Fast Detection of Size-Constrained Communities in Large Networks

Marek Ciglan and Kjetil Nørvåg

Dept. of Computer and Information Science, NTNU, Trondheim, Norway
marek.ciglan@idi.ntnu.no

Abstract. The community detection in networks is a prominent task in the graph data mining, because of the rapid emergence of the graph data; e.g., information networks or social networks. In this paper, we propose a new algorithm for detecting communities in networks. Our approach differs from others in the ability of constraining the size of communities being generated, a property important for a class of applications. In addition, the algorithm is greedy in nature and belongs to a small family of community detection algorithms with the pseudo-linear time complexity, making it applicable also to large networks. The algorithm is able to detect small-sized clusters independently of the network size. It can be viewed as complementary approach to methods optimizing modularity, which tend to increase the size of generated communities with the increase of the network size. Extensive evaluation of the algorithm on synthetic benchmark graphs for community detection showed that the proposed approach is very competitive with state-of-the-art methods, outperforming other approaches in some of the settings.

1 Introduction

Many real-world data sets have the form of graphs. The most straightforward examples are those of various types of networks; e.g., information networks, citation networks, social networks, communication networks, transportation networks or biological networks. This wide variety of existing graph data has triggered an increased attention to the graph analysis and the graph mining within the research community. One of the prominent tasks in graph mining is that of the community detection. The community detection aims to discover the community structure of a graph by dividing nodes of graph to clusters, where nodes of the same cluster are densely linked among themselves and less densely linked to the nodes of other communities.

Real-world networks exhibit the community structures, along with other properties like power law degree distribution and small-world property. Although the notion of a graph community structure has not been formally defined and is still intuitive for the most part (a number of diverse definition can be found through the literature), research on the community detection has gained a momentum in recent years. The most widely used approach in community detection algorithms is to maximize the modularity measure of the graph partition [16]. The problem of the modularity based algorithms is the resolution limit [8] – identified communities get bigger as the size of the network increases.

L. Chen, P. Triantafillou, and T. Suel (Eds.): WISE 2010, LNCS 6488, pp. 91–104, 2010.

In this paper we first formalize the problem; we define affinity partition of the network, based on the notion of community as defined by Hu et al in [12]. We then propose a new greedy algorithm for community detection in graphs. It builds on the label propagation algorithm [19] and overcome its drawback, which is the collapse of the result into a single community when the community structure of the given network is not very clear. Our algorithm is pseudo-linear in the execution time and it is designed to allow a user to specify the upper size limit of the communities being produced. As we will illustrate later on a real-world example, the communities identified in large networks are often of large sizes. This can be very impractical for certain classes of applications; in case the identified community structure is used by human expert (e.g. identification of a social group for a user of a social network), the community of several hundred thousands of nodes is not very useful. Another example of usefulness of size-constrained community detection is the task of partitioning the graph data for a distributed graph database into k balanced components.

The ability to constrain the size of communities is a feature that distinguish our approach from other methods. It is especially suited to identify small-scale communities in large networks and can be viewed as a complementary approach to the methods maximizing the modularity. The main contribution of the paper is the new algorithm for community detection, which has following distinctive features:

- Allows user to specify upper size limit of the produced communities
- Fast in execution time; as it is pseudo-linear in time complexity, it can be used also for clustering of large graphs
- Achieves very good quality in unveiling community structure for networks with communities of small sizes and has results competitive with state-of-the-art methods (often with higher time complexity)

We first present related work in Section 2. In the Section 3, we describe measures useful for the evaluation of the goodness of the identified graph partition into communities and we discuss other existing fast algorithms for community detection. In Section 4 we propose a new algorithm for community detection. Section 5 presents the evaluation of our approach. We conclude the paper in Section 6 where we also provide directions for future research.

2 Related Work

The problem of the community detection has been quite popular in recent years and a large number of different algorithms addressing the problem were proposed. The interest in the problem boomed after introduction of divisive methods based on betweenness centrality measures, mainly the Girvan-Newman algorithm [9]. Same authors have later proposed modularity measure [16] – a quality function that can be used to evaluate how good a partition of the network is. The underlying idea is to compare how community-like the given partition is compared to a random graph with the same node degrees. This is perhaps the most influential work in the community detection research. A number of proposed methods rely on maximizing the modularity when identifying the community structure; e.g., greedy techniques [1] [4], simulated annealing based method [11]

or extremal optimization [3]. As was shown in [8], the modularity function has a resolution limit that prevents communities of small sizes (compared to the graph size) to be identified, even if they are clearly defined. Variants of basic community detection problem were studied in literature, e.g. clustering of bipartite graphs [17], identification of overlapping communities [10, 23] or clustering based on graph structure combined with additional content associated with nodes [22]. The most related to our work are fast, greedy algorithms for community detection [19, 1, 20]. We discus those works in more detail in Section 3. A number of other approaches to the community detection task has been proposed; an extensive overview of the methods and graph clustering related problems can be found in surveys on the topic [7, 18].

