DIGRAC: DIGRAPH CLUSTERING BASED ON FLOW IMBALANCE
SUPPLEMENTARY INFORMATION

Anonymous authors
Paper under double-blind review

A LOSS AND OBJECTIVES

A.1 ADDITIONAL DETAILS ON PROBABILISTIC CUT AND VOLUME

Recall that the **probabilistic cut** from cluster $\mathcal{C}_k$ to $\mathcal{C}_l$ is defined as

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

where $\mathbf{P}_{(:,k)}, \mathbf{P}_{(:,l)}$ denote the k^{th} and l^{th} columns of the assignment probability matrix $\mathbf{P}$, respectively. The **imbalance flow** between clusters $\mathcal{C}_k$ and $\mathcal{C}_l$ is defined as

$$|W(\mathcal{C}_k, \mathcal{C}_l) - W(\mathcal{C}_l, \mathcal{C}_k)|,$$

for $k, l \in \{0, \ldots, K-1\}$. The loss functions proposed in the main paper can be understood in terms of a probabilistic notion of degrees, as follows. We define the probabilistic out-degree of node v_i with respect to cluster k by $\tilde{d}^{(\text{out})}_{i,k} = \sum_{j=1}^{n} \mathbf{A}_{i,j} \cdot \mathbf{P}_{j,k} = (\mathbf{A}\mathbf{P}_{(:,k)})_i$, where subscript i refers to the i^{th} entry of the vector $\mathbf{A}\mathbf{P}_{(:,k)}$. Similarly, we define the probabilistic in-degree of node v_i with respect to cluster k by $\tilde{d}^{(\text{in})}_{i,k} = (\mathbf{A}^T\mathbf{P}_{(:,k)})_i$, where $\mathbf{A}^T$ is the transpose of $\mathbf{A}$. The **probabilistic degree** of node v_i with respect to cluster k is $\tilde{d}_{i,k} = \tilde{d}^{(\text{in})}_{i,k} + \tilde{d}^{(\text{out})}_{i,k} = ((\mathbf{A}^T + \mathbf{A})\mathbf{P}_{(:,k)})_i = \sum_{j=1}^{n} (\mathbf{A}_{i,j} + \mathbf{A}_{j,i}) \cdot \mathbf{P}_{j,k}$.

For comparisons and ease of interpretation, it is advantageous to normalize the imbalance flow between clusters; for this purpose, we introduce the probabilistic volume of a cluster, as follows. The *probabilistic out-volume* for cluster $\mathcal{C}_k$ is defined as $VOL^{(\text{out})}(\mathcal{C}_k) = \sum_{i,j} \mathbf{A}_{j,i} \cdot \mathbf{P}_{j,k}$, and the *probabilistic in-volume* for cluster $\mathcal{C}_k$ is defined as $VOL^{(\text{in})}(\mathcal{C}_k)(\mathbf{A}^T\mathbf{P}_{(:,k)})_i$, where $\mathbf{A}^T$ is the transpose of $\mathbf{A}$. These volumes can be viewed as sum of probabilistic out-degrees and in-degrees, respectively; for example, $VOL^{(\text{in})}(\mathcal{C}_k) = \sum_{i=1}^{n} \tilde{d}^{(\text{in})}_{i,k}$. Then, it holds true that

$$VOL^{(\text{out})}(\mathcal{C}_k) = \sum_{i,j} \mathbf{A}_{i,j} \cdot \mathbf{P}_{i,k} \geq \sum_{i,j} \mathbf{A}_{i,j} \cdot \mathbf{P}_{i,k} \cdot \mathbf{P}_{j,l} = W(\mathcal{C}_k, \mathcal{C}_l), \tag{1}$$

since entries in $\mathbf{P}$ are probabilities, which are in $[0, 1]$, and all entries of $\mathbf{A}$ are nonnegative. Similarly, $VOL^{(\text{in})}(\mathcal{C}_k) \geq W(\mathcal{C}_l, \mathcal{C}_k)$.

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

$$\min(VOL(\mathcal{C}_k), VOL(\mathcal{C}_l)) \geq \max(W(\mathcal{C}_k, \mathcal{C}_l), W(\mathcal{C}_l, \mathcal{C}_k)) \geq |W(\mathcal{C}_k, \mathcal{C}_l) - W(\mathcal{C}_l, \mathcal{C}_k)|. \tag{2}$$

When there exists a strong imbalance, then $|W(\mathcal{C}_k, \mathcal{C}_l) - W(\mathcal{C}_l, \mathcal{C}_k)| \approx \max(W(\mathcal{C}_k, \mathcal{C}_l), W(\mathcal{C}_l, \mathcal{C}_k))$. As an extreme case, if $\mathbf{P}_{j,l} = 1$ for all nonnegative terms in the summations in Eq. (1), and $VOL^{(\text{in})}(\mathcal{C}_k) = 0$, then $|W(\mathcal{C}_k, \mathcal{C}_l) - W(\mathcal{C}_l, \mathcal{C}_k)| = VOL(\mathcal{C}_k)$.

A.2 VARIANTS OF NORMALIZATION

Recall that the imbalance term involved in most of our experiments, named $\text{CI}^{\text{vol\_sum}}$, is defined as

$$\text{CI}^{\text{vol\_sum}}(k,l) = 2\frac{|W(\mathcal{C}_k,\mathcal{C}_l) - W(\mathcal{C}_l,\mathcal{C}_k)|}{VOL(\mathcal{C}_k) + VOL(\mathcal{C}_l)} \in [0,1]. \tag{3}$$

An alternative, which does not take volumes into account, is given by

$$\text{CI}^{\text{plain}}(k,l) = \left|\frac{W(\mathcal{C}_k,\mathcal{C}_l) - W(\mathcal{C}_l,\mathcal{C}_k)}{W(\mathcal{C}_k,\mathcal{C}_l) + W(\mathcal{C}_l,\mathcal{C}_k)}\right| = 2\left|\frac{W(\mathcal{C}_k,\mathcal{C}_l)}{W(\mathcal{C}_k,\mathcal{C}_l) - W(\mathcal{C}_l,\mathcal{C}_k)} - \frac{1}{2}\right| \in [0,1]. \tag{4}$$

We call this cut flow imbalance CI^{plain} as it does not penalize extremely unbalanced cluster sizes.

To achieve balanced cluster sizes and still constrain each imbalance term to be in $[0,1]$, one solution is to multiply the imbalance flow value by the minimum of $VOL(\mathcal{C}_k)$ and $VOL(\mathcal{C}_l)$, and then divide by $\max_{(k',l')\in\mathcal{T}}(\min(VOL(\mathcal{C}_{k'}), VOL(\mathcal{C}_{l'})))$, where $\mathcal{T} = \{(\mathcal{C}_k,\mathcal{C}_l) : 0 \le k < l \le K-1, k,l \in \mathbb{Z}\}$. The reason for using $\mathcal{T}$ is that $\text{CI}^{\text{plain}}(k,l)$ is symmetric with respect to k and l, and $\text{CI}^{\text{plain}}(k,l) = 0$ whenever $k = l$. Note that the maximum of the minimum here equals the second largest volume among clusters. We then obtain $\text{CI}^{\text{vol\_min}}$ as

$$\text{CI}^{\text{vol\_min}}(k,l) = \text{CI}^{\text{plain}}(k,l) \times \frac{\min(VOL(\mathcal{C}_k), VOL(\mathcal{C}_l))}{\max_{(k',l')\in\mathcal{T}}(\min(VOL(\mathcal{C}_{k'}), VOL(\mathcal{C}_{l'})))}. \tag{5}$$

Another potential choice, denoted $\text{CI}^{\text{vol\_max}}$, whose normalization follows from the same reasoning as $\text{CI}^{\text{vol\_sum}}$, is given by

$$\text{CI}^{\text{vol\_max}}(k,l) = \frac{|W(\mathcal{C}_k,\mathcal{C}_l) - W(\mathcal{C}_l,\mathcal{C}_k)|}{\max(VOL(\mathcal{C}_k), VOL(\mathcal{C}_l))} \in [0,1]. \tag{6}$$

A.3 VARIANTS OF CHOOSING THE PAIRWISE IMBALANCE SCORES

We consider three variants for choosing the cluster pairs.
• (1) The ***"sort"*** variant picks the largest β pairwise cut 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. For example, when we have a "cycle" meta-graph with three clusters and no ambient nodes, then $\beta = 3$. When we have a "path" meta-graph with three clusters and ambient nodes, then $\beta = 1$.
• (2) The ***"naive"*** variant considers all possible $\binom{K}{2}$ pairwise cut imbalance values.
• (3) The ***"std"*** variant only considers pairwise cut imbalance values that are 3 standard deviations away from the imbalance values; the standard deviation is calculated under the null hypothesis that the between-cluster relationship has no direction preference, i.e. $\mathbf{F}_{k,l} = \mathbf{F}_{l,k}$, as follows.

Suppose two clusters have only noisy links between them (no edge in the meta-graph $\mathbf{F}$). Suppose also that the underlying network is fixed in terms of the number of nodes and where edges exist; the only randomness stems from the direction of an edge. Then, for each edge between these two clusters, say, clusters $\mathcal{C}_k$ and $\mathcal{C}_l$, the edge direction is random, i.e. the edge is from $\mathcal{C}_k$ to $\mathcal{C}_l$ with probability 0.5, and $\mathcal{C}_l$ to $\mathcal{C}_k$ with probability 0.5 also. Let $\mathcal{E}^{k,l}$ denote the set of edges between $\mathcal{C}_k$ and $\mathcal{C}_l$ if $\mathcal{E}^{k,l}$ is not empty, then for every edge $e \in \mathcal{E}^{k,l}$, define a Rademacher random variable X_e by

$$X_e = \begin{cases} 1 & \text{if the edge is from } \mathcal{C}_k \text{ to } \mathcal{C}_l, \\ -1, & \text{otherwise.} \end{cases} \tag{7}$$

Then $(X_e + 1)/2 \sim Ber(0.5)$ is a Bernoulli(0.5) random variable with mean $2 \times 0.5 - 1 = 0$ and variance $2^2 \times 0.5 \times (1 - 0.5) = 1$. In the case of unweighted edges, the total number of edges between $\mathcal{C}_k$ and $\mathcal{C}_l$ is $|\mathcal{E}^{k,l}| = W(\mathcal{C}_k,\mathcal{C}_l) + W(\mathcal{C}_l,\mathcal{C}_k)$, and that the sum of X_e terms is $\sum_{e\in\mathcal{E}^{k,l}} X_e = W(\mathcal{C}_k,\mathcal{C}_l) - W(\mathcal{C}_l,\mathcal{C}_k)$. In the case of weighted edges, with symmetric edge weights $w_{i,j} = w_{j,i}$ given and only edge direction random, it holds that $W(\mathcal{C}_k,\mathcal{C}_l) - W(\mathcal{C}_l,\mathcal{C}_k) = \sum_{e\in\mathcal{E}^{k,l}} X_e w_e$.

Let us assume that the edge indicators are independent and that $\sum_{e\in\mathcal{E}^{k,l}} w_e^2 > 0$. Under the null hypothesis that there is no meta-graph edge between $\mathcal{C}_k$ and $\mathcal{C}_l$, the random variable $\frac{\sum_{e\in\mathcal{E}^{k,l}} X_e w_e}{\sqrt{\sum_{e\in\mathcal{E}^{k,l}} w_e^2}}$

has mean 0 and variance 1. Assuming that the weights are bounded above and that $\sum_{e \in \mathcal{E}^{k,l}} w_e^2$ is bounded away from 0 with increasing network size, we can employ the Central Limit Theorem for sums of independent random variables, see for example Theorem 3.4 in Chen et al. (2010). Then, under the null hypothesis, approximately 99.7 % of the observations would fall within 3 standard deviations from 0. While this calculation makes many assumptions and ignores reciprocal edges, the resulting threshold is still a useful guideline for restricting attention to pairwise imbalance values which are very likely to capture a true signal.

A.4 SELECTION OF THE LOSS FUNCTION

Table 1: Naming conventions for objectives and loss functions

Selection variant / CI	$CI^{\text{vol\_sum}}$	$CI^{\text{vol\_min}}$	$CI^{\text{vol\_max}}$	CI^{plain}
sort	$\mathcal{O}^{\text{sort}}_{\text{vol\_sum}}, \mathcal{L}^{\text{sort}}_{\text{vol\_sum}}$	$\mathcal{O}^{\text{sort}}_{\text{vol\_min}}, \mathcal{L}^{\text{sort}}_{\text{vol\_min}}$	$\mathcal{O}^{\text{sort}}_{\text{vol\_max}}, \mathcal{L}^{\text{sort}}_{\text{vol\_max}}$	$\mathcal{O}^{\text{sort}}_{\text{plain}}, \mathcal{L}^{\text{sort}}_{\text{plain}}$
std	$\mathcal{O}^{\text{std}}_{\text{vol\_sum}}, \mathcal{L}^{\text{std}}_{\text{vol\_sum}}$	$\mathcal{O}^{\text{std}}_{\text{vol\_min}}, \mathcal{L}^{\text{std}}_{\text{vol\_min}}$	$\mathcal{O}^{\text{std}}_{\text{vol\_max}}, \mathcal{L}^{\text{std}}_{\text{vol\_max}}$	$\mathcal{O}^{\text{std}}_{\text{plain}}, \mathcal{L}^{\text{std}}_{\text{plain}}$
naive	$\mathcal{O}^{\text{naive}}_{\text{vol\_sum}}, \mathcal{L}^{\text{naive}}_{\text{vol\_sum}}$	$\mathcal{O}^{\text{naive}}_{\text{vol\_min}}, \mathcal{L}^{\text{naive}}_{\text{vol\_min}}$	$\mathcal{O}^{\text{naive}}_{\text{vol\_max}}, \mathcal{L}^{\text{naive}}_{\text{vol\_max}}$	$\mathcal{O}^{\text{naive}}_{\text{plain}}, \mathcal{L}^{\text{naive}}_{\text{plain}}$

Table 1 provides naming conventions of all the twelve pairs of variants of objectives and loss functions used in this paper. We select the loss functions for DIGRAC based on two representative models, and compare the performance of different loss functions. We use $d = 32$, hidden units, $h = 2$ hops, and no seed nodes. Figures 1(a) and 2 compare twelve choices of loss combinations on a DSBM with $n = 1000$ nodes, $K = 5$ blocks, $\rho = 1, p = 0.02$ without ambient nodes, with a complete meta-graph structure. The subscript indicates the choice of pairwise imbalance, and the superscript indicates the variant of selecting pairs. Figures 1(b) and 3 are based on a DSBM with $n = 1000$ nodes, $K = 5$ blocks, $\rho = 1, p = 0.02$ without ambient nodes, with a cycle meta-graph structure.

