RESEARCH Open Access

Graph neural network inspired algorithm for unsupervised network community detection

Stanislav Sobolevsky[1,2*] and Alexander Belyi[2]

*Correspondence: ss9872@nyu.edu

[1] Center For Urban Science+Progress, New York University, Brooklyn, NY, USA
[2] Department of Mathematics and Statistics, Faculty of Science, Masaryk University, Brno, Czech Republic

Abstract
Network community detection often relies on optimizing partition quality functions, like modularity. This optimization appears to be a complex problem traditionally relying on discrete heuristics. And although the problem could be reformulated as continuous optimization, direct application of the standard optimization methods has limited efficiency in overcoming the numerous local extrema. However, the rise of deep learning and its applications to graphs offers new opportunities. And while graph neural networks have been used for supervised and unsupervised learning on networks, their application to modularity optimization has not been explored yet. This paper proposes a new variant of the recurrent graph neural network algorithm for unsupervised network community detection through modularity optimization. The new algorithm's performance is compared against the state-of-the-art methods. The approach also serves as a proof-of-concept for the broader application of recurrent graph neural networks to unsupervised network optimization.

Keywords: Complex networks, Community detection, Network science

Introduction

The complex networks play a pivotal role in various fields such as physics, biology, economics, social sciences, and urban planning. Thus understanding the underlying community structure of the networks saw a wide range of applications, including social science (Plantié and Crampes 2013), biology(Guimerà and Nunes Amaral 2005), and economics (Piccardi and Tajoli 2012). In particular, partitioning the networks of human mobility and interactions is broadly applied to regional delineation (Ratti et al. 2010; Blondel et al. 2010; Sobolevsky et al. 2013; Amini et al. 2014; Hawelka et al. 2014; Kang et al. 2013; Sobolevsky et al. 2014; Belyi et al. 2017; Grauwin et al. 2017; Xu et al. 2021) as well as urban zoning (Sobolevsky et al. 2018; Landsman et al. 2020, 2021).

Over the last two decades, a large number of approaches and algorithms for community detection in complex networks have been suggested. Some of them are just straightforward heuristics such as hierarchical clustering (Hastie 2001) or the Girvan-Newman (Girvan and Newman 2002) algorithm, while the vast majority rely on optimization techniques based on the maximization of various objective functions. The first and the most well-known partition quality function is modularity (Newman and Girvan

2004; Newman 2006) assessing the relative strength of edges and quantifying the cumulative strength of the intra-community links. Many modularity optimization strategies have been suggested over the last two decades (Newman and Girvan 2004; Newman 2006, 2004; Clauset et al. 2004; Agarwal and Kempe 2008; Sun et al. 2009; Blondel et al. 2008; Guimera et al. 2004; Good et al. 2010; Duch and Arenas 2005; Lee et al. 2012; Aloise et al. 2012; Barber and Clark 2009; Liu and Murata 2010; Sobolevsky et al. 2014; Džamić et al. 2019; Traag et al. 2019; Biedermann et al. 2018). Comprehensive historical overviews are presented in Fortunato (2010); Fortunato and Hric (2016) as well as some later surveys (Khan and Niazi 2017; Javed et al. 2018).

And while the problem of finding the exact modularity maximum is known to be NP-hard (Brandes et al. 2006), most of the available modularity optimization approaches rely on specific discrete optimization heuristics (although, in some cases, an algorithmic optimality proof of the partition is possible Agarwal and Kempe 2008; Aloise et al. 2010; Sobolevsky et al. 2017; Belyi et al. 2019, 2021; Belyi and Sobolevsky 2022).

As we show below, modularity optimization can be formulated as a continuous matrix optimization problem. However, the direct application of generic gradient descent methods is inefficient due to a large number of local maxima, which gradient descent might not be able to overcome.

Recently, graph neural networks (GNNs) became increasingly popular for supervised classifications and unsupervised embedding of the graph nodes with diverse applications in text classification, recommendation systems, traffic prediction, computer vision and many others (Wu et al. 2020). GNNs were already successfully applied for community detection, including supervised learning of the ground-truth community structure (Chen et al. 2017) as well as unsupervised learning of the node features enabling representation modeling of the network, including stochastic block-model (Bruna and Li 2017) and other probabilistic models with overlapping communities (Shchur and Günnemann 2019) or more complex self-expressive representation (Bandyopadhyay and Peter 2020). However, existing GNN applications overlook unsupervised modularity optimization, which so far has been a major approach in classic community detection.

This paper aims to fill this gap by proposing a straightforward GNN-inspired algorithmic framework for unsupervised community detection through modularity optimization. We perform a comprehensive comparative evaluation of the performance of the proposed method against the state-of-the-art ADVNDS and Combo algorithms (capable of reaching the best known partition in most cases), a viral Louvain algorithm (which, despite its sub-optimal performance, is very fast and capable of handling the large-scale networks), and its successor, Leiden algorithm, that improves partition quality while preserving short execution time. We demonstrate that the method provides a reasonable balance between performance and speed for classic, synthetic and real-world networks, including temporal networks, and is sometimes capable of finding partitions with a higher modularity score that other algorithms cannot achieve.

More importantly, we believe the proposed approach serves as a proof of concept of leveraging GNN approaches for solving a broader range of network optimization problems. Such problems often arise when the aim is to reconstruct nodes' attributes based on their features and the network structure, including various types of unsupervised graph clustering (Kampffmeyer et al. 2019; Bianchi 2022). Following the work applying machine learning to combinatorial optimization problems by Bengio et al. (2021), several attempts to apply GNNs to hard combinatorial optimization problems were recently made. Some first promising results were obtained for the problems of minimum vertex cover, maximal clique, maximal independent set, and the satisfiability problem (Li et al. 2018). Furthermore, various GNN architectures were adapted to address the graph correlation clustering problem formulated as the minimum cost multicuts problem (Jung and Keuper 2022). Initial steps were even taken towards graph clustering via maximizing some variant of modularity function (Lobov and Ivanov 2019; Tsitsulin et al. 2020). However, their results still fall behind current state-of-the-art approaches in terms of modularity score. In this work, we show that with GNN-inspired techniques it is possible to achieve more practical results close to state-of-the-art.

In the following sections, first, we recall the formulation of community detection through modularity maximization and show how it could be framed as continuous quadratic optimization. Then we propose our GNN-inspired method and describe how to select its parameters. Lastly, we evaluate our approach using three benchmarks: classical real-world networks used previously in the literature to test modularity maximization algorithms, synthetic-networks benchmark, and a temporal network of taxi trips. The paper ends with conclusions and discussions.

The modularity optimization problem

The network modularity was among the first quality/objective functions proposed to assess and optimize the community structure (Newman 2006). However, it is now known to have certain shortcomings, including a resolution limit (Fortunato and Barthélémy 2007; Good et al. 2010) and the fact that it does not compare with a proper baseline and finds communities in random networks. Therefore alternative objective functions should be mentioned, e.g., Infomap description code length (Rosvall and Bergstrom 2007, 2008), Stochastic Block Model likelihood (Karrer and Newman 2011; Ball et al. 2011; Bickel and Chen 2009; Decelle et al. 2011, 2011; Yan et al. 2014), and Surprise (Aldecoa and Marìn 2011). Nevertheless, despite its limitations, modularity remains perhaps the most commonly used objective function so far.

In 2014, the authors proposed a novel optimization technique for community detection, "Combo" (Sobolevsky et al. 2014), capable of maximizing various objective functions, including modularity, description code length, and pretty much any other metric based on the link scoring and assessing the cumulative score of the intra-community links. At the time of publication, for modularity optimization, Combo outperformed other state-of-the-art algorithms, including a popular Louvain method (Blondel et al. 2008) in terms of the quality (modularity score) of the resulting partitioning, which could be achieved within a reasonable time for most real-world and synthetic networks of up to tens of thousands of nodes. The size limitation for the algorithm evaluation is due to the current implementation handling a full modularity matrix in the memory entirely. However, this is not a fundamental limitation and could be overcome by using sparse matrix operations. Recently, more precise algorithms were proposed, but they are slower and often designed to run on a distributed cluster (Džamić et al. 2019; Biedermann et al. 2018; Hamann et al. 2018; Lu et al. 2015). Moreover, their code is not available.

