igraph-community.js—An open-source library for community detection based on partial information

1st Artur Myszkowski
University of Warsaw
Warsaw, Poland
0000-0001-5473-3398

2nd Dorota Celińska-Kopczyńska
University of Warsaw
Warsaw, Poland
0000-0001-5910-0039

3rd Marcin Waniek
New York University Abu Dhabi
Abu Dhabi, UAE
mjwaniek@nyu.edu

4th Paweł Gołąb
University of Warsaw
Warsaw, Poland
0000-0002-2051-1292

5th Talal Rahwan
New York University Abu Dhabi
Abu Dhabi, UAE
tr72@nyu.edu

6th Tomasz P. Michalak
University of Warsaw
and IDEAS NCBR
Warsaw, Poland
tpm@mimuw.edu.pl

***Abstract*—While there are many community detection algorithms in the literature, they assume that there is no prior information about what community the node belongs to. However, there are real-life scenarios in which we have partial knowledge about the community membership, e.g., based on the information disclosed by social media users. Starting from scratch in such a situation may result in unnecessary loss of information, eventually leading to potentially improper assignments to communities.**

In this paper, we introduce refinements to three well-known community detection algorithms (Clauset-Newman-Moore, Louvain, and Girvan-Newman) that allow us to find communities based on partial information. We implement our algorithms in the form of an open-source JavaScript library called *igraph-community.js*. Not only does this library include our proposed algorithms, but it also comes with many of the widely-used functionalities of the *igraph* package in the form of JavaScript. Moreover, we develop a graphical user interface (GUI) that facilitates the visualization and analysis of social networks at a scale.

***Index Terms*—community detection, network analysis, prior information**

I. Introduction

Communities can be understood as subsets of vertices that are more densely connected to each other than to the rest of the network [1], [2]. The challenging task of discovering and analyzing the community structure of a network has attracted significant attention in the literature due to its numerous applications in academia and industry. In academia, community detection belongs to the class of fundamental social network analysis tools that are applicable in various disciplines, including medicine [3], computer science [4], and biology [5], just to name a few. In industry, community detection can, for instance, be used to improve the performance of an advertising campaign by targeting the most influential users of each community and exploiting the word-of-mouth effect [6]. Similarly, the knowledge of the affiliations of users to communities may significantly enhance the effectiveness of recommendations [7].

This paper was supported by the Polish National Science Centre grant 2016/23/B/ST6/03599.

There are many community detection algorithms in the literature that use various techniques. Some follow a greedy approach to approximate the quality of the community structure [8], [9]. Others are based on statistical mechanics [10], label propagation [11], random walk [12], simulated annealing [13], and the spectral analysis of the adjacency matrix [14], among others.

The common feature of nearly all of these algorithms is that they divide the network into communities from scratch, i.e., without any *a priori* assumptions about which nodes should be assigned to which communities. However, such an approach may be inadequate in many practical applications, and may even yield erroneous results. Consider, for instance, the application mentioned earlier in which the community membership of social network users are leveraged to provide them with improved recommendations based on common interests shared with their acquaintances [7]. Users of real-world social networks often publicly disclose their community membership (e.g., the university they attend or their place of employment [15]). By neglecting this information, the existing community detection algorithms may wrongly assign some nodes, even when information about their ground-truth community is readily available. We will often refer to the known ground-truth communities of some nodes as *source communities*.

Our paper addresses this limitation by proposing refinements of three well-known community detection algorithms, proposed earlier by Clauset-Newman-Moore [8], Louvain [9], and Girvan-Newman [16]. The key advantage of our refinements is that they ensure that the resulting community structure correctly clusters those nodes about which the community affiliations is known *a priori*. We also show that this information can help improve the quality of the identified community structure.

We implement our algorithms in the form of an open-source JavaScript library *igraph-community.js*, available at https://github.com/rthrs/igraph-community.js. In addition to providing access to the new algorithms presented in this work, our library also allows using many functionalities of the well-known *igraph* package [17], in the form of JavaScript. This includes a rich set of popular community detection algorithms in the literature, not yet available in JavaScript. The proposed library is also very efficient; compiling C/C++ code into JavaScript-interpreted WebAssembly modules makes it the performance up to 16 times faster compared to a pure JavaScript implementation, depending on the hardware and browser.

Furthermore, we created an online user interface in which one can provide the information about the community affiliation for the chosen nodes and then run the refined algorithms. Thanks to *WebGL*—a demanding but efficient graphics rendering engine for web browsers—the interface allows for visualizing large networks (for instance, up to half a million edges on a laptop equipped with an integrated graphics card). Hence, its capabilities are an order of magnitude better than alternative network analysis tools [1]. Furthermore, *Web Workers* technology makes the interface responsive, even in the case of long-lasting calculations.

Finally, we present numerical simulations, analyzing the refined algorithms. We examine in particular how the size and the composition of the group of nodes with an *a priori* known assignment to communities affect the performance. Based on the outcome of these analyses, one can determine whether it is worthwhile to invest resources in search for the ground-truth information regarding some nodes to improve the precision of the community detection algorithms.

The remainder of this paper is organized as follows. In the next section, we discuss related works. Next, we introduce the necessary notation used in the paper. In Section IV, we formally define the problem of community detection with partial knowledge and, in Section V, we present the refined algorithms, including their theoretical underpinnings. We then introduce the *igraph-community.js* library in Section VI. Sections VII and VIII present the setup and the results of numerical simulations. Conclusions follow.

II. Related Work

The literature on community detection with *a priori* affiliations knowledge is significantly smaller than the one that does not make any assumptions about the ground truth. Sun and Wang [18] present a partially supervised machine learning algorithm based on partial knowledge of whether a given pair of vertices should belong to one cluster or not. The general reasoning of the method can be summarized by two rules: *"my friend's friend is my friend"* and *"my enemy's friend is my enemy"*. The authors consider three algorithms: the non-negative matrix factorization [19] method, the spectral clustering [20] and the Infomap [13] algorithm. In our work we consider a different kind of available information—fragments of ground truth communities rather than pairwise information about nodes.

Tran et al. [21] consider networks where it is explicitly known that some nodes and edges are missing. The authors introduce community detection through the non-negative matrix factorization to solve this problem. The missing part of the network is estimated based on the Kronecker model. Similarly, Dahlin and Svenson [22] consider community detection algorithm for networks where edges are assigned a probability of existence. The method is based on sampling from a set of networks consistent with the available information. We do not consider missing information but rather partial knowledge about the ground truth community structure in our work.

The most popular approach to the problem of community detection when partial groups are known is *seed-set expansion*. The method is based on identifying central nodes (so-called *seed selection*), finding local communities for these particular nodes, and finally building the community structure based on the found communities. The central nodes can be selected randomly; centrality measures such as degree, closeness, and PageRank are typically employed to this end [23]. Expanding communities can be based on using the PageRank algorithm to find regions likely to be reached from the seed nodes [24], [25], adding nodes that improve the split quality measure [26], [27] modularity, or optimizing the number of neighbors inside and outside the node's community. Many algorithms using the seed-set expansion technique assume that the resulting communities can overlap, i.e., a given node can belong to more than one community [23], [28]–[30]. Kloumann and Kleinberg [31] consider a problem where the goal is to identify nodes belonging to known source communities, using the seed-set expansion technique with the PageRank-based community detection algorithm. However, the algorithms proposed in our work are not based on the seed-set expansion technique but instead on modularity optimization with the use of partial source communities.

III. Preliminaries

Let $G = (V, E)$ be a *network*, where V is a set of n nodes and E is a set of m edges. In this work we only consider *undirected* networks, where each edge is an unordered pair of nodes $\{u, v\}$, such that $u, v \in V$ and $u \neq v$. Notice that the last condition implies that we consider networks without any self-loops. We denote by $N(v)$ the set of *neighbors* of v in G, i.e., $N(v) = \{w \in V : \{v, w\} \in E\}$. The *degree* of v is denoted by $d(v)$, i.e., $d(v) = |N(v)|$.