These figures indicate that the "*sort*" variant generally provides the best test ARI performance and the best overall global imbalance scores, among which using normalizations $\text{CI}^{\text{vol\_sum}}$ and $\text{CI}^{\text{vol\_max}}$ perform the best. $\mathcal{L}^{\text{sort}}_{\text{vol\_min}}$ appears to behave worse than $\mathcal{L}^{\text{sort}}_{\text{vol\_sum}}$ and $\mathcal{L}^{\text{sort}}_{\text{vol\_max}}$, even when using the "sort" variant to select pairwise imbalance scores. One possible explanation is that $\mathcal{L}^{\text{sort}}_{\text{vol\_min}}$ does not penalize extreme volume sizes, and that it takes minimum as well as maximum which, as functions of the data, are not as smooth as taking a summation. Throughout our experiments in the main text, we hence use the loss function $\mathcal{L}^{\text{sort}}_{\text{vol\_sum}}$.

B IMPLEMENTATION DETAILS

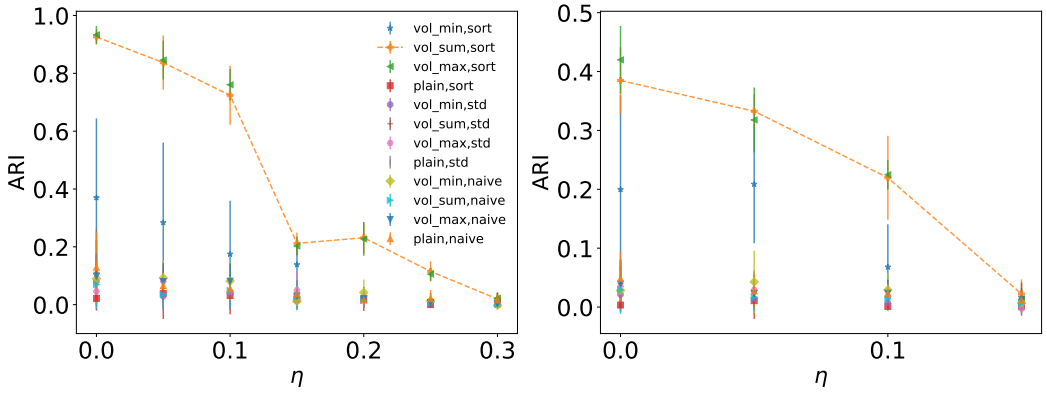
B.1 CODE

To fully reproduce our results, anonymized code and preprocessed data are available at `https://anonymous.4open.science/r/1b728e97-cc2b-4e6a-98ea-37668813536c`.

B.2 HARDWARE

Experiments were conducted on a compute node with 8 Nvidia RTX 8000, 48 Intel Xeon Silver 4116 CPUs and 1000GB RAM, a compute node with 4 NVIDIA GeForce RTX 2080, 32 Intel Xeon E5-2690 v3 CPUs and 64GB RAM, a compute node with 2 NVIDIA Tesla K80, 16 Intel Xeon E5-2690 CPUs and 252GB RAM, and an Intel 2.90GHz i7-10700 processor with 8 cores and 16 threads.

With this setup, all experiments for spectral methods, MagNet, DiGCN and DIGRAC can be completed within two days, including repeated experiments, to obtain averages over multiple runs. DGCN and DiGCN_ib have much longer run time (especially DGCN, which is space-consuming, and we cannot run many experiments in parallel), with a total of three days for both of them to finish. The slow speed stems from the competitor methods; some of the other GNN methods take a long time to run. Table 1 in the main text shows N/A values for Bi_sym and for DD_sym exactly for this reason. Empirically, DIGRAC is the fastest among all GNN methods to which it is compared.



(a) DSBM("complete", F, $n = 1000, K = 5, p = 0.02, \rho = 1$)

(b) DSBM("cycle", F, $n = 1000, K = 5, p = 0.02, \rho = 1$)

Figure 1: ARI comparison of loss functions on DSBM with 1000 nodes, 5 blocks, $\rho = 1, p = 0.02$ without ambient nodes, of cycle (left) and complete (right) meta-graph structures, respectively. The first component of the legend is the choice of pairwise imbalance, and the second component is the variant of selecting pairs. The naming conventions for the abbreviations in the legend are provided in Table 1.

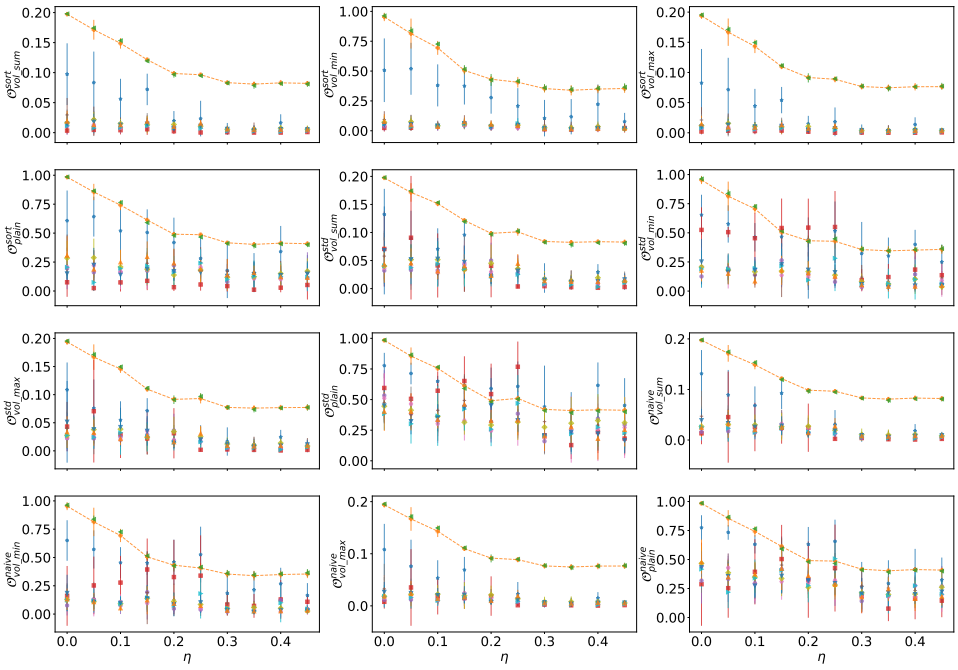


Figure 2: Imbalance scores comparison of loss functions on DSBM with 1000 nodes, 5 blocks, $\rho = 1, p = 0.02$ without ambient nodes, of the **complete meta-graph** structure. The legend is the same as Figure 1(a).

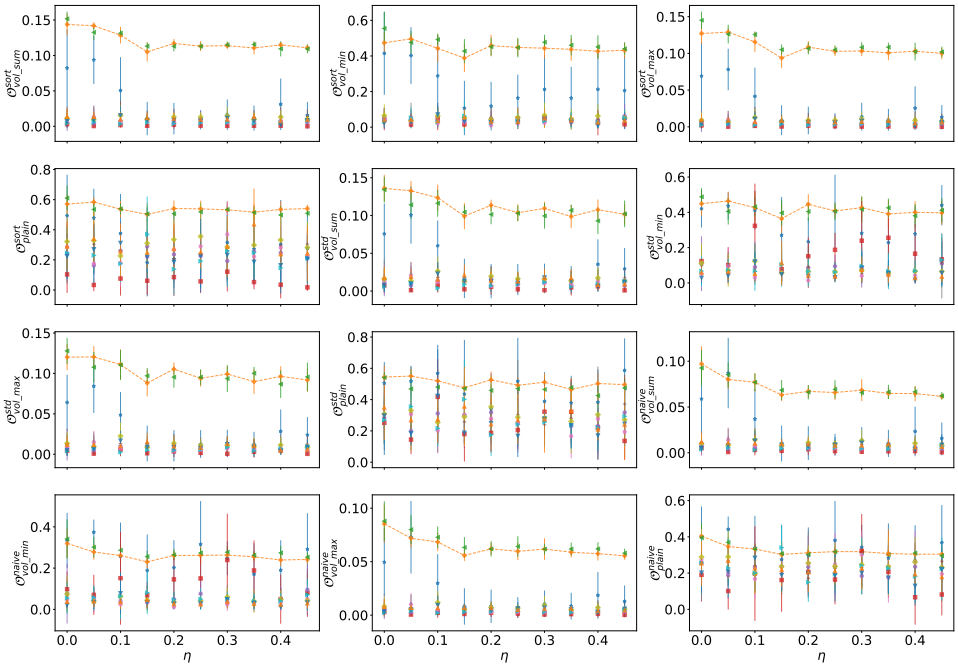


Figure 3: Imbalance scores comparison of loss functions on DSBM with 1000 nodes, 5 blocks, $\rho = 1, p = 0.02$ without ambient nodes, of the **cyclic meta-graph** structure. The legend is the same as Figure 1(a).

B.3 DATA

The results comparing DIGRAC with other methods on synthetic data are averaged over 50 runs, five synthetic networks under the same setting, each with 10 different data splits. For synthetic data, 10% of all nodes are selected as test nodes for each cluster (the actual number is the ceiling of the total number of nodes times 0.1, to avoid falling below 10% of test nodes), 10% are selected as validation nodes (for model selection and early-stopping; again, we consider the ceiling for the actual number), while the remaining roughly 80% are selected as training nodes (the actual number can never be higher than 80% due to using the ceiling for both the test and validation splits). To further clarify the training setup, we use 0% of the labels in training. As DIGRAC is a self-supervised method in principle we could use all nodes for training. However for a fair comparison with other GNN methods we use only 80% of the nodes for training. For supervised methods our split of 80% - 10% - 10% is a standard split. For the non-GNN methods, all nodes are used for training.

For both synthetic and real-world data sets, we extract the largest weakly connected component for experiments, as our framework could be applied to different weakly connected components, if the digraph is disconnected. Isolated nodes do not include any imbalance information. As customary in community detection, they are often omitted in real networks. When “ground-truth" is given, test results are averaged over 10 different data splits on one network. When no labels are available, results are averaged over 10 different data splits.

Averaged results are reported with error bars representing one standard deviation in the figures, and plus/minus one standard deviation in the tables.

Our synthetic data, DSBM, which we denote by DSBM $(\mathcal{M}, \mathbb{1}(\text{ambient}), n, K, p, \rho, \eta)$, is built similarly to Cucuringu et al. (2020) but with possibly unequal cluster sizes: • (1) Assign cluster sizes $n_0 \leq n_1 \leq \cdots \leq n_{K-1}$ with size ratio $\rho \geq 1$, as follows. If $\rho = 1$ then the first $K-1$ clusters have the same size $\lfloor n/K \rfloor$ and the last cluster has size $n - (K-1)\lfloor n/K \rfloor$. If $\rho > 1$, we set $\rho_0 = \rho^{\frac{1}{K-1}}$. Solving $\sum_{i=0}^{K-1} \rho_0^i n_0 = n$ and taking integer value gives $n_0 = \left\lfloor n(1-\rho_0)/(1-\rho_0^K) \right\rfloor$. Further, set $n_i = \lfloor \rho_0 n_{i-1} \rfloor$, for $i = 1, \cdots, K-2$ if $K \geq 3$, and $n_{K-1} = n - \sum_{i=0}^{K-2} n_i$. Then the ratio of

the size of the largest to the smallest cluster is approximately $\rho_0^{K-1} = \rho$. • (2) Assign each node randomly to one of K clusters, so that each cluster has the allocated size. • (3) For node $v_i, v_j \in \mathcal{C}_k$, independently sample an edge from node v_i to node v_j with probability $p \cdot \tilde{\mathbf{F}}_{k,k}$. • (4) For each pair of different clusters $\mathcal{C}_k, \mathcal{C}_l$ with $k \neq l$, 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 $p \cdot \tilde{\mathbf{F}}_{k,l}$.

For real-world data sets, we choose the number K of clusters in the meta-graph and the number β of edges between clusters in the meta-graph as follows. As they are needed as input for DIGRAC, we resort to Herm_rw (Cucuringu et al., 2020) as an initial view of the network clustering. When a suitable meta-graph is suggested in a previous publication, then we use that choice. Otherwise, the number K of clusters is determined using the clustering from Herm_rw. First, we pick a range of K, and for each K, we calculate the global imbalance scores and plot the predicted meta-graph flow matrix $\mathbf{F}'$ based on the clustering from Herm_rw. Its entries are defined as

$$\mathbf{F}'(k,l) = \mathbb{1}(W(\mathcal{C}_k, \mathcal{C}_l) + W(\mathcal{C}_l, \mathcal{C}_k) > 0) \times \frac{W(\mathcal{C}_k, \mathcal{C}_l)}{W(\mathcal{C}_k, \mathcal{C}_l) + W(\mathcal{C}_l, \mathcal{C}_k)}. \tag{8}$$

These entries can be viewed as predicted probabilities of edge directions. Then, we choose K from this range so that the predicted meta-graph flow matrix has the highest imbalance scores and strong imbalance in the predicted meta-graph flow matrix.

