Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Clustering Node Attributed Networks with Graph Neural Networks and Self Learning

Rodrigo Luna

rluna@cos.ufrj.br

Federal University of Rio de Janeiro https://orcid.org/0009-0002-3866-2491

Daniel Ratton Figueiredo

daniel@cos.ufrj.br

Federal University of Rio de Janeiro https://orcid.org/0000-0001-9341-6619

Research Article

Keywords: graph clustering, attributed networks, graph neural networks, self learning

Posted Date: June 4th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9902966/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Clustering Node Attributed Networks with Graph Neural Networks and Self Learning

Rodrigo de Sapienza Luna[a,*], Daniel Ratton Figueiredo[a,*]

[a]*Systems Engineering and Computer Science (PESC), Federal University of Rio de Janeiro (UFRJ), Technology Center, Block H, Room 319, 68511, Cidade Universitária, Rio de Janeiro, Brazil*

ARTICLE INFO

Keywords:
graph clustering, attributed networks, graph neural networks, self learning

ABSTRACT

Graph clustering – partitioning the node set of a graph into disjoint subsets that reflect some latent information – is a fundamental problem as it finds applications in a myriad of different scenarios. While this classic problem has been tackled for decades by different communities, a recent variation of the problem driven by real data considers the scenario where nodes have attributes that are also informative. This has triggered novel methods that simultaneously leverage network information (edges) and node information (attributed) in the design of novel clustering algorithms. This work proposes a novel framework that builds on prior works that have applied graph neural networks (GNN) to graph clustering. The proposed framework operates in rounds of self learning in a fully unsupervised setting. In each round, a GNN generates representations for nodes that are used to cluster the nodes. This clustering influences the graph used to generate the node representation in the next round. Moreover, a context graph built in each round using the original graph is used to generate the node representations. Empirical results show that the proposed methodology extracts information from both network edges and node attributes in synthetic data, outperforming algorithms focused solely on the network or attributes when neither are very informative. Multiple rounds of learning also improve the performance and always outperforms a long single round of training (i.e., classic GNN graph clustering). When considering real datasets, empirical results indicate that the proposed methodology is competitive to state-of-the-art methods when cluster sizes are balanced.

1. Introduction

Graph clustering or graph partitioning or community detection consists of partitioning the set of nodes of a graph into disjoint subsets that reflect some higher order structure (e.g., many more edges within the subsets than across subsets) [21, 9]. This is a fundamental problem in data science and network science since it finds applications in a myriad of contexts such as social networks, biological networks and information networks [17]. Not surprising, graph clustering has been investigated for decades by various communities using different problem formulations and approaches (e.g., fast heuristics for minimizing modularity in network science, or approximation algorithms for optimal ratio-cut in graph theory).

A related but even more fundamental problem is data clustering where a set of points in some space must be partitioned into disjoint subsets that reflect some higher order structure (e.g. points in a subset are much closer than points in different subsets) [3]. Data clustering finds even more applications as the input are just data points (and not a network) and an extensive literature in various communities. Indeed, the k-means algorithm was proposed over 50 years ago and is perhaps the most ubiquitous approach to tackle data clustering.

While both graph and data clustering are classic, only recently have they been unified into a single formulation [7, 22]. In attributed networks, each network node is associated with some feature that is representative of the node (e.g., the DNA of the person in a social network). Suppose such feature is a point in some space. Note that nodes of this network can be clustered using just the node attributes (data clustering), just the network structure (graph clustering), or both kinds of information (joint data and graph clustering). This work considers the latter problem.

Jointly using node attributes and network structure to cluster nodes of an attributed network requires mixing evidence from the attributes with evidence from the network structure. Intuitively, such approaches should outperform any approach that uses just one kind of information, and performance gains should be higher when both kinds of information are relatively noisy (and independent). In this scenario, the joint approach can use more reliable attribute information to correct for noisy network information, and vice-versa. Of course, designing methodologies and building algorithms that can leverage this intuition is not trivial, and has been an active area of research over the past few years [7, 22, 29, 23].

An important consideration concerning graph or data clustering is prior information, or supervision. In a supervised setting, the labels of nodes or data points are available and indicate their clusters (or partially available in a semi-supervised setting). This information can be used to train a model that determines the cluster of unseen nodes. In contrast, in an unsupervised setting there is no prior information concerning the clusters. In modern and large

[*] This work received financial support through research grants provided by CNPq and FAPERJ (Brazil).

[*]Corresponding author

rluna@cos.ufrj.br (R.d.S. Luna); daniel@cos.ufrj.br (D.R. Figueiredo)

https://www.cos.ufrj.br/~daniel/ (D.R. Figueiredo)

ORCID(s):

problem instances, it is often the case that label information is not available (or is not reliable). This work considers the unsupervised scenario. The sole input to the problem here considered is a single instance of an attributed network.

Graph Neural Network (GNN) can be interpreted as a framework to generate representation for nodes using both the network structure and node attributes [15]. Such representations are points in some dimensional space and reflect some higher order structure of the network nodes (their local network structure and attributes). GNN has been successfully applied to node classification and link prediction in various scenarios. Not surprising, GNN has also recently been applied to clustering nodes of attributed network (see Section 2). In a nutshell, the representations generated by the GNN can be used as the input to a data clustering problem. However, when representations are generated as to optimize a generic objective function, such naive approaches often do not exhibit good performance. Most alternative solutions use a supervised (or semi-supervised) scenario where the label information can be used in the objective function or in some other stage. More recently, fully unsupervised approaches have also been proposed (more details in Section 2).

This work proposes a novel fully unsupervised framework for clustering attributed networks, called Dynamic Context Self-Learning Graph Neural Network (DCSL-GNN). The framework operates in rounds and its main contribution is the self-learning component and generation of the context over the rounds. Using the representation and clusters of the previous round as well as the original network, the context (a set of nodes of fixed size) for each node is randomly generated. This context is used to generate a representation for nodes via a GNN (where "neighbors" are the nodes in the context). These representations are the input to a data clustering problem in Euclidean space, that can be solved with the classic k-means algorithm. These clusters and representations are then used to determine the context graph for the next round and their new representations. In a nutshell, representation and clusters found in the previous round influence the context generated for the nodes in the current round that is used by a GNN to generate new node representations, that are then used to determine the clusters of the current round.