The *community structure* is a partition of the network's set of nodes into disjoint subsets or *communities*. Formally, a community structure $\mathcal{C}$ is a set $\{C_1, \ldots, C_k\}$ that satisfies the following conditions: $\forall_i \, C_i \subseteq V$, $\bigcup_i C_i = V$, and $\forall_{i \neq j} \, C_i \cap C_j = \emptyset$. We denote by $\mathcal{C}(v)$ the community in $\mathcal{C}$ containing node v, i.e., $\mathcal{C}(v) = C_i \in \mathcal{C} : v \in C_i$.

The *community detection* problem is the task of generating the community structure based on the structure of a given network $G = (V, E)$. While there are many different ways of evaluating the quality of any given community structure, the most popular measure is modularity [16]. Intuitively, modularity prefers community structures with many intra-community edges (edges between members of the same community), and few inter-community edges (edges between members

[1]e.g. with http://networkrepository.com/graphvis.php

of different communities). Formally the modularity Q of a community structure $\{C_1, \ldots, C_k\}$ is defined as:

$$Q(\{C_1, \ldots, C_k\}) = \sum_{i=1}^{k} e_{ii} - a_i^2 \quad (1)$$

where e_{ij} is the fraction of edges connecting nodes from community C_i to the nodes in community C_j, i.e.:

$$e_{ij} = \frac{|\{v, v'\} \in E : v \in C_i \wedge v' \in C_j|}{m}$$

and a_i is the fraction of edges incident to the nodes from community C_i, i.e.:

$$a_i = \frac{|\{v, v'\} \in E : v \in C_i \vee v' \in C_i|}{m}.$$

Typically, the goal of the *community detection algorithms*, i.e., algorithms solving the community detection problem, is to maximize the modularity of the resulting community structure. In this work we consider the following community detection algorithms:

- **Clauset-Newman-Moore (CNM) algorithm** [8]—an algorithm based on greedy optimization of the modularity. It starts with a community structure consisting of singletons, i.e., each node forms a separate community. The algorithm then iteratively merges two communities that provide the greatest increase of modularity, until either all communities are merged, or none of the possible merges increases modularity.
- **Louvain algorithm** [9]—another algorithm based on greedy optimization of modularity. The Louvain algorithm starts with a community structures consisting of singletons, and moves the nodes to neighboring communities based on the greatest increase of modularity. When it reaches the local maximum, i.e., the moment when no moves offer better modularity, the Louvain algorithm melds all nodes in each community into a single node, and the process continues on this new network.
- **Girvan-Newman (GN) algorithm** [16]— algorithm iteratively removes from the network edges with the greatest *betweenness* value (edges that belong to a largest number of shortest paths between nodes), as the probable candidates for edges connecting communities. Each time the network splits into more connected components, they are marked as separate communities, and the process continues until all edges are removed from the network. Out of all the community structures found by this process, the one with the greatest modularity is returned as a result.

Table I summarizes the notation used in this paper.

TABLE I
THE SUMMARY OF THE KEY NOTATION

G	Undirected network
V	The set of nodes of the network
E	The set of edges of the network
M	Incidence matrix of the network
n	The number of nodes of the network
m	The number of network's edges
$N(v)$	The set of neighbors of node v
$d(v)$	The degree of node v
$C_i \subseteq V$	Community in the network
$\mathcal{C}$	Community structure of the network
$\mathcal{C}(v)$	the community in $\mathcal{C}$ containing node v
Q	Modularity, see formula 1
a_i	Fraction of all edges incident to nodes from C_i
e_{ij}	A fraction of edges connecting the nodes from C_i to the nodes from C_j
ΔQ_{ij}	Modularity change resulting from merging the communities C_i and C_j
$c_B(e)$	Edge betweenness centrality of the edge e
$v \mapsto_C C_i$	Modified community structure C where we move node v to community C_i.
CNM	Clauset-Newman-Moore algorithm for community detection, see Section III
CNM$_{\text{seed}}$	The modified Clauset-Newman-Moore algorithm for community detection with prior knowledge, see Algorithm 1
Louvain	The Louvain algorithm for community detection, see Section III
Louvain$_{\text{seed}}$	The modified Louvain algorithm for community detection with prior knowledge, see Algorithm 2
GN	The Girvan-Newman algorithm for community detection, see Section III
GN$_{\text{seed}}$	The modified Girvan-Newman algorithm for community detection with prior knowledge, see Algorithm 3

IV. PROBLEM DEFINITION

As mentioned in the previous section, most community detection algorithms' primary focus is identifying a community structure with maximal modularity for a given network. Nevertheless, in some situations, we might not be interested in high modularity but rather in the result of the community detection algorithm being as close as possible to certain *ground truth*, i.e., a particular division of the network of which we only have fragmentary knowledge. Notice that this ground truth division does not necessarily have to be the one with the greatest (or even particularly high) modularity.

However, none of the currently existing community detection algorithms allow us to take advantage of our *a priori* knowledge about the ground-truth community structure, as they take into consideration only the structure of the network. To fill this gap in the literature, we will design a modified version of each of the algorithms mentioned in Section III that utilizes knowledge about a given set of disjointed *source communities* $\mathcal{S} = \{C_1^{\mathcal{S}}, ..., C_k^{\mathcal{S}}\}$. In particular, we want the nodes belonging to the same source community to also belong to the same community in the outcome of our modified community detection algorithms. More formally, we will design our algorithms so that:

$$\forall_{C_i^{\mathcal{S}} \in \mathcal{S}} \exists_{C_j \in \mathcal{C}} C_i^{\mathcal{S}} \subseteq C_j$$

where $\mathcal{C} = \{C_1, ..., C_l\}$ is the outcome of the modified community detection algorithm. Notice that the condition allows for multiple source communities to be grouped into a single community, i.e., it may happen that

$$\exists_{C_i^{\mathcal{S}}, C_{i'}^{\mathcal{S}} \in \mathcal{S}} \exists_{C_j \in C} C_i^{\mathcal{S}} \subseteq C_j \wedge C_{i'}^{\mathcal{S}} \subseteq C_j.$$

As we focus on settings with a particular ground truth community structure, we will generally evaluate our algorithm based on how close the result is to said ground truth, rather than based on the modularity of the result. More formally, we define our problem as follows.

Definition 1 (Community Detection with Partial Knowledge)**.** *Given a tuple $(G, \mathcal{S}, \mathcal{T})$, where $G = (V, E)$ is a network, $\mathcal{S} = \{C^{\mathcal{S}}_1, \dots, C^{\mathcal{S}}_k\}$ is a set of disjoint source communities, and $\mathcal{T} = \{C^{\mathcal{T}}_1, \dots, C^{\mathcal{T}}_l\}$ is a ground truth community structure for G, identify a community structure $\mathcal{C} = \{C_1, \dots, C_w\}$ such that $\forall_{C^{\mathcal{S}}_i} \exists_{C_j \in \mathcal{C}} C^{\mathcal{S}}_i \subseteq C_j$ and $\gamma(\mathcal{C}, \mathcal{S})$ is maximal, where γ is a similarity measure between $\mathcal{C}$ and $\mathcal{T}$.*

V. The Refined Algorithms

Having defined the main problem of our study, we now propose algorithms explicitly designed to solve this problem. More specifically, we present a modified version of each of the three community detection algorithms discussed earlier to incorporate any partial *a priori* knowledge about the ground truth community structure.

A. Modified Clauset-Newman-Moore algorithm

In the standard Clauset-Newman-Moore algorithm the process begins with every node forming its own separate community. The nodes are then greedily merged according to the modularity improvement. The main idea behind our modification of the algorithm is to start the process with a community structure that is consistent with the *a priori* knowledge about source communities $\mathcal{S} = \{C^{\mathcal{S}}_1, \dots, C^{\mathcal{S}}_k\}$. Importantly, since Clauset-Newman-Moore algorithm only merges communities from the initial structure, but never splits them, we are guaranteed that the condition from Definition 1 is satisfied.

The pseudocode of the modified version of the Clauset-Newman-Moore algorithm is presented as Algorithm 1. In line 1 we create the initial community structure consistent with the prior knowledge. In lines 2-7 we compute the values of e_{ij}. While a naive implementation would have a complexity of $\mathcal{O}(n^2)$, thanks to the use of a hash map we are able to compute the values of e_{ij} in time $\mathcal{O}(m)$. In lines 8-9 we compute the values of a_i. Then, in lines 10-14 we compute the change in modularity ΔQ_{ij} after merging communities C_i and C_j. We compute the value according to the following lemma.

