DIGRAC: DIGRAPH CLUSTERING BASED ON FLOW IMBALANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

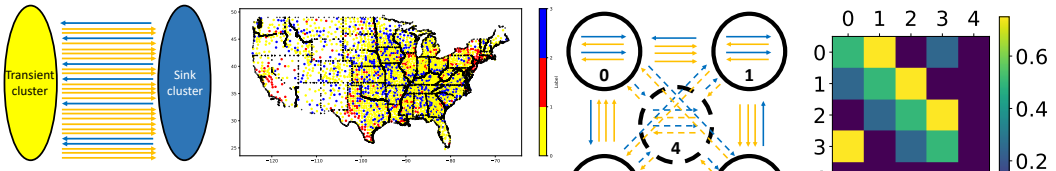
Node clustering is a powerful tool in the analysis of networks. We introduce a graph neural network framework to obtain node embeddings for directed networks in a self-supervised manner, including a novel probabilistic imbalance loss, which can be used for network clustering. Here, we propose *directed flow imbalance* measures, which are tightly related to directionality, to reveal clusters in the network even when there is no density difference between clusters. In contrast to standard approaches in the literature, in this paper, directionality is not treated as a nuisance, but rather contains the main signal. DIGRAC optimizes directed flow imbalance for clustering without requiring label supervision, unlike existing GNN methods, and can naturally incorporate node features, unlike existing spectral methods. Experimental results on synthetic data, in the form of directed stochastic block models, and real-world data at different scales, demonstrate that our method, based on flow imbalance, attains state-of-the-art results on directed graph clustering, for a wide range of noise and sparsity levels and graph structures and topologies.

1 INTRODUCTION

Revealing an underlying community structure of *directed* networks (*digraphs*) is an important problem in many application domains (Malliaros & Vazirgiannis, 2013), such as detecting influential social groups (Bovet & Grindrod, 2020) and analyzing migration patterns (Perry, 2003); node clustering in digraphs can be used for this community detection task (Palmer & Zheng, 2021). While most existing methods that could be applied to directed clustering use local edge densities as main signal and directionality as additional signal, we argue that even in the absence of any edge density differences, directionality can play a vital role in directed clustering as it can reveal latent properties of network flows. Therefore, instead of finding relatively dense groups of nodes in digraphs which have a relatively small amount of flow between the groups, as in Girvan & Newman (2002); Newman (2006); Leskovec et al. (2008); Leicht & Newman (2008); Chen & Baker (2021); Jia et al. (2017), our main goal is to recover clusters with *strong and imbalanced flow* among them, in the spirit of Cucuringu et al. (2020a); Laenen & Sun (2020), where directionality (i.e, edge orientation) is the main signal.

In contrast to standard approaches focusing on edge density, here edge directionality is not a nuisance but the main piece of information to uncover the latent structure. The underlying intuition is that homogeneous clusters of nodes form *meta-nodes* in a *meta-graph*, with the meta-graph directing the flow between clusters; directed core-periphery structure is such an example (Elliott et al., 2020). Figure 1(a) is an example of flow imbalance between two clusters, here on an unweighted network for simplicity: while 75% of the edges flow from the *Transient* cluster to the *Sink* cluster, only 25% flow in the other direction. As a real-world example, Figure 1(b) shows the strongest flow imbalances between clusters in a network of US migration flow (Perry, 2003); most edges flow from the red cluster (label 1) to the blue one (label 2). Figures 1(c-d) show examples on a synthetic meta-graph.

While most competitive state-of-the-art methods proposed for node clustering in digraphs mainly rely on edge density and overlook the role of directionality, those that do lay emphasis on directionality, usually by incorporating cut imbalance among clusters, are usually spectral methods, for which incorporating features is non-trivial, or GNN methods that require labeling information. In general, community detection methods for networks are not able to find communities in networks in which the sole structure lies in the directionality of edges. An exception is InfoMap (Rosvall & Bergstrom, 2008), a method for community detection in networks which uses directed random walks; however, it still relies on some edge density information within clusters.



(a) Flow imbalance example: most of the edges flow from *Transient* to *Sink*.

(b) Strongest imbalanced flow on *Migration* data, along with the geographic locations of the counties and state boundaries (in black).

(c) Meta-graph.

(d) Meta-graph adjacency matrix $\mathbf{F}$.

Figure 1: Visualization of cut flow imbalance and meta-graph: (a) 75% of edges flow from Transient to Sink, while 25% of edges flow in the opposite direction; (b) top pair imbalanced flow on *Migration* data (Perry, 2003): most edges flow from red (1) to blue (2); (c) & (d) are for a Directed Stochastic Block Model with a cycle meta-graph with ambient nodes, for a total of 5 clusters. 75% of the edges flow in direction $0 \to 1 \to 2 \to 3 \to 0$, while 25% flow in the opposite direction. Cluster 4 is the ambient cluster. In (c), dashed lines indicate flows with random equally likely directions; these flows do not exist in the meta-graph adjacency matrix $\mathbf{F}$. For (d), the lighter the color, the stronger the flow.

In this paper, we introduce a graph neural network (GNN) framework, denoted DIGRAC, to obtain node embeddings for clustering digraphs (potentially weighted, possibly with self-loops, but no multiple edges). In a self-supervised manner, a novel *probabilistic imbalance loss* is proposed to act on the digraph induced by all training nodes. The global imbalance score, one minus whom is the self-supervised loss function, is aggregated from pairwise normalized cut imbalances. The method is end-to-end in combining embedding generation and clustering without an intermediate step. To the best of our knowledge, this is the first GNN method that derives node embeddings for digraphs that directly maximizes flow imbalance between pairs of clusters. For the implementation of the framework, we devise a directed mixed path aggregation scheme that we abbreviate DIMPA, which is simple and effective. Other aggregation schemes which detect direction information could be used. With an emphasis on the use of a direction-based flow imbalance objective, experimental results on synthetic data and real-world data at different scales demonstrate that our method can achieve state-of-the-art performance for a wide range of network densities and topologies. Compared to its competitors, for synthetic data, our method achieves superior performance (with respect to the Adjusted Rand Index (ARI) (Hubert & Arabie, 1985)); for real-world data, our method clearly outperforms them, using imbalance scores as outcome success measures.

DIGRAC's main novelty is the ability to cluster based on direction-based flow imbalance, instead of using classical criteria such as maximizing relative densities within clusters. Compared with prior methods that focus on directionality, DIGRAC can easily consider node features and also does not require known labels. DIGRAC complements existing approaches in various aspects: (1) Our results show that DIGRAC complements classical community detection by detecting alternative patterns in the data, such as meta-graph structures, which are otherwise not detectable by existing methods. This aspect of detecting novel structures in directed graphs has also been emphasized in Cucuringu et al. (2020a). (2) DIGRAC complements existing spectral methods, through the possibility of including exogenous information, in the form of node-level features or labels, thus borrowing their strength. (3) DIGRAC complements existing GNN methods by introducing an imbalance-based objective.

