On the Detection of Overlapped Network Communities Via Weight Redistributions

Stavros I. Souravlas and Angelo Sifaleras

Abstract A community is an important attribute of networking, since people who join networks tend to join communities. Community detection is used to identify and understand the structure and organization of real-world networks, thus, it has become a problem of considerable interest. The study of communities is highly related to network partitioning, which is defined as the division of a network into a set of groups of approximately equal sizes with minimum number of edges. Since this is an NP-hard problem, unconventional computation methods have been widely applied.

This work addresses the problem of detecting overlapped communities (communities with common nodes) in weighted networks with irregular topologies. These communities are particularly interesting, firstly because they are more realistic, i.e., researchers may belong to more than one research community, and secondly, because they reveal hierarchies of communities: i.e., a medical community is subdivided into groups of certain specialties. Our strategy is based on weight redistribution: each node is examined against all communities and weights are redistributed between the edges. At the end of this process, these weights are compared to the total connectivity of each community, to determine if overlapping exists.

1 Introduction

Several systems of high interest to the scientific community can be successfully represented as networks. Network examples are the Internet, the World-Wide Web,

Stavros Souravlas,
University of Macedonia, Department of Applied Informatics, 156 Egnatias Str., Thessaloniki 54636, Greece, e-mail: sourstav@uom.gr

Angelo Sifaleras
University of Macedonia, Department of Applied Informatics, 156 Egnatias Str., Thessaloniki 54636, Greece, e-mail: sifalera@uom.gr

and the social networks [5]. Perhaps the latest are the type of networks in which researchers are most interested nowadays. Examples of social networks are Facebook (1.6 billion users), Instagram (400 million users), or Twitter (320 million users) [22]. Typically, social networks are organized in groups of users [3]. These users join a network, create their own profile, publish information and find other users with the same interests. In this way, small or larger groups of users are formed within networks. Such groups are referred to as *communities*. Although there is no universally acceptable definition of a community, one can define it as a set of nodes and links in a network such that, its internal connections are more than its external connections [6]. The nodes of a community are considered similar to each other, dissimilar to the other nodes of the network [21], and represent its users. The edges represent the connections between the users of one community or between users of different communities.

A lot of effort has focused on detecting communities in social and data networks. Communities give a more realistic approach of the networks and they can also be used to help the relay of messages within a network [7, 8, 10, 15]. The majority of the research papers focused on this subject state that common experience shows that communities really exist in social and data networks [2, 3]. The study of community structures is highly related to the problem of network partitioning. Typically, the network partitioning problem is defined as the partitioning of a network into a set of groups of approximately equal sizes with minimum number of edges [18, 21]. This problem is NP-hard and thus, heuristics and approximation algorithms are usually employed [14]. Much of this work involves parallel computations to enhance performance. Specifically, the nodes of the system represent computations while the edges represent communications and the general idea is to equally assign nodes among the processors, so that the communications (edges) between them are minimized. However, network partitioning is not the ideal method for the analysis of networks and for community detection. This is firstly due to the fact that in real networks, the communities formed rarely have approximately the same size, secondly because the connections are not minimized and finally because network partitioning does not consider the similarities between nodes (or users), which are inherited in a social network.

Most of the papers found in the literature agree that the members of a community are more strongly connected between them than they are connected to members of other communities. These papers introduced a number of algorithms to identify communities and measures to test and compare these algorithms [12, 18]. The most popular metric in the literature, which is used to quantify the strength of a community structure is *modularity*. This metric was defined by Newman and Girvan [18] and it measures the difference between the density of edges inside the communities and the density that would have occurred if the edges were randomly distributed over the entire network. A number of papers have been influenced by this metric [2, 11, 16, 17, 24, 26], although it has some flaws like the resolution limit, which can be partially overcomed by techniques like [1].