The proposed algorithms, including Combo, are often quite efficient and, in some cases, are able to reach the theoretical maximum of the modularity score as revealed by a suitable upper bound estimate (Sobolevsky et al. 2017). However, in general, finding the theoretically optimal solution may not be feasible, and one has to rely on heuristic algorithmic solutions without being certain of their optimality. Instead, an empiric assessment of their performance in comparison with other available algorithms could be performed.

The modularity function

In short, the modularity (Newman and Girvan 2004; Newman 2006) function of the proposed network partition quantifies how relatively strong all the edges between the nodes attached to the same community are. Specifically, if the network edge weights between each pair of nodes i, j are denoted as $e_{i,j}$, then the modularity of the partition $com(i)$ (expressed as a mapping assigning community number com to each node i) can be defined as

$$M = \sum_{i,j,com(i)=com(j)} q_{i,j}, \tag{1}$$

where the quantity $q_{i,j}$ for each edge i, j (call q a modularity score for an edge) is defined as its normalized relative edge weight in comparison with the random network model with the same node strengths. Namely,

$$q_{i,j} = \frac{e_{i,j}}{T} - \frac{w^{out}(i)w^{in}(j)}{T^2},$$

where $w^{out}(i) = \sum_k e_{i,k}$, $w^{in}(j) = \sum_k e_{k,j}$, $T = \sum_i w^{out}(i) = \sum_j w^{in}(j) = \sum_{i,j} e_{i,j}$.

Rewrite the modularity optimization problem in a vector form: let $Q = \left(q_{i,j}\right)$ be the matrix of all the modularity scores for all the edges (call it a modularity matrix). Let C be an $n \times k$ matrix, where n is the number of network nodes and k is the number of communities we are looking to build. Each element $c_{i,p}$ of the matrix can be zero or one depending on whether the node i belongs to the community p or not, i.e., whether $com(i) = p$. If the communities are not overlapping, then each row of the matrix has one single unit element, and the rest of its elements are zeros.

More generally, if we admit uncertainty in community attachment, then the elements $c_{i,p}$ of the matrix C could represent the probabilities of the node i to be attached to the community p. This way, $c_{i,p} \in [0, 1]$ and the sum of each row of the matrix C equals 1.

Then the modularity score M in the case of a discrete community attachment could be represented as a trace of matrix product

$$M = tr(C^T QC), \tag{2}$$

where tr denotes the trace of the matrix – a sum of all of its diagonal elements.

This way, finding the community structure of up to k communities optimizing the network modularity could be expressed as a constrained quadratic optimization problem of finding the $n \times k$ matrix C maximizing the trace of matrix product $M = tr(C^T QC)$, such that all $c_{i,p} \in \{0, 1\}$ and the sum of each row of the matrix C equals 1 (having a single unit element).

Replacing the binary attachment constraint $c_{i,p} \in \{0, 1\}$ with a continuous attachment $c_{i,p} \in [0, 1]$ relaxes the optimization problem to finding probabilistic community attachments. It could be easily shown that the optimal solution of the binary attachment problem could be derived from the optimal solution of the probabilistic attachment problem after assigning $q_{i,i} = 0$ for all the diagonal elements of the matrix Q. As diagonal elements $q_{i,i}$ are always included in the sum M since $com(i) = com(j)$ for $i = j$, the values of the diagonal elements serve as constant adjustment of the objective function M and do not affect the choice of the optimal partition, so we are free to null them without loss of generality. At the same time, for each given i, once $q_{i,i} = 0$, if we fix the community attachments of all the other nodes $j \neq i$, the objective function M becomes a linear function of the variables $c_{i,p}$ subject to constraints $\sum_p c_{i,p} = 1$ and $c_{i,p} \in [0, 1]$. Obviously, the maximum of the linear function with linear constraints is reached at one of the vertices of the domain of the allowed values for $c_{i,p}$, which will involve a single $c_{i,p}$ being one and the rest being zeros. This way, we have proven the following:

Proposition *The optimal probabilistic attachment $c_{i,p} \in [0, 1]$ maximizing (2) in the case of $q_{i,i} = 0$ represents a binary attachment $c_{i,p} \in \{0, 1\}$ maximizing (2) for an arbitrary original Q.*

So the discrete community detection problem through modularity optimization could be solved within the continuous constrained quadratic optimization framework. This allows the application of methods and techniques developed for continuous optimization, such as gradient descent, for example. However, despite its analytic simplicity, the quadratic programming problem with indefinite matrix Q is still NP-hard. In particular, the dimensionality of the problem leads to multiple local maxima challenging direct application of the standard continuous optimization techniques, like gradient descent. Indeed, any discrete partition, such that no single node could be moved to a different community with a modularity gain, will become such a local maximum. Unfortunately, finding such a local maximum rarely provides a plausible partition – such solutions could have been obtained with a simple greedy discrete heuristic iteratively adjusting the single node attachments, while we know that the modularity optimization, being NP-hard, generally requires more sophisticated non-greedy heuristics, like Sobolevsky et al.

(2014). To address this challenge, we introduce a new heuristic method that efficiently finds high-quality solutions to the described quadratic optimization problem, although without guaranteed achievement of the global optimum.

The GNNS method

In this section, we present a GNN-style method (GNNS) for unsupervised network partition through modularity optimization inspired by recurrent graph neural network models (in the definition of Wu et al. (2020)) as well as an older Weisfeiler-Lehmann graph node labeling algorithm Weisfeiler and Leman (1968). Ideas similar to the Weisfeiler-Lehmann algorithm have already found their application to community detection in a well-known label propagation algorithm Raghavan et al. (2007). Their development and application to modularity maximization were proposed and studied in detail in consequent works (Barber and Clark 2009; Liu and Murata 2010). However, they still were discrete optimization heuristics in their spirit and could not take into account recent advances in graph neural networks. Unlike these methods, we propose a continuous optimization technique that considers current nodes' attachments combined with attachments of their neighbors. Namely, we propose a simple iterative process starting with a random initial matrix $C = C_0$ and at each step $t = 1, 2, 3, ..., N$ performing an iterative update of the rows c_i of the matrix C representing the node i community attachments as follows:

$$\tilde{c}_i^t = F\left(c_i^{t-1}, Q_i C^{t-1}\right), \tag{3}$$

where $Q_i = \left(q_{i,j} : j = 1, 2, ..., n\right)$ is the i-th row of the modularity matrix Q representing the outgoing edges from the node i. This way, the term $Q_i C^{t-1}$ collects information about the neighbor nodes' community attachments (this could be viewed as a development of ideas discussed in Barber and Clark (2009)), and the equation (3) updates the node community attachments with respect to their previous attachments as well as the neighbor node attachments. In order to ensure the conditions $\sum_p c_{i,p} = 1$, a further normalization $c_{i,p}^t = \tilde{c}_{i,p}^t / \sum_{p^*} \tilde{c}_{i,p^*}^t$ needs to be applied at each iteration.

A simple form for an activation function F could be a superposition of a linear function subject to appropriate scale normalization and a rectified linear unit $ReLU(x) = \begin{cases} 0, x \leq 0 \\ x, x > 0 \end{cases}$, leading to

$$\tilde{c}_{i,p}^t = ReLU\left(f_1 c_{i,p}^{t-1} + f_2 Q_i C_p^{t-1} / \tau_i^t + f_0\right), \; c_{i,p}^t = \tilde{c}_{i,p}^t / \sum_{p^*} \tilde{c}_{i,p^*}^t, \tag{4}$$

where f_0, f_1, f_2 are the model parameters, $C_p = \left(c_{j,p} : j = 1, 2, ..., n\right)$ is the p-th column of the matrix C representing all the node attachments to the community p, and the $\tau_i^k = \left|\max_{p^*} Q_i C_{p^*}^{t-1}\right|$ are the normalization coefficients ensuring the same scale for terms of the formulae.