3 Preliminaries

In this section, we discuss the measures used for the evaluation of the goodness of identified community structures, when the community structure is known, e.g. is artificially planted in the network by a benchmark graph generator. We than discuss in more detail three algorithms closely related to our algorithm proposed in Section 4.

Community detection formalization. Although there is no consensus on the formal definition of the community detection task, we try to explain the problem at hand in a more formal manner, following the approach in [7]. Let the G, be a graph $G = \{V, E, f\}$, where V is a set of nodes, E is a set of edges, $f : V \times V \rightarrow E$. The partition of the node set is $P(V) = \{C_1, C_2, \ldots, C_k\}$ where $\bigcup_{\forall C_i \in P(V)} C_i = V$ and $\bigcap_{\forall C_i \in P(V)} C_i = \emptyset$. Let $E_{C_i,in}$ be a set of intra-cluster edges of cluster C_i : $E_{C_i,in} = \{e_{k,l} \in E : k \in C_i \wedge l \in C_i\}$ and let $E_{C_i,out}$ be a set of inter cluster edges: $E_{C_i,out} = \{e_{k,l} \in E : (k \in V \setminus C_i \wedge l \in C_i) \vee (k \in C_i \wedge l \in V\ C_i)\}$. Let $|V| = n$ and $|C_i| = n_{C_i}$; we can define

$$\delta_{in}(C_i) = \frac{|E_{C_i,in}|}{n_{C_i} \times (n_{C_i} - 1)/2} \quad \text{and} \quad \delta_{out}(C_i) = \frac{|E_{C_i,out}|}{n_{C_i} \times (n - n_{C_i})}$$

The goal of community detection is then finding a partition with a good balance between large $\delta_{in}(C_i)$ and small $\delta_{out}(C_i)$. We formalize out perception of the community structure in 4.1.

Comparing partitions of a network. Benchmark graph generators produce a definition of network and a division of nodes to communities – the 'ground truth' partition of the given network. The community detection algorithm also produces a partition of the given network. The problem is how to evaluate how good the partition provided by a community detection algorithm approximates the original partition. In the community detection literature, an established similarity measure for comparing network partitions is *Normalized Mutual Information* (NMI). This measure originates from the information theory and was first adopted for the community detection by Danon et al. in [6]. Lancichinetti et al. proposed in [15] a modification of the NMI measure able to compare graph partitions with overlapping communities. As argued by authors, although it does not reproduce exactly the same values as NMI, it is close. We will refer to this measure as cNMI. The cNMI was used as the similarity measure in comparative analysis of community detection algorithms in [14]. In order to be able to perform head-to-head

comparison of our approach with those evaluated in [14], we use NMI as well as cNMI in our experiments.

Greedy approaches to community detection. In the following, we discuss greedy algorithms for community detection with the pseudo-linear execution time. We discuss three algorithms, the Label Propagation by Raghavan et al. [19], heuristic method for modularity optimization by Bondel et al. [1] and multi-resolution community detection algorithm using Potts model proposed by Ronhovde and Nussinov [20] (we will refer to this algorithm as RN). Algorithms are similar in their basic operational principle, where the label propagation is a basic approach, the two other can be regarded as the extension and modification of the label propagation mechanism. Label Propagation algorithm [19] is based on the greedy assignment of a node to the community which contains the most of its neighbors. If several communities contain the same highest number of n's neighbors, the ties are broken uniformly at random. The algorithm is initialized by assigning unique labels to all nodes in the network. Labels are propagated through the network in iterations in which the community membership of all the nodes are updated in random order. The process should continue until no node changes its label during iteration. As the convergence of such a greedy approach might be hard to prove, one may use the constraint on the iterations number to ensure that the algorithm will stop. Resulting communities are created from the nodes with the same labels. This approach, as the result of randomizations, does not have a unique solution. The algorithm might reach the stop criterion for multiple different partitions of the network. Authors also propose to aggregate multiple different partition of the network by creating new labels for nodes based on the labels they received in different runs and re-run the algorithm on that initial setting. The problem of label propagation approach is that it's solution often collapses into one single community, in case the boundaries between communities are not clearly defined. Processing similar to label propagation is used by Bondel et al. in [1], where the computation is done in two phases. The first phase is similar to label propagation, with the difference in the greedy step, where the authors choose the community to join based on the gain of modularity. The second phase of their algorithm consist of contracting partition into a new network. Those two phases are repeated iteratively until no gain in modularity can be achieved. Ronhovde and Nussinov in [20] use the processing of the label propagation style, with different decision function for changing the node's community membership. Their decision function, referred to as absolute Potts model (APM), can be parameterized to produce communities at different resolutions. Their proposed multi-resolution algorithm computes partition at different resolutions and they compute correlation among multiple partitions, identifying significant structures by strong correlations.