The multiple rounds allows the framework to learn the right context for the nodes, leading to a better representation and consequently to a more effective clustering. In essence, DCSL-GNN employs a self-reinforcement mechanism: better representations generate better clusters, while better clusters generate better representations. The idea of node context is related to attention mechanisms in GNNs where edges incident to a node are weighted to focus on a subset of its neighbors. However, in DCSL-GNN any node can be in the context of another node, and not only a subset of its neighbors (as in attention mechanisms). This allows for nodes that are central in a cluster to be in the context of other nodes that belong to this cluster even when they are not neighbors in the original graph.

The framework is implemented and results on synthetic networks indicate its effectiveness in extracting information from both the network and attributes to generate effective clusters across the rounds of self-learning. Results clearly indicate that multiple rounds of self-learning are beneficial, and performance can be good even when one of the sources of information is very noisy (network or attributes). An evaluation using three benchmark datasets, DCSL-GNN outperforms all state-of-the-art algorithms in one of the datasets, indicating its potential in delivering good performance in the wild.

2. Related Work

In this section, we review the related work in two categories that are related to our framework.

2.1. Community Detection

Community detection (or graph clustering) is one of most important topics of modern network science, with practical application across various domains. Researchers from different fields have explored this problem for many years, employing various problem formulations and approaches, such as fast heuristics for minimizing modularity in network or approximating algorithms for optimal ratio-cut in graph theory [8].

Modularity is a metric to measure the quality of a community (cluster) partition of a network using the difference between the number of edges within communities and the expected number of such edges across all pairs of nodes [18]. Modularity optimization is the most used technique in community detection, and Newman and Girvan [18] proposed the first heuristic to find communities that optimize modularity. Another popular community detection algorithm is the Louvain algorithm proposed by Blondel et al. [4] which employs a greedy and efficient strategy to optimize the network modularity.

2.2. Graph Neural Network for clustering

Recently, Graph Neural Networks (GNNs) have been introduced to address community detection on attributed graphs. Many approaches employ GNNs as node encoders to leverage both structural and node attribute information. The learning process is achieved through the decoder process, where the model parameters are learned by comparing the difference between the original graph and the reconstructed graph. After obtaining the node embeddings through the encoder-decoder process, these embeddings are typically used as input to a cluster algorithm to detect communities within the graph. For example, Chu Wan et al. [24] utilize a GNN encoder-decoder architecture for community detection. However, some approaches have introduced contrastive methods with specifics loss functions with the objective to learning community structures. Zhang et al. [29] introduced SENet, which utilizes a spectral clustering loss to capture the global cluster structure and learn node embedding. Bo et al. [6] developed a graph reconstruction-based clustering method that employs dual self-supervised

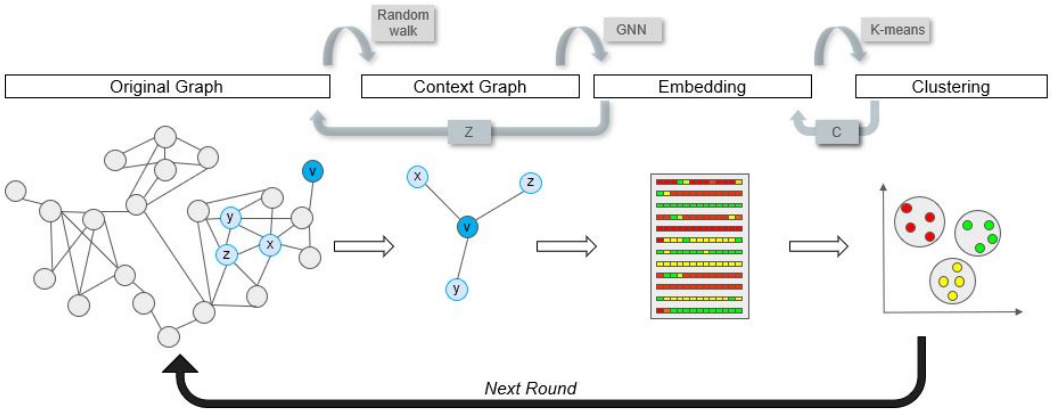


Figure 1: Schematic representation of DCSL-GNN framework. Note that the clustering result is used to determine the edge weights of the original graph in the next round that is then used to generate the next context graph. Thus, embeddings influence clustering, and clustering influences embeddings.

learning to optmize the model jointly with the encoder-decoder. CommDGI [27] proposes a combined objective of three loss functions: mutual information, modularity, and contrastive loss. DMoN [23] is another framework that learns the model parameters through a loss function based on modularity. Additionally, SEComm [2] combines the self-expressiveness with a self-supervised GNN for community detection through the optimization of two combined loss function. Last, DCOM-GNN [25] proposes a methodology to enhance the embeddings generated by an arbitrary GNN. The idea is to learn weights and matrices that manipulate the original GNN embedding to better discriminate inter-cluster and intra-cluster representations.

Our work introduces a novel approach to learning node representations for community detection in attributed graphs. Unlike most methods in the literature, the proposed framework does not rely on specific loss functions related to clustering. Instead, we employ a dynamic graph and a self-learning module that iterative optimize the node embeddings. This approach allows to refine the dynamic graph based on the evolving node representations, enhancing community detection performance.

3. Framework

In this section, we present our proposed framework Dynamic Context Self-Learning Graph Neural Network (DCSL-GNN). As shown in Fig 1 our model contains three main modules: Dynamic Context, Embedding and Clustering.

These modules are jointly learn in end-to-end manner to benefits each other.

3.1. Problem Formulation

Given an undirected attributed graph $G = (V, E, X)$, where $V = \{v_1, v_2, \dots, v_n\}$ is a set of n nodes and E is a set of edges with $e_{ij} = (v_i, v_j) \in E$ if an edge exists between nodes v_i and v_j, and $X = \{x_1, x_2, \dots, x_n\} \in \mathbb{R}^{n \times F}$ is the attribute matrix, where each node v_i is associated by an attribute vector x_i with dimension F. Our goal is to partition the nodes into disjoint sets accordingly to the topological structure and the nodes attributes. Thus, we want to learn a function $f : V \rightarrow C$ where $C = \{C_1, C_2, \dots, C_K\}$ is the set of communities (or clusters), which map each node into a community C_i. We want to achieve the partitioning in the unsupervised way, i.e, without having any ground truth community information of a node, just knowing the number of communities.