Lemma 1. *Let $\mathcal{C}$ be a community structure. The modularity change after merging two communities $C_i \in \mathcal{C}$ and $C_j \in \mathcal{C}$ is $\Delta Q_{ij} = e_{ij} - 2a_i a_j$.*

Proof. The modularity change after merging communities C_i and C_j is:

$$\Delta Q_{ij} = Q(\mathcal{C} \setminus \{C_i, C_j\} \cup \{C_i \cup C_j\}) - Q(\mathcal{C})$$

Notice that in the definition of modularity the value of $e_{ii} - a_i^2$ depends only on the nodes in community C_i, hence when computing ΔQ_{ij} the values for communities other than C_i and C_j will cancel out. Let $\hat{Q}^{\mathcal{C}}_i = e_{ii} - a_i^2$ be the contribution of community C_i to modularity $Q(\mathcal{C})$. We then have:

Algorithm 1 The CNM$_{\text{seed}}$ algorithm

```
Input: The network $G = (V, E)$, the source communities $\mathcal{S} = \{C^{\mathcal{S}}_1, \dots, C^{\mathcal{S}}_k\}$.
1:  $\mathcal{C} \leftarrow \mathcal{S} \cup \left\{\{v\} : v \in V \wedge \forall_{C^{\mathcal{S}}_i \in \mathcal{S}}\, v \notin C^{\mathcal{S}}_i\right\}$
2:  $e_{ij} \leftarrow$ EmptyHashMap()
3:  for $\{v, w\} \in E$ do
4:      $(C_i, C_j) \leftarrow (\mathcal{C}(v), \mathcal{C}(w))$
5:      if $e_{ij}$ does not exist then
6:          $e_{ij} \leftarrow 0$
7:      $e_{ij} \leftarrow e_{ij} + \frac{1}{m}$
8:  for $C_i \in \mathcal{C}$ do
9:      $a_i \leftarrow \frac{1}{2m} \sum_{v \in C_i} d(v)$
10: for $C_i, C_j \in \mathcal{C}$ do
11:     if $\exists_{v \in C_i, w \in C_j} \{v, w\} \in E$ then
12:         $\Delta Q_{ij} \leftarrow e_{ij} - 2a_i a_j$
13:     else
14:         $\Delta Q_{ij} \leftarrow 0$
15: $H \leftarrow$ InitHeap($G$, $\Delta Q$)
16: while $|\mathcal{C}| > 1$ do
17:     $i, j \leftarrow \arg\max_{(i,j) \in H} \Delta Q_{ij}$
18:     if $\Delta Q_{ij} > 0$ then
19:         $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C_i, C_j\} \cup \{C_i \cup C_j\}$
20:         $\Delta Q \leftarrow$ UpdateMatrix($G$, $\Delta Q$, $i$, $j$)
21:         $H \leftarrow$ UpdateHeap($\Delta Q$)
22:         $a_j \leftarrow a_i + a_j$
23:     else
24:         break
25: return $\mathcal{C}$
```

$$\begin{aligned}\Delta Q_{ij} =& [\underbrace{(e_{ii} + e_{jj} + e_{ij})}_{(*)} - (\underbrace{a_i + a_j}_{(**)})^2] - [\hat{Q}^{\mathcal{C}}_i + \hat{Q}^{\mathcal{C}}_j] \\ =& [e_{ii} + e_{jj} + e_{ij} - (a_i^2 + 2a_i a_j + a_j^2)] \\ & - [e_{ii} - a_i^2 + e_{jj} - a_j^2] \\ =& \cancel{e_{ii}} + e_{ij} + \cancel{e_{jj}} - \cancel{a_i^2} - 2a_i a_j \\ & - \cancel{a_j^2} - \cancel{e_{ii}} + \cancel{a_i^2} - \cancel{e_{jj}} + \cancel{a_j^2} \\ =& e_{ij} - 2a_i a_j,\end{aligned}$$

where $(*)$ is the fraction of edges within communities C_i and C_j, as well as between the communities C_i and C_j, while $(**)$ is the fraction of edges incident with node in communities C_i and C_j. □

Finally, in lines 15-25 we continue the process according to the normal course of the standard Clauset-Newman-Moore algorithm, see Clauset et al. [8] for more details. The time complexity of the algorithm is $\mathcal{O}(mn \log n)$—the same as the standard version.

It is worth noting that Algorithm 1 is an *anytime* algorithm, i.e., its solution quality grows monotonically over time. Hence, if it is forced to terminate prematurely, it would still produce a valid (albeit not necessarily optimal) outcome.

Algorithm 2 The Louvain$_{\text{seed}}$ algorithm

```
Input: The network $G = (V, E)$, the source communities $\mathcal{S} = \{C_1^{\mathcal{S}}, \dots, C_k^{\mathcal{S}}\}$.
 1: $\mathcal{C} \leftarrow \mathcal{S} \cup \left\{\{v\} : v \in V \wedge \forall_{C_i^{\mathcal{S}} \in \mathcal{S}}\, v \notin C_i^{\mathcal{S}}\right\}$
 2: $(V^*, E^*) \leftarrow (V, E)$
 3: repeat
 4:     $(V^*, E^*) \leftarrow (\mathcal{C},$
            $\{\{C_i, C_j\} \subset \mathcal{C} : \exists_{\{v,w\} \in E^*} v \in C_i \wedge w \in C_j\})$
 5:     $\mathcal{C} \leftarrow \{\{v\} : v \in V^*\}$
 6:     $\pi \leftarrow$ SetNodesFirstPermutation($V^*$)
 7:     $\mathcal{C} \leftarrow$ MoveNodes($(V^*, E^*)$, $\mathcal{C}$, $\pi$)
 8: until $|\mathcal{C}| = |V^*|$
 9: return $\{\text{Flatten}(C_i) : C_i \in \mathcal{C}\}$

10: function MoveNodes(The network $G = (V, E)$, the community structure $\mathcal{P}$, the node permutation $\pi$)
11:     repeat
12:         $Q_{old} \leftarrow Q(\mathcal{P})$
13:         for $v \in \pi$ do
14:             $\mathcal{C} \leftarrow \underset{C' \in \mathcal{P} : \exists_{w \in C'} \{v,w\} \in E}{\arg\max} \Delta Q(v \mapsto_{\mathcal{P}} C')$
15:             if $\Delta Q(v \mapsto_{\mathcal{P}} \mathcal{C}) > 0$ then
16:                 $\mathcal{P} \leftarrow v \mapsto_{\mathcal{P}} \mathcal{C}$;
17:     until $Q(\mathcal{P}) = Q_{old}$
18:     return $\mathcal{P}$

19: function Flatten(Set $S$)
20:     if IsSet($S$) then
21:         return $S$
22:     else
23:         return $\bigcup_{s \in S}$ Flatten($s$)
```

B. Modified Louvain algorithm

The standard Louvain algorithm begins with a community structure consisting of singletons, and gradually merges them based on the greatest increase of modularity. The main idea behind our modified version of the algorithm is to start with the *a priori* source communities already merged together.

The pseudocode of the modified version of the Louvain algorithm is presented as Algorithm 2. We begin by initializing a community structure consistent with the prior knowledge about the ground truth in line 1. The main loop of the algorithm consists of lines 3-8. In line 4 we create the graph whose nodes correspond to communities of the best community structure found so far, while in line 5 we set the current community structure to the singletons of the nodes of the new network. Later on, in line 6 we randomly select a permutation of the nodes with elements consisting of merged nodes at the beginning. In line 7 we move nodes between communities by calling the function defined in lines 10-18. Finally, we return the result constructed using the function defined in lines 19-23, which recursively unpacks the previously merged nodes.