4 Size Constrained Greedy Community Detection

In this section, we propose a new community detection algorithm that allows a user to constrain the size of the communities being generated. It is named Size Constrained Greedy Community Detection algorithm (SizConCD). First, we describe the motivation for the work. We then provide the formalization of the problem, which correspond to our perception of the communities and the community structure (Subsection 4.1). and we provide the description of the proposed greedy algorithm.

Our work was motivated by the limitations of existing approaches to the problem of the community detection. In our view, there are the two major limitations. The first is the computational complexity of majority of methods which prevents them to be applied on large networks. The second is that algorithms with pseudo-linear computational time, which can be used on large networks, often produce partitions with very large communities. This is not very useful for the detailed analysis of a node (e.g. a community of 100 000 nodes is not particularly useful when one wants to identify social group of a user in a social network). Let us provide an illustrative example. We have tried to detect the community structure of the Wikipedia link graph. Our expectation was that semantically similar topics should be grouped together in communities. The size of the link graph extracted from Wikipedia XML dump was 3.1 million nodes and 91 million edges. The large size of the network limited our choice of a community detection algorithm only to those running in linear time. We have analyzed the link graph using Label Propagation [19] algorithm and the greedy modularity optimization [1]. The community structure produced by label propagation algorithm had the largest community with over 2.96 million of nodes. The size of the largest community in the partition identified by the greedy modularity optimization method was smaller (containing around 400 000); however, 20 largest communities of this partition comprised more than 95% of the nodes. In general, producing the partition with communities of limited sizes is useful for certain applications; e.g. when the community structure is analyzed by a human expert (inspecting social community of a user in social network, inspecting related concepts in semantic network). Another example of usefulness of size-constrained communities is the task of splitting the network into k groups, minimizing the number of edges connecting them. This task is useful for example for partition of the graph data for distributed graph database. This motivated us to develop a new algorithm for the community detection, which would allow us to constrain the community sizes.

4.1 Problem Formalization

Currently, there is no consensus on the formalization of the community detection problem. Several definitions can be found in the literature; often, the problem is not formalized at all and the definition of the task is provided as an informal, intuitive description.

In our work we adopt the definition of the community proposed by Hu et al., in [12]. Informally, every node of a community C should have higher or equal number of edges connecting it with other nodes of C, than number of edges connecting it with other communities. Let $G = (V, E, f, w)$ be a graph, with nodes V, edges E and function $f : V \times V \rightarrow E$ defining the mapping between nodes and edges and $w : E \rightarrow \mathbb{R}$ be the function defining the weights of the edges.

We will use the term *affinity* of a node n towards a cluster C to denote the sum of weights of edges connecting n with nodes of the cluster C:

$$\mathit{aff}(n, C) = \sum_{i \in C} w(e_{n,i}) : e_{n,i} \in E$$

Let us consider an example graph in Figure 1 and let all edges have the weight of 1. The affinity of n towards cluster A is 1 ($\mathit{aff}(n, A) = 1$); $\mathit{aff}(n, B) = 2$ and $\mathit{aff}(n, C) = 3$.

Based on work in [12], we define *affinity partition* of a graph to be $\gamma = C_1, C_2, .., C_m$, such that

$$\bigcup_{k \in \langle 1,\ldots,m \rangle} C_k = V \qquad \text{and} \qquad \bigcap_{k \in \langle 1,\ldots,m \rangle} C_k = \emptyset$$

and

$$\forall j \in C_k, aff(j, C_k) \geq max\{aff(j, C_l), C_l \in \gamma\}$$

It is obvious that there is more than one *affinity partition* of a graph. In fact, trivial partitions (single community containing all the nodes and partition where each node is a member of different community) also comply with the definition of the *affinity partition*. The authors in [12] propose to favor the partition which minimize number of intra communities edges, and provide mathematic formulation of the criterion. We do not adopt this criterion, as it favor partition with small number of large communities. (E.g., it can be shown that for GN-benchmark graphs [9], using this criterion, we would favor the partition of two communities instead the canonical four communities partition.) We propose a new criterion to compare *affinity partitions*. It is based on the following: if a node has the highest value of affinity towards multiple communities it should be assigned to the community where the ratio of the affinity towards the community and the number of community members is highest. This means that it should be assigned to the smallest community (of the candidate communities). We can define the *compactness* of the community as: $compactness(C) = \sum_{n \in C} \frac{aff(n,C)}{|C|^2}$ and we can define average compactness of a community partition as: $avg\_copactness(\gamma) = \frac{\sum_{C \in \gamma} compactenss(C)}{|\gamma|}$. Thus, the goal of our work is to approximate the affinity partition γ of a given network with highest value of $avg\_compactness(\gamma)$.