The choice of β is as follows. We plot the ranked pairs of CI^{plain} values from Herm_rw and select the β which is at least as large as $K-2$, to allow the meta-graph to be connected, and which corresponds to a large drop in the plot.

Here we provide a brief description for each of the data sets; Table 2 gives the number, n, of nodes, the number, $|\mathcal{E}|$, of directed edges, the number $|\mathcal{E}^r|$, of reciprocal edges (self-loops are counted once and for $u \neq v$, a reciprocal edge $u \to v, v \to u$ is counted twice) as well as their percentage among all edges, for the real-world networks, illustrating the variability in network size and density (defined as $|\mathcal{E}|/[n(n-1)]$).

• *Telegram* (Bovet & Grindrod, 2020) is a pairwise influence network between $n = 245$ Telegram channels with $|\mathcal{E}| = 8,912$ directed edges. It is found in Bovet & Grindrod (2020) that this network reveals a core-periphery structure in the sense of Elliott et al. (2020). Following Bovet & Grindrod (2020) we assume $K = 4$ clusters, and the core-periphery structures gives $\beta = 5$.

• *Blog* (Adamic & Glance, 2005) records $|\mathcal{E}| = 19,024$ directed edges between $n = 1,212$ political blogs from the 2004 US presidential election. In Adamic & Glance (2005) it is found that there is an underlying structure with $K = 2$ clusters corresponding to the Republican and Democratic parties. Hence we choose $K = 2$ and $\beta = 1$.

• *Migration* (Perry, 2003) reports the number of people that migrated between pairs of counties in the US during 1995-2000. It involves $n = 3,075$ countries and $|\mathcal{E}| = 721,432$ directed edges after obtaining the largest weakly connected component. We choose $K = 10$ and $\beta = 9$, following Cucuringu et al. (2020). Since the original digraph has entries extremely large, to cope with these outliers, we preprocess the input network by

$$\mathbf{A}_{i,j} = \frac{\mathbf{A}_{i,j}}{\mathbf{A}_{i,j} + \mathbf{A}_{j,i}} \mathbb{1}(\mathbf{A}_{i,j} > 0), \forall i, j \in \{1, \cdots, n\}, \tag{9}$$

which follows the preprocessing of Cucuringu et al. (2020). The results for not doing this preprocessing is provided in Table 10.

• *WikiTalk* (Leskovec et al., 2010) contains all users and discussion from the inception of Wikipedia until Jan. 2008. The $n = 2,388,953$ nodes in the network represent Wikipedia users and a directed edge from node v_i to node v_j denotes that user i edited at least once a talk page of user j. There are $|\mathcal{E}| = 5,018,445$ edges. We choose $K = 10$ clusters among candidates $\{2, 3, 5, 6, 8, 10\}$, and $\beta = 10$.

• *Lead-Lag* (Bennett et al., 2021) contains yearly lead-lag matrices from 269 stocks from 2001 to 2019. We choose $K = 10$ clusters based on the GICS industry sectors (Phillips & Ormsby, 2016), and choose $\beta = 3$ to emphasize the top three pairs of imbalance values. The lead-lag matrices are built from time series of daily price log returns, as detailed in (Bennett et al., 2021). The lead-lag metric for entry (i, j) in the network encodes a measure of the extent to which stock i leads stock j, and is obtained by applying a functional that computes the signed normalized area under the curve (auc) of the standard cross-correlation function (ccf). The resulting matrix is skew-symmetric, and

entry (i, j) quantifies the extent to which stock i leads or lags stocks j, thus leading to a directed network interpretation. Starting from the skew-symmetric matrix, we further convert negative entries to zero, so that the resulting digraph can be directly fed into other methods; note that this step does not throw away any information, and is pursued only to render the representation of the digraph consistent with the format expected by all methods compared, including DIGRAC. Note that the statistics given in Table 2 are averaged over the 19 years.

Table 2: Summary statistics for the real-world networks.

data set	n	$\lvert\mathcal{E}\rvert$	density	weighted	$\lvert\mathcal{E}^r\rvert$	$\frac{\lvert\mathcal{E}^r\rvert}{\lvert\mathcal{E}\rvert}(\%)$
Telegram	245	8,912	$1.28 \cdot 10^{-2}$	True	1,572	17.64
Blog	1,222	19,024	$1.49 \cdot 10^{-1}$	True	4,617	24.27
Migration	3,075	721,432	$7.63 \cdot 10^{-2}$	True	351,100	48.67
WikiTalk	2,388,953	5,018,445	$8.79 \cdot 10^{-7}$	False	723,526	14.42
Lead-Lag	269	29,159	$4.04 \cdot 10^{-1}$	True	0.00	0.00

As input features, after obtaining eigenvectors from Hermitian matrices constructed as in Cucuringu et al. (2020), we standardize each column vector so that it has mean zero and variance one. We use these features for all GNN methods except MagNet, since MagNet has its own way of generating random features of dimension one.

B.4 Hyperparameters

We conduct hyperparmeter selection via a greedy search. To explain the details, consider for example the following synthetic data setting: DSBM with 1000 nodes, 5 clusters, $\rho = 1$, and $p = 0.02$, without ambient nodes under different hyperparameter settings. By default, we use the loss function $\mathcal{L}^{\text{sort}}_{\text{vol\_sum}}$, $d = 32$ hidden units, hop $h = 2$, and no seed nodes. Instead of a grid search, we tune hyperparameters according to what performs the best in the default setting of the respective GNN method. The procedure starts with a random setting. For the next iteration, the hyperparameters are set to the current best setting (based on the last iteration), independently. For example, if we start with $a = 1, b = 2, c = 3$, and we find that under this default setting, the best a (when fixing $b = 2, c = 3$) is 2 and the best b (when fixing $a = 1, c = 3$) is 3, and the best c is 3 (when fixing $a = 1, b = 2$), then for the next iteration, we set $a = 2, b = 3, c = 3$. If two settings give similar results, we choose the simpler setting, for example, the smaller hop size. When we reach a local optimum, we stop searching. Indeed, just a few iterations (less than five) were required for us to find the current setting, as DIGRAC tends to be robust to most hyperparameters.

Figure 4, 5 and 6 are plots corresponding to the same setting but for three different meta-graph structures, namely the complete meta-graph structure, the cycle structure but with ambient nodes, and the complete structure with ambient nodes, respectively.

In theory, more hidden units give better expressive power. To reduce complexity, we use 32 hidden units throughout, which seems to have desirable performance. We observe that for low-noise regimes, more hidden units actually hurt performance. We can draw a similar conclusion about the hyperparameter selection. In terms of τ, DIGRAC seems to be robust to different choices. Therefore, we use $\tau = 0.5$ throughout.

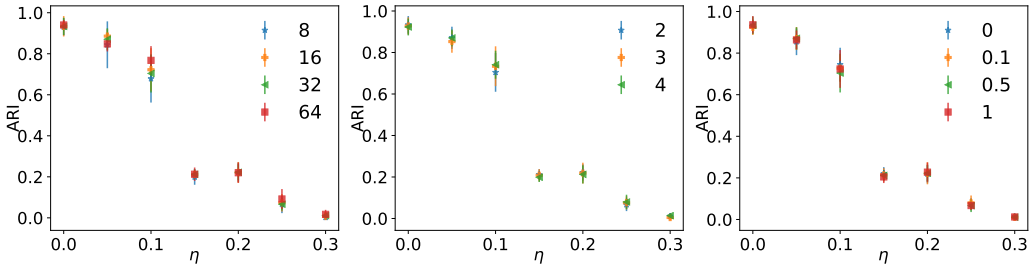
B.5 Use of seed nodes in a semi-supervised manner

B.5.1 Supervised loss

For seed nodes in $\mathcal{V}^{\text{seed}}$, similar to the loss function in Tian et al. (2019), we use as a supervised loss function the sum of a cross-entropy loss and a triplet loss. The cross-entropy loss is given by

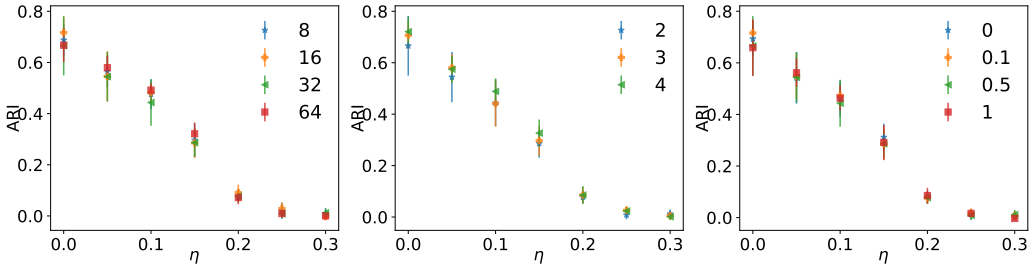
$$\mathcal{L}_{\text{CE}} = -\frac{1}{|\mathcal{V}^{\text{seed}}|} \sum_{v_i \in \mathcal{V}^{\text{seed}}} \sum_{k=1}^{K} \mathbb{1}(v_i \in \mathcal{C}_k) \log\left((\mathbf{p}_i)_k\right), \tag{10}$$

where $\mathbb{1}$ is the indicator function, $\mathcal{C}_k$ denotes the k^{th} cluster, and $(\mathbf{p}_i)_k$ denotes the k^{th} entry of probability vector $(\mathbf{p}_i)$. With the function $L : \mathbb{R}^2 \to \mathbb{R}$ given by $L(x, y) = [x - y]_+$ (where the



(a) Vary d. (b) Vary the hop h. (c) Vary τ.

Figure 4: Hyperparameter analysis on different hyperparameter settings on the **complete** DSBM with 1000 nodes, 5 clusters, $\rho = 1$, and $p = 0.02$ **without** ambient nodes.



(a) Vary d. (b) Vary the hop h. (c) Vary τ.

Figure 5: Hyperparameter analysis on different hyperparameter settings on the **complete** DSBM with 1000 nodes, 5 clusters, $\rho = 1$, and $p = 0.02$ **with** ambient nodes.

subscript $+$ indicates taking the maximum of the expression value and 0), the triplet loss is defined as

$$\mathcal{L}_{\text{triplet}} = \frac{1}{|\mathcal{S}|} \sum_{(v_i, v_j, v_k) \in \mathcal{S}} L(\text{CS}(\mathbf{z}_i, \mathbf{z}_j), \text{CS}(\mathbf{z}_i, \mathbf{z}_k), \tag{11}$$

where $\mathcal{S} \subseteq \mathcal{V}^{\text{seed}} \times \mathcal{V}^{\text{seed}} \times \mathcal{V}^{\text{seed}}$ is a set of node triplets: v_i is an anchor seed node, and v_j is a seed node from the same cluster as the anchor, while v_k is from a different cluster; and $\text{CS}(\mathbf{z}_i, \mathbf{z}_j)$ is the cosine similarity of the embeddings of nodes v_i and v_j. We choose cosine similarity so as to avoid sensitivity to the magnitude of the embeddings. The triplet loss is designed so that, given two seed nodes from the same cluster and one seed node from a different cluster, the respective embeddings of the pairs from different clusters should be farther away than the embedding of the pair within the same cluster.

We then consider the weighted sum $\mathcal{L}_{\text{CE}} + \gamma_t \mathcal{L}_{\text{triplet}}$ as the supervised part of the loss function for DIGRAC, for some parameter $\gamma_t > 0$. The parameter γ_t arises as follows. The cosine similarity between two randomly picked vectors in d dimensions is bounded by $\sqrt{\ln(d)/d}$ with high probability. In our experiments $d = 32$, and $\sqrt{\ln(2d)/(2d)} \approx 0.25$. In contrast, for fairly uniform clustering, the cross-entropy loss grows like $\log n$, which in our experiments ranges between 3 and 17. Thus some balancing of the contribution is required. Following Tian et al. (2019), we choose $\gamma_t = 0.1$ in our experiments.

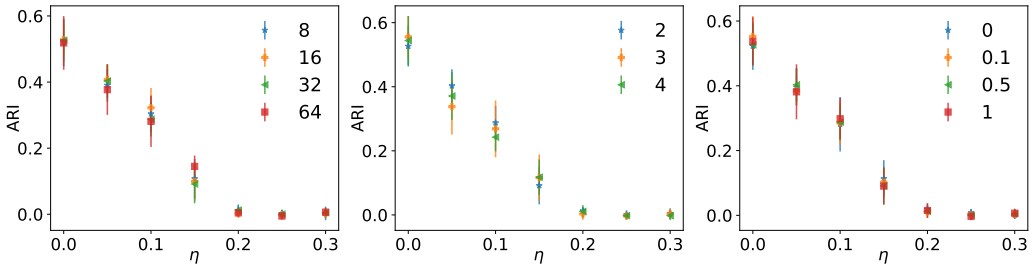
B.5.2 Overall objective function

We recall the objective function and the loss function

$$\mathcal{O}^{\text{sort}}_{\text{vol\_sum}} = \frac{1}{\beta} \sum_{(\mathcal{C}_k, \mathcal{C}_l) \in \mathcal{T}(\beta)} \text{CI}^{\text{vol\_sum}}(k, l), \quad \text{and} \quad \mathcal{L}^{\text{sort}}_{\text{vol\_sum}} = 1 - \mathcal{O}^{\text{sort}}_{\text{vol\_sum}}, \tag{12}$$

from the main text. By combining Eq. (10), Eq. (11), and Eq. (12), our objective function for semi-supervised training with known seed nodes minimizes

$$\mathcal{L} = \mathcal{L}^{\text{sort}}_{\text{vol\_sum}} + \gamma_s (\mathcal{L}_{\text{CE}} + \gamma_t \mathcal{L}_{\text{triplet}}), \tag{13}$$



(a) Vary d. (b) Vary the hop h. (c) Vary τ.

