA	BDPA	DPA
karate	34	78	0.574	0.578	**0.660**
dolphins	62	159	0.714	0.762	**0.774**
books	105	441	0.737	0.803	**0.805**
football	115	616	0.878	0.896	**0.897**
elegans	453	2025	0.610	0.615	**0.618**
jazz	198	2742	0.602	0.748	**0.808**

larger networks, where hierarchical core extraction prevails (i.e., DPA).

We have also analyzed the number of core extractions (Sec. III C) made by DPA on these networks (Table I). Core extraction does not gain on networks with less than thousands of nodes or edges, where the average number is commonly close to 0. However, when networks become larger, a (single) core extraction produces a significant gain in modularity (on these networks). Interestingly, even on a network with several millions of nodes and several tens of millions of edges (i.e., *live*), the number of extractions is still 1 (on average).

Next, we have thoroughly compared the time complexity of a simple LPA and DPA (and also LPAM [12]). On each iteration of the algorithms, each edge of the network is visited (at most) twice. Thus the time complexity of a single iteration equals $O(m)$, with m being the number of edges. The complexity for DPA is even lower, after the core has been extracted; however, due to simplicity, we consider each iteration to have complexity $O(m)$.

Iterative algorithms (such as label propagation) are commonly assessed only on smaller networks, where the number of iterations can be bounded by a small constant. In this context, both LPA and DPA exhibit nearly linear complexity, $O(m)$. However, on networks with thousands or millions of nodes and edges, this "constant" indeed increases—even for simple LPA, which is known for its speed, the number of iterations notably increases on larger networks. We have thus analyzed the total number of iterations made by the algorithms on real-world networks (Table I). The results are shown in Fig. 6 (the number of edges m is chosen to represent the size of the network). Note that the number of iterations for DPA corresponds to the sum of the iterations made by all of the algorithms run within (i.e., DDALPA, ODALPA, and BDPA).

As discussed earlier, DPA (and LPA) scales much better than LPAM—the average number of iterations on a network with tens of millions of edges is 147 and 78 for DPA and LPA, respectively, while LPAM already exceeds 300 iterations on networks with tens of thousands of edges. Furthermore, results also show that DPA scales even better than simple LPA [i.e., $O(m^{1.19})$, as opposed to $O(m^{1.23})$]; however, it is outperformed by LPA because of a larger constant. Nevertheless, the analysis shows promising results for future analyses of large complex networks.

In the context of analyzing large networks, it should be mentioned that by far the fastest convergence is obtained by using the defensive propagation algorithm DDALPA (Sec. III B). On the largest of the networks (i.e., *live*), the algorithm converges in only 25 iterations (3 times faster than LPA); still, the modularity of the revealed community structure is only 0.470.