Notations	Definition
$G = (V, E, X)$	Attributed graph
$G_c = (V_c, E_c, X)$	Context graph
$X \in \mathbb{R}^{n \times F}$	Node feature matrix
$H(\cdot)$	Embedding for node v
W	Learning matrix
$C^l \in \mathbb{R}^K$	Community sets at l-th round
$\|\|X, Y\|\|$	Euclidean distance between X and Y
$L(\theta)$	Triplet loss function
K	Number of communities
$N(v)$	Neighborhood of node v
$s(v)$	Silhouette value of node v
n_e	Number of epochs
n_u	Mini-batch size

Table 1
Notation.

Important notation is described in Table 1.

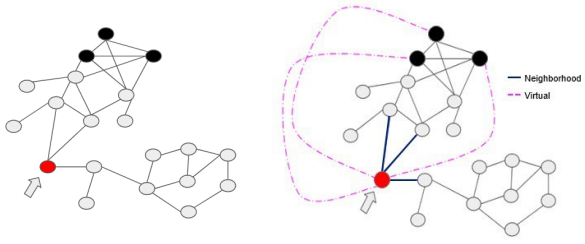
3.2. Dynamic Context

GNNs aim to learn each node's embeddings by integrating their attributes with graph structure. In GNN, a layer can be viewed as a message passing between nodes, where each node updates its latent representation by aggregating the messages from its direct neighbors.

In the context of the graph clustering problem, nodes most representative of a cluster could be more than K hop distance. This imposes a challenge for GNNs to effectively

propagate information from these nodes to others. The reason behind this challenge arises from the fact that, with the objective of allowing a node to receive information from nodes within a radius of K, the depth of GNN layer needs to be at least K. Without this depth, the information propagation is constrained, resulting in a phenomenon known as under-reaching [5]. Nevertheless, unlike conventional neural network algorithms, when additional layers are stacked in GNNs, the nodes representation tends to become increasingly indistinguishable. Consequently, this leads to a significant degradation in prediction accuracy and overall performance, known as over-smoothing [12, 1, 13]. For that reason, most GNN models have very few layers, such as 2 or 3, and often deploy mechanism to avoid over-smoothing.

In order to avoid under-reaching and over-smoothing, we proposed changes in the graph that feeds the GNN. The original node neighborhood may not adequately represent the propagation of cluster signals. Thence, we design a flexible neighborhood sampling strategy biased by the similarity of nodes in representation space. To capture cluster information, we add virtual edges to the closest nodes of cluster center mass. This approach allows both the information from cluster center through virtual edges and local structure of a node from original edges to be directly aggregated into a node, this new graph built is known as Context Graph (G_c). As mentioned, DCSL-GNN is a self-learning framework trained in end-to-end manner. This means in order to achieve better clustering performance, the nodes embeddings and soft-labels assignment in previous round are employed to guide the optimization procedure of current round. These estimated parameters in current round are use to generate the context of nodes in next round.



(a) Original Graph (b) Random walk perspective

Figure 2: In each step of the random walk, a decision is made to either follow a virtual edge towards the cluster center nodes or explore the neighborhood of the current node.

DCSL-GNN utilizes random walks as a mechanism to generate the sequence of nodes that define the context of a given node, a technique adopted by severals works [10, 26, 19]. This context will determine a new set of edges

We propose a random walk which reflects the similarities between node clusters, meaning that a node will transition to another node if they exhibit a higher degree of similarity in terms of their cluster membership. Figure 2 introduces the concept of virtual edges, where at each step of the random walk, a decision is made to either traverse a real edge in original graph or select a virtual edge leading to set of nodes nearest at the cluster center of mass $CC(\cdot)$. This choice is determined by a Bernoulli process, where an original edge is taken with a probability of p, and a virtual edge is chosen with probability $(1 - p)$. If a virtual edge is selected, the node to which the random walk transitions is chosen uniformly at random from the set of nodes closest to the cluster's mass center, with the $|CC(\cdot)|$ defined by a parameter. Otherwise, when selecting an original edge, the random walk explores the neighborhood based on proximity in the embedding space, resulting in a biased random walk where edge weights are inversely proportional to distances between their representations.

To guide the random walk in the original graph, an edge weight function denoted by $w_H(\cdot)$ use the nodes representations generated by the previous round of DCSL-GNN, to bias the walk. It's worth noting that in the first round, transitions are made uniformly within the node's neighborhood. The $w_H(\cdot)$ value rises as the Euclidean distance between node representations diminishes. This characteristic steers the random walk towards nodes that are more inclined to connect to nodes with similar embedding, ensuring that the exploration of the random walk focuses on nodes that are closer in the embedding space.

The Equation 1 presents the edge weight function in embedding space. Note that the value $\max_{z \in N(v)} ||H(v) - H(z)||$ represents the longest distance from node v to its neighbors in the embedding space. The value tends to be higher for nodes closer to v, approaching $\max_{z \in N(v)} ||H(v) - H(z)||$, and tends to be lower for nodes that are farther away, approaching 0 (zero for the farthest node). In essence, the weight function $w_H(\cdot)$ corresponds to the Euclidean distance relative to node v.

$$w_H(v,u) = \max_{z \in N(v)} (||H(v) - H(z)||) - ||H(v) - H(u)|| \quad (1)$$

$H(\cdot)$ is is the latent node representation for the nodes in euclidean space.

Given a node u, we simulate multiples random-walks of length L biased by $w_H(\cdot)$ in order to select nodes to compose the context for node u. Let c_i denote the i-th node in the walk, starting with $c_0 = u$. Nodes c_i are generated by the following distribution:

$$P(c_i = k | c_{i-1} = v) = \begin{cases} w_H(v,k)/Z_v, & k \in N(v), \text{w.p. } p \\ 1/T, & k \in CC(v), \text{w.p. } 1-p \end{cases} \quad (2)$$

where $CC(u)$ is the set of nodes nearest to the cluster mass center of node u, and Z_v is the normalization factor for vertex v given by:

$$Z_v = \sum_{i \in N(v)} w_H(v,i) \quad (3)$$

To determine the neighborhood of node u in the Context Graph (Gc), denoted as $Nc(u)$, we adopted Importance-based neighborhoods, as proposed in the PinSage algorithm [26]. This method defines the neighborhood of a node u as the T nodes that exert the most influence on node u. As the embeddings are influenced by both the network structure and attributes, using them to bias the random walks generates a sequence of nodes that are strongly correlated with the clusters they belong to, due to the homophily phenomenon. In particular, the most visited nodes will be the neighbors of node u. Thus, the visits counts of the random walks are used to select the most important nodes. Since each source node $u \in V_G$ is associated with an independent non-identically visit count distribution, we employ a threshold value to determine if a visited node will be included in the new neighborhood. This threshold value is defined as the k-th percentile of the distribution, determined by a parameter. Consequently, the neighborhood of node u, denoted as $N_c(u)$, in the Context Graph is comprised of nodes that have been visited at least the value of the k-th percentile.