Figure 6: Hyperparameter analysis on different hyperparameter settings on the **cycle** DSBM with 1000 nodes, 5 clusters, $\rho = 1$, and $p = 0.02$ **with** ambient nodes.

where $\gamma_s, \gamma_t > 0$ are weights for the supervised part of the loss and triplet loss within the supervised part, respectively. We set $\gamma_s = 50$ as we want our model to perform well on seed nodes. The weights could be tuned depending on how important each term is perceived to be.

B.6 Training

For all synthetic data, we train DIGRAC with a maximum of 1000 epochs, and stop training when no gain in validation performance is achieved for 200 epochs (early-stopping). For real-world data, no "ground-truth" labels are available; we use all nodes to train and stop training when the training loss does not decrease for 200 epochs, or when we reach the maximum number of epochs, 1000.

For the two-layer MLP, we do not have a bias term for each layer, and we use Rectified Linear Unit (ReLU) followed by a dropout layer with 0.5 dropout probability between the two layers, following Tian et al. (2019). We use Adam (Kingma & Ba, 2014) as the optimizer and ℓ_2 regularization with weight decay $5 \cdot 10^{-4}$ to avoid overfitting. We use as learning rate 0.01 throughout.

B.7 Implementation details for the comparison methods

In our experiments, we compare DIGRAC against five spectral methods and five GNN-based supervised methods on synthetic data, and spectral methods on real data. The reason we are not able to compare DIGRAC with the above GNNs on these data sets is due to the fact that these data sets do not have labels, which are required by the other GNN methods. We use the same hyperparameter settings stated in these papers. Data splits for all models are the same; the comparison GNNs are trained with 80% nodes under label supervision.

For MagNet, we use $q = 0.25$ for the phase matrix as in (Zhang et al., 2021), because it is mentioned that $q = 0.25$ lays the most emphasis on directionality, which is our main focus in this paper. Code for MagNet is from `https://github.com/matthew-hirn/magnet`. We use the code from `https://github.com/flyingtango/DiGCN/blob/main/code/digcn.py` to obtain the log of probability matrix $\mathbf{P}$ for the methods DiGCN and DiGCN_app. The only difference between these two methods is whether or not to use approximate Laplacian based on personalized PageRank. The "adj_type" options for them correspond to "or" and "appr", respectively.

For DiGCN_ib, we use the code from `https://github.com/flyingtango/DiGCN/blob/main/code/digcn_ib.py` with option "adj_type" equals "ib". As a recommended option in Tong et al. (2020a), we use three layers for DiGCN_ib and two layers for DiGCN and DiGCN_app. All other settings are the same as in the original paper (Tong et al., 2020a).

B.8 Algorithm and Complexity analysis

To avoid computationally expensive and space unfriendly matrix operations, as described in Eq. (1) in the main text, DIGRAC uses an efficient sparsity-aware implementation, described in Algorithm 1, without explicitly calculating the sets of powers $\mathcal{A}^{s,h}$ and $\mathcal{A}^{t,h}$. We omit the subscript $\mathcal{V}$ for ease of notation. The algorithm is efficient in the sense that it takes sparse matrices as input, and never explicitly computes a multiplication of two $n \times n$ matrices. Therefore, for input feature dimension

d_{in} and hidden dimension d, if $d' = \max(d_{\text{in}}, d) \ll n$, time and space complexity of DIMPA, and implicitly DIGRAC, is $\mathcal{O}(|\mathcal{E}|d'h^2 + 2nd'K)$ and $\mathcal{O}(2|\mathcal{E}| + 4nd' + nK)$, respectively (Harrison & Joseph, 2018; Greiner & Jacob, 2010).

Indeed, it is a current shortcoming of DIGRAC that it does not scale well to very large networks; however, this limitation is also shared by all the GNN competitors compared against in the paper, and some of the spectral methods. DIGRAC scales well in the sense that when the underlying network is sparse, the sparsity is preserved throughout the pipeline. In contrast, Bi_sym and DD_sym (Satuluri & Parthasarathy, 2011) construct derived dense matrices for manipulation, rendering the methods no longer scalable. These methods resulted in N/A values in Table 1 in the main text. For large-scale networks, DIMPA is amenable to a minibatch version using neighborhood sampling, similar to the minibatch forward propagation algorithm in Hamilton et al. (2017); Markowitz et al. (2021). We are also aware of a framework (Fey et al., 2021) for scaling up graph neural networks automatically, where theoretical guarantees are provided, and ideas there will be exploited in future. We expect that the theoretical guarantees could be adapted to our situation.

Algorithm 1: Weighted Multi-Hop Neighbor Aggregation (DIMPA).

Input : (Sparse) row-normalized adjacency matrices $\overline{\mathbf{A}}^s, \overline{\mathbf{A}}^t$; initial hidden representations $\mathbf{H}^s, \mathbf{H}^t$; hop $h (h \geq 2)$; lists of scalar weights $\Omega^s = (\omega^s_{\mathbf{M}}, \mathbf{M} \in \mathcal{A}^{s,h}), \Omega^t = (\omega^t_{\mathbf{M}}, \mathbf{M} \in \mathcal{A}^{t,h})$.

Output : Vector representations $\mathbf{z}_i$ for all $v_i \in \mathcal{V}$ given by $\mathbf{Z}$.

$\tilde{\mathbf{X}}^s \leftarrow \overline{\mathbf{A}}^s \mathbf{H}^s$; $\quad$ $\tilde{\mathbf{X}}^t \leftarrow \overline{\mathbf{A}}^t \mathbf{H}^t$;

$\mathbf{Z}^s \leftarrow \Omega^s[0] \cdot \mathbf{H}^s + \Omega^s[1] \cdot \tilde{\mathbf{X}}^s$; $\quad$ $\mathbf{Z}^t \leftarrow \Omega^t[0] \cdot \mathbf{H}^t + \Omega^t[1] \cdot \tilde{\mathbf{X}}^t$;

for $i \leftarrow 2$ **to** h **do**

$\tilde{\mathbf{X}}^s \leftarrow \overline{\mathbf{A}}^s \tilde{\mathbf{X}}^s$; $\quad$ $\tilde{\mathbf{X}}^t \leftarrow \overline{\mathbf{A}}^t \tilde{\mathbf{X}}^t$;

$\mathbf{Z}^s \leftarrow \mathbf{Z}^s + \Omega^s[i] \cdot \tilde{\mathbf{X}}^s$; $\quad$ $\mathbf{Z}^t \leftarrow \mathbf{Z}^t + \Omega^t[i] \cdot \tilde{\mathbf{X}}^t$;

end

$\mathbf{Z} = \text{CONCAT}\,(\mathbf{Z}^s, \mathbf{Z}^t)$;

C More results on synthetic data

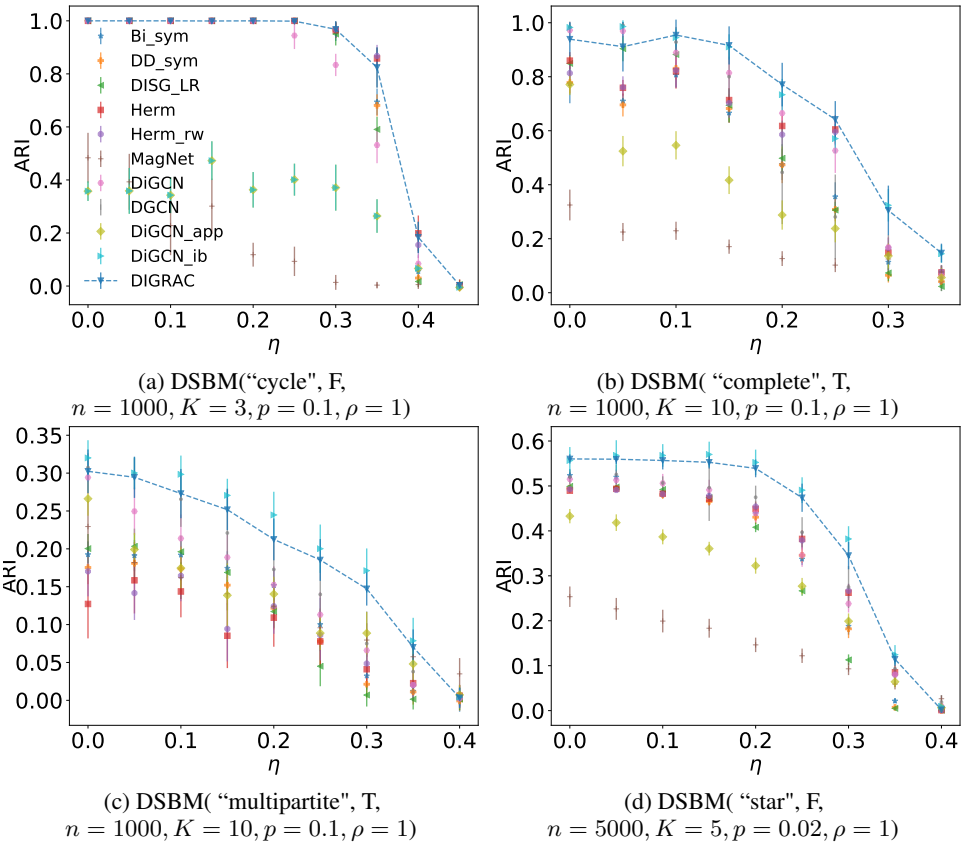
C.1 An additional meta-graph structure

Recall that the Directed Stochastic Block Models used in our experiments depend on a meta-graph adjacency matrix $\mathbf{F}$ and a filled version of it, $\tilde{\mathbf{F}}$, for some number of clusters, K, and noise level $\eta \leq 0.5$. The meta-graph adjacency matrix $\mathbf{F}$ is generated from some meta-graph structure, called $\mathcal{M}$. Based on $\mathcal{M}$, the filled meta-graph $\tilde{\mathbf{F}}$ replaces every zero in $\mathbf{F}$ that is not part of the imbalance structure with 0.5, independently of the choice of η. It is the filled meta-graph $\tilde{\mathbf{F}}$ which we feed into the DSBM generation process.The filled meta-graph creates a number of *ambient nodes* which correspond to entries which are not part of the imbalance structure and thus are not part of a meaningful cluster; the set of *ambient nodes* is also called the *ambient cluster*.

Here, we introduce an additional meta-graph structure, called "multipartite", following Elliott et al. (2019). First, when there are no ambient nodes: we divide the index set into three sets; setting $i_1 = \lfloor \frac{K}{9} \rfloor, i_2 = \lfloor \frac{3K}{9} \rfloor + i_1$, let

$$\begin{aligned} \mathbf{F}_{k,l} \quad = \quad & (1-\eta)\mathbb{1}(k < i_1, i_1 \leq l < i_2) + \eta \mathbb{1}(i_1 \leq k < i_2, l \geq i_2) \\ & + (1-\eta)\mathbb{1}(k \geq i_2, i_1 \leq l < i_2) + \eta \mathbb{1}(i_1 \leq k < i_2, l < i_1). \end{aligned}$$

When we have ambient nodes, the construction involves two steps, with the first step the same as the above but with the following changes: divide the indices into three sets, with set boundaries given by $i_1 = \lfloor \frac{K-1}{9} \rfloor, i_2 = \lfloor \frac{3(K-1)}{9} \rfloor + i_1$. The second step is to assign 0 (respectively, 0.5) to the last row and the last column of $\mathbf{F}$ (respectively, $\tilde{\mathbf{F}}$).



(a) DSBM("cycle", F, $n = 1000, K = 3, p = 0.1, \rho = 1$)

(b) DSBM("complete", T, $n = 1000, K = 10, p = 0.1, \rho = 1$)

(c) DSBM("multipartite", T, $n = 1000, K = 10, p = 0.1, \rho = 1$)

(d) DSBM("star", F, $n = 5000, K = 5, p = 0.02, \rho = 1$)

Figure 7: Node clustering test ARI comparison on four additional synthetic data sets. Dashed lines highlight DIGRAC's performance. Error bars are given by one standard error. Abbreviations for all the methods are given in Section 4 in the main text.

C.2 Additional comparison plots and analysis

Figure 7 compares the numerical performance of DIGRAC with other methods except InfoMap, which performs rather poorly in our synthetic experiments in the main text, on four more settings of synthetic data, namely, a cycle structure with three clusters, a complete structure with ten clusters, a multipartite structure with ten clusters, and a star structure with five clusters. Indeed, Infomap clusters all nodes into one big cluster for all of our synthetic experiments. Considering the results in Section 5 and Figure 7, we remark that DIGRAC gives state-of-the-art results on a wide range of network densities and noise levels, on different scales of the networks, and with different meta-graph structures, whether or not ambient nodes exist.