DIGRAC's applicability extends beyond settings where the input data is a digraph; for example, with time series data as input, the digraph construction mechanism can accommodate any procedure that encodes a pairwise directional association between the corresponding time series, such as lead-lag relationships and Granger causality (Shojaie & Fox, 2021), with applications such as in the analysis of information flow in brain networks (Dhamala et al., 2008), biology (Runge et al., 2019), finance (Bennett et al., 2021; Sornette & Zhou, 2005) and earth sciences (Harzallah & Sadourny, 1997).

Our method for extracting flow-driven clusters in digraphs based on higher-order meta-graphs can facilitate tasks in time series analysis, ranking, and anomaly detection, as it allows one to extrapolate from *local* pairwise (directed) interactions to a *global* structure inference, in the high-dimensional low signal-to-noise ratio regime. As another use case, consider a social network where a set of fake accounts $\mathcal{A}$ have been created, which are likely to target another subset $\mathcal{B}$ of real accounts by sending them messages. Most likely, there would be many more messages from $\mathcal{A}$ to $\mathcal{B}$ than from $\mathcal{B}$ to $\mathcal{A}$,

hinting that $\mathcal{A}$ is most likely comprised of fake accounts. In ranking applications, where the match results (or preference relationships between items) are encoded in a digraph, DIGRAC would allow for the detection of subsets of players where the relative strength of players in $\mathcal{A}$ is superior to that of players in $\mathcal{B}$, thus facilitating the extraction of partial rankings (Cucuringu, 2016).

Main contributions. Our main contributions are as follows. • (1) We propose a GNN framework for self-supervised end-to-end node clustering on digraphs, which are possibly attributed and weighted, explicitly taking into account the directed flow imbalance. • (2) We propose a probabilistic version of the global imbalance score to serve as a self-supervised loss function. To the best of our knowledge, this is the first method directly maximizing cut flow imbalance for node clustering in digraphs using GNNs. This new imbalance score is the key novelty in this paper. • (3) We extend our method to the semi-supervised setting when label information is available.

Paper outline. Section 2 reviews existing work. Section 3 introduces our DIGRAC method. Section 4 validates its strength through extensive experiments at different scales, on synthetic and real-world data. Section 5 draws conclusions and discusses limitations and future works. In Supplementary Information (SI), Section A expands on variants of our model; Section B discusses implementation details; Sections C, D & E provide additional results.

Anonymized codes and preprocessed data are available at `https://anonymous.4open.science/r/1b728e97-cc2b-4e6a-98ea-37668813536c`.

2 Related Work

Directed clustering has been explored by spectral and GNN-based methods. Satuluri & Parthasarathy (2011) performs directed clustering that hinges on symmetrizations of the adjacency matrix, but is not scalable as it requires large matrix multiplication. Rohe et al. (2016) proposes a spectral co-clustering algorithm for asymmetry discovery that relies on in-degree and out-degree. Whenever direction is the sole information, such as in a complete network with lead-lag structure derived from time series (Bennett et al., 2021), a purely degree-based method cannot detect the clusters. While Zhang et al. (2021a) produces two partitions of the node set, one based on out-degree and one based on in-degree, here we produce a partition that simultaneously takes both directions into account. Cucuringu et al. (2020a) seeks to uncover clusters characterized by a strongly imbalanced flow circulating among them, based on eigenvectors of the Hermitian matrix $(\mathbf{A} - \mathbf{A}^T) \cdot i$, where $\mathbf{A}$ is the (normalized) adjacency matrix and i the imaginary unit. Zhang et al. (2021b) builds upon Cucuringu et al. (2020a); Mohar (2020) and introduces a complex Hermitian matrix that encodes undirected geometric structure in the magnitude of its entries, and directional information in their phase. Laenen & Sun (2020) uncovers higher-order structural information among clusters in digraphs, while maximizing the imbalance of the edge directions, but its definition of the flow ratio restricts the underlying meta-graph to a path.

Ma et al. (2019) and Palmer & Zheng (2021) introduce *directed graph Laplacians*, but these methods are only applicable to strongly connected digraphs, which is hardly the case in sparse networks arising in applications. Monti et al. (2018) utilizes convolution-like anisotropic filters based on local subgraph structures (motifs) for semi-supervised node classification tasks in digraphs, but relies on pre-defined structures and fails to handle complex networks. DGCN (Tong et al., 2020b) uses first and second order proximity, constructs three Laplacians, but the method is space and speed inefficient. DiGCN (Tong et al., 2020a) simplifies DGCN, builds a directed Laplacian based on PageRank, and aggregates information dependent on higher-order proximity. In addition, Zhang et al. (2021b); Monti et al. (2018); Tong et al. (2020b;a) all require known labels, which are not generally available for real-world data. Satuluri & Parthasarathy (2011); Rohe et al. (2016); Cucuringu et al. (2020a); Palmer & Zheng (2021); Laenen & Sun (2020) could not trivially incorporate node attributes or node labels. In contrast, we propose an efficient GNN-based method that maximizes a probablistic flow imbalance objective, in a self-supervised manner, and which can naturally analyze attributed weighted digraphs.

InfoMap by Rosvall & Bergstrom (2008) assumes that there is a "map" underlying the network, similar to a meta-graph in DIGRAC. However, InfoMap aims to minimize the expected description length of a random walk and is recommended for networks where edges encode patterns of movement among nodes. While related to DIGRAC, InfoMap still relies on some signals within modules.

To avoid potential misunderstanding, we briefly mention some related papers that we are aware of but do not compare against in our experiments in the main text. Lancichinetti et al. (2011) proposes the OSLOM method, which is very flexible but is based on a density heuristic and hence a comparison to DIGRAC on networks without density signal would not be fair to begin with. Dugué & Perez

(2015) introduces directionality in the Louvain algorithm. This algorithm optimizes a modularity-type function that compares the number of edges within communities to the expected number of edges under a specified model. It is thus an approach that aims to find denser than expected groups of vertices. When all groups have the same density, as in our synthetic data sets, and the only structure lies in the directionality of the edges, this method simply cannot be expected to perform well. Traag et al. (2019), the Leiden algorithm, also builds on the Louvain algorithm, again optimizing a modularity-type function that compares the number of edges within communities to the expected number of edges under a specified model. It is a powerful method for that task, but cannot be fairly compared to DIGRAC which is tailored to find imbalances.

We also do not compare DIGRAC against graph pooling methods (Bianchi et al., 2020), which are inspired by pooling in CNN and developed to discard information which is superfluous for the task at hand. A partition of the nodes which can be interpreted as clustering is only a byproduct of graph pooling. Although not often stated explicitly, graph pooling methods are usually developed only for undirected networks. While graph matching as in Xu et al. (2019a;b); Chowdhury & Needham (2021) and Xu (2020) can be viewed as a clustering method of networks, matching the graph of interest to a disconnected graph by connecting each node in the observed graph with an isolated node of the disconnected graph, this approach is not developed for directed networks. The underlying idea of these papers is complementary to the meta-graph idea which underpins DIGRAC; in the meta-graph, the components are connected, and estimating the directionality of these connections is the main focus. Hence this work addresses a very different task. We emphasize that these are all excellent methods, but they address different objectives and tasks. As confirmed by our experiments in SI E, comparing these methods to DIGRAC is not appropriate. DIGRAC is tailored to detect an imbalance signal in directed networks, and such a signal cannot be present in an undirected network. As it is based on imbalance, DIGRAC will not be able to detect a signal in an undirected network, thus rendering it not applicable to undirected networks.