Members in the same community share the same interests, viewpoints, preferences, hobbies, professions, newsgroups, and, as far as social networks are concerned, they are more likely to comment or “like” topics of the same interest. In most of the papers, this whole activity is modeled via analyzing the link weights [4, 9, 16, 15, 19, 20, 21] of the network. In order for a node to be considered as member of a community, its internal/external links to this community should follow some well-defined properties. Some algorithms also detect *overlapping* communities [4, 9, 13, 15, 25]

The main problem in these techniques is that, the aforementioned analysis usually suffers high complexity times, although there are also linear-time approaches [21, 23]. Lu et al. [15] proposed a strategy which can detect both overlapping and non-overlapping communities by adding one node to a community at each *expanding step*. In each expanding step, the authors initially compute the *belonging degree* of a number of nodes to a community C and they pick the one with the highest belonging degree. This node is temporarily attached to C, forming $C^{'}$. Then, if the *conductance* of $C^{'}$ is lower compared to the conductance of C, the selected node is finally attached to C. The worst time complexity can grow as large as n^2, where n is the number of nodes in the network.

Another interesting strategy proposed by Newman and Girvan [18] repeatedly calculates the *betweeness* scores between all the edges in the network and remove the edge with the highest score. Again, this strategy operates in worst-case time that can be as large as $O(n^3)$. More strategies for modeling the relationships between the nodes of a network have quadratic or worse complexities [16, 19, 20]. An interesting linear-complexity approach was presented by Raghavan et al. [21], which uses a simple label propagation algorithm. Unlike most of the strategies found in the literature this strategy does not use a predefined objective function (like the conductance mentioned in the previous paragraph) to identify the communities. Every node is initialized with a unique label and at every step each node adopts the label that most of its neighbors currently have. There are two concerns regarding this technique. First, it is assumed that each node joins the community in which most of its neighbors belong and second there is no clear updating strategy that shows what would happen when new nodes entering the network have equal number of neighbors between many communities or when some nodes leave the network temporarily or permanently.

This paper presents a weight-redistribution technique used to find overlapping communities in networks with irregular topologies. Our strategy performs a depth-first based search over the network and redistributes the data weights across the links. Finally, the values computed by the redistribution function are compared to the connectivity values of each community to determine if there are overlaps. The remaining of this paper, is organized as follows: Section 2 briefly sets up the problem studied, presents the weight-redistribution strategy, and performs a theoretical analysis. Section 3 concludes the paper and offers aspects for future research.

2 Overlapped Communities Detection Strategy

In this section, we typically describe the proposed strategy and briefly describe some aspects of the problem studied.

2.1 Problem Aspects

Consider a small network of irregular topology, like the one shown in Fig. 1, where there are three communities C_1, C_2, and C_3. The problem we address is to find whether there is overlapping between the communities. Fig. 1 reveals some aspects that need to be considered before trying to uncover these overlaps:

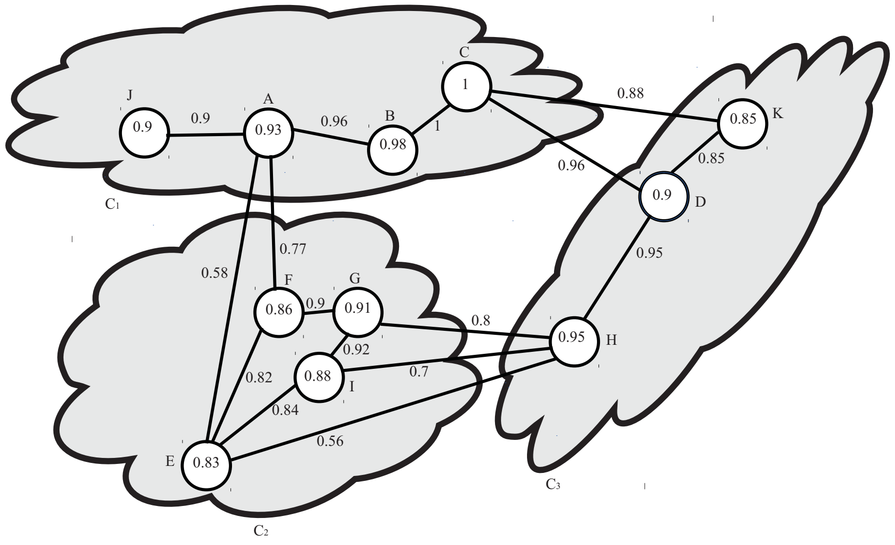

Fig. 1 A network with three communities, C_1, C_2, and C_3, depicting nodes similarities (numbers next to each edge) and network connectivity degrees (numbers inside each node)