Note that the multipartite, the cycle and the star settings correspond to the intuition behind Tong et al. (2020a) which assumes that nodes are similar if their set of k^{th}-order neighbourhoods are similar; here the second-order neighbourhoods are similar by design. For networks with underlying meta-graph structure "star", "cycle" or "multipartite", clusters could be determined by grouping nodes that share similar in-neighbors and out-neighbors together, which aligns well with the second-order proximity used in DGCN and DiGCN_ib from Tong et al. (2020b). Therefore, these methods are naturally well-suited for dealing with the such synthetic data. We also note that although DIGRAC does not explicitly use second-order proximity, it can achieve comparable performance with DGCN and DiGCN_ib. This indicates DIGRAC's flexibility to adapt to directed networks with different underlying topologies, without explicitly utilizing higher-order proximity. On the other hand, DiGCN_ib is fully supervised, and takes much more space and time to implement, than DIGRAC. This is partially due to the use of the so-called *inception blocks* in DiGCN_ib, where multi-scale directed structure features are encoded and fused with a fusion function. As stated in Tong et al. (2020a), the worst space complexity is $\mathcal{O}(k'n^2)$, where k' is the order of proximity to consider (we

use $k' = 2$ throughout). The eigenvalue decomposition in the preprocessing step is $\mathcal{O}(n^3)$. We also remark that the approximate Laplacian based on personalized PageRank, when no inception blocks are used, performs no better than the simpler implementation without the approximation. We conclude that overall DIGRAC is a fast method for general directed clustering when directionality is the main signal, which performs as well as custom-tailored methods when the proximity neighborhood heuristic holds, while outperforming all tested methods on the complete meta-graph, where the proximity neighborhood heuristic does not hold.

D ADDITIONAL RESULTS ON REAL-WORLD DATA

D.1 EXTENDED RESULT TABLES

Tables 3, 4, 5 and 6 provide a detailed comparison of DIGRAC with spectral methods. Since no labeling information is available and all of the other competing GNN methods require labels, we do not compare DIGRAC with them on these real data sets.

In Tables 3, 4, 5 and 6, we report 12 combinations of global imbalance scores by data set. The naming convention of these imbalance scores is provided in Table 1. To assess how balanced our recovered clusters are in terms of sizes, we also report the size ratio, which is defined as the size of the largest predicted cluster to the smallest one, and the standard deviation of sizes, size std, in order to show how varied the sizes of predicted clusters are. For a relatively balanced clustering, we expect the latter two terms to be small.

Table 3: Performance comparison on *Telegram*. The best is marked in **bold red** and the second best is marked in underline blue.

Metric/Method	InfoMap	Bi_sym	DD_sym	DISG_LR	Herm	Herm_rw	DIGRAC
$\mathcal{O}^{\text{sort}}_{\text{vol\_sum}}$	0.04±0.00	0.21±0.00	0.21±0.00	0.21±0.01	0.20±0.01	0.14±0.00	**0.32±0.01**
$\mathcal{O}^{\text{sort}}_{\text{vol\_min}}$	0.47±0.00	0.67±0.00	0.61±0.00	0.66±0.02	0.66±0.02	0.19±0.00	**0.79±0.06**
$\mathcal{O}^{\text{sort}}_{\text{vol\_max}}$	0.03±0.00	0.20±0.00	0.20±0.00	0.20±0.01	0.19±0.01	0.12±0.00	**0.29±0.01**
$\mathcal{O}^{\text{sort}}_{\text{plain}}$	**1.00±0.00**	0.80±0.00	0.75±0.00	0.78±0.03	0.76±0.04	0.59±0.00	0.96±0.01
$\mathcal{O}^{\text{std}}_{\text{vol\_sum}}$	0.01±0.00	0.26±0.00	0.26±0.00	0.26±0.01	0.25±0.02	**0.35±0.00**	0.28±0.01
$\mathcal{O}^{\text{std}}_{\text{vol\_min}}$	0.16±0.00	**0.84±0.00**	0.76±0.00	0.82±0.03	0.82±0.03	0.49±0.00	0.73±0.03
$\mathcal{O}^{\text{std}}_{\text{vol\_max}}$	0.01±0.00	0.25±0.00	0.25±0.00	0.25±0.01	0.24±0.02	**0.29±0.00**	0.25±0.01
$\mathcal{O}^{\text{std}}_{\text{plain}}$	0.68±0.00	**1.00±0.00**	0.94±0.00	0.98±0.04	0.95±0.04	0.99±0.00	0.90±0.05
$\mathcal{O}^{\text{naive}}_{\text{vol\_sum}}$	0.01±0.00	0.26±0.00	0.26±0.00	0.26±0.01	0.25±0.02	0.23±0.00	**0.27±0.01**
$\mathcal{O}^{\text{naive}}_{\text{vol\_min}}$	0.11±0.00	**0.84±0.00**	0.76±0.00	0.82±0.03	0.82±0.03	0.32±0.00	0.72±0.04
$\mathcal{O}^{\text{naive}}_{\text{vol\_max}}$	0.00±0.00	**0.25±0.00**	**0.25±0.00**	**0.25±0.01**	0.24±0.02	0.20±0.00	0.24±0.01
$\mathcal{O}^{\text{naive}}_{\text{plain}}$	0.63±0.00	**1.00±0.00**	0.94±0.00	0.98±0.04	0.95±0.04	0.99±0.00	0.89±0.06
size ratio	24.750	242.000	242.000	242.000	242.00	53	**3.090**
size std	35.57	104.360	104.360	104.360	104.360	63.460	**26.39**

Tables 3, 4, 5, 6, 7, 8 and 9 reveal that DIGRAC provides competitive global imbalance scores in all of the 12 objectives introduced, and across all the real data sets, usually outperforming all the other methods. Among the tables, Table 9 provides results in terms of the distance to the best yearly performance, averaged across the 19 years; DIGRAC usually outperforms all the other methods across all the years. Note that Bi_sym and DD_sym are not able to generate results for *WikiTalk*, as large $n \times n$ matrix multiplication with its transpose causes memory issue, when $n = 2,388,953$. Small values of the size ratio and size standard deviation suggest that the normalization in the loss function penalizes tiny clusters, and that DIGRAC tends to predict balanced cluster sizes.

Table 4: Performance comparison on *Blog*. The best is marked in **bold red** and the second best is marked in underline blue.

Metric/Method	InfoMap	Bi_sym	DD_sym	DISG_LR	Herm	Herm_rw	DIGRAC
$\mathcal{O}^{\text{sort}}_{\text{vol\_sum}}$	0.07±0.00	0.07±0.00	0.00±0.00	0.05±0.00	0.37±0.00	0.00±0.00	**0.44±0.00**
$\mathcal{O}^{\text{sort}}_{\text{vol\_min}}$	0.02±0.00	0.33±0.00	0.05±0.00	0.31±0.00	0.78±0.01	**0.89±0.00**	0.76±0.00
$\mathcal{O}^{\text{sort}}_{\text{vol\_max}}$	0.05±0.00	0.05±0.00	0.00±0.00	0.04±0.00	0.26±0.00	0.00±0.00	**0.40±0.00**
$\mathcal{O}^{\text{sort}}_{\text{plain}}$	**1.00±0.00**	0.33±0.00	0.05±0.00	0.31±0.00	0.78±0.01	0.89±0.00	0.76±0.00
$\mathcal{O}^{\text{std}}_{\text{vol\_sum}}$	0.00±0.00	0.07±0.00	0.00±0.00	0.05±0.00	0.37±0.00	0.00±0.00	**0.44±0.00**
$\mathcal{O}^{\text{std}}_{\text{vol\_min}}$	0.00±0.00	0.33±0.00	0.05±0.00	0.31±0.00	0.78±0.01	**0.89±0.00**	0.76±0.00
$\mathcal{O}^{\text{std}}_{\text{vol\_max}}$	0.00±0.00	0.05±0.00	0.00±0.00	0.04±0.00	0.26±0.00	0.00±0.00	**0.40±0.00**
$\mathcal{O}^{\text{std}}_{\text{plain}}$	0.73±0.00	0.33±0.00	0.05±0.00	0.31±0.00	0.78±0.01	**0.89±0.00**	0.76±0.00
$\mathcal{O}^{\text{naive}}_{\text{vol\_sum}}$	0.00±0.00	0.07±0.00	0.00±0.00	0.05±0.00	0.37±0.00	0.00±0.00	**0.44±0.00**
$\mathcal{O}^{\text{naive}}_{\text{vol\_min}}$	0.00±0.00	0.33±0.00	0.05±0.00	0.31±0.00	0.78±0.01	**0.89±0.00**	0.76±0.00
$\mathcal{O}^{\text{naive}}_{\text{vol\_max}}$	0.00±0.00	0.05±0.00	0.00±0.00	0.04±0.00	0.26±0.00	0.00±0.00	**0.40±0.00**
$\mathcal{O}^{\text{naive}}_{\text{plain}}$	0.76±0.00	0.33±0.00	0.05±0.00	0.31±0.00	0.78±0.01	**0.89±0.00**	0.76±0.00
size ratio	**1.270**	8.700	2.450	6.100	11.93	44.26	1.860
size std	**64.50**	485	256.200	439	516.500	584	183.20

Table 5: Performance comparison on *Migration*. The best is marked in **bold red** and the second best is marked in underline blue. InfoMap results are omitted here as it predicts a single huge cluster and could not generate imbalance results.

Metric/Method	Bi_sym	DD_sym	DISG_LR	Herm	Herm_rw	DIGRAC
$\mathcal{O}^{\text{sort}}_{\text{vol\_sum}}$	0.03±0.00	0.01±0.00	0.02±0.00	0.04±0.00	0.02±0.00	**0.05±0.00**
$\mathcal{O}^{\text{sort}}_{\text{vol\_min}}$	**0.19±0.00**	0.08±0.00	0.08±0.00	0.15±0.02	0.05±0.00	0.18±0.03
$\mathcal{O}^{\text{sort}}_{\text{vol\_max}}$	0.03±0.00	0.01±0.00	0.01±0.00	0.03±0.00	0.02±0.00	**0.04±0.00**
$\mathcal{O}^{\text{sort}}_{\text{plain}}$	0.24±0.00	0.20±0.00	0.17±0.00	0.40±0.01	**0.49±0.06**	0.29±0.04
$\mathcal{O}^{\text{std}}_{\text{vol\_sum}}$	0.01±0.00	0.01±0.00	0.01±0.00	0.02±0.00	0.02±0.00	**0.04±0.01**
$\mathcal{O}^{\text{std}}_{\text{vol\_min}}$	0.10±0.00	0.05±0.00	0.05±0.00	0.08±0.01	0.04±0.00	**0.16±0.03**
$\mathcal{O}^{\text{std}}_{\text{vol\_max}}$	0.01±0.00	0.01±0.00	0.01±0.00	0.01±0.00	0.01±0.00	**0.03±0.01**
$\mathcal{O}^{\text{std}}_{\text{plain}}$	0.13±0.00	0.12±0.00	0.11±0.00	0.20±0.01	0.20±0.01	**0.26±0.01**
$\mathcal{O}^{\text{naive}}_{\text{vol\_sum}}$	0.01±0.00	0.01±0.00	0.01±0.00	0.02±0.00	0.01±0.00	**0.04±0.01**
$\mathcal{O}^{\text{naive}}_{\text{vol\_min}}$	0.09±0.00	0.04±0.00	0.04±0.00	0.08±0.01	0.01±0.00	**0.16±0.03**
$\mathcal{O}^{\text{naive}}_{\text{vol\_max}}$	0.01±0.00	0.00±0.00	0.01±0.00	0.01±0.00	0.00±0.00	**0.03±0.01**
$\mathcal{O}^{\text{naive}}_{\text{plain}}$	0.12±0.00	0.10±0.00	0.08±0.00	0.19±0.00	0.19±0.03	**0.26±0.01**
size ratio	7.780	6.070	**4.360**	36.05	1035.90	4.420
size std	135.210	132.76	**103.43**	335.790	353.060	264.500

Table 6: Performance comparison on *WikiTalk*. The best is marked in **bold red** and the second best is marked in underline blue. InfoMap results are omitted here as its large number of predicted clusters leads to memory error in imbalance calculation.

Metric/Method	DISG_LR	Herm	Herm_rw	DIGRAC
$\mathcal{O}^{\text{sort}}_{\text{vol\_sum}}$	0.18±0.03	0.15±0.02	0.00±0.00	**0.24±0.05**
$\mathcal{O}^{\text{sort}}_{\text{vol\_min}}$	0.10±0.03	0.22±0.05	0.26±0.00	**0.28±0.13**
$\mathcal{O}^{\text{sort}}_{\text{vol\_max}}$	0.16±0.03	0.09±0.01	0.00±0.00	**0.19±0.04**
$\mathcal{O}^{\text{sort}}_{\text{plain}}$	0.87±0.08	0.99±0.01	0.98±0.00	**1.00±0.00**
$\mathcal{O}^{\text{std}}_{\text{vol\_sum}}$	**0.17±0.04**	0.06±0.01	0.01±0.00	0.14±0.02
$\mathcal{O}^{\text{std}}_{\text{vol\_min}}$	0.09±0.02	0.09±0.02	**0.27±0.00**	0.18±0.08
$\mathcal{O}^{\text{std}}_{\text{vol\_max}}$	**0.15±0.04**	0.04±0.00	0.00±0.00	0.11±0.02
$\mathcal{O}^{\text{std}}_{\text{plain}}$	0.72±0.03	0.70±0.05	**0.98±0.00**	0.84±0.06
$\mathcal{O}^{\text{naive}}_{\text{vol\_sum}}$	0.10±0.02	0.04±0.00	0.00±0.00	**0.12±0.01**
$\mathcal{O}^{\text{naive}}_{\text{vol\_min}}$	0.06±0.03	0.07±0.02	**0.26±0.00**	0.15±0.07
$\mathcal{O}^{\text{naive}}_{\text{vol\_max}}$	**0.09±0.02**	0.03±0.00	0.00±0.00	**0.09±0.01**
$\mathcal{O}^{\text{naive}}_{\text{plain}}$	0.64±0.04	0.61±0.04	**0.98±0.00**	0.76±0.06
size ratio	1190162.25	2217434.50	**250.48**	71765.14
size std	713813.72	660060.33	657941.88	**643220.37**

Table 7: Performance comparison on *Lead-Lag* for year 2015. The best is marked in **bold red** and the second best is marked in underline blue. InfoMap results are omitted here as it usually predicts a single huge cluster and could not generate imbalance results.