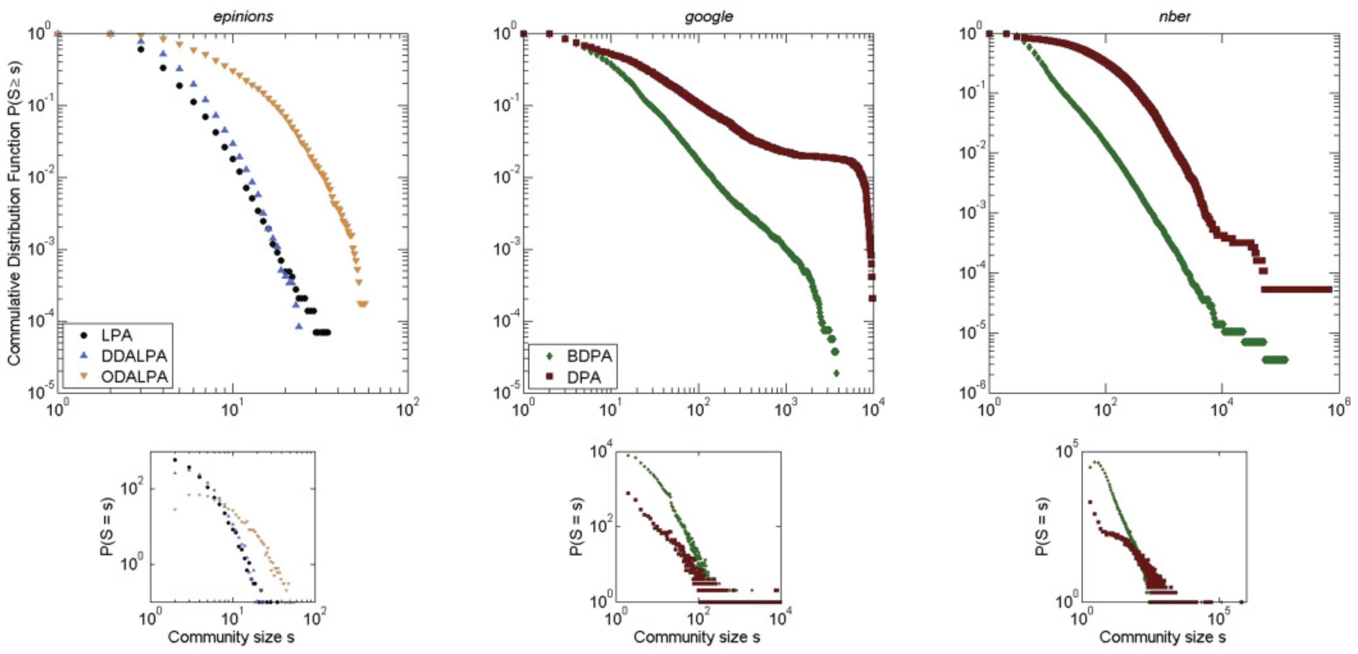

FIG. 7. (Color online) (Cumulative) distributions of the community sizes for three real-world networks from Table I (for the *epinions* network, the results were averaged over 10 runs). Note some particularly large communities revealed by DPA in the case of the *google* and *nber* networks (with around 10^4 and 10^6 nodes, respectively). Interestingly, these coincide with the low-conductance [55] communities reported in [3].

Last, we have also studied the stability of DPA (and BDPA), and compared it with simple LPA. The latter is known to find a large number of distinct community structures in each network [12,18,23], while Tibély and Kertész [23] have argued that these structures are relatively different among themselves. Indeed, on the *zachary* network LPA revealed 628 different community structures (in 10 000 iterations), while this number equals 159 and 124 for BDPA and DPA, respectively. However, as the number of distinct communities can be misleading, we have rather directly compared the identified community structures.

In Table II we show mean pairwise NMI of (distinct) community structures that were identified by the algorithms on a selected set of real-world networks. DPA (and BDPA) appears to be more stable than LPA; moreover, the identified community structures are relatively similar for all of the algorithms considered (in most networks analyzed). Interestingly, the results also seem to correlate well with the modularities shown in Table I—the clearer the community structure of the network, the more stable the algorithms appear. Nevertheless, as indicated by various previous authors [18,23], the number of different community structures can be very high, especially in larger networks (e.g., 1116 and 1330 for DPA applied to the *football* and *jazz* networks, respectively).

(Cumulative) distributions of sizes of communities, obtained with the proposed algorithms on three real-world networks, are shown in Fig. 7.

V. CONCLUSION

This paper proposes an advanced label propagation community detection algorithm that combines two unique strategies of community formation. The algorithm analyzes the network in a hierarchical manner that recursively extracts the core of the network and identifies whisker communities. The algorithm employs only local measures for community detection and does not require the number of communities to be specified beforehand. The proposition was rigorously analyzed on benchmark networks with planted partition and on a wide range of real-world networks, with up to several millions of nodes and tens of millions of edges. The performance of the algorithm is comparable to the current state-of-the-art community detection algorithms; moreover, the algorithm exhibits almost linear time complexity (in the number of edges of the network) and scales even better than the basic label propagation algorithm. The proposal thus gives prominent grounds for future analysis of large complex networks.