Algorithm 3 The GN$_{\text{seed}}$ algorithm

```
Input: The network $G = (V, E)$, the source communities $\mathcal{S} = \{C_1^{\mathcal{S}}, \dots, C_k^{\mathcal{S}}\}$.
 1: $R \leftarrow []$
 2: $\eta \leftarrow |\{\{v, w\} \in E : \exists_{C_i^{\mathcal{S}} \in \mathcal{S}} \{v, w\} \subseteq C_i^{\mathcal{S}}\}|$
 3: while $|E| > \eta$ do
 4:     $c_B \leftarrow$ EdgeBetwenness(G)
 5:     $\{v^*, w^*\} \leftarrow \underset{\{v,w\} \in E : \neg\exists_{C_i^{\mathcal{S}} \in \mathcal{S}} \{v,w\} \subseteq C_i^{\mathcal{S}}}{\arg\max} c_B(\{v, w\})$
 6:     $R \leftarrow [\{v^*, w^*\} \mid R]$;
 7:     $G \leftarrow (V, E \setminus \{v^*, w^*\})$
 8: $\mathcal{C} \leftarrow \mathcal{S} \cup \left\{\{v\} : v \in V \wedge \forall_{C_i^{\mathcal{S}} \in \mathcal{S}}\, v \notin C_i^{\mathcal{S}}\right\}$
 9: $\mathcal{C}^* \leftarrow \mathcal{C}$
10: while not Empty(R) do
11:     $\{v, w\} \leftarrow$ Head($R$)
12:     $R \leftarrow$ Tail($R$)
13:     $C_i \leftarrow \mathcal{C}(v)$
14:     $C_j \leftarrow \mathcal{C}(w)$
15:     if $C_i \neq C_j$ then
16:         $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C_i, C_j\} \cup \{C_i \cup C_j\}$
17:         if $Q(C) > Q(C^*)$ then
18:             $\mathcal{C}^* \leftarrow \mathcal{C}$
19: return $\mathcal{C}^*$
```

C. Modified Girvan-Newman algorithm

The standard Girvan-Newman community detection algorithm repeatedly removes edges from the graph and records the modularity of community structures corresponding to the sets of connected components of the network. The main idea behind our modification of the algorithm is that we do not allow the removal of edges between nodes belonging to the same *a priori* source community.

The pseudocode of the modified version of the Girvan-Newman algorithm is presented as Algorithm 3. The list R, initialized in line 1 collects edges being removed from the network. In line 2 we compute η, the number of edges inside the source communities, i.e., edges that will not be removed from the network. In the loop in lines 3-7 we iteratively remove from the network edges with the greatest edge betweenness centrality (see Newman and Girvan [16] for more details) that are not contained within the source communities. Then, in line 8 we initialize the current community structure $\mathcal{C}$ consistent with the source communities, while in line 9 we initialize the best found community structure $\mathcal{C}^*$. Finally, in lines 10-18 we analyze the community structures resulting from the removal of the edges collected in the list R, and select the community structure $\mathcal{C}^*$ with the greatest modularity, which is then returned in line 19. The time complexity of the algorithm is $\mathcal{O}(nm^2)$—the same as the standard version.

VI. System Overview

We implement the community detection algorithms with prior knowledge presented in the previous section as a JavaScript library called *igraph-community.js*. Besides providing access

TABLE II
COMPARISON OF ALGORITHMS AVAILABLE IN OUR LIBRARY, THE *igraph* LIBRARY, AND AVAILABLE JAVASCRIPT ALTERNATIVES.

Algorithm	Our library JavaScript	*igraph* C/C++	Alternative JavaScript
EdgeBetweenness [16] (GN)	✓	✓	✗
FastGreedy [8] (CNM)	✓	✓	✓ [34]
Infomap [13]	✓	✓	✓ [35]
LabelPropagation [11]	✓	✓	✓ [36]
LeadingEigenvector [14]	✓	✓	✗
Multilevel [9] (Louvain)	✓	✓	✓ [37]
Leiden [38]	✓	✓	✗
Optimal [1]	✓	✓	✗
Spinglass [10]	✓	✓	✗
Walktrap [12]	✓	✓	✗
FluidCommunities [39]	✗	✓	✗
CNMSeed (Algorithm 1)	✓	✗	✗
LouvainSeed (Algorithm 2)	✓	✗	✗
GNSeed (Algorithm 3)	✓	✗	✗

to the new algorithms presented in this work, our library also allows the use of many functionalities of the well-known *igraph* package [17] in JavaScript. Below we describe the implementation of the library, as well as a way to access it via a web interface.

A. JavaScript Library

In 2021 JavaScript maintains it's position of world's most popular programming language for a ninth year in a row according to the comprehensive Stack Overflow survey [32]. Although it is mainly used as an interface creation tool, its simplicity and universality prompts development teams to apply it in a wider range of scenarios. The source code of our library *igraph-community.js* is available on GitHub [33].

The core of our solution is the *igraph* package [17] implemented in C/C++. It is a collection of various network analysis tools, used both in industrial applications and in research [40]. In this work, we are particularly interested in its capability to perform community detection in networks.

We used the Emscripten tool [41] to compile the C/C++ source code to LLVM bytecode and then to WebAssembly [42]. The programs compiled using Emscripten can run on any JavaScript engine compatible with the EcmaScript 5 specification, i.e., on over 99 % of all browsers according to *Can I Use* data [43]. WebAssembly is an open, portable binary format for stacked virtual machines. Experiments show that decoding this format can be significantly faster than just parsing JavaScript code [44].

Using the above-mentioned tools we were able to port ten out of eleven community detection algorithms available in *igraph* to JavaScript. The only algorithm that we omitted is the *FluidCommunities* algorithm, as it requires knowledge of the number of communities in advance, which was unsuitable for our applications. We then added to the algorithms from *igraph* the implementations of our algorithms for community detection with partial knowledge. Table II provides an overview of the algorithms implemented in the *igraph-community.js* library. The table also provides information about the alternative implementations of community detection algorithms available in JavaScript.

B. Web Interface

While one can use our library directly in their JavaScript code, its functionality is also available via a web interface at https://network-centrality.com. The application works on most modern browsers, however, we suggest using one of the Chromium-family browsers for the best performance.

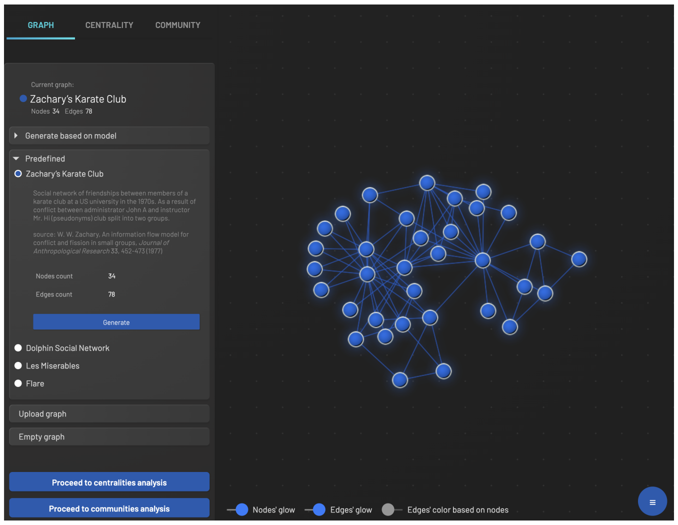
The user may choose to generate a network using three popular models, namely Erdős-Rényi [45], Barabási-Albert [46], and Watts-Strogatz [47]. The network to analyze can also be imported from a file in *tgf*, *GraphViz*, or *Pajek* format. Finally, the user can select one of multiple pre-defined networks.

The application allows the user to visualize the network, run community detection algorithms listed in Table II, and compute centrality measures, including Shapley value-based centrality [48]. In the implementation of the web application, we used the React library [49] supported by the Redux [50] state management software. For the network visualization we used WebGL [51] API. The network layouts are calculated using d3js [52] force-directed layout implementation. In order to enhance the visualisations for communities, we applied the forceInABox [53] library which guarantees a reasonable separation of the detected communities in the final layout. Thanks to several network rendering optimization, our web application allows for the visualization of a network with half a million edges on laptops with an integrated graphics card. As executing centrality measures and community detection algorithms may be computationally demanding, we decided to use WebWorkers [54] to parallelize these tasks so that the interface remains responsive even during heavy computations. Figure 1 presents an overview of our web interface, while Figure 2 presents an example of network visualization.