Metric/Method	Bi_sym	DD_sym	DISG_LR	Herm	Herm_rw	DIGRAC
$\mathcal{O}^{\text{sort}}_{\text{vol\_sum}}$	0.07±0.00	0.07±0.00	0.06±0.00	0.07±0.00	0.06±0.01	**0.15±0.00**
$\mathcal{O}^{\text{sort}}_{\text{vol\_min}}$	**0.53±0.06**	0.50±0.02	0.45±0.07	0.50±0.03	0.46±0.06	0.50±0.02
$\mathcal{O}^{\text{sort}}_{\text{vol\_max}}$	0.07±0.00	0.06±0.00	0.06±0.00	0.06±0.00	0.06±0.00	**0.15±0.01**
$\mathcal{O}^{\text{sort}}_{\text{plain}}$	0.65±0.03	**0.67±0.03**	0.59±0.03	0.65±0.03	0.65±0.02	0.55±0.07
$\mathcal{O}^{\text{std}}_{\text{vol\_sum}}$	0.04±0.00	0.04±0.00	0.04±0.00	0.04±0.00	0.04±0.00	**0.11±0.02**
$\mathcal{O}^{\text{std}}_{\text{vol\_min}}$	0.27±0.03	0.27±0.02	0.24±0.02	0.27±0.02	0.26±0.04	**0.35±0.04**
$\mathcal{O}^{\text{std}}_{\text{vol\_max}}$	0.03±0.00	0.03±0.00	0.03±0.00	0.03±0.00	0.03±0.00	**0.10±0.02**
$\mathcal{O}^{\text{std}}_{\text{plain}}$	0.39±0.02	0.39±0.01	0.37±0.02	0.39±0.02	**0.40±0.02**	0.38±0.04
$\mathcal{O}^{\text{naive}}_{\text{vol\_sum}}$	0.03±0.00	0.03±0.00	0.03±0.00	0.03±0.00	0.03±0.00	**0.08±0.03**
$\mathcal{O}^{\text{naive}}_{\text{vol\_min}}$	0.20±0.02	0.20±0.02	0.17±0.03	0.20±0.02	0.20±0.03	**0.25±0.08**
$\mathcal{O}^{\text{naive}}_{\text{vol\_max}}$	0.02±0.00	0.03±0.00	0.02±0.00	0.03±0.00	0.03±0.00	**0.08±0.03**
$\mathcal{O}^{\text{naive}}_{\text{plain}}$	0.29±0.01	0.29±0.01	0.26±0.02	0.30±0.01	0.30±0.01	**0.31±0.05**
size ratio	3.070	3.110	3.060	**2.89**	2.95	15.640
size std	8.390	7.94	8.680	**7.28**	8.050	18.680

Table 8: Performance comparison on *Lead-Lag*. Results in each year is averaged over ten runs. Mean and standard deviation (after ±) are calculated over the 19 years. The best is marked in **bold red** and the second best is marked in underline blue. InfoMap results are omitted here as it usually predicts a single huge cluster and could not generate imbalance results.

Metric/Method	Bi_sym	DD_sym	DISG_LR	Herm	Herm_rw	DIGRAC
$\mathcal{O}^{\text{sort}}_{\text{vol\_sum}}$	0.07±0.01	0.07±0.01	0.07±0.01	0.07±0.02	0.07±0.02	**0.15±0.03**
$\mathcal{O}^{\text{sort}}_{\text{vol\_min}}$	**0.51±0.10**	0.48±0.09	0.47±0.10	**0.51±0.11**	0.50±0.10	0.47±0.09
$\mathcal{O}^{\text{sort}}_{\text{vol\_max}}$	0.07±0.01	0.06±0.01	0.06±0.01	0.07±0.01	0.07±0.01	**0.14±0.03**
$\mathcal{O}^{\text{sort}}_{\text{plain}}$	**0.66±0.09**	0.64±0.08	0.63±0.08	**0.66±0.09**	0.65±0.09	0.53±0.09
$\mathcal{O}^{\text{std}}_{\text{vol\_sum}}$	0.04±0.01	0.04±0.01	0.04±0.01	0.04±0.01	0.04±0.01	**0.12±0.03**
$\mathcal{O}^{\text{std}}_{\text{vol\_min}}$	0.27±0.04	0.27±0.04	0.25±0.04	0.27±0.03	0.27±0.03	**0.38±0.07**
$\mathcal{O}^{\text{std}}_{\text{vol\_max}}$	0.04±0.00	0.03±0.00	0.03±0.00	0.03±0.00	0.03±0.00	**0.11±0.02**
$\mathcal{O}^{\text{std}}_{\text{plain}}$	0.40±0.05	0.39±0.05	0.38±0.05	0.40±0.05	0.40±0.05	**0.44±0.07**
$\mathcal{O}^{\text{naive}}_{\text{vol\_sum}}$	0.03±0.01	0.03±0.01	0.03±0.01	0.03±0.01	0.03±0.01	**0.08±0.04**
$\mathcal{O}^{\text{naive}}_{\text{vol\_min}}$	0.20±0.05	0.19±0.05	0.18±0.05	0.19±0.04	0.19±0.04	**0.26±0.10**
$\mathcal{O}^{\text{naive}}_{\text{vol\_max}}$	0.03±0.01	0.02±0.01	0.02±0.01	0.02±0.00	0.02±0.00	**0.08±0.03**
$\mathcal{O}^{\text{naive}}_{\text{plain}}$	0.30±0.06	0.28±0.06	0.27±0.06	0.29±0.05	0.29±0.05	**0.32±0.11**
size ratio	3.67	**3.34**	3.900	4.110	3.880	8.070
size std	9.31	**9.14**	10.090	10.490	10.360	17.060

Table 9: Performance comparison on *Lead-Lag*, where we evaluate the performance distance to the best one in each year. Results in each year is averaged over ten runs. Mean and standard deviation (after ±) are calculated over the 19 years. The best is marked in **bold red** and the second best is marked in underline blue. InfoMap results are omitted here as it usually predicts a single huge cluster and could not generate imbalance results.

Metric/Method	Bi_sym	DD_sym	DISG_LR	Herm	Herm_rw	DIGRAC
$\mathcal{O}^{\text{sort}}_{\text{vol\_sum}}$	0.07±0.02	0.08±0.02	0.08±0.02	0.07±0.02	0.07±0.02	**0.00±0.00**
$\mathcal{O}^{\text{sort}}_{\text{vol\_min}}$	**0.01±0.01**	0.05±0.03	0.06±0.03	0.02±0.02	0.02±0.02	0.06±0.04
$\mathcal{O}^{\text{sort}}_{\text{vol\_max}}$	0.07±0.02	0.07±0.02	0.07±0.02	0.07±0.02	0.07±0.02	**0.00±0.00**
$\mathcal{O}^{\text{sort}}_{\text{plain}}$	**0.01±0.02**	0.03±0.03	0.05±0.03	**0.01±0.02**	0.02±0.02	0.14±0.03
$\mathcal{O}^{\text{std}}_{\text{vol\_sum}}$	0.08±0.02	0.08±0.02	0.08±0.02	0.08±0.02	0.08±0.02	**0.00±0.00**
$\mathcal{O}^{\text{std}}_{\text{vol\_min}}$	0.10±0.05	0.11±0.04	0.13±0.05	0.11±0.05	0.11±0.05	**0.00±0.00**
$\mathcal{O}^{\text{std}}_{\text{vol\_max}}$	0.07±0.02	0.08±0.02	0.08±0.02	0.08±0.02	0.08±0.02	**0.00±0.00**
$\mathcal{O}^{\text{std}}_{\text{plain}}$	0.04±0.03	0.05±0.04	0.06±0.04	0.04±0.04	0.04±0.03	**0.00±0.00**
$\mathcal{O}^{\text{naive}}_{\text{vol\_sum}}$	0.05±0.03	0.06±0.03	0.06±0.03	0.05±0.03	0.05±0.03	**0.00±0.00**
$\mathcal{O}^{\text{naive}}_{\text{vol\_min}}$	0.06±0.07	0.07±0.06	0.08±0.07	0.07±0.08	0.07±0.08	**0.00±0.00**
$\mathcal{O}^{\text{naive}}_{\text{vol\_max}}$	0.05±0.03	0.05±0.03	0.05±0.03	0.05±0.03	0.05±0.03	**0.00±0.00**
$\mathcal{O}^{\text{naive}}_{\text{plain}}$	0.03±0.06	0.05±0.05	0.06±0.06	0.04±0.06	0.04±0.06	**0.01±0.02**
size ratio	1.04	**0.71**	1.270	1.480	1.250	5.440
size std	0.58	**0.41**	1.360	1.770	1.630	8.340

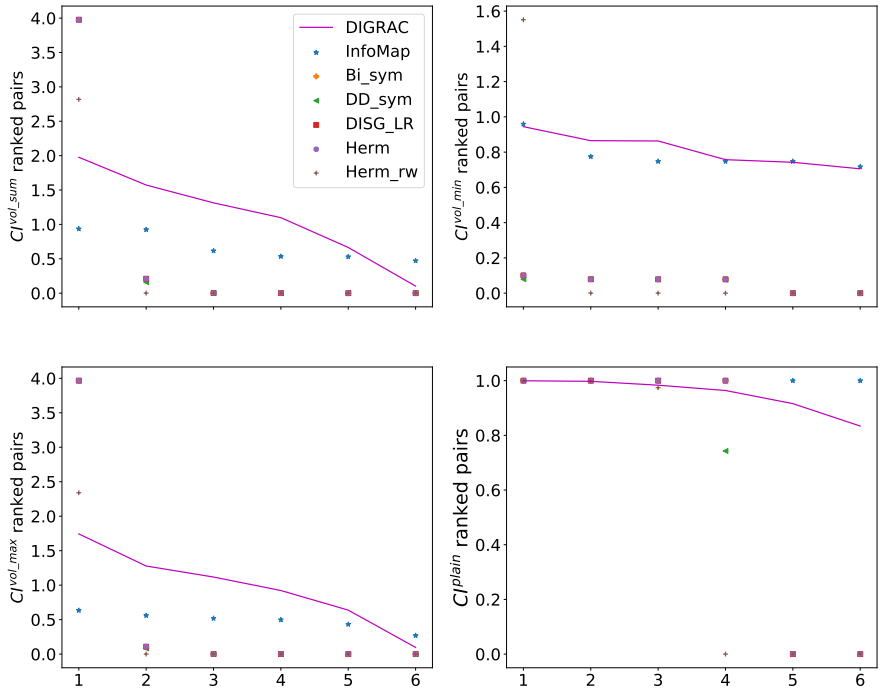


Figure 8: Ranked pairs of pairwise imbalance recovered by comparing methods for different choices of normalization on the *Telegram* data set. Lines are used to highlight DIGRAC's performance.

D.2 Ranked pairwise imbalance scores

We also plot the ranked pairwise imbalance scores for all data sets except *Blog*, which has only one possible pairwise imbalance score. For *Lead-Lag*, we only plot the year 2015 as an example; the plots for the other years are similar. Figures 8, 9, 10 and 11 illustrate that DIGRAC is able to provide comparable or higher pairwise imbalance scores for the leading pairs, especially on $\mathrm{CI}^{\mathbf{vol\_min}}$ pairs. We also observe that except for $\mathrm{CI}^{\mathbf{plain}}$, DIGRAC has a less rapid drop in pairwise imbalance scores after the first leading pair compared to Herm and Herm_rw, which can have a few pairs with higher imbalance scores than DIGRAC.

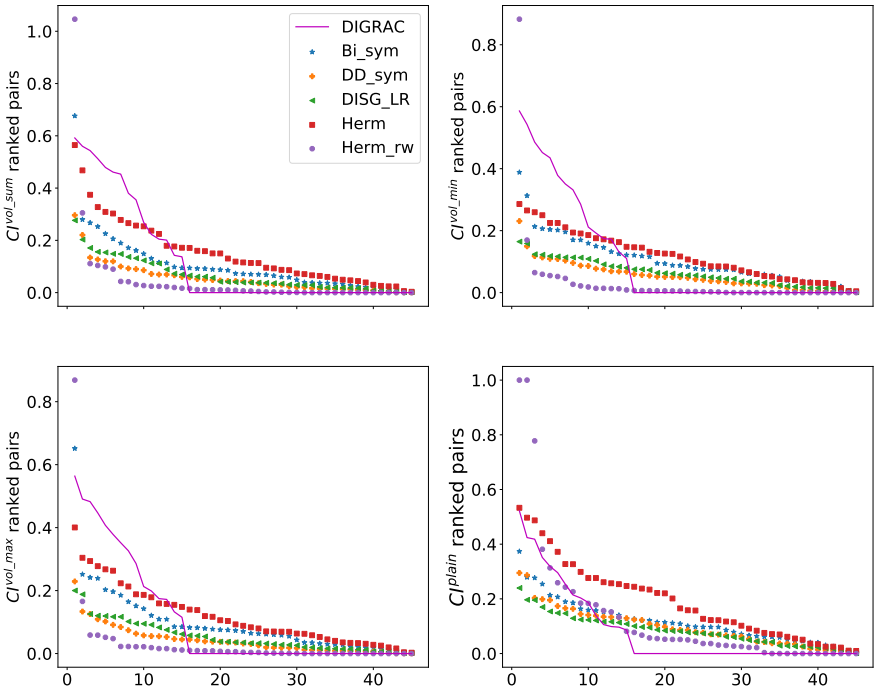


Figure 9: Ranked pairs of pairwise imbalance recovered by comparing methods for different choices of normalization on the *Migration* data set. Lines are used to highlight DIGRAC's performance. InfoMap results are omitted as it predicts one single huge cluster and could not produce imbalance results.