This work also provides further understanding on the dynamics of label propagation, in particular, on how different propagation strategies can alter the dynamics of the process and reveal community structures with unique properties.

ACKNOWLEDGMENTS

The authors wish to thank (anonymous) reviewers for comments and criticisms that helped improve the paper. This work has been supported by the Slovene Research Agency ARRS within Research Program No. P2-0359.

APPENDIX A: CORE-PERIPHERY STRUCTURE

Leskovec *et al.* [3] have conducted an extensive analysis of large social and information—and some other—networks. They have observed that these networks can be clearly divided into the central core and remaining periphery (i.e., the core-periphery structure). The periphery is constituted of many small, well-defined communities (in terms of conductance [55]) that are only weakly connected to the rest of the network. When they are connected by a single edge, they are called whiskers (or 1-whiskers). On the other hand, the core of the network consists of larger communities that are well connected, and thus only loosely defined in the sense of communities. Their analysis has thus shown that the best communities (according to conductance) reside in the periphery of these networks (i.e., whiskers) and have a characteristic size of around 100 nodes. For further discussion, see [3,56].

APPENDIX B: ALGORITHMS

In this section we give the pseudocode of all the algorithms proposed in this paper (Figs. 8–10) and discuss some of the implementation issues.

Because of the nature of label propagation, it may be that when the algorithm converges, two (disconnected) communities share the same label. This happens when a node propagates its label in two direction but is itself relabeled in the later stages of the algorithm. Nevertheless, disconnected communities can be detected at the end using a simple breath-first search.

Each run of BDPA or DPA (Figs. 9 and 10) unfolds several sets of communities and the best are returned at the end

```
Input: Graph $G(N,E)$ with weights $W$
Output: Communities $C$ (i.e. node labels)
 1: $\delta \leftarrow 0$
 2: for $n \in N$ do
 3:     $c_n \leftarrow l_n$ {Unique label.}
 4:     $d_n \leftarrow 0$
 5:     $p_n \leftarrow 1/|N|$
 6: end for
 7: $shuffle(N)$
 8: while not converged do
 9:     for $n \in N$ do
10:         $c_n \leftarrow \mathrm{argmax}_l \sum_{i \in \mathcal{N}^l(n)} p_i(1-\delta d_i) w_{ni}$
11:         if $c_n$ has changed then
12:             $d_n \leftarrow (\min_{i \in \mathcal{N}^{c_n}(n)} d_i) + 1$
13:             $p_n \leftarrow \sum_{i \in \mathcal{N}^{c_n}(n)} p_i / k_i^{c_n}$
14:         end if
15:     end for
16:     $\delta \leftarrow$ proportion of labels changed
17:     if $\delta \geq \delta_{max}$ then
18:         {$\delta_{max}$ is fixed to $\frac{1}{2}$.}
19:         $\delta \leftarrow 0$
20:     end if
21: end while
22: return $C$
```

FIG. 8. Defensive label propagation algorithm with (dynamic) hop attenuation (DDALPA). In the offensive version (ODALPA), the node preference p_i is replaced by $1 - p_i$ (line 10) and the degree $k_i^{c_n}$ is replaced by k_i (line 13).

```
Input: Graph $G(N, E)$ with weights $W$
Output: Communities $C$ (i.e. node labels)
  $C \leftarrow DDALPA(G, W)$
  for $c \in C$ do
    {Retain community cores.}
    $m_c \leftarrow median(\{p_n | n \in N \wedge c_n = c\})$
    for $n \in N \wedge c_n = c \wedge p_n \leq m_c$ do
      $c_n \leftarrow l_n$ {Unique label.}
      $d_n \leftarrow 0$
      $p_n \leftarrow 0$ {Maximal preference.}
    end for
  end for
  $C \leftarrow ODALPA(G, W)$
  return $C$ {Returns best communities.}
```

FIG. 9. Basic diffusion and propagation algorithm (BDPA).