Intuitive considerations allow defining possible ranges for the model coefficients f_0, f_1, f_2. Defining the coefficient f_1 within the range $f_1 \in [0,1]$ would ensure decay scaling of the community attachment at each iteration unless confirmed by the strength of the node's attachment to the rest of the community expressed by $Q_i C_p^{t-1}$. A free term $f_0 \in [-1,0]$ provides some additional constant decay of the community attachment at each iteration, while the term $Q_i C_p^{t-1}/\tau_i^t$ strengthens the attachment of node i to those communities having positive modularity scores of the edges between node i and the rest of the community. Normalization term τ_i^t ensures that the strongest community attachment gets a maximum improvement of a fixed scale f_2.

A consistent node attachment that cannot be improved by assigning the given node i to a different community p (i.e., having $Q_i C_p^{t-1} = \tau_i^{t-1} = \max_{p^*} Q_i C_{p^*}^{t-1}$) should see the fastest increase in the attachment score $\tilde{c}_{i,p}^t$, eventually converging to the case of $c_{i,p} = 1$. Any weaker attachment should see a decreasing community membership score $c_{i,p}^t$, eventually dropping to zero. This could be ensured by the balancing equation $f_1 + f_2 + f_0 = 1$, allowing to define appropriate f_2 given f_0 and f_1.

Training the GNNS

The sequence of the GNNS iterations (4) depends on the choice of the model parameters f_0, f_1, as well as the initial community attachments. The final convergence also often depends on those choices. Given that, a good strategy is to simulate multiple iteration sequences with different initial attachments and choose the best final result. Also, it turns out that the method demonstrates reasonable performance for a broad range of parameter values $f_0 \in [-1,0]$, $f_1 \in [0,1]$, so rather than trying to fit the best choice for all the networks or a given network, one may simply include the random choice for f_0, f_1 along with a random choice of the initial community attachments.

So the proposed GNNS algorithm starts with a certain number of S random partitions and parameter choices. Then the GNNS performs 10 iterations of the partition updates according to (4). Among those, a batch of the best $\lfloor S/3 \rfloor$ partitions (with the highest achieved modularity scores) is selected and further supplemented with another $S - \lfloor S/3 \rfloor$ configurations derived from the selected batch by randomly shuffling partitions and assigning new random parameters. Another 10 iterations are performed. Another batch of $\lfloor S/9 \rfloor$ best partitions is selected and shuffled, creating a total of $\lfloor S/3 \rfloor$ samples. Then another 30 iterations are performed with those, and for small S ($S \leq 1000$), the best resulting partition is selected as the final outcome of the algorithm. For larger S ($S > 1000$), another batch of $\lfloor S/30 \rfloor$ best partitions is selected and shuffled, creating a total of $\lfloor S/10 \rfloor$ samples. Then the final 100 iterations are performed with those, and the best resulting partition is selected as the final outcome of the algorithm. Finally, the partition is discretized by assigning each node to the cluster with maximum probability. The following algorithm describes the whole process.

Algorithm 1 GNNS

```
Input: Graph $G$ with $n$ nodes, number random attachments and parameter choices $S$, maximum
    number of communities $k$
Output: Partition $P$
 1: $Q = \text{CalculateModularityMatrix}(G)$
 2: $\mathcal{B} = \{(C^{(s)} \in [0,1]^{n\times k}, f_0^{(s)} \in [-1,0], f_1^{(s)} \in [0,1], f_2^{(s)} = 1 - f_0^{(s)} - f_1^{(s)})|s = 1..S\}$ ▷ initialize
    $\mathcal{B}$ as a set of $S$ random attachments and parameters
 3: $iterations\_per\_stage = [10, 10, 30]$
 4: if $S > 1000$ then append 100 to $iterations\_per\_stage$
 5: for $i = 1..\text{length}(iterations\_per\_stage)$ do
 6:     foreach attachment $C$ and associated parameters $f_0, f_1, f_2$ from $\mathcal{B}$ do
 7:         foreach raw $C_i$ in $C$ do $C_i = C_i / \sum_p C_{ip}$
 8:         repeat $iterations\_per\_stage[i]$ times                    ▷ apply formula 4
 9:             $QXC = Q \times C$
10:             foreach raw $QXC_i$ in $QXC$ do $QXC_i = QXC_i / \max_p |QXC_{ip}|$
11:             $C = ReLU(f_0 + f_1 \cdot C + f_2 \cdot QXC)$
12:             foreach raw $C_i$ in $C$ do $C_i = C_i / \sum_p C_{ip}$
13:         $modularity = tr(C^T QC)$
14:     if $i <$ length($iterations\_per\_stage$) then
15:         $nextS = \lfloor S \cdot iterations\_per\_stage[0] / iterations\_per\_stage[i+1] \rfloor$
16:         select best $\lfloor nextS/3 \rfloor$ attachments based on $modularity$
17:         fill the rest $nextS - \lfloor nextS/3 \rfloor$ attachments by randomly picking from those best $\lfloor nextS/3 \rfloor$
    attachments and assign them new random parameters $f_0, f_1$
18: select partition $C$ that gives the highest modularity
19: foreach raw $C_i$ in $C$ do $P[i] = \arg\max_p C_{ip}$                 ▷ discretize partition
20: return $P$
```

If matrix Q is stored as a sparse graph matrix and two separate vectors for in- and out-degrees, this algorithm has a time complexity of $O(Simk)$, where S is the number of attempts with different initial attachments and parameter values, i is the number of iterations, m is the number of edges, and k is the maximum number of communities. Value k is usually selected as an (educated) guess of the expected maximum number of communities or as the maximum feasible value. One nice property the GNNS inherits from neural networks is that it is easily parallelizable by modern frameworks.

Below we evaluate the three versions of the GNNS: a fast version with $S = 100$ denoted GNNS100, a slower but more precise version with $S = 2500$ denoted GNNS2500, and a slow but very precise version with $S = 25000$ denoted GNNS25000.

Comparative evaluation

We implemented the proposed GNNS modularity optimization algorithm in Python and ran our experiments in Google Colab[1] (results of Combo runs on the three largest networks were obtained on a laptop because of the time limits in Colab). The source code is available on GitHub[2]. We evaluate our algorithm against other state-of-the-art

1 https://colab.research.google.com/

2 https://github.com/Alexander-Belyi/GNNS

Table 1 The best modularity scores reached by the Combo and Louvain method after a different number of attempts for the Email network

Method/attempts	1	5	10	20	50
Combo	0.581918	0.582751	0.582751	0.582829	0.582829
Louvain	0.563761	0.570319	0.573820	0.574912	0.574912

techniques mentioned above: a fast and popular Louvain method Blondel et al. (2008), its successor, the Leiden algorithm Traag et al. (2019), Combo algorithm Sobolevsky et al. (2014), often capable of reaching the best known modularity score over a wide range of networks, and the ADVNDS method claimed to be state-of-the-art method in 2017 Džamić et al. (2019). Both papers introducing Combo and ADVNDS make a thorough comparison with other methods available at their time and conclude that those methods outperform their competitors. We also tried all other modularity maximization methods implemented in the CDLib library developed specifically for evaluating community detection methods Rossetti et al. (2019). However, their performance was much worse both in terms of running time and achieved modularity scores. Some more recent methods claim to be new state-of-the-art, but they are designed to be run on distributed systems, and their results cannot be fairly compared Biedermann et al. (2018). Thus, first, we shall compare our method with four selected approaches over a sample of classic network examples. And then, all those methods but ADVNDS (whose implementation is not available to us) shall also be evaluated over the series of the two types of random graphs often used for benchmarking the community detection algorithms: Lancichinetti et al. (2008) and Block-model graphs (Karrer and Newman 2011).