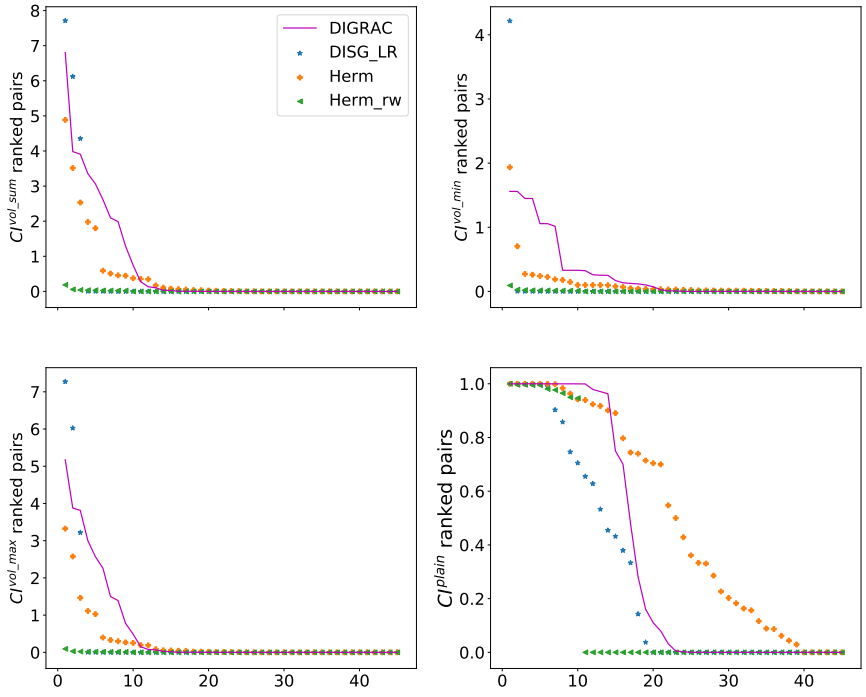


Figure 10: Ranked pairs of pairwise imbalance recovered by comparing methods for different choices of normalization on *WikiTalk* data set. Lines are used to highlight DIGRAC's performance. InfoMap results are omitted here because it triggers memory error due to the large number of predicted clusters.

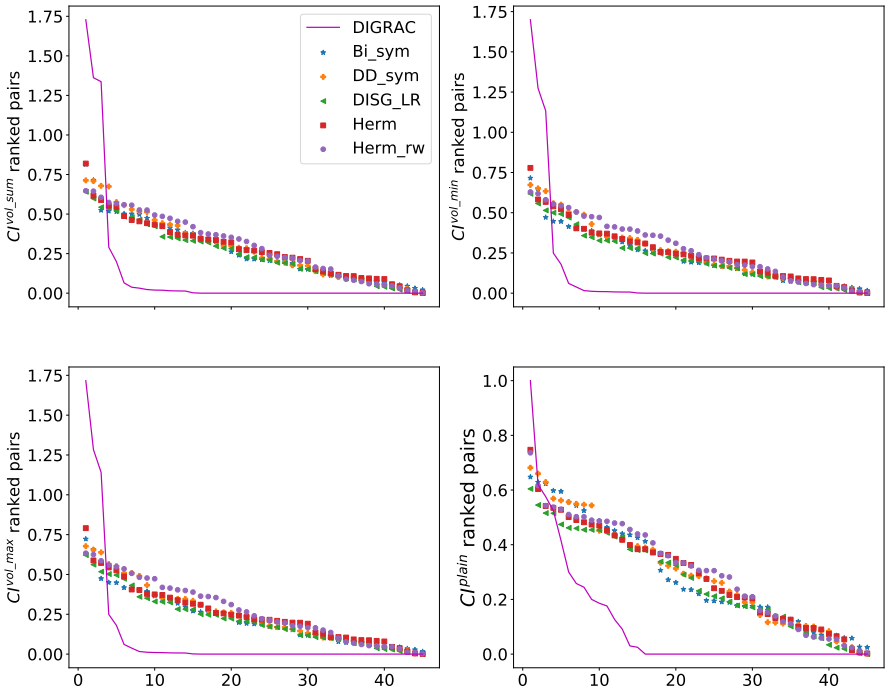


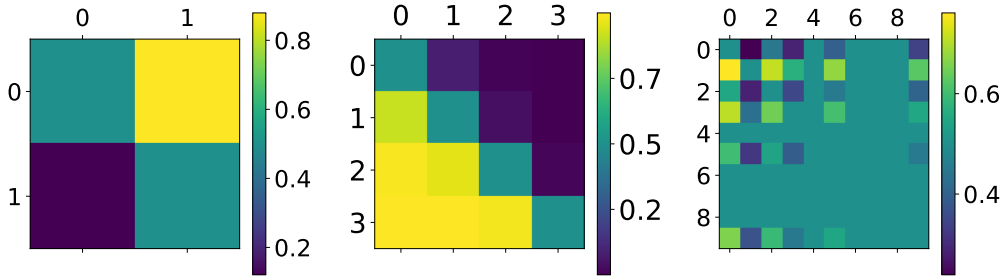
Figure 11: Ranked pairs of pairwise imbalance recovered by comparing methods for different choices of normalization on *Lead-Lag* data set. Lines are used to highlight DIGRAC's performance. InfoMap results are omitted here because it only predicts a single cluster.

D.3 Predicted meta-graph flow matrix plots

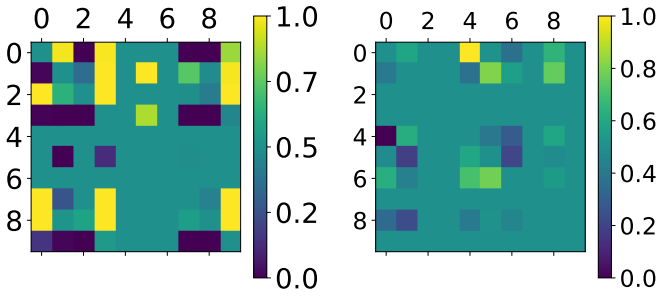
For each data set, we plot the predicted meta-graph flow matrix $\mathbf{F}'$ defined in Eq. (8).

From Figure 12, we conclude that DIGRAC is able to recover a directed flow imbalance between clusters in all of the selected data sets. Figure 12a shows a clear cut imbalance between two clusters, possibly corresponding to the Republican and Democratic parties. Figure 12b plots imbalance flows in the real data set *Telegram*, where cluster 3 is a core-transient cluster, cluster 0 is a core-sink cluster, cluster 2 is a periphery-upstream cluster, while cluster 1 is a periphery-downstream cluster (Elliott et al., 2020; Bovet & Grindrod, 2020). For *WikiTalk*, illustrated in Figure 12d, the lower-triangular part entries are typically source nodes for edges, while the upper-triangular part are target nodes. For *Lead-Lag*, taking the year 2015 as an example, DIGRAC is also able to recover high imbalance in the data.

We also note that DIGRAC would not necessarily predict the same number of clusters as assumed, so that we do not need to specify the exact number of clusters before training DIGRAC; specifying the maximum number of possible clusters suffices.



(a) *Blog* (b) *Telegram* (c) *Migration*



(d) *WikiTalk* (e) *Lead-Lag* 2015

Figure 12: Predicted meta-graph flow matrix from DIGRAC of five real-world data sets.

D.4 Migration plots

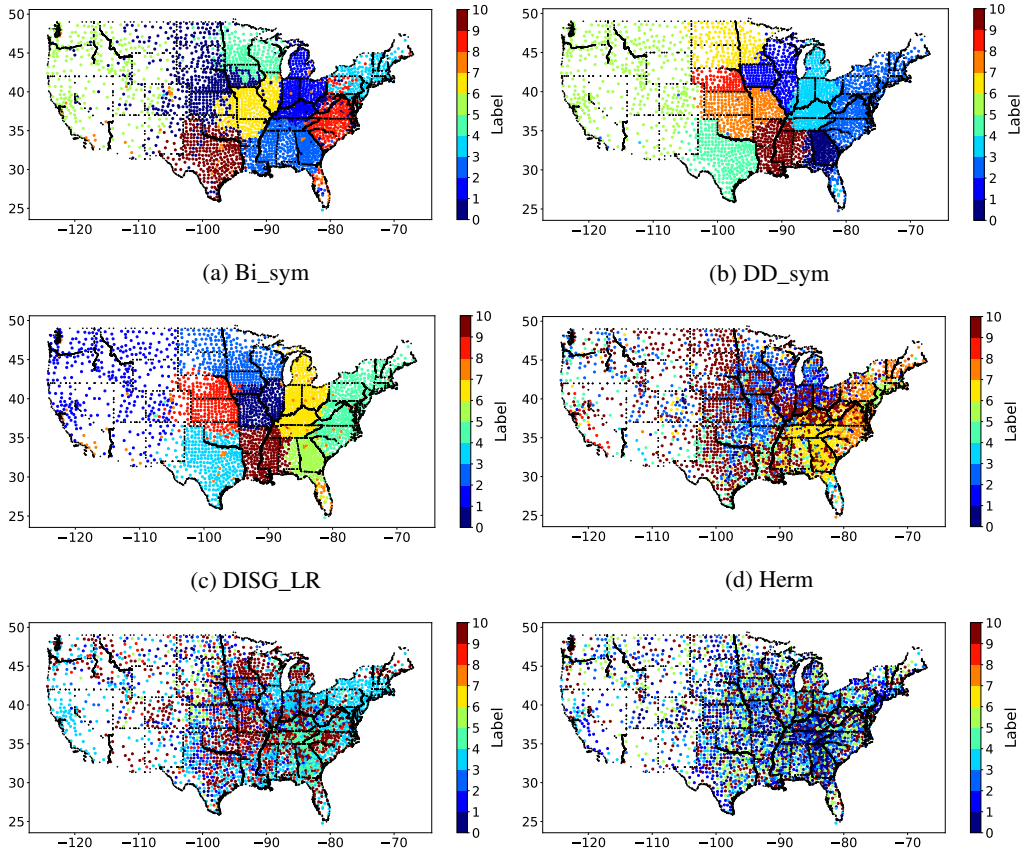
We compare DIGRAC to five spectral methods for recovering clusters for the US migration data set, and plot the recovered clusters on a map, in Figure 13.

The visualization in Figure (a-c) shows that clusters align particularly well with the political and administrative boundaries of the US states, as previously observed in Cucuringu et al. (2013). This outcome is not deemed too insightful, as it trivially reveals the fact there there is significant intra-state and inter-state migration, and does not uncover any of the information on latent migration patterns between far-away states, and more generally, between regions which are not necessarily geographically cohesive.

D.5 Coping with Outliers

As mentioned in Section B.3, the preprocessing step to use ratio of migration instead of absolute migration numbers is a way to cope with outliers (here, extremely large entries in the original digraph) in *Migration*. To validate the effectiveness of this approach to cope with outliers, Table 10 provides imbalance results for *Migration* when we do not transform the nonzero entries into ratios. Comparing with Table 5, we witness an overall decrease in the performance. In this case InfoMap no longer predicts a single huge cluster. However, its predicted number of clusters is about 44, which is too large. This also implies that InfoMap is very sensitive to the magnitude of digraph entries, while DIGRAC is not. Indeed, InfoMap gives 43 (too many) clusters for *Blog*, 19 (too many) for *Telegram*, 1 (too small) for *Migration*, and 17498 (far too many) for *WikiTalk*.

We compare DIGRAC to five spectral methods as well as InfoMap for recovering clusters for the US migration data set without the preprocessing step discussed earlier, and plot the recovered clusters on a map in Figure 15. Note that all methods, except DIGRAC, recover either clusters which are trivially small in size or contain one very large dominant cluster (as in (a), (b), (e) and to some extent, also (f)). The DISG_LR clustering and InfoMap clustering provide clear geographic boundaries, but



(a) Bi_sym (b) DD_sym

(c) DISG_LR (d) Herm

(e) Herm_rw (f) DIGRAC

Figure 13: US migration predicted clusters, along with the geographic locations of the counties as well as state boundaries (in black). InfoMap results are omitted here because it only produces one huge cluster. The input data is normalized, following Eq. (9).