(according to some measure of goodness of communities). For the analysis in Sec. IV, the algorithms reported community structures that obtained the highest modularity (computed on the original network). Thus, the results might be attributed to modularity's resolution limit problem [57] or other limitations [58]; still, this is not a direct artifact of the algorithms.

```
Input: Graph $G(N, E)$ with weights $W$
Output: Communities $C$ (i.e. node labels)
  $C \leftarrow DDALPA(G, W)$
  $C_C \leftarrow ODALPA(G_C, W_C)$
  if $C_C$ contains one community then
    $C \leftarrow BDPA(G, W)$
  else
    {Recursion on core $c$ in $C_C$.}
    $C \leftarrow (C_C - \{c\}) \cup DPA(G_C(c), W_C(c))$
  end if
  return $C$ {Returns best communities.}
```

FIG. 10. Diffusion and propagation algorithm (DPA).

An additional note should be made for the offensive propagation algorithm ODALPA (Fig. 8). When used on networks with several thousands of nodes or less, diffusion values p_n should only be updated (line 13) after the first iteration; otherwise the algorithm might not converge. The reason is that during the first iteration, communities are still rather small (because of the size of the network) and thus all of the nodes lie in the border of the communities. Hence, updating the diffusion values results in applying propagation preference to all of the nodes.

[1] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Nature (London) **435**, 814 (2005).
[2] E. Ravasz and A. L. Barabási, Phys. Rev. E **67**, 026112 (2003).
[3] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, Internet Mathematics **6**, 29 (2009).
[4] M. E. J. Newman and E. A. Leicht, Proc. Nat. Acad. Sci. USA **104**, 9564 (2007).
[5] S. Pinkert, J. Schultz, and J. Reichardt, PLoS Comput. Biol. **6**, e1000659 (2010).
[6] M. Girvan and M. E. J. Newman, Proc. Nat. Acad. Sci. USA **99**, 7821 (2002).
[7] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
[8] A. Clauset, M. E. J. Newman, and C. Moore, Phys. Rev. E **70**, 066111 (2004).
[9] M. E. J. Newman, Phys. Rev. E **74**, 036104 (2006).
[10] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, J. Stat. Mech. (2008) P10008.
[11] M. J. Barber and J. W. Clark, Phys. Rev. E **80**, 026129 (2009).
[12] X. Liu and T. Murata, Physica A **389**, 1493 (2009).
[13] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, Proc. Natl. Acad. Sci. USA **101**, 2658 (2004).
[14] C. Pang, F. Shao, R. Sun, and S. Li, in *Proceedings of the International Symposium on Neural Networks*, edited by W. Yu, H. He, and N. Zhang (Springer, Wuhan, 2009), pp. 839–846.
[15] L. Donetti and M. A. Muñoz, J. Stat. Mech. (2004) P10012.
[16] M. Rosvall and C. T. Bergstrom, Proc. Natl. Acad. Sci. USA **105**, 1118 (2008).
[17] A. Firat, S. Chatterjee, and M. Yilmaz, Comput. Stat. Data Anal. **51**, 6285 (2007).
[18] U. N. Raghavan, R. Albert, and S. Kumara, Phys. Rev. E **76**, 036106 (2007).
[19] I. X. Y. Leung, P. Hui, P. Liò, and J. Crowcroft, Phys. Rev. E **79**, 066107 (2009).
[20] P. Ronhovde and Z. Nussinov, Phys. Rev. E **81**, 046114 (2010).
[21] X. Liu and T. Murata, in *Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, Vol. 1 (IEEE, Washington, DC, 2009), pp. 50–57.
[22] S. Fortunato, Phys. Rep. **486**, 75 (2010).
[23] G. Tibély and J. Kertész, Physica A **387**, 4982 (2008).
[24] P. Schuetz and A. Caflisch, Phys. Rev. E **77**, 046112 (2008).
[25] M. E. J. Newman, [http://www-personal.umich.edu/~mejn/netdata/].
[26] Throughout the article we refer to core and whiskers of the network as being the result of the algorithm, although this might not necessarily coincide with the analysis of Leskovec *et al.* [9].
[27] A. Lancichinetti, S. Fortunato, and F. Radicchi, Phys. Rev. E **78**, 046110 (2008).
[28] The conclusion naturally depends on the definition of research communities, which are, in this case, communities revealed by the algorithm.
[29] S. Gregory, e-print arXiv:0910.5516.
[30] L. Freeman, Sociometry **40**, 35 (1977).
[31] L. C. Freeman, Soc. Networks **1**, 215 (1979).
[32] D. J. Watts and S. H. Strogatz, Nature (London) **393**, 440 (1998).
[33] W. W. Zachary, J. Anthropol. Res. **33**, 452 (1977).
[34] The estimation could also be done by using the label propagation itself; however, we believe that using the floating-point counterpart of the approach would produce more accurate results in practice.