`Topic 1:` User A is directly connected to users E and F in C_2. Thus, there is a chance that A belongs to C_2 as well.

`Topic 2:` User A is connected to C_3 via E or via the rout B,C. Thus, there is a chance that A belongs to C_3 as well. In this case, the relays E or B,C may also belong to C_3.

`Topic 3:` Continuing Topic 2, if there is a high percentage of overlapping between two communities, chances are that they are actually one community.

`Topic 4:` Not all users of a community are necessarily related (in Facebook for example, two users may have common friends while they are not related). In Fig. 1, A is not directly connected H, it is connected to E. In its turn, E is connected to H. To continue with the Facebook analogy, this is a case that a user (for example A) is unaware of a community C_3 until he/she visits the page of another user that has some relationship to members of this community (like E). A way to quantify the *similarities* (see next section) between unrelated users, like A and H is required.

The four topics referred in this section will be considered in our proposed strategy, which is typically presented in the next section.

2.2 Finding Community Overlaps

Let $G = (V, E)$ be a weighted, undirected graph, where V is the set of nodes and E is the set of edges. Nodes represent users and edges represent the connections between two users (for example friendship in Facebook, etc.). The *similarity* between users i and j is given by $w_{i,j}$, that is, the weight of the edge that connects i and j. This value lies in the interval $[0 \dots 1]$. For example, a value of 0.78 shows that the views of i and j can be considered quite converging, at a percentage of 78%. Bu et al. [3] present a table of user phrases that indicate supportive or opposing attitude towards a comment or opinion, etc. These phrases are accompanied by a value that shows the degree of support.

The network nodes are also weighted: the weight of a node i indicates its *network connectivity degree*, that is, how well the preferences, likes, views of a user are fitted to a community. The network connectivity degree is also between $[0 \dots 1]$. The letters are the node names, the edge values indicate view similarities and the node values indicate similarity degrees. The *Network Connectivity Degree* of a user i, NCD_i, is computed as follows:

$$NCD_i = \frac{\sum w_{i,j}}{m} \tag{1}$$

where $w_{i,j}$ is the weight of any edge that connects user i with any user j that lies in the same community and m is the number of such edges. For example, consider community C_2. User G has two internal links, namely with users F and I and one external, with user H. To compute NCD_G, we only consider the internal links and we have $NCD_G = \frac{(0.9+0.92)}{2} = 0.91$. This value is stored in node G. The values of the external links will be used to determine overlapping, as will be seen later.

Topics 1 and 2 of Section 2.1 indicate that a user can be connected to a community, either *directly*, through a relationship with a user or many users of the community or *indirectly*, through a relationship with a user that is related to a member of the community. Thus, we need to quantify the similarities between unrelated nodes (Topic 4), in order to uncover possible overlaps. Since there is no relationship (thus we can't use data like the ones proposed by Bu et al. [3]), we have to intuitively describe similarity. Our approach suggests that the similarity of two unconnected users is generally higher when the number of links between them is small (for example a friend of a friend in Facebook probably has more similarities compared to a friend of a friend . . . of a friend . . .), and when the weights on the links that connect them are as large as possible. Thus, the largest possible similarities are likely to be found in the shortest paths from each node to each community. The depth-first based procedure described next, finds the highest similarities between unrelated nodes. During this procedure, we also redistribute the weights:

Procedure 1: Finding the Similarities Between Unrelated users

```
Step 1: Start from a user node, which is examined for possible overlaps with
        a community $C$. This user will be denoted as the root. Set $a = 0$ (a
        variable to keep the current highest similarity).
Step 2: Find all the users connected to the root, such that the weight of their
        corresponding connecting links is $> a$.
        If there are direct connections to the desired community
            Step 2.1: Select the link with the highest weight value, assign it as
                      the working link and move to Step 3
        else
            Step 2.2: Select the link from the root to an unprocessed node $j$
                      that has the highest $w_{root,j}$ and mark $j$ as processed. Set
                      $j$ as the root. The chosen link is the working link. Move
                      to Step 3.
Step 3: Update the weight of the working link using the computed product of
        the previous link (or 1 for the first iteration) and the weight of the work-
        ing link.
Step 4: If $C$ is reached then,
            Step 4.1: If the working link weight is $> a$, then
                Step 4.1.a: Update $a$
                Step 4.1.b: Exclude from further examination all
                            the links with weight less than $a$ (they
                            will never produce a similarity $> a$).
            Step 4.2: Return to the root and set the link leading to the root as
                      the working link.
            Step 4.3: If there is a link from the root to a node $j$ (out of the
                      examined community) such that $w_{root,j} > a$, then go
                      to Step 2, else go to Step 5.
        else return to Step 2.
Step 5: Work backwards until there is no unprocessed node and link.
```

Let us use the example of Fig.1 to see how this procedure will compute the similarity between node A and the nodes in community C_3. From Step 1, user A is set as the root. From `Step 2`, there are 4 users connected to the root, namely B, J, F, E (in decreasing order of link weights). The weights of all links are greater thatn a ($a = 0$ from `Step 1`) and there is no direct connection to a member of C_3. So, according to `Step 2.2`, user B is selected, and $w_{A,B} = 0.96$. User B is marked as processed and becomes the root. The link from A to B is the working link. In `Step 3`, a weight equal to 1 is assigned to the working link (first iteration, there is no previous link), and its newly assigned weight will be $0.96 \times 1 = 0.96$. From `Step 4`, C_3 is not reached, so we have to return to `Step 2`. From the root, user B, there is only one connection to user C. There is no direct connection to C_3, so we move to `Step 2.2`. Since $w_{B,C} = 1 > a$, C is marked as processed and becomes the new root The link from B to C becomes the working link. From `Step 3`, the weight of the previous node is redistributed to the working link, and the product is $0.96 \times 1 = 0.96$. So $w_{B,C} = 0.96$. From `Step 4`, C_3 is not reached, so we repeat `Step 2`. The users connected to the root are D and K. The link weights are greater than a and there is direct connection to C_3. So (`Step 2.1`), we choose the link to user D, since its weight is 0.96, while the weight of link to user K is 0.88. The chosen link is the working link. From `Step 3`, the weight from the previous link is redistributed, so a weight of 0.96 is multiplied with $w_{C,D}$. Thus, the new $w_{C,D} = 0.96 \times 0.96 \approx 0.92$. Now, the condition of `Step 4` is satisfied. The working link's weight is greater than a, so a becomes 0.92. Now, all the links with weights less than 0.92 have to be excluded from further searching, since multiplications with values less than 1 will always be producing similarities less than a. Thus, the algorithm must terminate and the largest similarity of a to a node of C_3 is 0.92.

We now show the importance of eliminating some links in `Step 4.1.b`. Assume that, we let the algorithm continue, thus we ignore `Step 4.1.b`. The algorithm will continue, subsequently assigning as roots nodes F and G. Then, it reaches node H inside C_3 and computes $W_{GH} = 0.77 \times 0.9 \times 0.8 \approx 0.55$. Since C is reached, we move to `Step 4.1` but the condition is false. So we return to the root, G, assign the link connecting F and G as the working link and move to user I. From I, there is direct connection to a member of C_3, namely H. Thus, $w_{I,H} = 0.77 \times 0.9 \times 0.92 \times 0.7 \approx 0.44$. Similarly, the algorithm will compute $w_{E,H} = 0.77 \times 0.9 \times 0.92 \times 0.56 \approx 0.35$. Assuming that the condition $w_{root,j} > a$ of `Step 4.3` is also not checked, the algorithm will take us to `Step 5` and from E we will move to the initial root A, since this is the only unprocessed node, when H is the root. When we reconsider A as root, there are two possible options: to compute the similarity for E to H, $w_{E,H} = 0.58 \times 0.56 \approx 0.32$ and to move to user J that has no further connections. Apparently, `Step 4.1.b` saves us from many unnecessary computations.