3 THE DIGRAC METHOD

Problem definition. Denote a (possibly weighted) digraph with node attributes as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w, \mathbf{X}_{\mathcal{V}})$, with $\mathcal{V}$ the set of nodes, $\mathcal{E}$ the set of directed edges or links, and $w \in [0, \infty)^{|\mathcal{E}|}$ the set of edge weights. $\mathcal{G}$ may have self-loops, but no multiple edges. The number of nodes is $n = |\mathcal{V}|$, and $\mathbf{X}_{\mathcal{V}} \in \mathbb{R}^{n \times d_{\text{in}}}$ is a matrix whose rows encode the nodes' attributes [1]. Such a network can be represented by the attribute matrix $\mathbf{X}_{\mathcal{V}}$ and the adjacency matrix $\mathbf{A} = (A_{ij})_{i,j \in \mathcal{V}}$, with $\mathbf{A}_{ij} = 0$ if no edge exists from v_i to v_j; if there is an edge e from v_i to v_j, we set $A_{ij} = w_e$, the edge weight.

Digraphs often lend themselves to interpreting weighted directed edges as flows, with a meta-graph on clusters of vertices describing the overall flow directions; see Figure 1. A clustering is a partition of the set of nodes into K disjoint sets (clusters) $\mathcal{V} = \mathcal{C}_0 \cup \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_{K-1}$. Intuitively, nodes within a cluster should be similar to each other with respect to flow directions, while nodes across clusters should be dissimilar. In a self-supervised setting, only the number of clusters K is given. In a semi-supervised setting, for each of the K clusters, a fraction set $\mathcal{V}^{\text{seed}} \subseteq \mathcal{V}^{\text{train}} \subset \mathcal{V}$ of the set $\mathcal{V}^{\text{train}}$ of all training nodes are selected as seed nodes, for which the cluster membership labels are known before training. The goal of semi-supervised clustering is to assign each node $v \in \mathcal{V}$ to a cluster containing some known seed nodes, without knowledge of the underlying flow meta-graph. The corresponding self-supervised clustering task does not use seed nodes.

3.1 SELF-SUPERVISED LOSS FOR CLUSTERING

Our self-supervised loss function is inspired by Cucuringu et al. (2020a), aiming to cluster the nodes by maximizing a normalized form of cut imbalance across clusters. We first define probabilistic versions of cuts, imbalance flows, and probabilistic volumes. For K clusters, the *assignment probability matrix* $\mathbf{P} \in \mathbb{R}^{n \times K}$ has as row i the probability vector $\mathbf{P}_{(\mathbf{i},:)} \in \mathbb{R}^K$ with entries denoting the probabilities of each node to belong to each cluster; its kth column is denoted by $\mathbf{P}_{(:,k)}$.

• The **probabilistic cut** from cluster $\mathcal{C}_k$ to $\mathcal{C}_l$ is defined as

$$W(\mathcal{C}_k, \mathcal{C}_l) = \sum_{i,j \in \{1,\ldots,n\}} \mathbf{A}_{i,j} \cdot \mathbf{P}_{i,k} \cdot \mathbf{P}_{j,l} = (\mathbf{P}_{(:,k)})^T \mathbf{A} \mathbf{P}_{(:,l)}.$$

[1] If no attributes are available, one could use any feature matrix generated from $\mathbf{A}$, such as stacking the eigenvectors of the Hermitian matrix introduced in Cucuringu et al. (2020a), to construct the feature matrix.

• The **imbalance flow** between clusters $\mathcal{C}_k$ and $\mathcal{C}_l$ is defined as

$$|W(\mathcal{C}_k, \mathcal{C}_l) - W(\mathcal{C}_l, \mathcal{C}_k)|, \quad \forall k, l \in \{0, \ldots, K-1\}.$$

For interpretability and ease of comparison, we normalize the imbalance flows to obtain an imbalance score with values in $[0, 1]$ as follows (we defer additional details to SI A.1).

• The **probabilistic volume** for cluster $\mathcal{C}_k$ is defined as

$$VOL(\mathcal{C}_k) = VOL^{(\text{out})}(\mathcal{C}_k) + VOL^{(\text{in})}(\mathcal{C}_k) = \sum_{i,j} (\mathbf{A}_{i,j} + \mathbf{A}_{j,i}) \cdot \mathbf{P}_{j,k}.$$

Then $VOL(\mathcal{C}_k) \geq W(\mathcal{C}_k, \mathcal{C}_l)$ for all $l \in \{0, \ldots, K-1\}$ and

$$\min(VOL(\mathcal{C}_k), VOL(\mathcal{C}_l)) \geq \max(W(\mathcal{C}_k, \mathcal{C}_l), W(\mathcal{C}_l, \mathcal{C}_k)) \geq |W(\mathcal{C}_k, \mathcal{C}_l) - W(\mathcal{C}_l, \mathcal{C}_k)|. \quad (1)$$

The imbalance term, which is used in most of our experiments, denoted $\text{CI}^{\text{vol_sum}}$, is defined as

$$\text{CI}^{\text{vol_sum}}(k, l) = 2\frac{|W(\mathcal{C}_k, \mathcal{C}_l) - W(\mathcal{C}_l, \mathcal{C}_k)|}{VOL(\mathcal{C}_k) + VOL(\mathcal{C}_l)} \in [0, 1]. \quad (2)$$

The aim is then to find a partition which maximizes the imbalance flow, to capture groups of nodes which would represent clusters in the meta-graph. The normalization by the volumes is carried out to penalize partitions that put most nodes into a single cluster. The range $[0, 1]$ follows from Eq. (1). Additional variants are discussed in SI A.2.