Note that constructing node neighborhoods based on importance has several advantages. First, the selection of the most important nodes is based by visits counts, it enables nodes with high visit counts being direct neighbors in the Context Graph, even when the node is many hops away, consequently, it reduces the noise when passing messages between them. Second, a few layers stacked in GNN in the original graph could already incur unwelcome messages between clusters, however this is less likely to occurs in the context graph, because the neighborhood contains only similar nodes that are likely to belongs to the same clusters. Third, as only the most important nodes are chosen, it allows to reduce the neighbourhood size without loss of generality and helps to decrease the algorithm memory's consumption.

3.3. Weights in Context Graph

Initially, all edges in G_c are assigned a weight value of 1, implying that all messages from the neighborhood hold equal importance during the aggregation step. However, due the stochastic nature of context graph across multiples rounds, uncertain edges may be included in the neighborhood in the neighborhood, potentially introducing noise ad mixing their latent representations. Furthermore, nodes that exhibit high similarity in their attributes should contribute with greater importance during the aggregation step. In this way, an edge $e(u, v)$ is deemed uncertain when the attributes from nodes u and v lack correlation, and at least one of them is highly correlated with the attributes of the cluster to which it belongs.

The Silhouette coefficient algorithm [20] is employed to quantify the cohesion of attributes within a cluster. This coefficient measures the similarity of a data point to its own cluster compared to other clusters. It considers both the distance between the data point and other points within its own cluster (cohesion) and the distance between the data point and points in other clusters (separation).

Assuming the node embeddings have been clustered into C clusters using any clustering algorithm, the silhouette coefficient for each data point i can be defined as follows:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{4}$$

Where $a(i)$ is the average intra-cluster distance, i.e the average distance between each point within the cluster of node i and $b(i)$ is the average inter-cluster distance, which is the average distance between the node i and all points in other clusters, defined as follows:

$$a(i) = \frac{1}{|C_{z_i}| - 1} \sum_{j \in C_{z_i}, i \neq j} d(i, j) \tag{5}$$

$$b(i) = \min_{y \neq z_i} \frac{1}{C_y} \sum_{j \in C_y} d(i, j) \tag{6}$$

Where C_{z_i} is the cluster of node i and z_i is the cluster number of node i.

For each node u in a cluster $-1 \leq s(u) \leq 1$. A silhouette value closer to 1 indicates that the data is appropriately clustered, a value closer by -1 indicates poor clustering, and a value closer to 0 suggests the value is on the border between multiples clusters.

The dynamic nature of context graph enables the utilization of the previous clustering results to guide the current optmization process. DCSL-GNN evaluates the G_c edges weigths in current process with the cluster obtained by the previous round. However, this optmization process might take a considerable amount of rounds to discover consistent clusters. Thus, an attribute node i $X[i]$ is considered properly clustered only if $s(i) \geq \beta$, where is β is a threshold value. If one of the nodes from an edge satisfies this condition, the edge weight value will depend on the inverse Euclidean distance between the attribute values of the nodes, otherwise the weight is considered 1. The following function captures this idea:

$$e(j, i) = \begin{cases} \min\left(\frac{1}{\|X[i], X[j]\|}, \alpha\right), & \text{if } s(i) \geq \beta \text{ or } s(j) \geq \beta \\ 1 \end{cases} \tag{7}$$

An edge weight $e(i, j)$ with nodes attributes $X[i]$ and $X[j]$ assumes values between $[0, \alpha]$. It is important to note that α is a parameter which controls the upper bound of the weight, in order to prevent the edge weight from approaching infinity and dominating the integration. Therefore, when $e(i, j) < 1$, the edge is penalized as it reduces the importance between the nodes. Conversely, when $e(i, j) > 1$, it increases the influence in the messages between them. Finally, when $edge(i, j) = 1$ the nodes clustering are not reliable with their own clusters and the edge do not contribute the latant representation. Last, these edge weights is part of the input to the graph neural network, along with node attributes.

3.4. Embedding

The encoder proposed in [12] was adopted in the Embedding Module. Given a graph G with a adjacency matrix A a stack layer of the model is defined as follow:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{8}$$

Where $\tilde{A} = A + I$, where $I \in \mathbb{R}^{n\times n}$ is the identity matrix with goal to add self-loops and the degree diagonal matrix with $\tilde{D}_{ii} = \sum_{j\in V}\tilde{A}_{ij}$. The matrix W^l of the l-th layer is a trainable parameter in GCN and σ is the activation function, we use $ReLu$ in ours experiments. In the last layer of embedding matrix $H^l \in \mathbb{R}^{n\times F}$ is the representation matrix and output from the previous layer and contains in each row the corresponding node representation with dimension F.

The operator can be defined in the perspective of a node as follow:

$$H_i^{(l+1)} = W^{(l)}\sum_{j\in N(i)}\frac{e_{j,i}}{\sqrt{\tilde{d}_j\tilde{d}_i}}H_j^{(l)} \tag{9}$$

where $e_{j,i}$ denotes the weight edge value from source node j to target node i and $\tilde{d}_j$ denotes the node j degree.