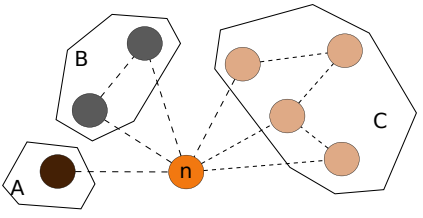


Fig. 1. Example of node's affinity towards clusters

4.2 SizConCD Algorithm

The label propagation algorithm uses the function identical to our definition of *affinity* of a node towards a community to resolve node's community membership in its greedy step. However, when the community structure is not very clear, the label propagation algorithm often fails, and produces a single community comprising all the nodes as a result. The reason is the following: let there be a canonical affinity partition of a network; if we modify this partition, e.g. by margining two communities into one, the balance of the affinity partition can be broken and applying the greedy label propagation algorithm on this setting can lead to collapsing, node after node, the solution into a single community. We want to keep low computational complexity of the greedy approach, but we want to prevent the collapse of the community structure being produced into a single community. To achieve that, we first build seed groups containing small number of nodes.

Algorithm 1 SizConCD function: ComputeOneLevel

Require: int *iteration_limit* > 0
Require: Graph G
Require: int *upper* {Upper limit for community size}
1: int *current_iteration* $= 0$
2: int *moves* $= -1$
{create partition where each node is in separate community}
3: Set of Set of Node C = *initialize_partition(G)*
4: **while** (*current_iteration* $<$ *iteration_limit*) $\vee$ (*moves* $= 0$) **do**
5: List l = *randomizeNumbering(G)* {create randomized list of nodes}
6: int *moves* $= 0$
7: **for** $\forall n \in l$ **do**
8: *moves* += *move(n, G, C, upper)* {greedy step - choose community for n}
9: **end for**
10: **end while**
11: **return** C

Building seed groups. We first introduce the basic mechanics of the algorithm and than describe it in a step by step manner. Basic procedure of the algorithm is the iteration over all nodes of the graph; for every node we consider all of its neighboring communities and select the best one to join. To identify the best cluster for node n, we define a *gain* function *seed_gain*(n, C). The gain function in seed groups building phase is:

$$\textit{seed\_gain}(n, C) = \textit{aff}(n, C) \times \log\left(\frac{\textit{UpperLimit}}{|C|}\right) \tag{1}$$

where *UpperLimit* is the user provided upper limit for community sizes, this parameter is set in seed groups building phase to N (number of nodes in the graph); Once we compute the gain function for all neighboring clusters, we choose to assign node n with the community with highest gain value. In case when multiple clusters have the same (highest) gain value, we choose the smallest one. If there is still multiple candidates, we break the ties uniformly at random. For the example depicted in Figure 1, given the *UpperLimit* $= 100$, the gain values in SizConCD algorithm are: *seed_gain*$(n, A) = 1.699$, *seed_gain*$(n, B) = 3.045$ and *seed_gain*$(n, C) = 3.903$. Thus, the node n would be assigned to community C in this particular iteration.

We now discuss the whole algorithm step by step. The algorithm is initialized by crating communities for all the nodes – i.e., each node is initially placed in a separate cluster. The processing is done in iterations (Algorithm 1).Each iteration begins by creating a list of nodes l and randomize their order in the list (line 5). We than traverse nodes in the list l and process each node separately (lines 7–9). The node processing is as follows (pseudocode is presented in Algorithm 2): we remove the node from it's original community (line 5) and compute the gain function for all the neighboring communities (lines 7–17). We assign the node to the community with the highest gain value (lines 18–19). If the community the node is assigned to is different from it's original community, we say that the node moves between communities (return value of Algorithm 2 indicates whether the node has moved). After iteration over the list l is finished, we decide whether we continue with another iteration or finish the computation.