VII. EXPERIMENTAL SETUP

For the experimental evaluation, we utilize four popular real-world networks with known ground-truth: Zachary Karate Club [55] ($n = 34$, $m = 78$, $|\mathcal{C}| = 2$), Dolphins [56] ($n = 62$, $m = 159$, $|\mathcal{C}| = 2$), Football [57] ($n = 115$, $m = 613$, $|\mathcal{C}| = 12$), and Polbooks [58] ($n = 105$, $m = 441$, $|\mathcal{C}| = 3$). Arguably, these networks are the gold standard in the literature; nearly every paper on community detection uses at least one of them.

We may perceive community detection as a generalization of the clustering problem for graphs. To evaluate the results of our simulation, we need to generalize the standard terminology. We will check if the algorithms correctly separate or aggregate pairs of vertices into communities. True Positives (TP) denote correctly classified pairs of vertices from the same community. Analogously, the number of True Negatives (TN) measures the number of correctly separated pairs of vertices from different communities. A False Positive (FP) occurs when the algorithm predicts two vertices from different



Social network of friendships between members of a karate club at a US university in the 1970s. As a result of conflict between administrator John A and instructor Mr. Hi (pseudonyms) club split into two groups.

source: W. W. Zachary, An information flow model for conflict and fission in small groups, *Journal of Anthropological Research* 33, 452-473 (1977)

Fig. 1. An overview of the web interface.

An overview of the web interface

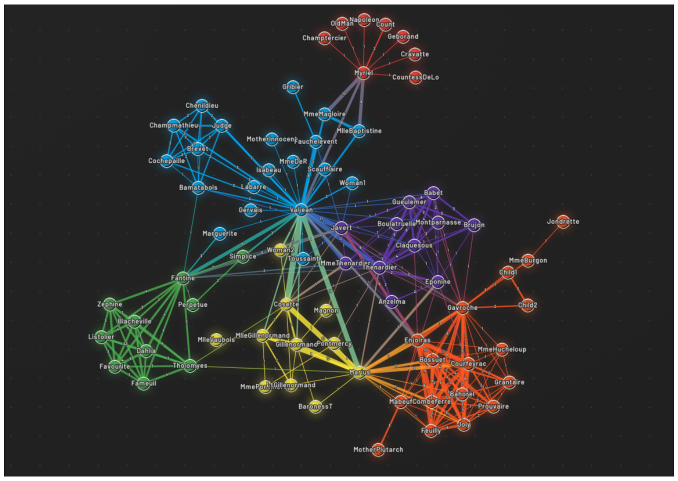


Fig. 2. A sample network visualization.

A sample network visualization

ground-truth communities to belong to the same community; it signals problems with separation. In contrast, a False Negative (FN) is when the algorithm assigns two vertices from the same ground-truth community to different communities, signalling problems with aggregation.

The setup of the experiments is as follows. We start with the base (no information at all) algorithms' clusterings, for which we find the sets of TP, TN, FP, and FN. Then, for every algorithm and every graph, we reveal partial information. First, we decide for how many communities we will reveal information (the number falls within the range from one to the number of ground-truth communities); then we decide what percentage of each source community nodes is revealed (5%, 10%, 25% or 50%). The revealed nodes can be selected randomly (we draw unconnected vertices) or from a consistent induced subgraph. In the second case, the vertices are selected using the BFS algorithm: priority is given to vertices with the least connections to ground-truth other than their own community. Then the degree of the vertex counts (the higher the better), and in the case of a tie, we randomize.

While drawing the revealed nodes, we also control for the composition factor—the percentage of vertices in the source community that were not TP for base algorithm: 0%, 25%, 50%, 75% and 100%. Note that it may not always be possible to select a sufficient number of vertices to match the declared composition factor, e.g., the TP set might not be sufficiently large. This is why we will rather use the real composition ratios than the declared ones. In extreme cases, no nodes were revealed; this happened in about 6,900 cases (3% of the original dataset). We decided to remove those observations from the analysis. For each combination (algorithm $\times$ graph $\times$ given number of communities revealed $\times$ given percentage of nodes in community revealed $\times$ composition factor) we repeat the simulation 100 times.

To evaluate the results, we will use the standard measurements of the quality of the division: **Adjusted Rand Index** (ARI), **Normalized Mutual Information** (NMI) [59], and the modularity of the division. Our implementation also returns data on the resulting classification and a graph that can be imported into our interface (see Section VI-B).

Let S be the set of n items. Clustering $\mathbf{U}$ of the set S is a way of dividing S into r disjoint subsets, where $\bigcup_{i=1}^{r} U_i = S$ and $U_i \cap U_j = \emptyset$. We would like to measure the overlapping information of two clusterings $\mathbf{U} = \{U_1, U_2, ..., U_n\}$ and $\mathbf{V} = \{V_1, V_2, ..., V_m\}$. In our problem, communities are clusters.

Let N_{TP} be the number of pairs from the same cluster in $\mathbf{U}$. $\mathbf{V}$; N_{FP} is the number of pairs from the same cluster in $\mathbf{U}$ but different clusters in $\mathbf{V}$; N_{TN} is the number of pairs that belong to different clusters in $\mathbf{U}$ but to the same clusters in $\mathbf{V}$, and N_{FN} is the number of pairs that belong to different clusters in $\mathbf{U}$ and $\mathbf{V}$. Adjusted Rand Index (ARI; defined in range [0,1]) is given by the formula:

$$ARI(\mathbf{U}, \mathbf{V}) = \frac{2(N_{FN}N_{TP} - N_{FP}N_{TN})}{\theta},$$

where

$$\theta = (N_{TP} + N_{FP})(N_{FP} + N_{TP}) + (N_{FN} + N_{TN})(N_{TN} + N_{TP}).$$

Given two clusterings $\mathbf{U}$ and $\mathbf{V}$, the reciprocal MI information measures the information that $\mathbf{U}$ and $\mathbf{V}$ share. It proxies for how much knowing one clustering reduces the uncertainty about the other. Formally, the mutual information of two random variables X, Y is expressed by the formula $MI(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$, where $H(X)$ is the Shannon entropy [23]. In our simulation, we use normalized MI (NMI; defined in range [0,1]), as suggested in [60]. NMI of 1 means that the compared clusterings are identical, and in our case, that the algorithm has flawlessly discovered the true division into communities.

TABLE III
BASIC COMPARISON OF QUALITY METRICS FOR MODIFIED ALGORITHMS AGAINST THE BASELINE ONES

	% of cases in which the modified algorithm was...		
	worse than	equal to	better than baseline
ARI	23.69	31.73	44.58
NMI	19.66	31.72	48.62
modularity	47.21	25.26	27.53

The codes and the datasets we used for the analysis of experiments are available at https://github.com/tehora/community-detection-benchmark.

VIII. EXPERIMENTAL RESULTS

The distributions of ARI, NMI, and modularity are multimodal (Fig. 3, we present Zachary Karate Club as a remarkable example); that is why a typical, simple description in terms of mean (expected value) and standard deviation may be misleading. TableIII provides basic facts on the comparison of modified algorithms and baseline ones for the whole sample. Despite a noticable share of cases in which the outcome of our proposition was worse than the baseline algorithm, we would like to stress that it is unsurprising and not necessarily a pessimistic result. The success of the algorithm depends on the quality of data it is provided; if given messy data it may not work correctly. The results of this section lead to practical implications: despite we could suppose "the more information, the better", it is crucial to obtain valuable information. If we know what makes information valuable, we are able to be cost-effective. That is why in this section we want to find out we found out the relationships between the efficiency of our modified algorithms and the characteristics of the networks and nodes. To this end, we utilized binary logit models [61] as they do not imply assumptions on the independent values. We defined efficiency as a binary variable (0—modified algorithms scored lower in terms of an analyzed characteristic; 1—the results of the modified algorithm were no worse than the original, for brevity, we will say that the algorithms failed or succeeded, respectively).