As the GNNS chooses the best partition among multiple runs, we consider its three configurations involving a) 100, b) 2500, and c) 25000 initial random samples of partitions plus model configurations. We shall refer to those as GNNS100, GNNS2500, and GNNS25000. All other algorithms in our comparison also involve random steps, and the best partition they converge to is not perfectly stable. Thus their performance could also benefit from choosing the best partition among multiple runs, especially for the Louvain method. It often takes up to 10-20 attempts to find the best partition they are capable of producing. E.g. applying Combo and Louvain to the classic case of the Email network leads to the following performance reported in Table 1 below. As we see, both reach their best performance after 20 iterations (although results of the Combo for this network are significantly better compared to Louvain) but do not further improve over the subsequent 30 iterations. Based on that, in the further experiments, we shall report the best performance of the Combo, Louvain, and Leiden algorithms achieved after 20 attempts for each. Since the implementation of the ADVNDS algorithm is not available to us, we provide results reported in its original paper after ten runs with running time calculated as the reported average multiplied by ten Džamić et al. (2019).

Classic examples

Most of the classic instances were taken from the clustering chapter of the 10th DIMACS Implementation Challenge[3] and were often reused in community detection

[3] All networks were downloaded from http://www.cc.gatech.edu/dimacs10/downloads.shtml

literature Sanders et al. (2014). Table 2 reports the sources and details of those networks. Since the complexity of our method is proportional to the maximum number of communities, we limited this dataset to the networks with less than three hundred communities, according to the ADVNDS paper Džamić et al. (2019). All originally directed networks were symmetrized, and self-loops were removed. The largest network, Krong500slogn16, is synthetic, while all other networks represent real-world data.

Tables 3 and 4 report the performance of the proposed approach compared with ADVNDS Džamić et al. (2019), Leiden Traag et al. (2019), Louvain Blondel et al. (2008), and Combo methods Sobolevsky et al. (2014). Missing values in the ADVNDS column mean that the corresponding networks were not present in the original paper. For the Leiden and Louvain methods, we used their implementation in the Python igraph package, setting the number of iterations to −1 allowing the Leiden algorithm to run until the best modularity is achieved. The missing value in the Combo column corresponds to the network too large for the current implementation.

According to the results reported in Tables 3 and 4, Louvain is the fastest algorithm, closely followed by the GNNS100 method, especially for larger networks, with GNNS100 often providing higher modularity scores, especially for smaller networks,

Table 2 List, with sources, of the networks we used in our benchmark

No	Name	Nodes	Edges	Weighted	Description
1	karate	34	78	NO	Zachary's Karate network Zachary (1977)
2	chesapeake	39	170	NO	Chesapeake Bay Mesohaline Network Baird and Ulanowicz (1989)
3	dolphins	62	159	NO	Dolphins' Social Network Lusseau et al. (2003)
4	lesmis	77	254	YES	Co-appeareance of characters in Les Miserables Knuth (1993)
5	polbooks	105	441	NO	Amazon.com Co-purchases of political books[a]
6	adjnoun	112	425	NO	Common adjective and noun adjacencies in David Copperfield Newman (2006)
7	football	115	613	NO	American College Football games in year 2000 Girvan and Newman (2002)
8	jazz	198	2742	NO	Network of Jazz Musicians Gleiser and Danon (2003)
9	C.Elegans N.	297	2148	YES	Neural network of C. Elegans White et al. (1986)
10	US Airports1997	332	2126	YES	US Aiports network from 1997[b]
11	C.Elegans M.	453	2025	NO	Metabolic Network of C. ElegansDuch and Arenas (2005)
12	email	1133	5451	NO	Email Networks University of Tarragona Guimerà et al. (2003)
13	polblogs	1490	16715	NO	Connections among political blogs Adamic and Glance (2005)
14	US Airports2010	1858	17215	YES	Complete network of US airports in 2010[c]
15	power	4941	6594	NO	A network of the Western States Power Grid of the US Watts and Strogatz (1998)
16	PGPgiantcompo	10680	24316	NO	Giant component of the network of users of the PGP algorithm Boguñá et al. (2004)
17	Krong500slogn16	65536	2456071	NO	A synthetic graphs created with the Kronecker generator[d]

[a] Valdis Krebs, data available online at http://www.orgnet.com

[b] Vladimir Batagelj and Andrej Mrvar (2006): Pajek datasets. Airports. http://vlado.fmf.uni-lj.si/pub/networks/data/mix/USAir97.net

[c] Data from the Bureau of Transportation Statistics - details at http://toreopsahl.com/datasets/#usairports

[d] https://graph500.org

Table 3 Execution time (in seconds) of the ADVNDS, Leiden, Louvain, Combo, and GNNS algorithms over the classic network examples

No	Name	ADVNDS	Leiden	Louvain	Combo	GNN100	GNN2500	GNN25000
1	karate	2.5	0.02	**0.01**	0.02	0.04	0.23	4.81
2	chesapeake	3.3	0.02	**0.01**	0.03	0.03	0.23	4.86
3	dolphins	0.0	0.07	**0.01**	0.06	0.03	0.26	5.05
4	lesmis	0.0	0.03	**0.01**	0.20	0.03	0.26	5.28
5	polbooks	0.0	0.08	**0.01**	0.42	0.03	0.27	5.26
6	adjnoun	2.5	0.12	**0.03**	0.65	**0.03**	0.30	5.61
7	football	10.8	0.09	**0.02**	0.41	0.03	0.35	6.20
8	jazz	30.5	0.23	0.16	0.88	**0.07**	0.67	6.10
9	C.Elegans N	1.5	0.46	0.29	4.64	**0.10**	0.71	7.50
10	US Airports1997	–	0.58	0.19	6.08	**0.08**	0.88	6.97
11	C.Elegans M	16.6	1.00	0.26	24.73	**0.08**	1.38	12.93
12	email	297.3	2.72	0.72	141.41	**0.28**	2.71	31.24
13	polblogs	1.2	1.62	**0.72**	56.63	0.75	15.58	195.01
14	US Airports2010	–	2.06	1.01	170.47	**0.85**	19.08	237.23
15	power	16439.4	6.28	**1.12**	2200.80	1.43	39.60	491.33
16	PGPgiantcompo	18000.0	11.41	**2.59**	25773.10	3.32	101.95	1256.97
17	Krong500slogn16	17610.5	5983.00	298.22	–	**45.78**	1061.30	13736.06

The best results are in bold

Table 4 Best modularity scores of the ADVNDS, Leiden, Louvain, Combo, and GNNS algorithms over the classic network examples