To obtain a **global probabilistic imbalance score**, based on $\text{CI}^{\text{vol_sum}}$ from Eq. (2), we average over pairwise imbalance scores of different pairs of clusters. Since the scores discussed are symmetric and the cut difference before taking absolute value is skew-symmetric, we only need to consider the pairs $\mathcal{T} = \{(\mathcal{C}_k, \mathcal{C}_l) : 0 \leq k < l \leq K-1, k, l \in \mathbb{Z}\}$. We consider a "*sort*" variant to select these pairs; concretely, we choose the pairs of clusters with the largest β pairwise cut flow imbalance values, where β is half of the number of nonzero entries in the off-diagonal entries of the meta-graph adjacency matrix $\mathbf{F}$ (if the meta-graph is known), or can be approximated otherwise. With $\mathcal{T}(\beta) = \{(\mathcal{C}_k, \mathcal{C}_l) \in \mathcal{T} : \text{CI}^{\text{vol_sum}}(k, l) \text{ is among the top } \beta \text{ values}\}$, where $1 \leq \beta \leq \binom{K}{2}$, we set

$$\mathcal{O}^{\text{sort}}_{\text{vol_sum}} = \frac{1}{\beta} \sum_{(\mathcal{C}_k, \mathcal{C}_l) \in \mathcal{T}(\beta)} \text{CI}^{\text{vol_sum}}(k, l), \quad \text{and} \quad \mathcal{L}^{\text{sort}}_{\text{vol_sum}} = 1 - \mathcal{O}^{\text{sort}}_{\text{vol_sum}}, \quad (3)$$

as the corresponding loss function. For example, for a "*cycle*" meta-graph with three clusters and no ambient nodes, we have $\beta = 3$. When we consider a "path" meta-graph with three clusters and ambient nodes, we have $\beta = 1$. Definitions of meta-graph structures are discussed in Section 4.1. In SI A.3, additional variants for selecting pairs of clusters in $\mathcal{T}$ are considered, including an "*std*" variant based on hypothesis testing, and a "*naive*" variant which includes all pairs in $\mathcal{T}$. The corresponding scores and loss functions for these variants are defined analogously.

3.2 Directed Mixed Path Aggregation (DIMPA)

To implement DIGRAC, we devise a simple yet effective directed mixed path aggregation scheme, to obtain the probability assignment matrix $\mathbf{P}$ and feed it to the loss function, as a digraph extension of the successful *KernelGCN* method introduced by Tian et al. (2019). Thus, in order to build node embeddings, we capture local network information by taking a weighted average of information from neighbors within h hops. To this end, we row-normalize the adjacency matrix, $\mathbf{A}$, to obtain $\overline{\mathbf{A}}^s$. Similar to the regularization discussed in Kipf & Welling (2016), we add a weighted self-loop to each node and normalize by setting $\overline{\mathbf{A}}^s = (\tilde{\mathbf{D}}^s)^{-1}\tilde{\mathbf{A}}^s$, where $\tilde{\mathbf{A}}^s = \mathbf{A} + \tau\mathbf{I}$, and the diagonal matrix $\tilde{\mathbf{D}}^s(i, i) = \sum_j \tilde{\mathbf{A}}^s(i, j)$, for a small value τ, such as 0.5.

The h-hop **source** matrix is given by $(\overline{\mathbf{A}}^s)^h$. We denote the set of *up-to-h-hop* source neighborhood matrices as $\mathcal{A}^{s,h} = \{\mathbf{I}, \overline{\mathbf{A}}^s, \ldots, (\overline{\mathbf{A}}^s)^h\}$. Similarly, for aggregating information when each node is construed as a **target** node of a link, we carry out the same procedure for $\mathbf{A}^T$. We denote the set of up-to-h-hop target neighborhood matrices as $\mathcal{A}^{t,h} = \{\mathbf{I}, \overline{\mathbf{A}}^t, \ldots, (\overline{\mathbf{A}}^t)^h\}$, where $\overline{\mathbf{A}}^t$ is the row-normalized target adjacency matrix calculated from $\mathbf{A}^T$.

Next, we define two feature mapping functions for source and target embeddings, respectively. Assume that for each node in $\mathcal{V}$, a vector of features is available, and summarize these features in the input feature matrix $\mathbf{X}_{\mathcal{V}}$. The source embedding (the superscript s stands for source) is given by

$$\mathbf{Z}_{\mathcal{V}}^{s} = \left(\sum_{\mathbf{M} \in \mathcal{A}^{s,h}} \omega_{\mathbf{M}}^{s} \cdot \mathbf{M} \right) \cdot \mathbf{H}_{\mathcal{V}}^{s} \in \mathbb{R}^{n \times d}, \tag{4}$$

where for each $\mathbf{M}$, $\omega_{\mathbf{M}}^{s}$ is a learnable scalar, d is the dimension of this embedding, and $\mathbf{H}_{\mathcal{V}}^{s} = \mathbf{MLP}^{(s,l)}(\mathbf{X}_{\mathcal{V}})$. Here, the hyperparameter l controls the number of layers in the multilayer perceptron (MLP) with ReLU activation; we fix $l = 2$ throughout. Each layer of the MLP has the same number d of hidden units. The target embedding $\mathbf{Z}_{\mathcal{V}}^{t}$ is defined similarly, with s replaced by t in Eq. (4). Different parameters for the MLPs for different embeddings are possible. After these two decoupled aggregations, we concatenate the embeddings to obtain the final node embedding as a $n \times (2d)$ matrix $\mathbf{Z}_{\mathcal{V}} = \mathrm{CONCAT}\left(\mathbf{Z}_{\mathcal{V}}^{s}, \mathbf{Z}_{\mathcal{V}}^{t}\right)$. The embedding vector $\mathbf{z}_i$ for a node v_i is the ith row of $\mathbf{Z}_{\mathcal{V}}$, $\mathbf{z}_i := (\mathbf{Z}_{\mathcal{V}})_{(i,:)} \in \mathbb{R}^{2d}$. An efficient implementation of DIMPA is shown in Algorithm 1 in SI B.8.

After obtaining the embedding matrix $\mathbf{Z}_{\mathcal{V}}$, we apply a linear layer (an affine transformation) to $\mathbf{Z}_{\mathcal{V}}$, so that the resulting matrix has K columns. Next, we apply the unit *softmax* function to the rows and obtain the assignment probability matrix $\mathbf{P}$. Figure 2 gives an overview of this implementation.

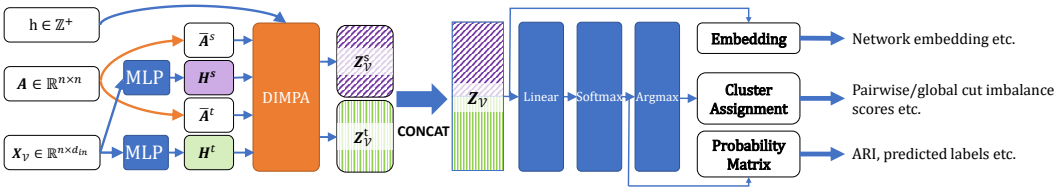


Figure 2: DIGRAC overview: from feature matrix $\mathbf{X}_{\mathcal{V}}$ and adjacency matrix $\mathbf{A}$, we first compute the row-normalized adjacency matrices $\overline{\mathbf{A}}^{s}$ and $\overline{\mathbf{A}}^{t}$. Then, we apply two separate MLPs on $\mathbf{X}_{\mathcal{V}}$, to obtain hidden representations $\mathbf{H}^{s}$ and $\mathbf{H}^{t}$. Next, we compute their decoupled embeddings using Eq. (4), and its equivalent for target embeddings. The concatenated decoupled embeddings are the final embeddings. For node clustering tasks, we add a linear layer followed by a unit *softmax* to obtain the probability matrix $\mathbf{P}$. Applying *argmax* on each row of $\mathbf{P}$ yields node cluster assignments.