3.5. Clustering

We apply the K-means clustering algorithm on the data points provided by the Embedding module. Given a set observations $X = \{x_1, \ldots, x_n\}$ this algorithm aims to partition the data points into K disjoint sets $C = \{C_1, \ldots, C_k\}$ with go to minimize the objective function bellow:

$$L = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}||x_n - \mu_k||^2 \tag{10}$$

K-means aims to optimize the parameters μ_k and r_{nk} that minimizes the objective function.These parameters are, respectively, the mean data points that belongs to the cluster k and a variable indicator, as described, respectively, in Equations 11 and 12. Note that if one of the parameters is known, another one can be inferred. For this, an iterative method is applied in order to optimize one by one.

$$\mu_k = \frac{\sum_n r_{nk}x_n}{\sum_n r_{nk}}, \text{for } k = 1, \ldots, K \tag{11}$$

$$r_{nk} = \begin{cases} 1 \text{ if } k = \arg\min_j ||x_n - \mu_j||^2, \text{for } k = 1, \ldots, K \\ 0 \text{ otherwise} \end{cases} \tag{12}$$

Standard optimization methods are used to solve the K-means problem in order to determine the cluster assignment matrix r_{nk}.

3.6. Model training

An important contribution of this work is the self-learning component in the proposed framework. In the training stage the process involves passing through each module, Dynamic Context, Embedding, and Clustering, in iterative rounds. In each round, the model in the Embedding module is trained for a fixed number of epochs to optimize and learn the weights matrices and bias for each layer in GNN. For each epoch, (Context Graph) is considered, as opposed to a sample the entire network. However, we evaluate the loss function and update the parameters by $\frac{n}{n_b}$ times using mini-batches of size n_b which are uniformly sampled without replacement from the graph nodes. Despite using all nodes and edges in message passing, only the nodes in the batch are used to evaluate the loss function at each mini-batch. This approach allows us to make efficient use of the entire network while still updating the model parameters in a batch-wise manner, an approach known to be lead to better convergence.

Algorithm 1 illustrates the DCSL-GNN framework. In the very first round, before the clusters are formed, each node is considered to be in its own individual cluster (cluster C^0). Thus, during the first random walk procedure, each node in the neighborhood has an equal probability of being visited, and the walk can return to the given node (as the node is the cluster center) with probability $(1 - p)$.

Algorithm 1 DCSL-GCN: model training using mini-batches for the evaluation function

```
for r = 1, ..., n_r do
  G_c ← GENERATECONTEXT(G, H^{r-1}, C^{r-1})
  for e = 1, ..., n_e do
    H^r = GenerateEmbeddings(G_c, W)
    for i = 1, ..., n/n_b do
      S ← GENERATESEEDS(V_c, n_u)
      L ← COMPUTELOSS(S, H^r)
      UPDATEWEIGHTS(W,L)
    end for
  end for
  C^r ← GENERATECLUSTERS(H^r)
end for
```

For training the GNN model, a loss function based on triplet loss was adopted, as shown in Equation 13. In this function a sample data point (known as anchor) is compared with the matching input (positive sample) and non-matching input (negative sample). Nodes that belong to the same cluster as the anchor node are considered positive samples, denoted as x_n^+. Nodes that do not belong to the same cluster are considered negative samples, denoted as x_n^-. This function aims to minimize the distance between an anchor and a positive sample; and maximizes the distance between the anchor and a negative sample. The hyper-parameter Δ enforces a margin distance between positive and negative

pairs.

$$L(\Theta) = \frac{1}{N}\sum_{n=1}^{N} \max\left(||x_n - x_n^+||^2 - ||x_n - x_n^-||^2 + \Delta, 0\right) \tag{13}$$

This loss function was adopted because it aims to minimize the distances between similar data points (positive pairs) and increase the distances between dissimilar data points (negative pairs). This behavior is aligned with the procedure to construct the Context Graph, as the random walks are biased by the Euclidean distance in the latent space. Therefore, in every round, similar nodes in embedding space becomes closer and distinct nodes becomes farther apart away. This process contributes to create a better context in the subsequent round.

Finally, the relatively simple GCN architecture proposed by [12] was adopted. It a simple model with only a few trainable parameters (only the weights and bias matrices). A GCN model with only two layers is considered, and thus, we need to learn two matrices, $W^{(0)}$ and $W^{(1)}$ as seen in the Equation 8. In each round of training, the parameters of the GCN for round r are initialized with the parameter values obtained from the previous round $r-1$. The computational cost of a simple model is reduced and the framework becomes relevant in scenarios where the graph has a large number of vertices and edges. Moreover, this procedure must be efficient since a model will be trained in every round of the proposed framework.

4. Evaluation

This section provides an evaluation of DCSL-GNN in various scenarios using synthetic attributed networks and benchmarks datasets as well as a direct comparison with other algorithms.

4.1. Attributed Stochastic Block Model

The Stochastic Block Model (SBM) [11] is a generative model for random graphs which produces networks containing cluster structure. The model is defined by number of clusters K, the number of nodes in each cluster, $n_i, i = 1, \dots K$, and a symmetric matrix $B \in [0,1]^{K\times K}$ specifying the connectivity probabilities between clusters. Thus, $B[i,j]$ is the probability that an edge is present between a pair of nodes u and v that belong to clusters i and j, respectively. To model homophily and induce clusters, $B[i,i] > B[i,j], i \neq j$.

The classic SBM has been recently extended to incorporate a model for node attributes that can be correlated to network structure[22, 7]. A simple model for node attributes assumes that all nodes in the same cluster have a random attribute that comes from the same distribution, in particular, the attributes of such nodes are independently and identically distributed. The Gaussian distributions with parameters μ_i and $\sigma_i^2, i = 1, \dots K$, is used to model the attributes for nodes in cluster i. Note that conditioned on its cluster, the edges incident to a node and its attribute are independent.

This model allows to explore the role of network information (edges) and node information (attributes) in the identification of clusters. In particular, by increasing the variance σ_i^2 (for all i) the information on the nodes becomes more noisy and less reliable. Similarly, by decreasing $B[i,i]$ (for all i) the network information becomes more noisy and less reliable.

The Normalized Mutual Information (NMI) is a classic metric used to evaluate the quality of a clustering with respect to a ground truth. NMI values range from $[0,1]$, where values close to 0 indicates poor cluster quality, while a value approaching 1 indicates good quality of the clusters. This metric will be used to assess performance of DCSL-GNN and other algorithms.

To ensure statistical significance given the stochastic nature of DCSL-GNN and the synthetic networks, multiple independent experiments using different sampled networks were conducted for each scenario. The reported results are the mean values obtained from the multiples runs (50 runs).