No	Name	ADVNDS	Leiden	Louvain	Combo	GNN100	GNN2500	GNN25000
1	karate	**0.419790**	**0.419790**	**0.419790**	**0.419790**	**0.419790**	**0.419790**	**0.419790**
2	chesapeake	**0.265796**	**0.265796**	**0.265796**	**0.265796**	**0.265796**	**0.265796**	**0.265796**
3	dolphins	**0.528519**	**0.528519**	*0.527728*	0.526799	**0.528519**	**0.528519**	**0.528519**
4	lesmis	**0.566688**	**0.566688**	**0.566688**	**0.566688**	**0.566688**	**0.566688**	**0.566688**
5	polbooks	**0.527237**	**0.527237**	*0.526967*	**0.527237**	**0.527237**	**0.527237**	**0.527237**
6	adjnoun	**0.313367**	0.310563	0.303934	*0.311839*	0.308758	0.310967	**0.313367**
7	football	**0.604570**	**0.604570**	**0.604570**	**0.604570**	**0.602872**	**0.604570**	**0.604570**
8	jazz	**0.445144**	**0.445144**	**0.445144**	*0.444469*	**0.445144**	**0.445144**	**0.445144**
9	C.Elegans N	**0.503782**	0.503485	0.498211	**0.503782**	0.502002	*0.503736*	**0.503782**
10	US Airports1997	–	0.214360	0.204418	**0.214688**	0.212227	*0.214531*	**0.214688**
11	C.Elegans M	**0.453248**	0.452867	0.446379	*0.453209*	0.441670	0.446602	0.448080
12	email	**0.582829**	0.582636	0.577651	*0.582792*	0.568329	0.576863	0.578422
13	polblogs	**0.427105**	**0.427105**	*0.427098*	0.427096	0.426937	0.427059	0.427081
14	US Airports2010	–	**0.275479**	0.273658	*0.275478*	0.274653	0.275458	0.275476
15	power	**0.940974**	*0.940714*	0.936614	0.939311	0.818490	0.880699	0.915185
16	PGPgiantcompo	**0.886647**	*0.886640*	0.884784	0.881017	0.838746	0.865027	0.874771
17	Krong500slogn16	*0.065661*	0.063362	0.059964	–	0.058143	0.064676	**0.066697**

The best results are in bold, and the second-best are in italic while ADVNDS, Combo, and GNNS25000 are by far the slowest, demonstrating however superior performance. GNNS2500 finds pretty good partitions for some networks, and it works much faster than ADVNDS and Combo. In general, its performance is comparable to Leiden. Both algorithms work quickly and find partitions with modularity just a bit below the best-known. Moreover, Combo could not handle the largest network, while GNNS2500 found a partition better than Leiden and did so almost six times faster. For that network, GNNS25000 finds the highest modularity score, outperforming all other methods, including ADVNDS.

While ADVNDS reported the highest known modularity scores for all the networks where it was applied except the largest one, it is much slower than GNNS, and its implementation is not available, so we could not test it on other networks. Besides, the current GNNS implementation uses pure Python, while implementing it in C++ (as done for all other algorithms) could provide further speed improvements.

Overall, while no single heuristic is the best solution for all the cases, a GNNS algorithm often finds a plausible solution, sometimes the best-known one, and provides a flexible parameter-controlled trade-off between speed and performance ranging from the fastest to the close-to-optimal performance, which makes it a valuable addition to an existing collection of algorithms. More importantly, since this simple GNN-style heuristic can perform comparably to the state-of-the-art, that serves as a proof-of-concept for considering more sophisticated GNN architectures, configurations, and learning techniques that could provide further improvement in solving community detection problem as well as other complex network optimization problems like minimum vertex cover, maximal independent set Li et al. (2018), clique partitioning, correlation clustering Jung and Keuper (2022), spectral clustering Bianchi et al. (2020), and others Bengio et al. (2021), Yow and Luo (2022).

Synthetic networks

In the next test, the methods were applied to the sets of synthetic networks – Lancichinetti–Fortunato–Radicchi (LFR) Lancichinetti et al. (2008) and Stochastic Block-Model (SBM) Holland et al. (1983). We built two sets of ten LFR networks of size 250 each, with overall average node degrees of $\overline{d} = 29.6$ and $\overline{d} = 13.3$, and three sets of ten SBM networks of size 300 for each value of the parameter ν, defining the ratio of the probability for the model graph to have inner-community edges (three communities of size 100 each) divided by the probability of the inter-community edges. For all SBM, networks the average probability of an edge was 0.1, leading to the average node degree $\overline{d} = 30$. The Python package networkx was used to generate these synthetic networks.

Table 5 shows the average values of modularity, normalized mutual information (NMI), and execution time obtained after partitioning ten instances of each type using the Leiden, Louvain, Combo (best of the 20 runs), and GNNS100 algorithms. As we can see, Combo demonstrates superior performance in terms of modularity in all sets of networks except for the last SBM set with the highest $\nu = 3$, where all algorithms performed achieve the same score. While GNNS100 demonstrates suboptimal performance

Table 5 Comparative evaluation of the Leiden, Louvain, Combo, and GNNS algorithms over the synthetic networks

Model	Leiden	Louvain	Combo	GNNS100
	Avg. modularity score			
LFR $\overline{d} = 29.6$	0.231638	0.230503	**0.231827**	0.231349
LFR $\overline{d} = 13.2$	0.323335	0.318014	**0.323993**	0.320207
SBM $\nu = 1.5$	0.159593	0.152365	**0.165762**	0.159826
SBM $\nu = 2.0$	0.175234	0.164898	**0.180759**	0.174756
SBM $\nu = 2.5$	**0.224523**	0.223458	**0.224523**	0.224508
SBM $\nu = 3.0$	**0.263608**	**0.263608**	**0.263608**	**0.263608**
	Avg. NMI			
LFR $\overline{d} = 29.6$	0.651277	0.641316	0.631941	**0.652583**
LFR $\overline{d} = 13.2$	**0.557155**	0.544109	0.511360	0.536949
SBM $\nu = 1.5$	0.050049	0.049271	0.036201	**0.052641**
SBM $\nu = 2.0$	0.335723	0.275103	**0.456881**	0.322522
SBM $\nu = 2.5$	**0.877335**	0.859546	**0.877335**	0.876657
SBM $\nu = 3.0$	**0.972523**	**0.972523**	**0.972523**	**0.972523**
	Avg. time, sec			
LFR $\overline{d} = 29.6$	0.534	0.268	6.309	**0.045**
LFR $\overline{d} = 13.2$	0.498	0.140	6.200	**0.043**
SBM $\nu = 1.5$	0.902	0.448	7.580	**0.051**
SBM $\nu = 2.0$	0.958	0.480	7.842	**0.060**
SBM $\nu = 2.5$	0.552	0.434	2.342	**0.041**
SBM $\nu = 3.0$	0.446	0.293	1.655	**0.050**

The best results are in bold

for those networks compared to Combo, it works at unparalleled speed, several times faster than Louvain and Leiden and two orders of magnitude faster than Combo. Based on that, GNNS100 proves itself to be the fastest solution for partitioning the synthetic networks demonstrating reasonable performance in terms of the modularity and NMI scores achieved.

Temporal networks

Earlier works established the applicability of the GNN architecture for capturing dynamic properties of the evolving graphs Ma et al. (2020). As GNNS is well-suited for the iterative partition improvement, it could be suggested for active learning of the temporal network partition. For example, initial warm-up training could be performed over the first temporal layers with subsequent tuning iterations while moving from a current temporal layer to the next one.

Below we apply the approach to the temporal network of the daily taxi mobility between the taxi zones in New York City (NYC). We use the 2016-2017 data provided by the NYC Taxi and Limousine Commission[4] to build the origin-destination network of yellow and green taxi ridership between the NYC taxi zones (edges of the network are weighted by the number of trips). The results of the temporal GNNS (GNNStemp)

[4] https://www1.nyc.gov/site/tlc/about/data.page

for each daily ridership network are compared against single runs (for the sake of speed) of the Louvain and Combo algorithms as the fastest and the most precise from the available algorithms. GNNStemp uses an initial warm-up over the year 2016 aggregated network and then performs a single run of 20 fine-tune iterations for each daily temporal layer in 2017, starting with the previously achieved partition. The achieved best modularity scores fluctuate slightly between the daily layers. The 2017 yearly average of the ratios of the daily scores achieved by each algorithm to the best score of all three algorithms for that day look as follows: 97.97% for Louvain, 99.99% for Combo, and 99.78% for GNNStemp. At the same time, the total elapsed time is as follows: 1.71 sec for Louvain, 16.04 sec for Combo, and 8.40 sec for GNNStemp. Furthermore, GNNStemp managed to find the best modularity score not reached by two other algorithms on 11.2% of the temporal layers.

So the performance of GNNStemp in terms of the achieved modularity score falls right in the middle between Louvain and Combo, while GNNStemp is nearly twice as fast as the single run of Combo.