Having found the highest similarity between a node and a community, we have to define the conditions under which the node can be considered as member of the community. We define the *Average Community Connectivity* (*ACC*) for a community C as the average of the *NCD*s of all the nodes in the C, that is:

$$ACC_C = \frac{\sum_{i=1}^{n} NCD_i}{n} \tag{2}$$

where n is the number of nodes. For example, the *ACC* of C_3 is $ACC_{C_3} = \frac{0.85+0.9+0.95}{3} = 0.9$.

To allow a user to become a member of the community, we must test if its inclusion will improve the community's *ACC*. For example, we have computed the similarity of A and D, which is 0.92. Since A's similarity to C is greater than 0.9, A can be considered as member of C_3. The checking procedure is described into two steps:

Procedure 2: Checking a Node's Inclusion to A Community

```
Step 1: From Procedure 1, find the highest similarity of a node $i$ to a member $j$
        of a community $C$.
Step 2: If $w_{i,j} > ACC_C$, then
            Step 2.1: Include the node to the community.
            Step 2.2: Run Procedure 1 for all the nodes on the path from $i$ to $j$
                      to check for more overlaps.
        else check another node.
```

`Step 2.2` indicates that there is a high probability that more nodes on the path from the examined node i to a community member j can also be added to the community. Generally, if two communities have several common nodes, they can be considered as a larger community.

2.3 Performance of the Communities Detection Strategy

To evaluate performance, we have to analyze Procedure 1. Proposition 1 gives us an upper bound for the complexity of this procedure.

Proposition 1: The time required for Procedure 1 to complete its computations is at most $O(m)$, where m is the number of edges.

Proof: Assume that the procedure starts from a node i. Each node is marked as unprocessed once and each link is processed two times, one during the redistribution step and one in case we return back to the root, and there are still unprocessed nodes. Thus, the total time is dictated by the number of edges and it is limited to this value, since the procedure involves a number of edges (and consequently nodes), which are not examined at all (see `Step 4.1.b` of the procedure). Thus, we can conclude that the proposed method is completed at linear time, which is a great advantage if compared to the times of other well-known strategies [16, 19, 20] among others.

3 Conclusions - Future Work

In this paper, we presented a computational method to find overlaps between communities in a network. Communities are important, since they give a realistic view of a network. Moreover, they can be used to develop forwarding (routing) algorithms (for example see [16]). Our technique is based on a depth-first based search through the network, in combination with a redistribution of weights on the links, to compute similarities between nodes that are not connected via a link.

This work gives rise to several issues that need to be further investigated. First, the adaptability to dynamic networks, where nodes arbitrarily enter and leave. This is a very complex problem, since any newly inserted node may be included in more than one communities, while a node leaving the network will necessarily force some reorganization over the network. Another point of interest is to test the strategy on a real network with real users and data and compare our results with other well-known strategies.