Algorithm 2 SizConCD function: move

Require: Graph G
Require: Node $n \in G$
Require: Set of Set of Node C {graph partition}
Require: int *upper* {Upper limit for community size}
1: double *max_gain* $= 0$
2: Set of Node *orig_comm* = *getCommunityForNode(n, C)* {get community n belongs to}
3: int *orig_comm_id* = *getCommunityId(orig_comm)*
4: Set of Set of Node *candidate_comms* $= \emptyset$
5: *orig_comm* = *orig_comm* $\setminus n$ {First, remove n from original community}
6: Set of Set of Node *neighbor_comms* = *getNeighboringCommunities(n, G, C)* {get communities adjacent to n}
7: **for** $\forall comm \in$ *neighbor_comms* **do**
8: double *affinity* = *getAffinity(n, comm)* {compute affinity of n towards comm}
9: double *gain* = *affinity* $\times$ $log\left(\frac{upper}{|comm|}\right)$ {compute gain}
10: **if** *gain* = *max_gain* **then**
11: *candidate_comms* $=$ *candidate_comms* $\cup$ $comm$
12: **end if**
13: **if** *gain* > *max_gain* **then**
14: *max_gain* = *gain*
15: *candidate_comms* $= \{comm\}$
16: **end if**
17: **end for**
18: Set of Node *best_comm* = *selectCommunity(candidate_comms)* {select smallest community from the candidates}
19: *best_comm* = *best_comm* $\cup$ n {add node to the best fitted community}
20: **if** *getCommunityId(best_comm)* = *orig_comm_id* **then**
21: **return** 0
22: **end if**
23: **return** 1

The stop criteria are: a) no node has moved in the whole iteration; b) user specified maximum number of iteration has been reached. The condition b) is introduced to ensure the convergence of the process and avoid the oscillation of nodes between communities with equal gain value for the node (we remind that we break ties by randomly picking one of the communities with the highest score). In our experiments, we have set the maximum number of iteration to 25. The Algorithm 1 handles iterations and terminal conditions, while Algorithm 2 is the greedy step, where we select the community the given node will join, based on current state of the intermediate partition of the network. We omit the definitions of functions used only for manipulation with data structures, as we consider them to be quite simple and unnecessary for the comprehension of algorithm's mechanism. Considering the definition of the gain function, our expectation is to receive communities of smaller sizes and rather balanced in sizes. This expectation is based on modification of affinity introduced in *seed_gain* function.

Approximation of affinity partition and size constrained community detection. The algorithm for building seed groups is very accurate itself for identifying good approximations of the affinity partition of a network with small communities. However, it fails when the range in community sizes is high. The algorithm, in this case, identifies large number of small communities. On the other hand, if we use $aff(n, C)$ as a gain function instead of gain function 1, the algorithm collapses into a single community, when the community structure is not very clear. The following function was proposed as a gain function for size constrained community detection, which allows a user to impose constraint on the community sizes. Let *UpperLimit* be the desired size limit (user provided parameter).

$$sizcon\_gain(n, C) = \frac{aff(n, c)}{\lfloor \frac{|C|}{UpperLimit} \rfloor + 1}$$

The function returns value equal to $aff(n, c)$ for the groups smaller then *UpperLimit*; the gain of joining communities larger then *UpperLimit* is purposely lowered. When the user does not wish to constrain the size of the communities, he/she uses *UpperLimit* equal to N (number of node in the network).

Experimentation with the use of different gain functions lead us to the following solution, the SizConCD algorithm: we first build the seed groups as described in Subsection 4.2; we then continue in the iterations with altering gain functions, switching $sizcon\_gain(n, C)$ and $seed\_gain(n, C)$ as the gain function. The intuition is that use of $sizcon\_gain(n, C)$ as a gain function pushes the intermediate result towards the state of *affinity partition*, while the use of the $seed\_gain(n, C)$ prevents the procedure from collapsing the result into a single large community. This greedy heuristics leads to very good results on synthetic benchmark graphs.

Time complexity. We first express time complexity of a single iteration of the algorithm. Let n be the number of nodes and m be the number of edges. The randomization of the order in which the nodes are processed takes $O(n)$ steps. In the subsequent loop we process all the edges when computing the gain function for each node, taking $O(m)$ steps. The time complexity of an iteration is then $O(n + m)$. To ensure convergence of the algorithm, we use the limit on the number of iterations, a constant; we thus perform at most k iterations. This means that the resulting time complexity of the algorithm stays $O(n + m)$.

5 Evaluation

In this section, we report on our experiments with the proposed SizConCD algorithm. We first describe the benchmark used for the evaluation. Our experiments include evaluation on artificial benchmark graphs and experiments on the Wikipedia link graph as a real-world network. To be able to take advantage of the comparative analysis of community detection algorithms conducted by Lancichinetti et al. in [14], we have redone their experiments with the use of our method. This allows us a head-to-head comparison with a number of popular community detection algorithms. In our experiments, we

have used NMI similarity measure, which is dominant in the literature. We have used the cNMI measure as well , as it has been used in the comparative analysis paper. We perform thorough evaluation of our approach on networks with various properties.

5.1 Benchmarks

For research purposes, it is practical to compare results of community detection algorithms with a ground truth, analyzing networks with the known community structure. For those purposes, community detection benchmarks were proposed. Benchmarks generate artificial networks containing communities.