[35] S. Brin and L. Page, Comput. Networks ISDN **30**, 107 (1998).

[36] J. M. Kleinberg, J. ACM **46**, 604 (1999).

[37] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. Barabási, Nature (London) **407**, 651 (2000).

[38] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, Behav. Ecol. Sociobiol. **54**, 396 (2003).

[39] V. Krebs, [http://www.orgnet.com/].

[40] P. Gleiser and L. Danon, Adv. Complex Syst. **6**, 565 (2003).

[41] H. Jeong, S. P. Mason, A. Barabási, and Z. N. Oltvai, Nature (London) **411**, 41 (2001).

[42] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas, Phys. Rev. E **68**, 065103 (2003).

[43] L. A. Adamic and N. Glance, in *Proceedings of the International Workshop on Link Discovery*, edited by J. Adibi, M. Grobelnik, D. Mladenic, and P. Pantel (ACM, New York, 2005), pp. 36–43.

[44] M. Boguná, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas, Phys. Rev. E **70**, 056122 (2004).

[45] M. E. J. Newman, Proc. Natl. Acad. Sci. USA **98**, 404 (2001).

[46] See [http://www.sigkdd.org/kddcup/].

[47] M. Hoerdt, M. Jaeger, A. James, D. Magoni, J. Maillard, D. Malka, and P. Merindol, [http://www.labri.fr/perso/magoni/nec/].

[48] M. Richardson, R. Agrawal, and P. Domingos, in *Proceedings of the International Semantic Web Conference*, edited by D. Fensel, K. P. Sycara, and J. Mylopoulos, Vol. 2 (Springer, Berlin, 2003), pp. 351–368.

[49] J. Leskovec, L. A. Adamic, and B. A. Huberman, ACM Trans. Web **1**, (2007).

[50] R. Albert, H. Jeong, and A. Barabási, Nature (London) **401**, 130 (1999).

[51] B. H. Hall, A. B. Jaffe, and M. Tratjenberg, *The NBER Patent Citation Data File: Lessons, Insights and Methodological Tools*, Tech. Rep. (National Bureau of Economic Research, 2001).

[52] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, J. Stat. Mech. (**2005**) P09008.

[53] A. Lancichinetti and S. Fortunato, Phys. Rev. E **80**, 056117 (2009).

[54] The analysis on networks with hundreds of millions or even billions of edges was bounded due to limited memory resources.

[55] B. Bollobás, *Modern Graph Theory* (Springer-Verlag, Berlin, 1998).

[56] J. Leskovec, K. J. Lang, and M. W. Mahoney, in *Proceedings of the ACM International Conference on World Wide Web*, edited by M. Rappa, P. Jones, J. Freire, and S. Chakrabarti (ACM, New York, 2010), pp. 631–640.

[57] S. Fortunato and M. Barthelemy, Proc. Natl. Acad. Sci. USA **104**, 36 (2007).

[58] B. H. Good, Y. A. de Montjoye, and A. Clauset, Phys. Rev. E **81**, 046106 (2010).