For the experiments, networks generated have $K = 4$ clusters $n_i = 100, i = 1, \dots, 4$ (a total of $N = 400$ nodes). The intra-cluster probability $B[i,i] = p_I$ was the same for all clusters and varied, while the inter-cluster probability fixed at $B[i,j] = 0.1$, $i \neq j$. Thus, the average number of edges between a node and nodes in different clusters is 30 (10 edges per cluster). The attributes for each cluster followed a Gaussian distribution with mean $[10, 20, 30, 40]$, respectively, and had identical variance σ^2 that varied in the experiments.

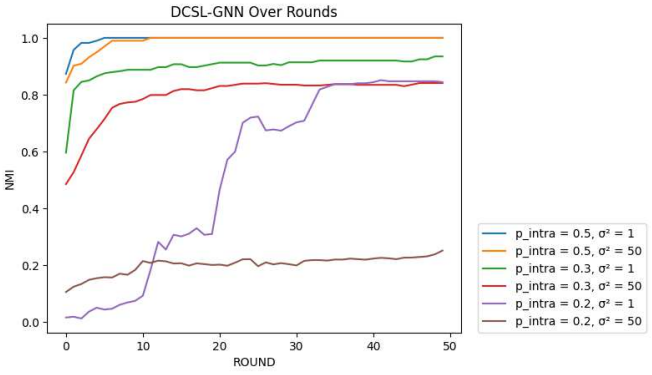


Figure 3: Normalized Mutual Information over 50 rounds in different scenarios.

4.2. Self-learning potential

DCSL-GNN utilizes self-supervised mechanism to learn through multiples rounds. The goal is to understand the behavior of the model across the multiples rounds under different scenarios for noise in the network structure and node attributes. Figure 3 shows the evolution of the NMI as a function of the learning rounds for the different scenarios. Note that in scenarios where the network information has little noise ($p_I = 0.5$), DCSL-GNN exhibits rapid convergence towards a perfect clustering, after a few rounds of self-learning. This occurs even when node information

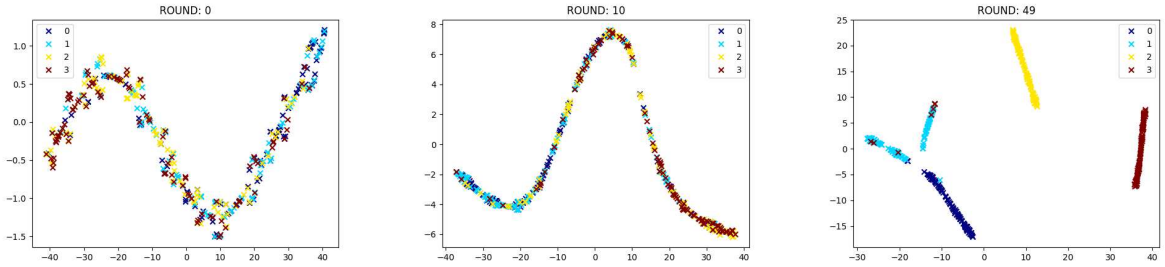


Figure 4: Projection of node's embedding at the end of different rounds for the scenario where $p_I = 0.2$ and $\sigma^2 = 1$ (two dimensional projection via t-SNE).

is very noisy ($\sigma^2 = 50$). However, when the network information becomes more noisy ($p_I = 0.3$) the model's performance improves over the rounds of self-learning and converges but not to a perfect clustering. This occurs even when node attributes have little noise ($\sigma^2 = 1$). Last, in the challenging scenarios where the network information is very noisy ($p_I = 0.2$), when attributes have little noise ($\sigma^2 = 1$), DCSL-GNN improve its performance over many rounds of self-learning eventually converging and delivering good performance. However, when attributes are also very noisy ($\sigma^2 = 50$), the multiple rounds of self-learning do not improve performance and the model converges to a low cluster quality (similar to a random guess). Indeed, in such scenario there is no information to perform an effective clustering.

Figure 4 shows the t-SNE projection in two dimensions of the node embeddings generated by DCSL-GNN at the end of a round in the scenario where $p_I = 0.2$ and $\sigma^2 = 1$. Initially the projections do not reflects any cluster structure. But, as the rounds progress the clusters begin to form.

4.3. Network and node noise

The performance DCSL-GNN in different scenarios of noise for the network and node attributes is evaluated. This is compared with the performance of algorithms that consider only the structure network (Louvain modularity) and only the node attributes (K-means) to cluster the data in the exact same networks. Additionally, DCSL-GNN is evaluated without the self-learning component (denoted "NSL" in the results) which is trained with only one round but with the same total number of epochs as the self-learning model.

Figure 5 compares the performance of the algorithms under a fixed variance (each plot) but a varying intra-cluster probabilities. Observe that in regimes with high intra-cluster probability ($p_I \geq 0.4$) the model achieves similar results as the Louvain algorithm, even when the attributes are very noisy ($\sigma^2 = 30$). Note that in all scenarios the performance decays as network information becomes (p_I is reduced). However, this decay depends on the quality of the attribute information, exhibiting better performance when the attribute is less noisy ($\sigma^2 = 0.5$). Interestingly, DCSL-GNN consistently outperforms the model without the self-learning component in all scenarios, indicating the effectiveness of the self-learning mechanism in improving the performance.

Dataset	Nodes	Edges	F	K	cc	$< k >$	Imbalance
Cora	2708	5278	1433	7	0.24	3.89	4.54
Citeseer	4230	4732	5337	6	0.11	2.52	1.48
PubMed	19717	44324	500	3	0.06	4.49	1.91

Table 2
Network characteristics of the benchmark datasets. F represents the dimension of the node attributes, K is number of clusters, cc denotes the average clustering coefficient of the network, $< k >$ is the average node degree, and imbalance is the ratio between the largest and smallest cluster.

Figure 6 shows the result for a fixed intra-cluster probability is fixed while the attribute variance is varied. Note that in all scenarios, the performance of DCSL-GNN decays with the increase in the variance. However, this decay depends on the p_I. When the network information has little noise ($p_I = 0.6$), DCSL-GNN provides perfect clustering even under very high attribute noise ($\sigma^2 = 50$). Note that DCSL-GNN outperforms K-means when attribute information is noisy but network information has little noise. Moreover, DCSL-GNN outperforms Louvain (modularity) when network information is noisy but attribute information has little noise. This indicates that DCSL-GNN is flexible and capable of extracting information from both the network structure and node attributes to perform more effective clusterings.