Details of the day-by-day GNNStemp performance compared to the best of the three algorithms are shown in fig. 1. On some days, one may notice somewhat visible differences between the top algorithm performance and that of the GNNStemp, but GNNStemp captures the temporal pattern of the modularity score dynamics pretty well, with a correlation of 99.44% between the GNNStemp score and the top algorithm score timelines.

By performing time-series periodicity and trend-seasonality analysis, one could notice some interesting temporal patterns in the strength of the network community structure, as also presented in fig. 1. The community structure quantified by means of the best achieved modularity score demonstrates a strong weekly periodicity with a stronger community structure over the weekends, including Fridays. The strength of the community structure also shows noticeable seasonality, with stronger communities over the winter and weaker over the summer. One may relate this observation

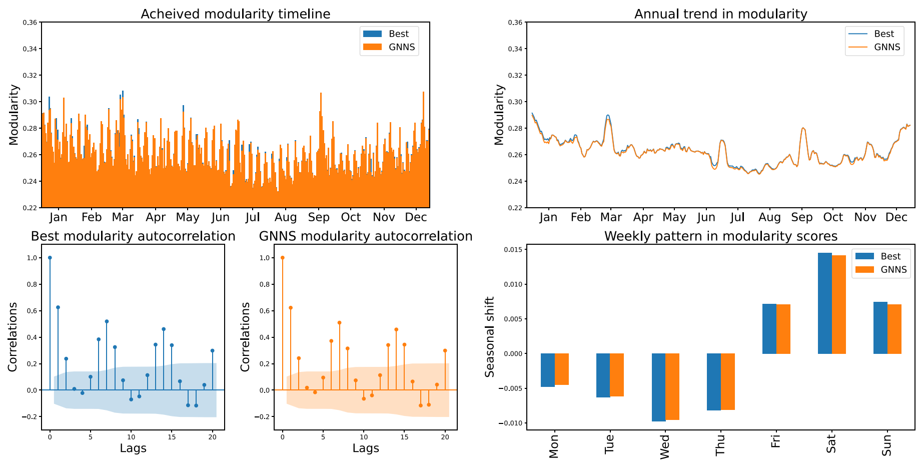

Fig. 1 Comparative performance of GNNS vs. the best of GNNS, Louvain, and Combo for community detection on the temporal network of daily taxi ridership in NYC. Subplots depict achieved network modularity and its time-series properties: autocorrelation, seasonality, and periodicity obtained by seasonal decomposition using moving averages

with the people exploring more destinations during their weekend and holiday time. As seen in fig. 1, the GNNStemp accurately reproduces the patterns discovered for the best partition timeline.

In conclusion, while Combo runs could demonstrate higher partition accuracy at certain temporal layers, the GNNStemp is capable of reaching similar performance much faster while adequately reproducing the qualitative temporal patterns. So GNNStemp could be trusted as a fast and efficient solution for extracting insights into the dynamics of the temporal network structure.

Conclusions

We proposed a novel recurrent GNN-inspired framework for unsupervised learning of the network community structure through modularity optimization. A simple iterative algorithm depends on only two variable parameters, and we propose an integrated technique for tuning those so that parameter selection and modularity optimization are performed within the same iterative learning process.

The algorithm's performance has been evaluated on classic network examples, synthetic network models, and a real-world temporal network case. Despite its simplicity, the new algorithm reaches similar and, in some cases, higher modularity scores compared to the more sophisticated discrete optimization state-of-the-art algorithms. One of the possible limitations of the proposed method is its dependence on the number of runs. However, this could be viewed as an advantage since it allows flexible adjustment of the algorithm's complexity, tuning the model parameters to find the right balance between speed and performance. At the low-complexity settings, it can significantly outperform alternative methods in terms of running time while maintaining reasonable performance in terms of partition quality. Thus, the algorithm is efficiently applicable in both scenarios—when the execution time is of the essence as well as when the quality of the resulting partition is a paramount priority.

Furthermore, the algorithm enables a special configuration for the active learning of the community structure on temporal networks, reconstructing all the important longitudinal patterns.

But more importantly, we believe the algorithm serves as a successful proof of concept for applying more advanced GNN-type techniques for unsupervised network learning, and opens possibilities for solving a broader range of network optimization problems. Similar approaches could work for such related problems as clique partitioning and correlation clustering. Applying more sophisticated model architectures along with constantly developing new methods for neural network training can possibly further improve the quality of found solutions. Developing these methods will constitute our future work in this direction.

Appendix. Results for directed networks

The analysis presented in the paper includes 17 classic sample networks. Their sources and characteristics (network size and whether the network is weighted and/or directed) are presented in the Table 2. Six of the 17 sample networks are directed in their original form. However Python iGraph implementations of the Leiden and Louvain methods handle only undirected networks, but Combo and GNNS

Table 6 Achieved modularity scores for the original directed configurations of the classical network examples

No	Name	best modularity score			
		Combo	GNNS100	GNNS2500	GNNS25000
1	Jazz	0.444787	0.445550	**0.445627**	**0.445627**
2	C.Elegans N.	**0.507642**	0.500875	**0.507642**	**0.507642**
3	US Airports1997	0.217195	0.212497	0.217545	**0.217800**
4	C.Elegans M.	**0.451758**	0.435311	0.448785	0.447451
5	polblogs	**0.432472**	0.432288	0.432366	0.432408
6	US Airports2010	**0.275524**	0.275496	0.275506	0.275518

The best results are in bold

implementations are capable of handling directed versions of the networks and in the Table 6 below we present the achieved modularity scores by Combo, GNNS100, GNNS2500, and GNNS25000 for the original directed versions of the sample networks. One can see that except of Jazz and US Airports1997, where Combo underperforms, all three methods reach closely similar modularity scores with a slight lead of Combo.

Abbreviations

GNN	Graph neural network
GNNS	GNN-style method
NYC	New York City
LFR	Lancichinetti–Fortunato–Radicchi
SBM	Stochastic block-model
NMI	Normalized mutual information

Author Contributions

SS designed the algorithm, performed initial analysis and wrote the first draft. AB implemented the algorithm, performed comparison with other methods, contributed to writing of the final paper. All authors read and approved the final manuscript.

Funding

This work was partially supported by the MUNI Award in Science and Humanities (MASH) of the Grant Agency of Masaryk University under the Digital City project (MUNI/J/0008/2021). This research/work/article was supported by ERDF "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16_019/0000822).

Availability of data and materials

All the data (sample networks) are taken from publicly available sources.

Declarations

Competing interests

The authors declare no competing interests.

Received: 24 October 2021 Accepted: 15 August 2022
Published online: 05 September 2022

References

Agarwal G, Kempe D (2008) Modularity-maximizing graph communities via mathematical programming. Eur Phys J B 66(3):409–418. https://doi.org/10.1140/epjb/e2008-00425-1

Aldecoa R, Marìn I (2011) Deciphering network community structure by surprise. PLoS one 6(9):e24195. https://doi.org/10.1371/journal.pone.0024195

Aloise D, Cafieri S, Caporossi G, Hansen P, Perron S, Liberti L (2010) Column generation algorithms for exact modularity maximization in networks. Phys Rev E 82(4):46112. https://doi.org/10.1103/PhysRevE.82.046112