The set of independent variables contained:

- **network-based** variables: the number of communities that we revealed information from (*count_com*), percent of nodes revealed per community (*size*); and
- **node-based** variables: median degree (*med_deg*), median closeness (*med_closeness*, and median betweenness (*med_betwenness*) of the revealed nodes. We also use dummies if any node from the top 25% of nodes in terms of a respective centrality measure was revealed (*deg_big, closeness_big, betweenness_big*). Centrality measures were normalized to [0,1]. We chose medians, because they are more immune to the presence of outliers that characterized the centrality measures' distributions (Fig. 4).

We controlled also for the type of the network, the average composition factor (*avg_cf*), the algorithm, and the selection of nodes (*random_structure*). Eventually, we added squares of *count_com*, *size*, and *avg_cf* to check the possible non-linearity in the impact of information.

TABLE IV
THE COEFFICIENTS OF BINARY LOGIT MODELS ON THE EFFICIENCY OF THE MODIFIED ALGORITHMS

	ARI	NMI	modularity
(Intercept)	1.7406 ⋆⋆⋆	4.9856 ⋆⋆⋆	3.5499 ⋆⋆⋆
football	0.3078 ⋆⋆⋆	1.2432 ⋆⋆⋆	1.748 ⋆⋆⋆
karate	0.289 ⋆⋆⋆	1.7059 ⋆⋆⋆	1.8053 ⋆⋆⋆
polbooks	-0.7032 ⋆⋆⋆	0.463 ⋆⋆⋆	0.4663 ⋆⋆⋆
CNMSeed	0.3282 ⋆⋆⋆	0.0505 ⋆⋆⋆	0.9863 ⋆⋆⋆
LouvainSeed	-0.5402 ⋆⋆⋆	-0.7052 ⋆⋆⋆	-1.2782 ⋆⋆⋆
random_structure	-0.7762 ⋆⋆⋆	-0.4587 ⋆⋆⋆	-0.9551 ⋆⋆⋆
count_com	-0.1386 ⋆⋆⋆	-0.1094 ⋆⋆⋆	-0.2259 ⋆⋆⋆
count_com^2	0.005 ⋆⋆⋆	0.0068 ⋆⋆⋆	0.0091 ⋆⋆⋆
size	1.5633 ⋆⋆⋆	1.9194 ⋆⋆⋆	-0.9595 ⋆⋆⋆
size^2	-2.2146 ⋆⋆⋆	-2.5315 ⋆⋆⋆	-0.0036
deg_median	-0.7198 ⋆⋆⋆	0.4315 ⋆⋆⋆	1.7428 ⋆⋆⋆
deg_big	-0.1781 ⋆⋆⋆	-0.1728 ⋆⋆⋆	0.1999 ⋆⋆⋆
closeness_median	2.8273 ⋆⋆⋆	-10.5666 ⋆⋆⋆	-8.688 ⋆⋆⋆
closeness_big	-0.0389⋆	-0.181 ⋆⋆⋆	-0.037 ⋆
betweenness_median	-1.3046 ⋆⋆⋆	8.3544 ⋆⋆⋆	1.2595 ⋆⋆⋆
betweenness_big	0.1173 ⋆⋆⋆	-0.0229	-0.4034 ⋆⋆⋆
avg_cf	-4.261 ⋆⋆⋆	-2.2754 ⋆⋆⋆	-9.5261 ⋆⋆⋆
avg_cf^2	4.2981 ⋆⋆⋆	2.4536 ⋆⋆⋆	9.21 ⋆⋆⋆

⋆⋆⋆, ⋆⋆, ⋆ denote significance at 0.01, 0.05, 0.1 level, respectively.

Coefficients in logit models have no direct interpretation; we will discuss the signs of the coefficients. For categorical variables, we will present the odds ratios. The variables in all regressions were jointly significant (p-values for F-tests 0.000).

When it comes to the graphs analyzed, our results (Table IV) suggest that Dolphins network was in the most cases the most problematic network for modified algorithms to work with. In comparison to working on Dolphins network, for each of the metrics, modified algorithms on Zachary Karate Club and Football had greater odds of success than the baseline ones (1.34 times higher for ARI, 3.46 times higher for NMI, and 5.74 times higher for modularity in the case of Football network; 1.36 times higher for ARI, 5.56 times higher for NMI, and 6.08 times higher for modularity in the case of Zachary Karate Club network). On the contrary, modified algorithms on Polbooks network had lower odds of getting a non-worse ARI (49% of chances of algorithm working on Dolphins network to succeed); working with that network increased odds of success by about 1.59 in terms of both NMI and modularity.

When we compare modified algorithms, LouvainSeed had the lowest chances to succeed. Its odds of success equaled 28% of GNSeed's odds success in terms of modularity. For ARI and NMI those chances were a bit higher: 49% and 58% of the GNSeed's odds to success in terms of NMI and ARI, respectively. CNMSeed had the highest chances to succeed of all. No matter the metrics, random selection of vertices had lower odds of success than the odds of success for drawing from a consistent induced subgraph. That effect differed in strength: it was the strongest in the case of modularity (0.38 times lower odds), and the mildest for NMI (0.63 times lower odds).

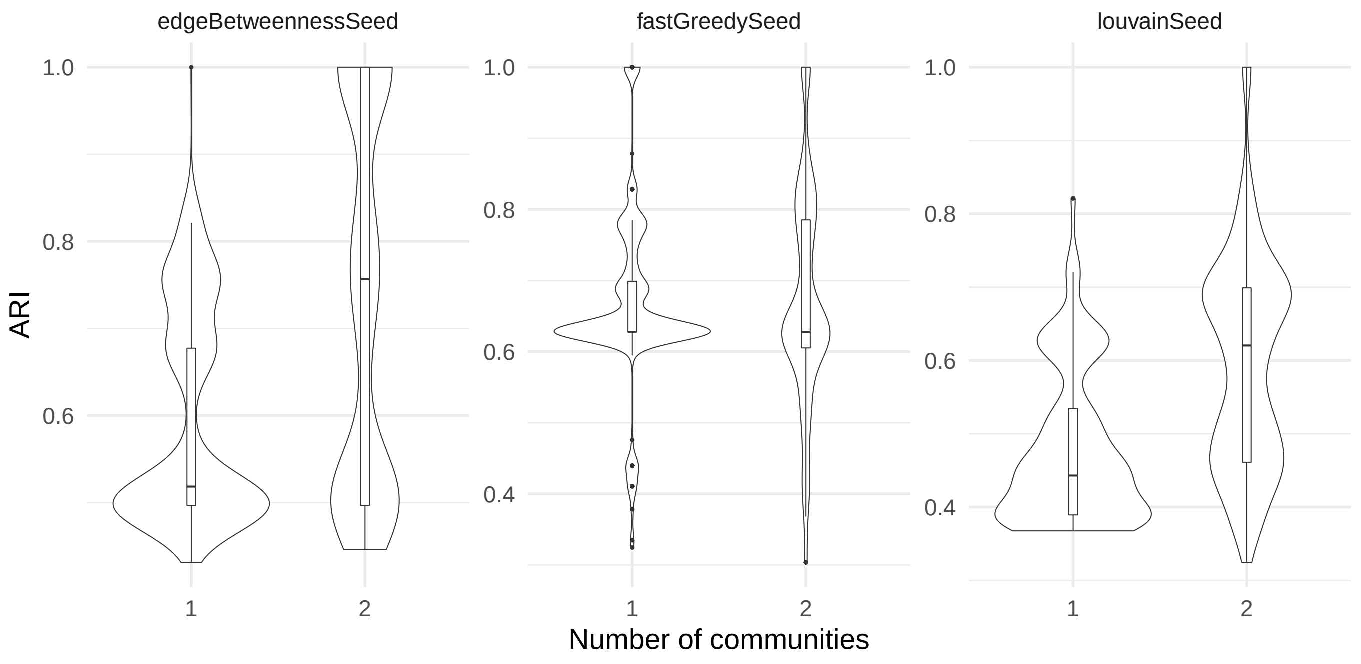


Fig. 3. The distribution of ARI for Karate Zachary Club with random selection of nodes and 25% of nodes revealed per community.