References

1. Berry, J.W., Hendrickson, B., LaViolette, R.A., Phillips, C.A.: Tolerating the community detection resolution limit with edge weighting. Physical Review. E **83**(5), 056119 (2011)
2. Bu, Z., Xia, Z., Wang, J.: A sock puppet detection algorithm on virtual spaces. Knowledge-Based Systems **37**, 366–377 (2013)
3. Bu, Z., Zhang, C., Xia, Z., Wang, J.: A fast parallel modularity optimization algorithm (FPMQA) for community detection in online social network. Knowledge-Based Systems **50**, 246–259 (2013)
4. Chen, D., Shang, M., Lv, Z., Fu, Y.: Detecting overlapping communities of weighted networks via a local algorithm. Physica A: Statistical Mechanics and its Applications **389**(19), 4177–4187 (2010)
5. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. Physical Review E **70**(6), 066111 (2004)
6. Fortunato, S.: Community detection in graphs. Physics Reports **486**(35), 75–174 (2010)
7. Gao, W., Cao, G., Porta, T.L., Han, J.: On exploiting transient social contact patterns for data forwarding in delay-tolerant networks. IEEE Transactions on Mobile Computing **12**(1), 151–165 (2013)
8. Gao, W., Li, Q., Zhao, B., Cao, G.: Social-aware multicast in disruption-tolerant networks. IEEE/ACM Transactions on Networking **20**(5), 1553–1566 (2012)
9. Gregory, S.: Finding overlapping communities in networks by label propagation. New Journal of Physics **12**(10), 103018 (2010)
10. Hui, P., Crowcroft, J., Yoneki, E.: Bubble rap: Social-based forwarding in delay-tolerant networks. IEEE Transactions on Mobile Computing **10**(11), 1576–1589 (2011)
11. Hui, P., Yoneki, E., Chan, S.Y., Crowcroft, J.: Distributed community detection in delay tolerant networks. In: Proceedings of 2nd ACM/IEEE International Workshop on Mobility in

the Evolving Internet Architecture, MobiArch ’07, pp. 7:1–7:8. ACM, New York, NY, USA (2007)
12. Lancichinetti, A., Fortunato, S.: Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. Physical Review E **80**(1), 016118 (2009)
13. Lancichinetti, A., Fortunato, S., Kertész, J.: Detecting the overlapping and hierarchical community structure in complex networks. New Journal of Physics **11**(3), 033015 (2009)
14. Leskovec, J., Lang, K.J., Mahoney, M.: Empirical comparison of algorithms for network community detection. In: Proceedings of the 19th International Conference on World Wide Web, WWW ’10, pp. 631–640. ACM, New York, NY, USA (2010)
15. Lu, Z., Sun, X., Wen, Y., Cao, G., Porta, T.L.: Algorithms and applications for community detection in weighted networks. IEEE Transactions on Parallel and Distributed Systems **26**(11), 2916–2926 (2015)
16. Lu, Z., Wen, Y., Cao, G.: Community detection in weighted networks: Algorithms and applications. In: IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 179–184 (2013)
17. Newman, M.E.: Analysis of weighted networks. Physical Review E **70**(5), 056131 (2004)
18. Newman, M.E., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E **69**(2), 026113 (2004)
19. Nguyen, N.P., Dinh, T.N., Tokala, S., Thai, M.T.: Overlapping communities in dynamic networks: Their detection and mobile applications. In: Proceedings of the 17th Annual International Conference on Mobile Computing and Networking, MobiCom ’11, pp. 85–96. ACM, New York, NY, USA (2011)
20. Nguyen, N.P., Dinh, T.N., Xuan, Y., Thai, M.T.: Adaptive algorithms for detecting community structure in dynamic social networks. In: IEEE International Conference on Computer Communications (INFOCOM), pp. 2282–2290 (2011)
21. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Physical review E **76**(3), 036106 (2007)
22. Statista: URL http://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users
23. Wu, F., Huberman, B.A.: Finding communities in linear time: a physics approach. European Physical Journal B **38**(2), 331–338 (2004)
24. Xia, Z., Bu, Z.: Community detection based on a semantic network. Knowledge-Based Systems **26**, 30–39 (2012)
25. Xie, J., Kelley, S., Szymanski, B.K.: Overlapping community detection in networks: The state-of-the-art and comparative study. ACM Computing Surveys **45**(4) (2013)
26. Zhao, Z., Feng, S., Wang, Q., Huang, J.Z., Williams, G.J., Fan, J.: Topic oriented community detection through social objects and link analysis in social networks. Knowledge-Based Systems **26**, 164–173 (2012)