4 Experiments

In this section, we describe the synthetic and real-world data sets used in this study, and illustrate the efficacy of our method. When ground truth is available, performance is measured by the Adjusted Rand Index (ARI) (Hubert & Arabie, 1985) for node clustering, and by accuracy for node classification and link direction prediction. Normalized Mutual Information (NMI) results give almost the same ranking for the best-performing methods as the ARI, with an average Kendall tau value of 84.2% and standard deviation 25.1%, for pairwise ranking comparison, on the methods compared in our experiments. We do not focus on NMI in the main text due to its shortcomings (Liu et al., 2019).

Clustering tasks will have different ground truths, depending on the pattern they are trying to detect. Many network clustering methods focus on detecting relatively dense clusters, and try to optimize classical network clustering measures, such as directed modularity or partition density. Ground truth for these clustering algorithms then relates to relatively densely connected subgroups in the data. DIGRAC is a novel method that addresses a novel task, namely that of detecting flow imbalances. To the best of our knowledge, real data sets with ground-truth flow imbalances are not available to date, and hence we introduce normalized imbalance scores to evaluate clustering performance based on flow imbalance. As ARI and NMI require ground-truth labels, they thus cannot be applied to the available real-world data sets. To address this shortcoming, for the real-world data sets, in Table 1, we include two performance measures which we introduce in the paper, and the SI contains an additional 12 performance measures. Implementation details are provided in SI B.

In our experiments, we compare DIGRAC against the most recent related methods from the literature for clustering digraphs. The **11** methods are • (1) InfoMap (Rosvall & Bergstrom, 2008), • (2) Bibliometric and • (3) Degree-discounted introduced in Satuluri & Parthasarathy (2011), • (4) DI_SIM (Rohe et al., 2016), • (5) Herm and • (6) Herm_sym introduced in Cucuringu et al. (2020a), • (7) MagNet (Zhang et al., 2021b), • (8) DGCN (Tong et al., 2020b), • (9) three variants of DiGCN (Tong et al., 2020a). The abbreviations of these methods, when reported in the numerical experiments, are Bi_sym, DD_sym, DISG_LR, Herm, Herm_sym, MagNet, DGCN,

DiGCN, DiGCN_app, DiGCN_ib, respectively. DGCN is the least efficient method in terms of speed and space complexity, followed by DiGCN_ib which involves the so-called *inception blocks* (hence the suffix *ib*). DiGCN denotes the method without using approximate Laplacian based on personalized PageRank, while DiGCN_app and DiGCN_ib use this approximation. We use the same hyperparameter settings stated in these papers. Methods (7), (8), (9) are trained with 80% nodes under label supervision. For MagNet, we use $q = 0.25$ for the phase matrix, which is suggested to capture directionality the most effectively.

4.1 DATA SETS

Synthetic data: Directed Stochastic Block Models A standard directed stochastic blockmodel (DSBM) is often used to represent a network cluster structure, see for example Malliaros & Vazirgiannis (2013). Its parameters are the number K of clusters and the edge probabilities; given the cluster assignment of the nodes, the edge indicators are independent. The DSBMs used in our experiments also depend on a meta-graph adjacency matrix $\mathbf{F} = (\mathbf{F}_{k,l})_{k,l=0,\ldots,K-1}$ and a *filled* version of it, $\tilde{\mathbf{F}} = (\tilde{\mathbf{F}}_{k,l})_{k,l=0,\ldots,K-1}$, and on a noise level parameters $\eta \leq 0.5$. The meta-graph adjacency matrix $\mathbf{F}$ is generated from the given meta-graph structure, called $\mathcal{M}$. To include an ambient background, the filled meta-graph adjacency matrix $\tilde{\mathbf{F}}$ replaces every zero in $\mathbf{F}$ that is not part of the imbalance structure by 0.5. The filled meta-graph thus creates a number of *ambient nodes* which correspond to entries which are not part of $\mathcal{M}$ and thus are not part of a meaningful cluster; this set of *ambient nodes* is also called the *ambient cluster*. First, we provide examples of structures of $\mathbf{F}$ without any ambient nodes, where $\mathbb{1}$ denotes the indicator function.

• (1) "*cycle*": $\mathbf{F}_{k,l} = (1-\eta)\mathbb{1}(l = ((k+1) \mod K)) + \eta\mathbb{1}(l = ((k-1) \mod K)) + \frac{1}{2}\mathbb{1}(l = k)$.

• (2) "*path*": $\mathbf{F}_{k,l} = (1-\eta)\mathbb{1}(l = k+1) + \eta\mathbb{1}(l = k-1) + \frac{1}{2}\mathbb{1}(l = k)$.

• (3) "*complete*": assign diagonal entries $\frac{1}{2}$. For each pair (k, l) with $k < l$, let $\mathbf{F}_{k,l}$ be η and $1-\eta$ with equal probability, then assign $\mathbf{F}_{l,k} = 1 - \mathbf{F}_{k,l}$.

• (4) "*star*", following Elliott et al. (2019): select the center node as $\omega = \lfloor \frac{K-1}{2} \rfloor$ and set $\mathbf{F}_{k,l} = (1-\eta)\mathbb{1}(k = \omega, l \text{ odd}) + \eta\mathbb{1}(k = \omega, l \text{ even}) + (1-\eta)\mathbb{1}(l = \omega, k \text{ odd}) + \eta\mathbb{1}(l = \omega, l \text{ even})$.

When ambient nodes are present, the construction involves two steps, with the first step the same as the above, but with the following changes: For "*cycle*" meta-graph structure, $\mathbf{F}_{k,l} = (1-\eta)\mathbb{1}(l = ((k+1) \mod (K-1))) + \eta\mathbb{1}(l = ((k-1) \mod (K-1))) + 0.5\,\mathbb{1}(l = k)$. The second step is to assign 0 (0.5, resp.) to the last row and the last column of $\mathbf{F}$ ($\tilde{\mathbf{F}}$, resp.). Figures 1(c-d) display a "*cycle*" meta-graph structure with ambient nodes (in cluster 4). The majority (75%) of edges flow in the form $0 \to 1 \to 2 \to 3 \to 0$, while 25% flow from the opposite direction. Figure 1(d) illustrate the meta-graph adjacency matrix corresponding to $\mathbf{F}$ shown in Figure 1(c).