The distribution of ARI for Karate Zachary Club with random selection of nodes and 25% of nodes revealed per community

Our results suggest that the relationships between the amount of information and the modified algorithms' efficiencies are non-linear. On the one hand, the effect of increasing the number of communities we reveal vertices from is U-shaped. This suggests, there is an extreme that changes the sign of the effect. As general marginal effects in binary logistic models with interactions are non-trivial to compute, let us approximate that minimum: for ARI and modularity it falls out of range of our possible number of communities. Therefore, increasing the seed number of communities decreases the modified algorithm's chance to succeed, however, revealing nodes from a lot of communities helps the modified algorithms in terms of NMI (minimum within the range of possible numbers of communities). Once we know from how many communities we reveal vertices, too much information still might be harmful. The effects of the increase of the share of revealed nodes per community are inverse-U shaped no matter the quality measure. For ARI and NMI, increasing the share of revealed vertices per community will start to deteriorate the probability of success once we revealed about 1/3 of nodes per community. Note, that in the case of modularity, the coefficient for the square is statistically insignificant. That means, increasing the share of nodes per community always decreases the probability for the algorithms to succeed in terms of the modularity (see the sign of the *size*'s coefficient).

Interestingly, revealing on average more nodes that were correctly classified by the base algorithms (composition factor) increases dramatically the probability of success for modified algorithms. No matter the metric, the approximate minimum is about 0.45. While this effect might seem counter-intuitive (one would expect that investing in revealing nodes algorithms had problems with would be beneficial), we might redefine that into removing a part of the task the given algorithm succeeds in, anyway. For comparison, let us draw some intuition from the non-network clustering algorithms. Clustering algorithms based on within-cluster and between-cluster variability usually correctly distinguish outliers from typical values. The between-cluster variability is maximized; however, the clustering on the typical values (that could be of our primary interest) might be messy and unsatisfactory. Removing outliers (the part of the information the discussed algorithms work with correctly, at the expense of other part) will improve the quality of the clustering. Note, that in the real-life use-cases we rarely know the ground-truth communities, so that result is interesting theoretically but of a limited practical use. Given that we analyzed the average composition factor, our result provides a nice practical implication. As, for e.g., two communities in one of which we revealed 100% of base algorithms' TPs, and in the second 0% of TPs we get an average of 0.5 (that is close to approximate minimum), we conclude that to increase the modified algorithms' efficiency we would rather avoid extreme differences with respect to base algorithms' TPs in choosing the nodes to reveal.

The amount of information we reveal is only a part of the decision problem. One should also take care of the information quality. To this end, we investigated the relationships between the efficiency of the modified algorithms and the centrality of the revealed nodes. Given the scale of the simulation (only

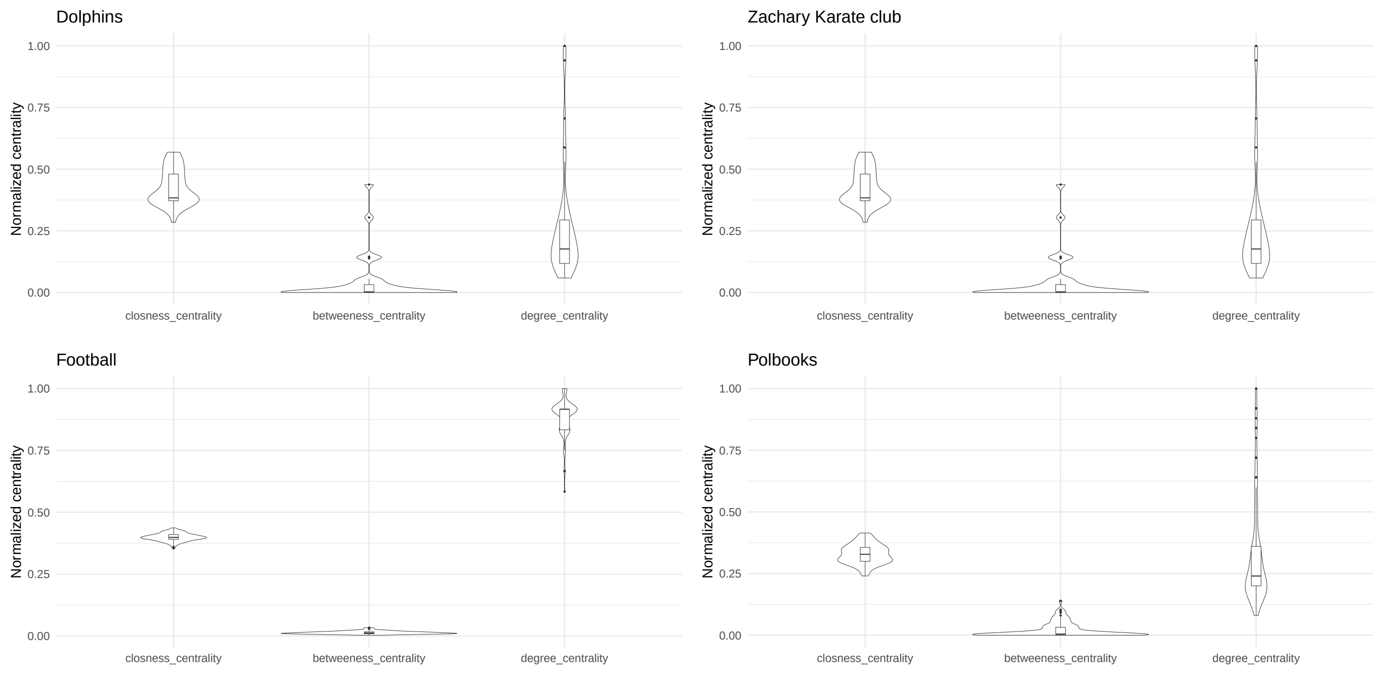


Fig. 4. Distribution of normalized centrality measures; Dolphins network (top left), Karate Zachary Club (top right), Football network (bottom left), and Polbooks network (bottom right).

four, rather small networks), we suggest to treat the results on the quality of the information in the set of nodes (based on their centrality measures) with caution. Exact estimation of the impact of the strategic revealing based on centralities on the community detection should require a more general simulation based on artificial networks and is out of the scope of this paper. That is why, we will discuss the unambiguous results. We found that revealing at least one node from the 25% top nodes with respect to degree centrality (that have more connections) decreases the probability of the modified algorithm to succeed in terms of NMI and ARI; however it improves chances of getting a better modularity of the result. Extending the revealed nodes set by at least one node from the 25% top nodes with respect to closeness centrality (nodes placed closer to the "middle" of the network) decreases the probability of the modified algorithm to succeed in terms of all characteristics. Eventually, according to our results, betweenness centrality is a beneficial aspect to consider. We want to somehow diverse the information on the bridges in our revealed set. Having at least one of top 25% nodes in terms of betweenness centrality revealed increases probability of gaining better efficiency in terms of ARI, for NMI the result is statistically insignificant. Improving median betweenness in revealed nodes set improves the probability of modified algorithms' success in terms of NMI and modularity. To sum up, we would say that one should not invest too much to determining the ground-truth for the extreme top betweenness nodes, but it should be worthwhile to consider at least a few of prominent ones. To gain some intuition, compare the "bridge" (a node with a considerable high betweenness centrality) placed more closer to the middle of the network with a possible node with an higher betwenness centrality placed on the network's boundaries (and the community), becuase it happens to connect a specific node that otherwise would be disconnected from the whole network. The first one might help us in determining the boundaries of the communities, while the second one does not add valuable information.

Note, that even if the modified algorithms yielded a higher number of worse results in terms of modularity (Table III), the general insights on the relationships between the efficiency and that metric were similar to the results on other metrics, according to which the modified algorithms generally performed at least not worse. Moreover, to check the generalizability of the results, we estimated also binary logit models, excluding cases in which modified algorithm scored exactly the same as the baseline. No matter the specification, the results were stable. The ordinal logit, an alternative possibility in which we could distinguish among worse, equal, and better efficiency, was found to be an incorrect model (Brandes tests jointly significant).