The most widely used approach for generating artificial networks with communities is *planted l-partition* model [5]. In this model, we generate defined number of groups of nodes; nodes are connected with probability of p_{in} to the other members of their group and with probability p_{out} to the nodes in other groups. Girvan and Newman used in their work [9] planted l-partition graphs with 128 nodes (each node having degree 16) divided into 4 disjoint communities, each having 32 nodes (GN-benchmark). This class of graphs becomes quite popular within community detection research, we refer to this type of benchmark graphs as GN-benchmark.

The criticism of the GN-benchmark is that all the nodes have the same degree and communities are of the same size, which makes them dissimilar to real-world networks where power-law distribution of node degrees and community sizes has been observed [2]. Lancichinetti et al. [13] have proposed LFR-benchmark that overcomes the drawbacks of GN-benchmark and generates networks with more realistic properties. LFR-benchmark is based on planted l-partition model; degrees of nodes and sizes of communities are assigned from a power law distribution, instead of probabilities p_{in}, p_{out} the mixing parameter μ is used. The value of μ defines the percentage of node's edges that connects it to the nodes outside it's own community. We believe that, to date, LFR-benchmark provides the most reliable way to test and compare community detection algorithms. Therefor, we have used the LFR-benchmark to evaluate the algorithm presented in this paper. Moreover, Lancichinetti et al. [14] have conducted a comparative analysis of several popular community detection algorithms on their benchmark. Thus, performing the evaluation on the graph with the same properties as the graphs used in [14] allows us the head-to-head comparison with numerous existing approaches to community detection.

5.2 Evaluation on the LFR Benchmark, on Small Undirected Graphs

We have performed experiments, using LFR-benchmark, with the proposed method on small undirected networks (1000 and 5000 nodes), with the same settings as in [14]. This benchmark test evaluates community detection methods on small undirected networks with small communities (10-50 nodes) and big communities (20-100 nodes). The results are depicted in Figure 2, each results represent the average of 100 trials, using SizConCD algorithm without constraining community sizes (upper limit: N).

SizConCD algorithm has clearly better performance on the networks with smaller communities. Surprising are consistently high values of NMI measure, even in case of network with high value of the mixing parameter (μ = 0.9). Those high values of NMI

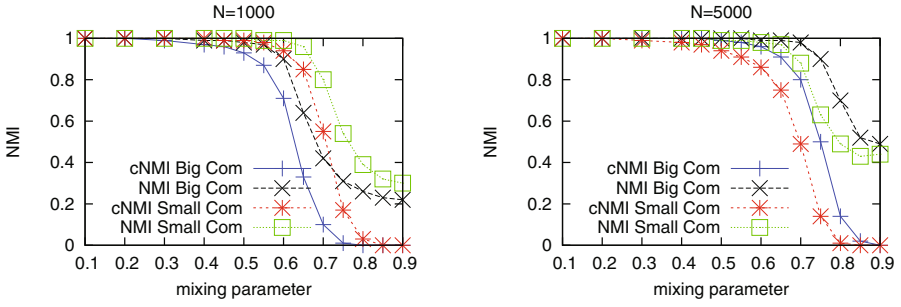


Fig. 2. LFR benchmark on undirected small graphs (1000 and 5000 nodes); small communities are of size 10–50, size of big communities ranges between 20–100

are caused by the bias of NMI measure towards partition with very small communities and do not indicate the accuracy of the proposed approach. SizConCD algorithm produced a large number of very small communities (3-5 nodes) when the community structure is very unclear (high values of μ); this is the cause of high NMI values. By comparing achieved results with the performance of those tested in [14], we can conclude that for LFR-benchmark on small undirected networks, the SizConCD has performance similar to the algorithm by Bondel et al. [1] and it is outperformed by algorithm introduced in [21] and RN [20] algorithms.

5.3 Evaluation on Small Directed Graphs

The next set of test was performed on small directed networks of LFR-benchmark. The experiment settings were identical to [14]. The results are depicted in Figure 3, using the SizConCD algorithm without size constraint (upper: N). We can observe better performance on networks with smaller communities. Comparison with benchmark results presented in [14] is rather favorable; in the comparative analysis paper, two algorithms were tested on benchmark for directed graphs – Infomap [21] and the modularity optimization via simulated annealing [11]. We observe higher values of cNMI for the partition produces by SizConCD than partitions by Infomap in all cases. Simulated annealing method has a slightly higher values for the case of 1000 nodes network with big communities, approach proposed in this paper is better in other settings. As the simulated annealing method optimizes the modularity, its results are rather poor for the case of 5000 node network with small communities due to the resolution limit of modularity. We can thus conclude that SizConCD method achieved the best results on this setting.