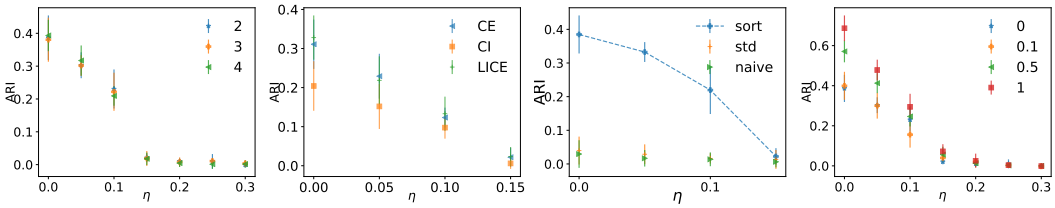
In our experiments, we choose the number of clusters, the (approximate) ratio, ρ, between the largest and the smallest cluster size, and the number, n, of nodes. To tackle the hardest clustering task, all pairs of nodes within a cluster and all pairs of nodes between clusters have the same edge probability, p. Note that for $\mathcal{M}$ ="*cycle*", even the expected in-degree and out-degree of all nodes are identical. Our DSBM, which we denote by DSBM $(\mathcal{M}, \mathbb{1}(\text{ambient}), n, K, p, \rho, \eta)$, is built similarly to Cucuringu et al. (2020a) but with possibly unequal cluster sizes, with more details in SI B.3. For each node $v_i \in \mathcal{C}_k$, and each node $v_j \in \mathcal{C}_l$, independently sample an edge from node v_i to node v_j with probability $p \cdot \tilde{\mathbf{F}}_{k,l}$. The parameter settings in our experiments are $p \in \{0.001, 0.01, 0.02, 0.1\}$, $\rho \in \{1, 1.5\}$, $K \in \{3, 5, 10\}$, $\mathbb{1}(\text{ambient}) \in \{\text{T, F}\}$ (True and False), $n \in \{1000, 5000, 30000\}$, and we also vary the direction flip probability η from 0 to 0.45, with a 0.05 step size.

Real-world data We perform experiments on five real-world digraph data sets, namely *Telegram* (Bovet & Grindrod, 2020), *Blog* (Adamic & Glance, 2005), *Migration* (Perry, 2003), *WikiTalk* (Leskovec et al., 2010), and *Lead-Lag* (Bennett et al., 2021), with details in SI B.3. We set the number of clusters K to be 4, 2, 10, 10, 10, respectively, and values of β to be 5, 1, 9, 10, 3, respectively. It is worth remarking that the *Lead-Lag* data set actually comprises of 19 separate networks, each one pertaining to a different year of financial time series data.

4.2 EXPERIMENTAL RESULTS

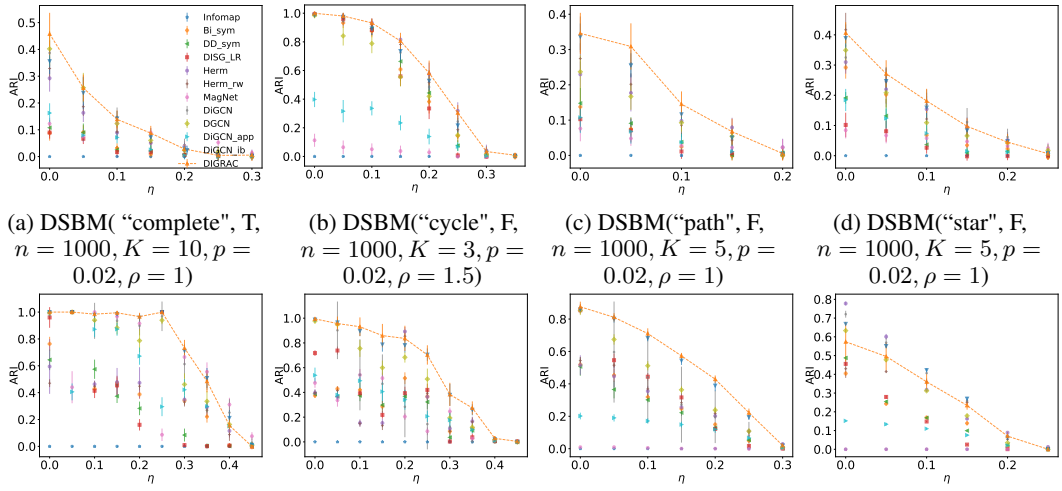
Training set-up and hyperparameter selection As training setup, we use 10% of all nodes from each cluster as test nodes, 10% as validation nodes to select the model, and the remaining 80% as training nodes. In each setting, unless otherwise stated, we carry out 10 experiments with different

data splits. Error bars are given by one standard error. Without node attributes, the matrix $\mathbf{X}_{\mathcal{V}}$ for DIGRAC is taken as the stacked eigenvectors corresponding to the largest K eigenvalues of the random-walk symmetrized Hermitian matrix used in the comparison method Herm_rw. The imbalance loss function acts on the subgraph induced by the training nodes. Hyperparameters are selected via greedy search, with details in SI B.4. Figure 3(a) compares DIGRAC on DSBM with $n = 1000$ nodes, $K = 5$ clusters, $\rho = 1, p = 0.02$ without ambient nodes, under different hyperparameter settings. Here hop $h = 2$ should be chosen to reduce complexity, as increasing h does not lead to increased performance. Therefore, our default setting is hop $h = 2, d = 32, \tau = 0.5$.



(a) Vary the hop h in DIMPA. (b) Vary loss for DiGCN: CE, CI or LICE. (c) Vary selection methods of imbalance pairs. (d) Vary seed ratio in a semi-supervised setting.

Figure 3: Hyperparemter analysis (a) and ablation study (b-d) on a "*cycle*" DSBM with $n = 1000$ nodes, $K = 5$ clusters, $\rho = 1$, and $p = 0.02$, without ambient nodes.



(a) DSBM("complete", T, $n = 1000, K = 10, p = 0.02, \rho = 1$) (b) DSBM("cycle", F, $n = 1000, K = 3, p = 0.02, \rho = 1.5$) (c) DSBM("path", F, $n = 1000, K = 5, p = 0.02, \rho = 1$) (d) DSBM("star", F, $n = 1000, K = 5, p = 0.02, \rho = 1$)

(e) DSBM("complete", F, $n = 1000, K = 3, p = 0.1, \rho = 1$) (f) DSBM("complete", T, $n = 1000, K = 5, p = 0.1, \rho = 1.5$) (g) DSBM("cycle", T, $n = 5000, K = 5, p = 0.01, \rho = 1.5$) (h) DSBM("cycle", F, $n = 30000, K = 5, p = 0.001, \rho = 1$)

Figure 4: Test ARI comparison on synthetic data. Dashed lines highlight DIGRAC's performance. Error bars are given by one standard error.

Node clustering results on synthetic data Figure 4 compares the numerical performance of DIGRAC with other methods on synthetic data. For this figure we generate 5 DSBM networks under each parameter setting and use 10 different data splits for each network, then average over the 50 runs. Error bars are given by one standard error. Additional implementation details are available in SI B.

We conclude that DIGRAC attains state-of-the-art results on a wide range of network densities and noise levels, on different network sizes, and with different meta-graph structures, whether or not there exist ambient nodes. DIGRAC outperforms its competitors by a large margin especially when there exist ambient nodes, which validates its strength of extracting the main directional signal planted in a larger graph. Additional results are reported in SI C, from which similar conclusions can be drawn.