4.4. Benchmark datasets

Experiments were performed on three real citation networks that are commonly used to evaluate different network analysis algorithms: Cora, Citeseer, and PubMed. These datasets are widely employed in community detection studies, serving as benchmark graphs for evaluating the performance of various frameworks and methods. DCSL-GNN was executed 20 times with the hyperparameters manually tuned according to the dataset. Table 2 describe important characteristics of these datasets.

Table 3 presents the evaluation of multiple frameworks for clustering attributed graphs, with the NMI metric used to measure the clusters quality. Note that for each dataset the performance of the different algorithms is very different. Moreover, the performance of each algorithm across

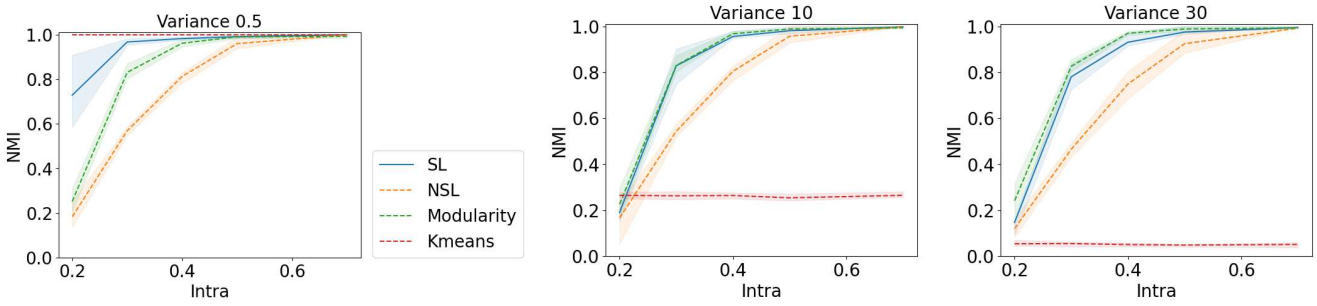


Figure 5: Performance (mean and standard deviation in shade) of different clustering algorithms on different scenarios as a function of p_I for different attribute variances.

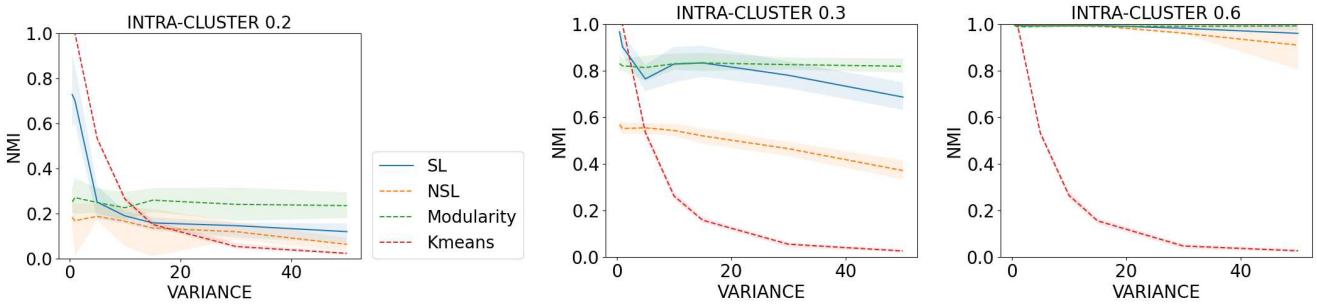


Figure 6: Performance (mean and standard deviation in shade) of different clustering algorithms on different scenarios as a function of attribute variances for different p_I.

	Cora	Citeseer	PubMed
Kmeans*	12.8 ± 3.5	26.54 ± 2.3	26.77 ± 0.2
Louvain*	41.49	28.62	21.81
GAE [2]	40.69	18.34	22.97
GIC [16]	53.70	45.30	31.90
AGC [28]	53.68	41.13	31.59
MGCCN [14]	60.20	43.20	–
DNENC-Att [24]	52.80	39.70	26.60
DMoN [23]	48.80	33.70	29.80
CommDGI [27]	57.90	41.90	35.70
SEComm [2]	56.04	42.53	36.50
DCSL-GNN*	46.95 ± 3.1	47.74 ± 2.8	24.27 ± 1.20

Table 3
Normalized Mutual Information (NMI) of clustering performance of different algorithms for Cora, Citeseer and PubMed datasets. Performance values were obtained from the corresponding papers (not reproduced in this work). Performance of algorithms marked by * were obtained in this work. Highlighted in red is the best result for each dataset, illustrating the lack of a single best algorithm across all datasets.

the datasets is also very different. In the Cora dataset, the MGCCN framework [14] achieves the highest NMI value and is considered state-of-the-art. Similarly, in the PubMed dataset, SEComm [2] is considered the state-of-the-art. Furthermore, the proposed framework DCSL-GNN shows superior performance for the Citeseer dataset. Clearly, there is no single algorithm that is best across all datasets.

While DCSL-GNN shows superior performance on the Citeseer dataset, its performance on the other two datasets is relatively far from the state-of-the-art. A possible cause for this low performance is cluster (class) imbalance which can significantly impacts performance of DCSL-GNN. In particular, cluster with fewer nodes contribute less to the objective function, which can result in a bias towards the dominant cluster. Indeed, for the Cora dataset, DCSL-GNN shows very good accuracy for the dominant cluster but negligible accuracy for the smallest cluster (results not shown here).

5. Conclusion

This paper proposed a novel fully unsupervised framework for clustering network nodes with attributes, called DCSL-GNN. The framework is based on two key ingredients: context graph and self-learning. The context graph is a weighted graph built from the original graph where edges indicate that nodes should belong to the same cluster (independent of their graph distance in the original graph). The self-learning allows DCSL-GNN to generate clusters and that generate new and better context graphs that can then be used to generate better clusters. A GNN is responsible for generating the node representations from the context graph which are in turn used by a clustering algorithm in Euclidean space. The GNN leverages the network and attribute information as well as a loss function that depends on clustering assignments of the previous self-learning round in order to generate meaningful representations. This iterative

self-learning approach enables DCSL-GNN to refine both the node representations and the clustering assignments in a mutually beneficial manner.