Amini A, Kung K, Kang C, Sobolevsky S, Ratti C (2014) The impact of social segregation on human mobility in developing and industrialized regions. EPJ Data Sci 3(1):6
Baird D, Ulanowicz RE (1989) The seasonal dynamics of the Chesapeake Bay ecosystem. Ecol Monogr 59(4):329–364
Ball B, Karrer B, Newman MEJ (2011) Efficient and principled method for detecting communities in networks. Phys Rev E 84:036103. https://doi.org/10.1103/PhysRevE.84.036103
Belyi A, Bojic I, Sobolevsky S, Sitko I, Hawelka B, Rudikova L et al (2017) Global multi-layer network of human mobility. Int J Geogr Inf Sci 31(7):1381–1402
Belyi A, Sobolevsky S, Kurbatski A, Ratti C (2019) Improved upper bounds in clique partitioning problem. J Belarusian State Univ Math Inf 2019(3):93–104. https://doi.org/10.33581/2520-6508-2019-3-93-104
Bengio Y, Lodi A, Prouvost A (2021) Machine learning for combinatorial optimization: a methodological tour d'horizon. European J Oper Res 290(2):405–421. https://doi.org/10.1016/j.ejor.2020.07.063
Bickel PJ, Chen A (2009) A nonparametric view of network models and Newman-Girvan and other modularities. Proceed Natl Acad Sci 106(50):21068–21073
Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008(10):P10008
Bruna J, Li X (2017) Community detection with graph neural networks. Stat 1050:27
Clauset A, Newman MEJ, Moore C (2004) Finding community structure in very large networks. Phys Rev E 70:066111. https://doi.org/10.1103/PhysRevE.70.066111
Decelle A, Krzakala F, Moore C, Zdeborová L (2011) Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. Phys Rev E 84:066106. https://doi.org/10.1103/PhysRevE.84.066106
Decelle A, Krzakala F, Moore C, Zdeborová L (2011) Inference and phase transitions in the detection of modules in sparse networks. Phys Rev Lett 107:065701. https://doi.org/10.1103/PhysRevLett.107.065701
Duch J, Arenas A (2005) Community detection in complex networks using extremal optimization. Phys Rev E 72:027104. https://doi.org/10.1103/PhysRevE.72.027104
Džamić D, Aloise D, Mladenović N (2019) Ascent-descent variable neighborhood decomposition search for community detection by modularity maximization. Ann Oper Res 272(1):273–287. https://doi.org/10.1007/s10479-017-2553-9
Fortunato S (2010) Community detection in graphs. Phys Rep 486:75–174
Fortunato S, Barthélémy M (2007) Resolution limit in community detection. Proceed Natl Acad Sci 104(1):36–41. https://doi.org/10.1073/pnas.0605965104
Fortunato S, Hric D (2016) Community detection in networks: a user guide. Phys Rep 659:1–44
Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci USA 99(12):7821–7826
Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proceed Natl Acad Sci 99(12):7821–7826. https://doi.org/10.1073/pnas.122653799
Gleiser PM, Danon L (2003) Community structure in Jazz. Adv Complex Syst 06(04):565–573. https://doi.org/10.1142/S0219525903001067
Good BH, de Montjoye YA, Clauset A (2010) Performance of modularity maximization in practical contexts. Phys Rev E 81:046106. https://doi.org/10.1103/PhysRevE.81.046106
Grauwin S, Szell M, Sobolevsky S, Hövel P, Simini F, Vanhoof M et al (2017) Identifying and modeling the structural discontinuities of human interactions. Sci Rep 7(1):1–11
Guimerà R, Nunes Amaral LA (2005) Functional cartography of complex metabolic networks. Nature 433(7028):895–900. https://doi.org/10.1038/nature03288
Guimerà R, Danon L, Díaz-Guilera A, Giralt F, Arenas A (2003) Self-similar community structure in a network of human interactions. Phys Rev E 68:065103. https://doi.org/10.1103/PhysRevE.68.065103
Guimera R, Sales-Pardo M, Amaral LAN (2004) Modularity from fluctuations in random graphs and complex networks. Phys Rev E 70(2):025101
Hamann M, Strasser B, Wagner D, Zeitz T (2018) Distributed graph clustering using modularity and map equation. In: Aldinucci M, Padovani L, Torquati M (eds) Euro-Par 2018: parallel processing. Springer International Publishing, Cham, pp 688–702
Hastie T (2001) The elements of statistical learning : data mining, inference, and prediction : with 200 full-color illustrations. Springer, New York
Hawelka B, Sitko I, Beinat E, Sobolevsky S, Kazakopoulos P, Ratti C (2014) Geo-located Twitter as proxy for global mobility patterns. Cartogr Geogr Inf Sci 41(3):260–271
Holland PW, Laskey KB, Leinhardt S (1983) Stochastic blockmodels: first steps. Soc Networks 5(2):109–137
Javed MA, Younis MS, Latif S, Qadir J, Baig A (2018) Community detection in networks: a multidisciplinary review. J Network Comput Appl 108:87–111
Kampffmeyer M, Løkse S, Bianchi FM, Livi L, Salberg AB, Jenssen R (2019) Deep divergence-based approach to clustering. Neural Networks 113:91–101. https://doi.org/10.1016/j.neunet.2019.01.015
Karrer B, Newman MEJ (2011) Stochastic blockmodels and community structure in networks. Phys Rev E 83:016107. https://doi.org/10.1103/PhysRevE.83.016107
Lancichinetti A, Fortunato S, Radicchi F (2008) Benchmark graphs for testing community detection algorithms. Phys Rev E 78(4):046110
Landsman D, Kats P, Nenko A, Sobolevsky S (2020) Zoning of St. Petersburg through the prism of social activity networks. Procedia Comput Sci 178:125–133
Lee J, Gross SP, Lee J (2012) Modularity optimization by conformational space annealing. Phys Rev E 85:056702. https://doi.org/10.1103/PhysRevE.85.056702
Liu X, Murata T (2010) Advanced modularity-specialized label propagation algorithm for detecting communities in networks. Phys A Stat Mech Appl 389(7):1493–1500. https://doi.org/10.1016/j.physa.2009.12.019
Lu H, Halappanavar M, Kalyanaraman A (2015) Parallel heuristics for scalable community detection. Parallel Comput 47:19–37. https://doi.org/10.1016/j.parco.2015.03.003