Node clustering results on real-world data For our real-world data sets, the node in- and out-degrees may not be identical across clusters. Moreover, as these data sets do not contain node attributes, DIGRAC considers the eigenvectors corresponding to the largest K eigenvectors of the Hermitian matrix from Cucuringu et al. (2020a) to construct an input feature matrix. Table 1 reveals

Table 1: Performance comparison on real-world data sets. The best is marked in **bold red** and the second best is marked in underline blue. The objectives are defined in Section 3.1.

Metric	Data set	InfoMap	Bi_sym	DD_sym	DISG_LR	Herm	Herm_rw	DIGRAC
$\mathcal{O}^{\text{sort}}_{\text{vol_sum}}$	*Telegram*	0.04±0.00	0.21±0.0	0.21±0.0	0.21±0.01	0.2±0.01	0.14±0.0	**0.32±0.01**
	Blog	0.07±0.00	0.07±0.0	0.0±0.0	0.05±0.0	0.37±0.0	0.0±0.0	**0.44±0.0**
	Migration	N/A	0.03±0.00	0.01±0.00	0.02±0.00	0.04±0.00	0.02±0.00	**0.05±0.00**
	WikiTalk	N/A	N/A	N/A	0.18±0.03	0.15±0.02	0.0±0.0	**0.24±0.05**
	Lead-Lag	N/A	0.07±0.01	0.07±0.01	0.07±0.01	0.07±0.02	0.07±0.02	**0.15±0.03**
$\mathcal{O}^{\text{naive}}_{\text{vol_sum}}$	*Telegram*	0.01±0.00	0.26±0.0	0.26±0.0	0.26±0.01	0.25±0.02	0.23±0.0	**0.27±0.01**
	Blog	0.00±0.00	0.07±0.0	0.0±0.0	0.05±0.0	0.37±0.0	0.0±0.0	**0.44±0.0**
	Migration	N/A	0.01±0.00	0.01±0.00	0.01±0.00	0.02±0.00	0.01±0.00	**0.04±0.01**
	WikiTalk	N/A	N/A	N/A	0.1±0.02	0.04±0.0	0.0±0.0	**0.12±0.01**
	Lead-Lag	N/A	0.30±0.06	0.28±0.06	0.27±0.06	0.29±0.05	0.29±0.05	**0.32±0.11**

that DIGRAC provides competitive global imbalance scores in both objectives discussed and across all real-world data sets, and outperforms all other methods in all instances, on the four data sets and two objective functions. The N/A entries for *WikiTalk* are caused by memory error, and the N/A entries for InfoMap on *Migration* and *Lead-Lag* are due to its prediction of only one single cluster. *Lead-Lag* results in each year are averaged over ten runs, while the mean and standard deviation values are calculated with respect to the 19 years. The experiments indicate that edge directionality contains an important signal that DIGRAC is able to capture. A comprehensive numerical comparison is available in SI D, revealing similar conclusions.

4.3 Ablation Study

Figure 3(b) compares the performance of DiGCN replacing the loss function by $\mathcal{L}^{\text{sort}}_{\text{vol_sum}}$ from Eq. (3), indicated by "CI", or LICE, sum of supervised and self-supervised loss, on a DSBM("*cycle*", T, $n = 1000, K = 5, p = 0.02, \rho = 1$) model. We find that replacing the supervised loss function with $\mathcal{L}^{\text{sort}}_{\text{vol_sum}}$ leads to comparable results, and that adding $\mathcal{L}^{\text{sort}}_{\text{vol_sum}}$ to the loss could be beneficial. Figure 3(c) compares the test ARI performance using three variants of loss functions on the same digraph. The current choice "*sort*" performs best among these variants, indicating a benefit in only considering top pairs of individual imbalance scores. More details on loss functions, comparison with other variants, and evaluation on additional metrics are discussed in SI A, with similar conclusions. As illustrated in Figure 3(d), again on the same digraph, we also experiment on adding seeds, with the seed ratio defined as the ratio of the number of seed nodes to the number of training nodes. A supervised loss, following Tian et al. (2019), is then applied to these seeds; SI B.5 contains additional details. In conclusion, seed nodes with a supervised loss function enhance performance, and we infer that our model can further boost its performance when additional label information is available.

5 Conclusion and Future Work

DIGRAC provides an end-to-end pipeline to create node embeddings and perform directed clustering, with or without available additional node features or cluster labels. Further work will include additional experiments in the semi-supervised setting, when there exist seed nodes with known cluster labels, or when additional information is available in the form of *must-link* and *cannot-link* constraints, popular in the *constrained clustering* literature (Basu et al., 2008; Cucuringu et al., 2016). Another future direction pertains to extending our framework to also detect the number of clusters (Riolo et al., 2017; Chen & Baker, 2021), instead of specifying it a-priori, as this is typically not available in real-world applications. The current framework requires additional preliminary analysis on how many pairwise imbalance scores to consider, such as by inspecting the (initial or fitted) meta-graph adjacency matrix. It would be interesting to build a more powerful framework that can automatically detect the value β used in the current model, to select the subset of influential pairs of imbalances.

Further research directions will address the performance in the sparse regime, where spectral methods are known to underperform, and various regularizations have been proven to be effective both on the theoretical and experimental fronts; for example, see regularization in the sparse regime for the undirected settings (Chaudhuri et al., 2012; Amini et al., 2013; Cucuringu et al., 2020b). Finally, adapting our pipeline for directed clustering in extremely large networks, possibly combined with sampling methods or mini-batch (Hamilton et al., 2017), is a direction worth exploring, rendering DIGRAC applicable to large scale industrial applications.

REFERENCES

Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, pp. 36–43, 2005.

Arash A. Amini, Aiyou Chen, Peter J. Bickel, and Elizaveta Levina. Pseudo-likelihood methods for community detection in large sparse networks. *The Annals of Statistics*, 41(4):2097–2122, 2013.

Sugato Basu, Ian Davidson, and Kiri Wagstaff. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. CRC Press, 2008.

Stefanos Bennett, Mihai Cucuringu, and Gesine Reinert. Detection and clustering of lead-lag networks for multivariate time series with an application to financial markets. *7th SIGKDD Workshop on Mining and Learning from Time Series (MiLeTS)*, 2021.

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*, pp. 874–883. PMLR, 2020.

Alexandre Bovet and Peter Grindrod. The Activity of the Far Right on Telegram. `https://www.researchgate.net/publication/346968575_The_Activity_of_the_Far_Right_on_Telegram_v21`, 2020.

Kamalika Chaudhuri, Fan Chung, and Alexander Tsiatas. Spectral clustering of graphs with general degrees in the extended planted partition model. In *25th Annual Conference on Learning Theory*, volume 23 of *Proceedings of Machine Learning Research*, pp. 35.1–35.23, Edinburgh, Scotland, 2012. JMLR Workshop and Conference Proceedings.

Yilin Chen and Jack W Baker. Community detection in spatial correlation graphs: Application to non-stationary ground motion modeling. *Computers and Geosciences*, 2021.