5.4 Evaluation on Large Undirected Graphs with Wide Range of Communities

In this experiment, we verify the performance of the algorithm on large networks of 50k and 100k nodes (community sizes: 20-1000, node degrees: 20-200). The results are shown in Figure 4(a). Again, the comparison of the proposed SizConCD algorithm with

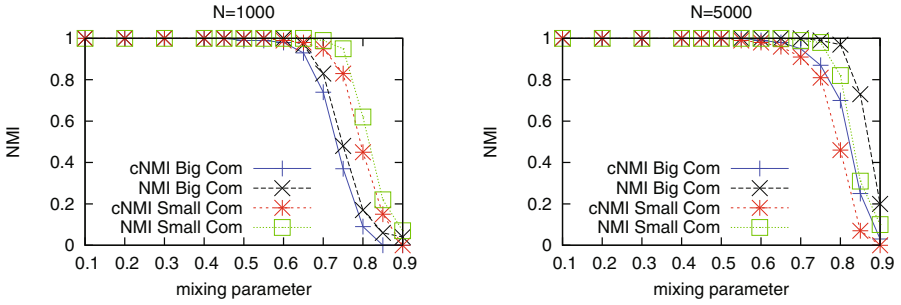
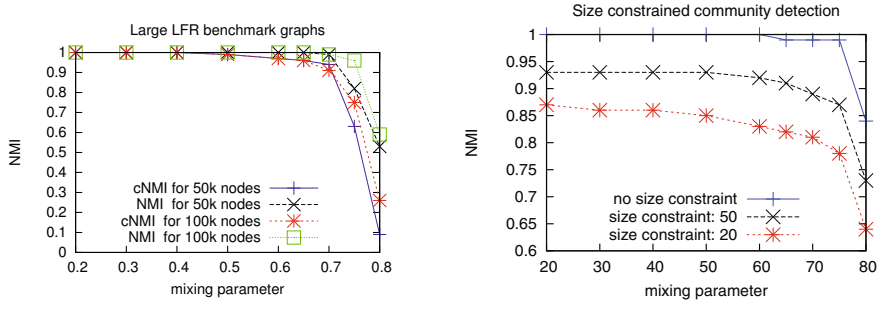


Fig. 3. LFR benchmark on directed small graphs (1000 and 5000 nodes); small communities are of size 10–50, size of big communities ranges between 20–100.



(a) Results on large LFR-benchmark graphs (b) Constraining the size of communities

Fig. 4. Left: On large LFR-benchmark graphs; Right: Effect of constraining the community size

the algorithms tested in [14] is rather favorable. Our approach achieves better cNMI values than all the algorithms tested in the comparative analysis study.

5.5 Applying Size Constraint

To evaluate the results of the SizConCD algorithm when constraining the community size, we have performed following experiment. Using LFR-benchmark, we have generated directed graphs of 10000 nodes, containing 10 communities, each of 100 nodes; the average node degree was set to 20, maximal,degree set to 50. We have then run the SizConCD algorithm a) without the size constraint (as a baseline), b) with the size constraint set to 50 nodes and c) with the size constraint set to 25 nodes. The results are depicted in Figure 4(b). High values of NMI indicates that algorithm constructs meaningful groupings when the size of generated communities is constraint.

5.6 Detecting Communities in Wikipedia

With promising results achieved on artificial networks, we wanted to apply the SizConCD algorithm on a real-world network. We have used proposed algorithm to cluster

the link graph of Wikipedia, where each node represents an article and edges represent hyperlinks between articles. We obtain the link graph by processing Wikipedia XML dump (from November 2009) by a custom script. Links to redirect pages were replaced by links to targets of the redirects. Resulting link graph contains 3.1 million nodes and 91 million edges. Thanks to the low computational complexity of SizConCD algorithm, we were able to analyze the graph of this size. We have first run the SizConCD algorithm on Wikipedia link graph without constraining the size of the communities. The resulting partition had 51 communities with more than 10000 nodes, with the largest community containing 170000 nodes. Nodes had on average 34% of edges going out of the community. We have rerun the experiments, setting upper size constraint to 100. The resulting partition had the largest community of 1900 nodes, nodes had on average 56% of edges going out of the community. After clustering Wikipedia link graph, one would expect nodes in communities to be somehow semantically related. As an example we provide the listing of titles of article sharing the community with article on 'Community structure'. We consider this clustering to be semantically correct, containing semantically close concepts. The members of the community are the following: *Duncan J. Watts, Six Degrees: The Science of a Connected Age, Simon model, Clustering coefficient, Modularity (networks), Random regular graph, Random graph, Preferential attachment, Luciano Pietronero, Watts and Strogatz model, Generalized scale-free model, Mixing patterns, Shlomo Havlin, Sexual network, Community structure, Small world experiment, Social-circles network model, Assortativity, Steven Strogatz, Average path length, Copying mechanism, Countability, Adilson E. Motter, Scale-free network, Degree distribution, Complex network zeta function, Fitness model (network theory), Reciprocity in network, Mark Newman, Giant component, Guido Caldarelli, Fitness model, Assortative mixing, Shortcut model, Triadic closure, Derek J. de Solla Price, Fractal dimension on networks, Erdős-Rényi model, Complex network, Small-world network, Barabási-Albert model, Eli Upfal.*