Lusseau D, Schneider K, Boisseau OJ, Haase P, Slooten E, Dawson SM (2003) The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. Behav Ecol Sociobiol 54(4):396–405. https://doi.org/10.1007/s00265-003-0651-y
Newman MEJ (2004) Fast algorithm for detecting community structure in networks. Phys Rev E 69:066133. https://doi.org/10.1103/PhysRevE.69.066133
Newman MEJ (2006) Finding community structure in networks using the eigenvectors of matrices. Phys Rev E 74:036104. https://doi.org/10.1103/PhysRevE.74.036104
Newman MEJ (2006) Modularity and community structure in networks. Proceed Nat Academ Sci 103(23):8577–8582
Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69(2):026113
Piccardi C, Tajoli L (2012) Existence and significance of communities in the World Trade Web. Phys Rev E. https://doi.org/10.1103/PhysRevE.85.066119
Raghavan UN, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E 76:036106. https://doi.org/10.1103/PhysRevE.76.036106
Ratti C, Sobolevsky S, Calabrese F, Andris C, Reades J, Martino M et al (2010) Redrawing the Map of Great Britain from a Network of Human Interactions. PLoS one 5(12):e14248. https://doi.org/10.1371/journal.pone.0014248
Rossetti G, Milli L, Cazabet R (2019) CDLIB: a python library to extract, compare and evaluate communities from complex networks. Appl Network Sci 4(1):1–26. https://doi.org/10.1007/s41109-019-0165-9
Rosvall M, Bergstrom CT (2007) An information-theoretic framework for resolving community structure in complex networks. Proceed Natl Acad Sci 104(18):7327–7331. https://doi.org/10.1073/pnas.0611034104
Rosvall M, Bergstrom CT (2008) Maps of random walks on complex networks reveal community structure. Proc Natl Acad Sci USA 105:1118–1123
Sobolevsky S, Szell M, Campari R, Couronné T, Smoreda Z, Ratti C (2013) Delineating geographical regions with networks of human interactions in an extensive set of countries. PloS one 8(12):e81707
Sobolevsky S, Campari R, Belyi A, Ratti C (2014) General optimization technique for high-quality community detection in complex networks. Phys Rev E 90(1):012811
Traag VA, Waltman L, Van Eck NJ (2019) From Louvain to Leiden: guaranteeing well-connected communities. Sci Rep 9(1):1–12. https://doi.org/10.1038/s41598-019-41695-z
Watts DJ, Strogatz SH (1998) Collective dynamics of 'small-world' networks. Nature 393(6684):440–442
Weisfeiler B, Leman A (1968) The reduction of a graph to canonical form and the algebra which appears therein. NTI, Series 2(9):12–16
White JG, Southgate E, Thomson JN, Brenner S (1986) The structure of the nervous system of the nematode caenorhabditis elegans. Philos Trans Royal Soc London B Biol Sci 314(1165):1–340. https://doi.org/10.1098/rstb.1986.0056
Xu Y, Li J, Belyi A, Park S (2021) Characterizing destination networks through mobility traces of international tourists - a case study using a nationwide mobile positioning dataset. Tour Manag. https://doi.org/10.1016/j.tourman.2020.104195
Yan X, Shalizi C, Jensen JE, Krzakala F, Moore C, Zdeborová L et al (2014) Model selection for degree-corrected block models. J Stat Mech Theory Exp 2014(5):P05007
Zachary WW (1977) An information flow model for conflict and fission in small groups. J Anthropol Res 33:452–473
Adamic LA, Glance N (2005) The political blogosphere and the 2004 U.S. election: divided they blog. In: Proceedings of the 3rd international workshop on Link discovery. LinkKDD '05. New York, NY, USA: ACM; p 36–43. Available from: http://doi.acm.org/10.1145/1134271.1134277. https://doi.org/10.1145/1134271.1134277
Aloise D, Caporossi G, Hansen P, Liberti L, Perron S, Ruiz M (2012) Modularity maximization in networks by variable neighborhood search. Graph Partitioning and Graph Clustering. 588(113)
Bandyopadhyay S, Peter V (2020) Self-expressive graph neural network for unsupervised community detection. arXiv preprint arXiv:2011.14078
Barber MJ, Clark JW (2009) Detecting network communities by propagating labels under constraints. Phys Rev E. 80, 026129. https://doi.org/10.1103/PhysRevE.80.026129
Belyi A, Sobolevsky S (2022) Network Size Reduction Preserving Optimal Modularity and Clique Partition. In: Gervasi O, Murgante B, Hendrix EMT, Taniar D, Apduhan BO (eds). Computational science and its applications – ICCSA 2022. Cham: Springer International Publishing; p 19–33. https://doi.org/10.1007/978-3-031-10522-7_2
Belyi A, Sobolevsky S, Kurbatski A, Ratti C (2021) Subnetwork Constraints for Tighter Upper Bounds and Exact Solution of the Clique Partitioning Problem. arXiv preprint arXiv:2110.05627
Bianchi FM (2022) Simplifying clustering with graph neural networks. arXiv preprint arXiv:2207.08779
Bianchi FM, Grattarola D, Alippi C (2020) Spectral Clustering with Graph Neural Networks for Graph Pooling. In: III HD, Singh A, editors. In: Proceedings of the 37th international conference on machine learning. vol. 119 of Proceedings of Machine Learning Research. PMLR; p 874–883. Available from: https://proceedings.mlr.press/v119/bianchi20a.html
Biedermann S, Henzinger M, Schulz C, Schuster B (2018) Memetic Graph Clustering. In: D'Angelo G, editor. 17th International Symposium on Experimental Algorithms (SEA 2018). vol. 103 of Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. p. 3:1–3:15. Available from: http://drops.dagstuhl.de/opus/volltexte/2018/8938. https://doi.org/10.4230/LIPIcs.SEA.2018.3
Blondel V, Krings G, Thomas I (2010) Regions and borders of mobile telephony in Belgium and in the Brussels metropolitan zone. Brussels Studies La revue scientifique électronique pour les recherches sur Bruxelles/Het elektronisch wetenschappelijk tijdschrift voor onderzoek over Brussel/The e-journal for academic research on Brussels
Boguñá M, Pastor-Satorras R, Díaz-Guilera A, Arenas A (2004 Nov) Models of social networks based on social distance attachment. Phys Rev E. 70:056122. https://doi.org/10.1103/PhysRevE.70.056122
Brandes U, Delling D, Gaertler M, Görke R, Hoefer M, Nikoloski Z, et&nbsp;al (2006) Maximizing modularity is hard. arXiv preprint physics/0608255
Chen Z, Li X, Bruna J (2017) Supervised community detection with line graph neural networks. arXiv preprint arXiv:1705.08415
Jung S, Keuper M (2022) Learning to solve minimum cost multicuts efficiently using edge-weighted graph convolutional neural networks. arXiv preprint arXiv:2204.01366

Kang C, Sobolevsky S, Liu Y, Ratti C (2013) Exploring human movements in Singapore: A comparative analysis based on mobile phone and taxicab usages. In: Proceedings of the 2nd ACM SIGKDD international workshop on urban computing. ACM; p&nbsp;1
Khan BS, Niazi MA (2017) Network community detection: a review and visual survey. arXiv preprint arXiv:1708.00977
Knuth DE (1993) The Stanford GraphBase: a platform for combinatorial computing. Addison-Wesley; Available from: http://www-cs-staff.stanford.edu/~uno/sgb.html
Landsman D, Kats P, Nenko A, Kudinov S, Sobolevsky S (2021) Social activity networks shaping St. Petersburg. In: Proceedings of the 54th Hawaii international conference on system sciences; p 1149
Li Z, Chen Q, Koltun V (2018) Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (eds). Advances in neural information processing systems. vol&nbsp;31. Curran Associates, Inc. p 1–10. Available from: https://proceedings.neurips.cc/paper/2018/file/8d3bba7425e7c98c50f52ca1b52d3735-Paper.pdf
Lobov I, Ivanov S (2019) Unsupervised community detection with modularity-based attention model. arXiv preprint arXiv:1905.10350
Ma Y, Guo Z, Ren Z, Tang J, Yin D (2020) Streaming graph neural networks. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval; p 719–728
Plantié M, Crampes M (2013) Survey on social community detection. In: Social media retrieval. Springer; p 65–85
Sanders P, Schulz C, Wagner D (2014) Benchmarking for graph clustering and partitioning. Encyclopedia of social network analysis and mining Springer
Shchur O, Günnemann S (2019) Overlapping community detection with graph neural networks. arXiv preprint arXiv:1909.12201
Sobolevsky S, Sitko I, Des&nbsp;Combes RT, Hawelka B, Arias JM, Ratti C (2014) Money on the move: Big data of bank card transactions as the new proxy for human mobility patterns and regional delineation. the case of residents and foreign visitors in spain. In: Big data (BigData Congress), 2014 IEEE international congress on. IEEE; p 136–143
Sobolevsky S, Belyi A, Ratti C (2017) Optimality of community structure in complex networks. arXiv preprint arXiv:1712.05110
Sobolevsky S, Kats P, Malinchik S, Hoffman M, Kettler B, Kontokosta C (2018) Twitter Connections Shaping New York City. In: Proceedings of the 51st Hawaii international conference on system sciences. p 1008–1016
Sun Y, Danila B, Josić K, Bassler KE (2009) Improved community structure detection using a modified fine-tuning strategy. EPL (Europhysics Letters). 86(2):28004. Available from: http://stacks.iop.org/0295-5075/86/i=2/a=28004
Tsitsulin A, Palowitch J, Perozzi B, Müller E (2020) Graph clustering with graph neural networks. arXiv preprint arXiv:2006.16904
Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY (2020) A comprehensive survey on graph neural networks. In: IEEE transactions on neural networks and learning systems
Yow KS, Luo S (2022) Learning-based approaches for graph problems: a survey. arXiv preprint arXiv:2204.01057

Publisher's Note