IX. Conclusions

We presented in this work refinements of three well-known community detection algorithms, proposed by Clauset-Newman-Moore, Louvain, and Girvan-Newman. We implemented these refinements in an open-source JavaScript library *igraph-community.js*. On top of our proposed algorithms, this library also includes many of the widely-used functionalities of the *igraph* package [17] as JavaScript. We also developed a graphical user interface (GUI) that facilitates the visualization and analysis of social networks at scale. Finally, we analyzed the performance of the refined algorithms via the means of numerical simultions.

There are various interesting direction for further research stemming from this work. Firstly, it would be interesting to extend other community detection algorithms to account for *a priori* information about node affiliation to communities. Next, a further development of the *igraph-community.js* library would be a natural step.

References

[1] Ulrik Brandes, Daniel Delling, Marco Gaertler, R. Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 20:172 – 188, 03 2008.

[2] Satu Schaeffer. Graph clustering. *Computer Science Review*, 1:27–64, 08 2007.

[3] Pall Jonsson, Tamara Cavanna, Daniel Zicha, and Paul Bates. Cluster analysis of networks generated through homology: Automatic identification of important protein communities involved in cancer metastasis. *BMC bioinformatics*, 7:2, 02 2006.

[4] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, Feb 2010.

[5] Alberto Pascual-García and Thomas Bell. functionink: An efficient method to detect functional groups in multidimensional networks reveals the hidden structure of ecological communities. *Methods in Ecology and Evolution*, 11(7):804–817, 2020.

[6] Jacob Goldenberg, Barak Libai, and Eitan Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing letters*, 12(3):211–223, 2001.

[7] Xujuan Zhou, Yue Xu, Yuefeng Li, Audun Josang, and Clive Cox. The state-of-the-art in personalized recommender systems for social networking. *Artificial Intelligence Review*, 37(2):119–132, 2012.

[8] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6), Dec 2004.

[9] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, Oct 2008.

[10] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1), Jul 2006.

[11] Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 76:036106, 10 2007.

[12] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *J. Graph Algorithms Appl.*, 10:191–218, 01 2006.

[13] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, Jan 2008.

[14] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3), Sep 2006.

[15] Alan Mislove, Bimal Viswanath, Krishna P Gummadi, and Peter Druschel. You are who you know: inferring user profiles in online social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 251–260, 2010.

[16] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), Feb 2004.

[17] Gabor Csardi, Tamas Nepusz, et al. The igraph software package for complex network research. *InterJournal, complex systems*, 1695(5):1–9, 2006.

[18] Kai-Di Sun and Si-Qi Wang. Enhanced community structure detection in complex networks with partial background information. *Scientific reports*, 3:3241, 11 2013.

[19] Daniel Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–91, 11 1999.

[20] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Adv. Neural Inf. Process. Syst*, 14, 04 2002.

[21] Cong Tran, Won-Yong Shin, and Andreas Spitz. Community detection in partially observable social networks, 2017.

[22] Johan Dahlin and Pontus Svenson. A method for community detection in uncertain networks. In *2011 European Intelligence and Security Informatics Conference*, pages 155 – 162, 10 2011.

[23] Zied Yakoubi and Rushed Kanawati. Licod: a leader-driven algorithm for community detection in complex networks. *Vietnam Journal of Computer Science*, 1:241–256, 01 2014.

[24] Glen Jeh and Jennifer Widom. Scaling personalized web search. *Proceedings of the 12th International Conference on World Wide Web, WWW 2003*, 08 2002.

[25] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, page 81–90, New York, NY, USA, 2004. Association for Computing Machinery.

[26] Aaron Clauset. Finding local community structure in networks. *Physical Review E*, 72(2), Aug 2005.

[27] F. Luo, J. Z. Wang, and E. Promislow. Exploring local community structures in large networks. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*, pages 233–239, 2006.

[28] Kun He, Yiwei Sun, David Bindel, John E Hopcroft, and Yixuan Li. Detecting overlapping communities from local spectral subspaces, 2015.

[29] Belfin R V, E. Grace Mary Kanaga, and Piotr Bródka. Overlapping community detection using superior seed set selection in social networks, 2018.

[30] Joyce Jiyoung Whang, David F. Gleich, and Inderjit S. Dhillon. Overlapping community detection using seed set expansion. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, page 2099–2108, New York, NY, USA, 2013. Association for Computing Machinery.

[31] Isabel M. Kloumann and Jon M. Kleinberg. Community membership identification from small seed sets. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 1366–1375, New York, NY, USA, 2014. Association for Computing Machinery.

[32] Stack Overflow. Developer survey results 2021. https://insights.stackoverflow.com/survey/2021. Accessed: 17.12.2021.

[33] Artur Myszkowski and Paweł Gołąb. igraph-community.js github. https://github.com/rthrs/igraph-community.js. Accessed: 17.12.2021.

[34] John Alexis Guerra Gomez. Netclustering.js. https://www.npmjs.com/package/netclustering. Accessed: 17.12.2021.

[35] Anton Eriksson and Daniel Edler. @mapequation/infomap. https://www.npmjs.com/package/@mapequation/infomap. Accessed: 17.12.2021.

[36] Luís Rita. layered-label-propagation. https://www.npmjs.com/package/layered-label-propagation. Accessed: 17.12.2021.

[37] John Granström. jlouvain. https://www.npmjs.com/package/jlouvain. Accessed: 17.12.2021.

[38] V. A. Traag, L. Waltman, and N. J. van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1), Mar 2019.

[39] Ferran Parés, Dario Garcia-Gasulla, Armand Vilalta, Jonatan Moreno, Eduard Ayguadé, Jesús Labarta, Ulises Cortés, and Toyotaro Suzumura. Fluid communities: A competitive, scalable and diverse community detection algorithm, 2017.

[40] Marcin Waniek, Tomasz P. Michalak, Michael J. Wooldridge, and Talal Rahwan. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2):139–147, Jan 2018.

[41] Emscripten Contributors. Emscripten. https://emscripten.org. Accessed: 17.12.2021.

[42] WebAssembly Contributors. Webassembly. https://webassembly.org. Accessed: 17.12.2021.

[43] Alexis Deveria. Can i use—support for the ecmascript 5 specification. https://caniuse.com/#feat=es5. Accessed: 17.12.2021.

[44] WebAssembly Contributors. Webassembly faq. https://webassembly.org/docs/faq/#why-create-a-new-standard-when-there-is-already-asmjs. Accessed: 17.12.2021.

[45] Paul Erdős and Alfréd Rényi. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

[46] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[47] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440–442, 1998.

[48] Tomasz P Michalak, Karthik V Aadithya, Piotr L Szczepanski, Balaraman Ravindran, and Nicholas R Jennings. Efficient computation of the shapley value for game-theoretic network centrality. *Journal of Artificial Intelligence Research*, 46:607–650, 2013.

[49] Facebook Open Source. React. https://reactjs.org/. Accessed: 17.12.2021.

[50] Dan Abramov. Redux. https://redux.js.org/. Accessed: 17.12.2021.

[51] Khronos Group. Webgl. https://www.khronos.org/webgl/. Accessed: 17.12.2021.

[52] Mike Bostock. Data-driven documents. https://d3js.org/. Accessed: 17.12.2021.

[53] John Guerra. Force in a box github. https://github.com/john-guerra/forceInABox. Accessed: 17.12.2021.

[54] WHATWG Community. Web workers specification. http://dev.w3.org/html5/workers/. Accessed: 17.12.2021.

[55] Wayne Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33, 11 1976.

[56] D Lusseau, K Schneider, O J Boisseau, P Haase, E Slooten, and S M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations - can geographic isolation explain this unique trait? *Behavioral Ecology and Sociobiology*, 54:396–405, 2003.

[57] Michelle Girvan and Mark Newman. Girvan, m. & newman, m. e. j. community structure in social and biological networks. proc. natl acad. sci. usa 99, 7821-7826. *Proceedings of the National Academy of Sciences of the United States of America*, 99:7821–6, 07 2002.

[58] V. Krebs. Political books network. http://www.orgnet.com.

[59] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854, December 2010.

[60] Leon Danon, Albert Díaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008–P09008, Sep 2005.

[61] David R. Cox. The regression analysis of binary sequences (with discussion). *Journal of the Royal Statistical Society. Series B (Methodological)*, 20:215–242, 1958.