Samir Chowdhury and Tom Needham. Generalized spectral clustering via gromov-wasserstein learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 712–720. PMLR, 2021.

Mihai Cucuringu. Sync-rank: Robust ranking, constrained ranking and rank aggregation via eigenvector and SDP synchronization. *IEEE Transactions on Network Science and Engineering*, 3(1): 58–79, 2016. ISSN 23274697. doi: 10.1109/TNSE.2016.2523761.

Mihai Cucuringu, Ioannis Koutis, Sanjay Chawla, Gary Miller, and Richard Peng. Scalable Constrained Clustering: A Generalized Spectral Method. *Artificial Intelligence and Statistics Conference (AISTATS) 2016*, 2016.

Mihai Cucuringu, Huan Li, He Sun, and Luca Zanetti. Hermitian matrices for clustering directed graphs: insights and applications. In *International Conference on Artificial Intelligence and Statistics*, pp. 983–992. PMLR, 2020a.

Mihai Cucuringu, Apoorv Vikram Singh, Déborah Sulem, and Hemant Tyagi. Regularized spectral methods for clustering signed networks. *arXiv:2011.01737*, 2020b.

Mukeshwar Dhamala, Govindan Rangarajan, and Mingzhou Ding. Analyzing information flow in brain networks with nonparametric Granger causality. *NeuroImage*, 41(2):354–362, 2008. ISSN 1053-8119. doi: https://doi.org/10.1016/j.neuroimage.2008.02.020. URL `https://www.sciencedirect.com/science/article/pii/S1053811908001328`.

Nicolas Dugué and Anthony Perez. *Directed Louvain: maximizing modularity in directed networks*. PhD thesis, Université d'Orléans, 2015.

Andrew Elliott, Paul Reidy Milton Martinez Luaces, Mihai Cucuringu, and Gesine Reinert. Anomaly detection in networks using spectral methods and network comparison approaches. *arXiv preprint arXiv:1901.00402*, 2019.

Andrew Elliott, Angus Chiu, Marya Bazzi, Gesine Reinert, and Mihai Cucuringu. Core–periphery structure in directed networks. *Proceedings of the Royal Society A*, 476(2241):20190783, 2020.

Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

A Harzallah and R Sadourny. Observed lead-lag relationships between indian summer monsoon and some meteorological variables. *Climate dynamics*, 13(9):635–648, 1997.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

Caiyan Jia, Yafang Li, Matthew B Carson, Xiaoyang Wang, and Jian Yu. Node attribute-enhanced community detection in complex networks. *Scientific Reports*, 7(1):1–15, 2017.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Steinar Laenen and He Sun. Higher-order spectral clustering of directed graphs. *Advances in Neural Information Processing Systems*, 2020.

Andrea Lancichinetti, Filippo Radicchi, José J Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. *PloS one*, 6(4):e18961, 2011.

Elizabeth A Leicht and Mark EJ Newman. Community structure in directed networks. *Physical Review Letters*, 100(11):118703, 2008.

Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th International Conference on World Wide Web*, pp. 695–704, 2008.

Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1361–1370, 2010.

Xin Liu, Hui-Min Cheng, and Zhong-Yuan Zhang. Evaluation of community detection methods. *IEEE Transactions on Knowledge and Data Engineering*, 32(9):1736–1746, 2019.

Yi Ma, Jianye Hao, Yaodong Yang, Han Li, Junqi Jin, and Guangyong Chen. Spectral-based graph convolutional network for directed graphs. *arXiv preprint arXiv:1907.08990*, 2019.

Fragkiskos D Malliaros and Michalis Vazirgiannis. Clustering and community detection in directed networks: A survey. *Physics reports*, 533(4):95–142, 2013.

Bojan Mohar. A new kind of Hermitian matrices for digraphs. *Linear Algebra and its Applications*, 584:343–352, 2020.

Federico Monti, Karl Otness, and Michael M. Bronstein. Motifnet: A motif-based graph convolutional network for directed graphs. In *2018 IEEE Data Science Workshop (DSW)*, pp. 225–228, 2018. doi: 10.1109/DSW.2018.8439897.

Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

William R. Palmer and Tian Zheng. Spectral clustering for directed networks. In Rosa M. Benito, Chantal Cherifi, Hocine Cherifi, Esteban Moro, Luis Mateus Rocha, and Marta Sales-Pardo (eds.), *Complex Networks & Their Applications IX*, pp. 87–99, Cham, 2021. Springer International Publishing. ISBN 978-3-030-65347-7.

Marc J Perry. *State-to-state Migration Flows, 1995 to 2000*. US Department of Commerce, Economics and Statistics Administration, US . . . , 2003.

Maria A Riolo, George T Cantwell, Gesine Reinert, and Mark EJ Newman. Efficient method for estimating the number of communities in a network. *Physical Review E*, 96(3):032310, 2017.

Karl Rohe, Tai Qin, and Bin Yu. Co-clustering directed graphs to discover asymmetries and directional communities. *Proceedings of the National Academy of Sciences*, 113(45):12679–12684, 2016.

Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences*, 105(4):1118–1123, 2008.

Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science advances*, 5 (11), 2019.

Venu Satuluri and Srinivasan Parthasarathy. Symmetrizations for clustering directed graphs. In *Proceedings of the 14th International Conference on Extending Database Technology*, pp. 343–354, 2011.

Ali Shojaie and Emily B Fox. Granger causality: A review and recent advances. *arXiv preprint arXiv:2105.02675*, 2021.

Didier Sornette and Wei-Xing Zhou. Non-parametric determination of real-time lag structure between two time series: the 'optimal thermal causal path'method. *Quantitative Finance*, 5(6):577–591, 2005.

Yu Tian, Long Zhao, Xi Peng, and Dimitris N Metaxas. Rethinking kernel methods for node representation learning on graphs. *Advances in Neural Information Processing Systems*, 32, 2019.

Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David Rosenblum, and Andrew Lim. Digraph inception convolutional networks. *Advances in Neural Information Processing Systems*, 33, 2020a.

Zekun Tong, Yuxuan Liang, Changsheng Sun, David S Rosenblum, and Andrew Lim. Directed graph convolutional network. *arXiv preprint arXiv:2004.13970*, 2020b.

Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.

Hongteng Xu, Dixin Luo, and Lawrence Carin. Scalable gromov-wasserstein learning for graph partitioning and matching. *Advances in neural information processing systems*, 32:3052–3062, 2019a.

Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin Duke. Gromov-wasserstein learning for graph matching and node embedding. In *International conference on machine learning*, pp. 6932–6941. PMLR, 2019b.

Hongtengl Xu. Gromov-wasserstein factorization models for graph clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6478–6485, 2020.

Jingnan Zhang, Xin He, and Junhui Wang. Directed community detection with network embedding. *Journal of the American Statistical Association*, pp. 1–11, 2021a.

Xitong Zhang, Yixuan He, Nathan Brugnone, Michael Perlmutter, and Matthew Hirn. Magnet: A neural network for directed graphs. *arXiv preprint arXiv:2102.11391*, 2021b.