Using a network model with clusters and attributes, DCSL-GNN is evaluated and compared to other algorithms with results indicating its potential in extracting information from both the network and attributes to generate effective clusters. In particular, multiple rounds of self-learning are always advantageous, and network or attribute information can be leveraged even when the other is very noisy. Finally, when considering benchmark datasets, DCSL-GNN achieved mixed results being superior to all state-of-the-art algorithms for one dataset.

CRediT authorship contribution statement

Rodrigo de Sapienza Luna: Conceptualization of this study, Methodology, Software, Writing. **Daniel Ratton Figueiredo:** Supervision, Conceptualization, Methodology, Writing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Daniel Ratton Figueiredo reports financial support was provided by National Council for Scientific and Technological Development (CNPq), Brazil

Data availability

Data will be made available on request

References

[1] et al., B., 2021a. Analyzing the expressive power of graph neural networks in a spectral perspective, in: International Conference on Learning Representations.

[2] et al., B., 2021b. Unsupervised constrained community detection via self-expressive graph neural network, in: Uncertainty in Artificial Intelligence, PMLR.

[3] et al., H., 2022. Data mining: concepts and techniques. Morgan kaufmann.

[4] et al., V.D.B., 2008. Fast unfolding of communities in large networks. Journal of Statistical Mechanics .

[5] Barceló, P., Kostylev, E.V., Monet, M., Pérez, J., Reutter, J., Silva, J.P., 2020. The logical expressiveness of graph neural networks, in: International Conference on Learning Representations.

[6] Bo, D., Wang, X., Shi, C., Zhu, M., Lu, E., Cui, P., 2020. Structural deep clustering network, in: Proceedings of the web conference 2020, pp. 1400–1410.

[7] Deshpande, Y., Sen, S., Montanari, A., Mossel, E., 2018. Contextual stochastic block models. Advances in Neural Information Processing Systems 31.

[8] Fortunato, S., 2010. Community detection in graphs. Physics reports 486.

[9] Fortunato, S., Hric, D., 2016. Community detection in networks: A user guide. Physics reports 659.

[10] Grover, A., Leskovec, J., 2016. node2vec: Scalable feature learning for networks, in: ACM SIGKDD.

[11] Holland, P.W., Laskey, K.B., Leinhardt, S., 1983. Stochastic blockmodels: First steps. Social networks 5.

[12] Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations.

[13] Li, Q., Han, Z., Wu, X.M., 2018. Deeper insights into graph convolutional networks for semi-supervised learning, in: Thirty-Second AAAI conference on artificial intelligence.

[14] Liu, L., Kang, Z., Ruan, J., He, X., 2022. Multilayer graph contrastive clustering network. Information Sciences 613.

[15] Liu, Z., Zhou, J., 2022. Introduction to graph neural networks. Springer Nature.

[16] Mavromatis, C., Karypis, G., 2021. Graph infoclust: Maximizing coarse-grain mutual information in graphs, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer.

[17] Newman, M., 2018. Networks. Oxford university press.

[18] Newman, M.E., Girvan, M., 2004. Finding and evaluating community structure in networks. Physical review E 69.

[19] Ribeiro, L.F., Saverese, P.H., Figueiredo, D.R., 2017. struc2vec: Learning node representations from structural identity, in: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining.

[20] Rousseeuw, P.J., 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics 20.

[21] Schaeffer, S.E., 2007. Graph clustering. Computer science review 1.

[22] Stanley, N., Bonacci, T., Kwitt, R., Niethammer, M., Mucha, P.J., 2019. Stochastic block models with multiple continuous attributes. Applied Network Science , 1–22.

[23] Tsitsulin, A., Palowitch, J., Perozzi, B., Muller, E., 2023. Graph clustering with graph neural networks. Journal of Machine Learning Research (JMLR) 24, 1–21. URL: http://jmlr.org/papers/v24/20-998.html.

[24] Wang, C., Pan, S., Celina, P.Y., Hu, R., Long, G., Zhang, C., 2022. Deep neighbor-aware embedding for node clustering in attributed graphs. Pattern Recognition .

[25] Yang, H., Wang, J., Duan, R., Yan, C., 2023. Dcom-gnn: A deep clustering optimization method for graph neural networks. Knowledge-Based Systems 279.

[26] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J., 2018. Graph convolutional neural networks for web-scale recommender systems, in: ACM SIGKDD.

[27] Zhang, T., Xiong, Y., Zhang, J., Zhang, Y., Jiao, Y., Zhu, Y., 2020. Commdgi: community detection oriented deep graph infomax, in: Proceedings of the 29th ACM international conference on information & knowledge management.

[28] Zhang, X., Liu, H., Li, Q., Wu, X.M., 2019. Attributed graph clustering via adaptive graph convolution. arXiv preprint arXiv:1906.01210 .

[29] Zhang, X., Liu, H., Wu, X.M., Zhang, X., Liu, X., 2021. Spectral embedding network for attributed graph clustering. Neural Networks .

Rodrigo de Sapienza Luna received a BS in Computer Science from the Federal University of Rio de Janeiro (UFRJ), Brazil, in 2020. Subsequently, he obtained a Master of Science in Engineering and Computer Science from the same institution in 2023, and is currently pursuing an PhD degree in Systems Engineering and Computer Science in the same university. His research interests include graph neural networks, machine learning and network science.

Daniel Ratton Figueiredo received a BS cum laude degree in Computer Science and MSc degree in Computer and Systems Engineering from the Federal University of Rio de Janeiro (UFRJ) Brazil, in 1996 and 1999, respectively. He received a MSc and PhD degrees in Computer Science from the University of Massachusetts Amherst (UMass) in 2005. He worked as a post-doc researcher at the Swiss Federal Institute of Technology, Lausanne (EPFL). In 2007, he joined the Department of Computer and Systems Engineering (PESC/COPPE) at the Federal University of Rio de Janeiro (UFRJ), Brazil where he currently holds an Associate Professor position. He has a Research Productivity Fellowship granted by CNPq (since 2009) and was member of the Young Scientists Program granted by FAPERJ (from 2010 to 2018). He was a Visiting Fulbright Scholar at Columbia University (6 months in 2017). His current main interests are in Network Science and Graph Learning.