6 Conclusion

We have proposed a new algorithm for the community detection in networks. A notable feature of the proposed algorithm is that a user can constrain the size of the communities being generated, which might be a practical feature for a number of applications. The algorithm has a pseudo-linear time complexity which makes it applicable also to large networks. Recent work on the LFR-benchmark for community detection algorithms allowed us to perform thorough evaluation of the performance of the proposed approach. The comparative analysis of community detection algorithms on the LFR-benchmark enabled direct, head-to-head comparison of our approach with other community detection algorithms. Evaluation showed very competitive performance of the proposed algorithm, which outperformed other approaches in several of the benchmarks.

As the algorithm has the potential to identify communities at different size resolutions (by varying size limit), the direction for the future work is to extend the algorithm for hierarchical clustering. Another direction for the future work is to identify multi-community membership of the nodes as a post-processing step of the algorithm.

References

1. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment (10) (2008)
2. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.-U.: Complex networks: Structure and dynamics. Physics Reports 424(4-5), 175–308 (2006)
3. Boettcher, S., Percus, A.G.: Optimization with extremal dynamics. Complex Adaptive systems: Part I 8(2), 57–62 (2002)
4. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E 70(6), 66111 (2004)
5. Condon, A., Karp, R.M.: Algorithms for graph partitioning on the planted partition model. Random Struct. Algorithms 18(2), 116–140 (2001)
6. Danon, L., Duch, J., Diaz-Guilera, A., Arenas, A.: Comparing community structure identification. Journal of Statistical Mechanics: Theory and Experiment (October 2005)
7. Fortunato, S.: Community detection in graphs. Physics Reports 486(3-5), 75–174 (2010)
8. Fortunato, S., Barthélemy, M.: Resolution limit in community detection. Proceedings of the National Academy of Sciences of the United States of America 104(1), 36–41 (2007)
9. Girvan, M., Newman, M.E.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA 99(12), 7821–7826 (2002)
10. Gregory, S.: A fast algorithm to find overlapping communities in networks. In: Daelemans, W., Goethals, B., Morik, K. (eds.) PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 408–423. Springer, Heidelberg (2008)
11. Guimera, R., Amaral, L.A.N.: Functional cartography of complex metabolic networks. Nature 433(7028), 895–900 (2005)
12. Hu, Y., Chen, H., Zhang, P., Li, M., Di, Z., Fan, Y.: Comparative definition of community and corresponding identifying algorithm. Phys. Rev. E 78(2), 26121 (2008)
13. Lancichinetti, A., Fortunato, S.: Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. Phys. Rev. E 80(1) (2009)
14. Lancichinetti, A., Fortunato, S.: Community detection algorithms: A comparative analysis. Phys. Rev. E 80(5), 56117 (2009)
15. Lancichinetti, A., Fortunato, S., Kertész, J.: Detecting the overlapping and hierarchical community structure in complex networks. New Journal of Physics 11(3), 33015 (2009)
16. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E 69(2), 26113 (2004)
17. Papadimitriou, S., Sun, J., Faloutsos, C., Yu, P.S.: Hierarchical, parameter-free community discovery. In: Daelemans, W., Goethals, B., Morik, K. (eds.) PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 170–187. Springer, Heidelberg (2008)
18. Porter, M.A., Onnela, J.-P., Mucha, P.J.: Communities in networks. CoRR, abs/0902.3788 (2009)
19. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E 76(3), 36106 (2007)
20. Ronhovde, P., Nussinov, Z.: Multiresolution community detection for megascale networks by information-based replica correlations. Phys. Rev. E 80(1), 16109 (2009)
21. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences 105(4), 1118–1123 (2008)
22. Yang, T., Jin, R., Chi, Y., Zhu, S.: Combining link and content for community detection: a discriminative approach. In: KDD 2009: Proceedings of the 15th ACM SIGKDD, pp. 927–936. ACM, New York (2009)
23. Zhang, Y., Wang, J., Wang, Y., Zhou, L.: Parallel community detection on large networks with propinquity dynamics. In: KDD 2009: Proceedings of the 15th ACM SIGKDD, pp. 997–1006. ACM, New York (2009)