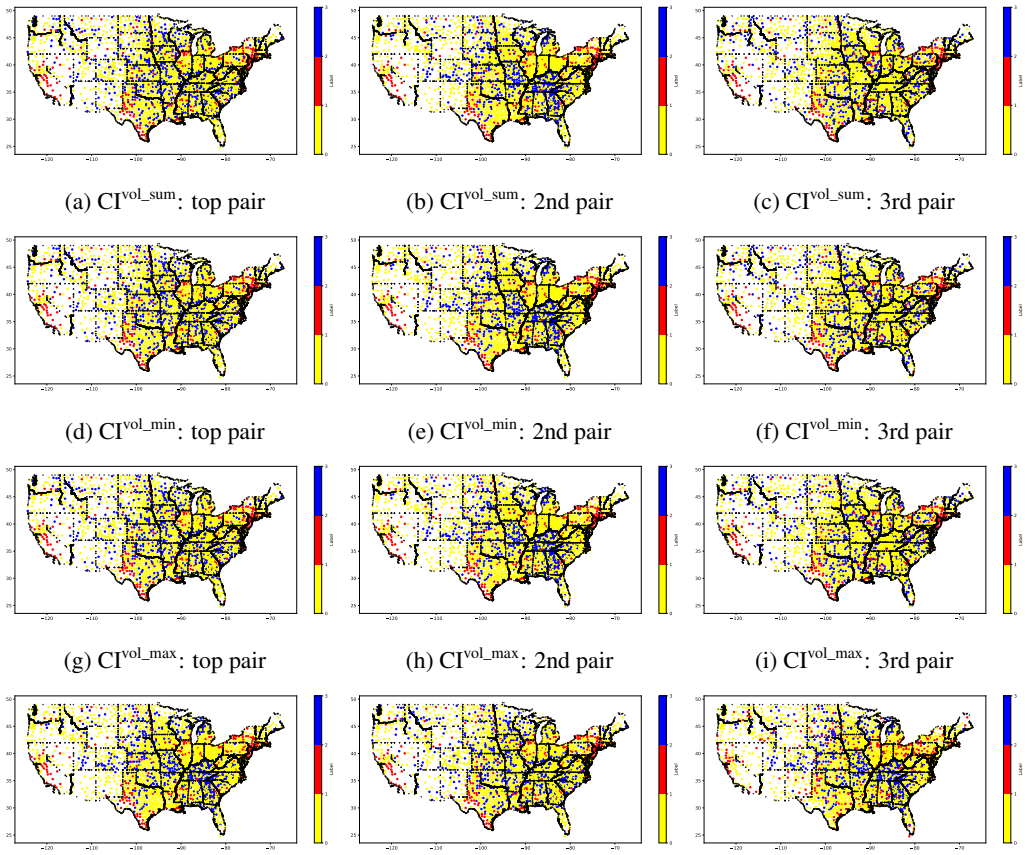
were not able to recover the imbalance among clusters. Other spectral methods generally have a dominant cluster containing most of the nodes, whereas DIGRAC has more balanced cluster sizes.

When employing methods that symmetrize the adjacency matrix (as in (a) and (b)), the migration flows between counties in different states will be lost in the process. Furthermore, the visualization in Figure (c) shows that clusters align particularly well with the political and administrative boundaries of the US states, as previously observed in Cucuringu et al. (2013). The same is for Figure (d). This outcome is not deemed very insightful, as it trivially reveals the fact that there is significant intra-state and inter-state migration, and does not uncover any of the information on latent migration patterns between far-away states, and more generally, between regions which are not necessarily geographically cohesive.

Figure 14 further plots the top three pairs of clusters based on four different imbalance scores given by DIGRAC. As shown in the figure, DIGRAC uncovers the migration trend from outside to inside, across states. This trend of the directed flow agrees with that discussed in Perry (2003), with many people migrating from New York and California to the inner states.

E Discussion of Related Methods that are not Compared Against in the Main Text

To further emphasize the importance of directionality, our synthetic data sets have no difference in density between clusters; their sole signal is in the directionality of the edges. If all edge directions



(a) CI^{vol_sum}: top pair (b) CI^{vol_sum}: 2nd pair (c) CI^{vol_sum}: 3rd pair

(d) CI^{vol_min}: top pair (e) CI^{vol_min}: 2nd pair (f) CI^{vol_min}: 3rd pair

(g) CI^{vol_max}: top pair (h) CI^{vol_max}: 2nd pair (i) CI^{vol_max}: 3rd pair

(j) CI^{plain}: top pair (k) CI^{plain}: 2nd pair (l) CI^{plain}: 3rd pair

Figure 14: US migration predicted cluster pairs with top imbalance, along with the geographic locations of the counties as well as state boundaries (in black). Red (label 1) is the sending cluster while blue (label 2) is the receiving cluster. Yellow (label 0) denotes all the other locations being considered. Subcaptions show the imbalance score and the rank based on that score.

were to be removed, then no algorithm out there should be available to detect the clusters. To further support our claim why some methods mentioned in Section 2 in the main text are not appropriate for comparison, we have applied the default setting versions of the Louvain method (Dugué & Perez, 2015), the Leiden algorithm (Traag et al., 2019) and OSLOM (Lancichinetti et al., 2011), to our synthetic data sets, and find that they do not detect the structure in the data, with ARI and NMI values very close to zero, and very low imbalance values. In particular, Louvain and Leiden tend to give a larger number of clusters than the ground truth which is designed to have small cluster sizes. OSLOM outputs clusters with extreme sizes, either a huge cluster containing (almost always) all the nodes, or every node forming a cluster by itself.

On the real-world data sets, these methods often give numbers of clusters that do not match our expectations. (*Blog* has two underlying parties, *Telegram* has a four-cluster core-periphery structure). Louvain clusters nodes from *Blog* into 8-13 clusters (too many), *Telegram* into 4-5 clusters (acceptable), *Migration* into 5-7 clusters (acceptable), *WikiTalk* into 150-219 clusters (too many), and *Lead-Lag* into 10-55 clusters (acceptable or a bit too many). Leiden gives 12 (too many) clusters for *Blog*, 4-5 for *Telegram*, 5-6 for *Migration*, 170-248 (too many) for *WikiTalk*, and 10-55 clusters (acceptable or a bit too many) for *Lead-Lag*. OSLOM gives 6 clusters for *Blog* (too many), 16 for *Telegram* (too many), and 46 for *Migration* (too many). It could not generate results for *WikiTalk* after running for 12 hours, and hence we omit its discussion here. On *Lead-Lag*, OSLOM places every node in a single cluster for most of the years, and clusters the rest of the years into either a huge single cluster or two clusters.

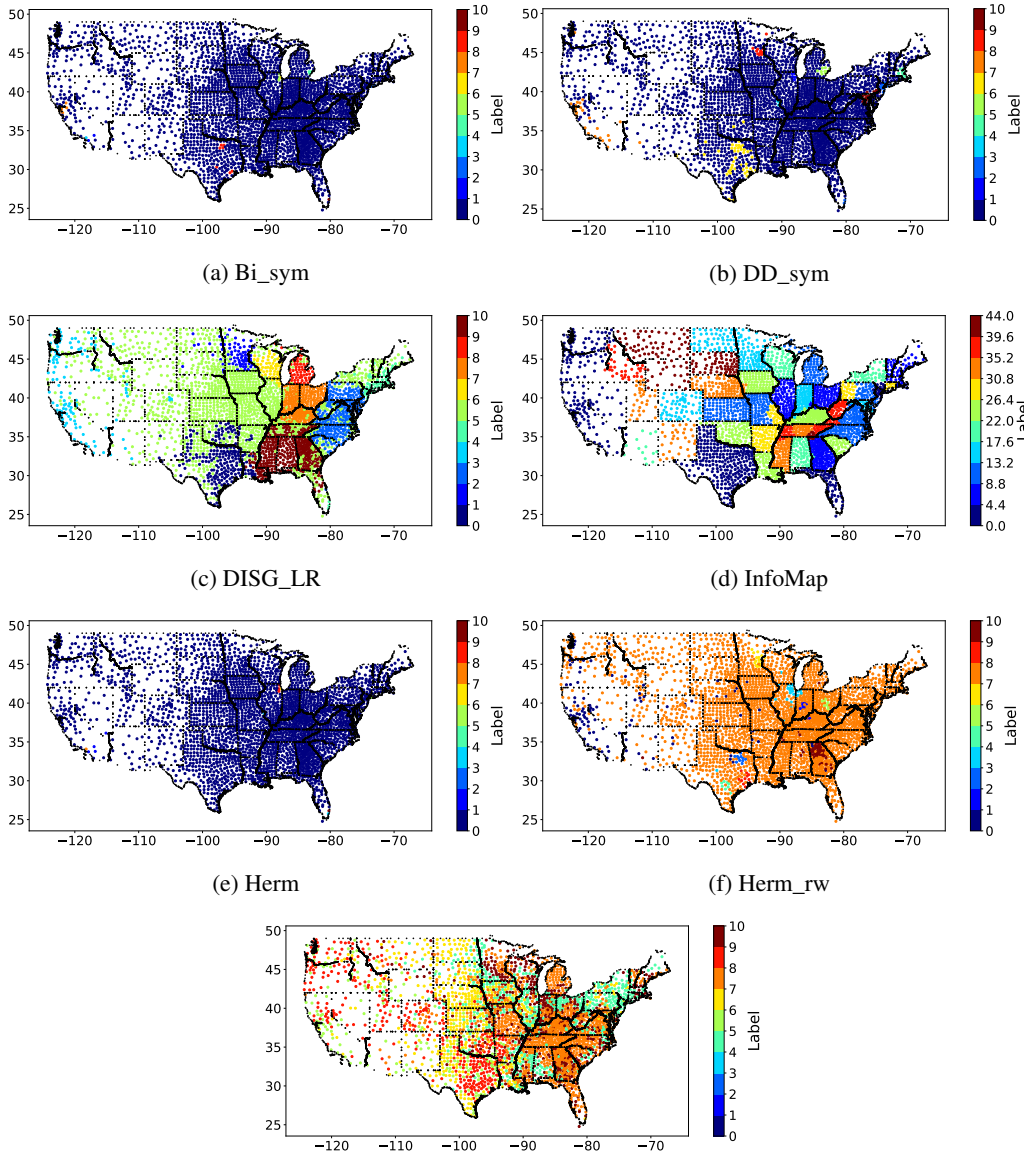
Table 10: Performance comparison on *Migration* (without preprocessing). The best is marked in **bold red** and the second best is marked in underline blue.

Metric/Method	InfoMap	Bi_sym	DD_sym	DISG_LR	Herm	Herm_rw	DIGRAC
$\mathcal{O}^{\text{sort}}_{\text{vol\_sum}}$	0.02±0.00	0.03±0.00	0.01±0.00	0.01±0.00	**0.07±0.00**	0.01±0.00	0.04±0.00
$\mathcal{O}^{\text{sort}}_{\text{vol\_min}}$	**0.24±0.00**	0.20±0.01	0.12±0.02	0.14±0.00	0.21±0.01	0.05±0.02	0.18±0.02
$\mathcal{O}^{\text{sort}}_{\text{vol\_max}}$	0.02±0.00	0.03±0.00	0.01±0.00	0.01±0.00	**0.06±0.00**	0.00±0.00	0.04±0.00
$\mathcal{O}^{\text{sort}}_{\text{plain}}$	0.61±0.00	0.46±0.00	0.29±0.02	0.26±0.00	**0.62±0.02**	0.40±0.00	0.32±0.11
$\mathcal{O}^{\text{std}}_{\text{vol\_sum}}$	0.00±0.00	0.01±0.00	0.00±0.00	0.00±0.00	0.02±0.00	0.00±0.00	**0.03±0.01**
$\mathcal{O}^{\text{std}}_{\text{vol\_min}}$	0.03±0.00	0.09±0.00	0.04±0.01	0.05±0.00	0.08±0.01	0.02±0.01	**0.11±0.03**
$\mathcal{O}^{\text{std}}_{\text{vol\_max}}$	0.00±0.00	0.01±0.00	0.00±0.00	0.00±0.00	0.01±0.00	0.00±0.00	**0.02±0.01**
$\mathcal{O}^{\text{std}}_{\text{plain}}$	0.19±0.00	0.23±0.00	0.14±0.01	0.12±0.00	**0.32±0.01**	0.25±0.01	0.21±0.03
$\mathcal{O}^{\text{naive}}_{\text{vol\_sum}}$	0.00±0.00	0.01±0.00	0.00±0.00	0.00±0.00	0.02±0.00	0.00±0.00	**0.03±0.01**
$\mathcal{O}^{\text{naive}}_{\text{vol\_min}}$	0.02±0.00	0.08±0.00	0.04±0.01	0.05±0.00	0.08±0.01	0.02±0.01	**0.11±0.04**
$\mathcal{O}^{\text{naive}}_{\text{vol\_max}}$	0.00±0.00	0.01±0.00	0.00±0.00	0.00±0.00	0.01±0.00	0.00±0.00	**0.02±0.01**
$\mathcal{O}^{\text{naive}}_{\text{plain}}$	0.16±0.00	0.22±0.00	0.13±0.01	0.11±0.00	**0.31±0.01**	0.22±0.00	0.21±0.03
size ratio	8.500	3043.80	722.620	25.780	**3059.20**	415.880	203.230
size std	**58.96**	912.100	861.280	409.900	917.230	844.750	342.38

None of the methods outperform DIGRAC on our chosen performance measures from Table 1, except on the *Lead-Lag* data set. With regards to the 12 imbalance measures from SI Table 4, leaving out OSLOM as before, Louvain and Leiden perform poorly on all of the real data sets, except on *Lead-Lag*. Indeed, for *Lead-Lag*, the number of clusters we use for DIGRAC is ten according to the GICS sector memberships. However, if we use the sector memberships as labels, the imbalance values are poor, which implies that ten may not be a desirable choice of the number of clusters. Further, DIGRAC usually clusters the nodes into smaller number of clusters, while Louvain and Leiden usually cluster the nodes into a larger number of clusters (usually around 30, and sometimes above 50 clusters).

Table 11: Performance comparison on *Lead-Lag*, including Louvain and Leiden. Results in each year is averaged over ten runs. Mean and standard deviation (after ±) are calculated over the 19 years. The best is marked in **bold red** and the second best is marked in underline blue. InfoMap results are omitted here as it usually predicts a single huge cluster and could not generate imbalance results. Louvain and Leiden yield essentially identical results and often attain the highest objectives, while DIGRAC almost always places either first or second across all methods considered.

Metric/Method	Louvain/Leiden	Bi_sym	DD_sym	DISG_LR	Herm	Herm_rw	DIGRAC
$\mathcal{O}^{\text{sort}}_{\text{vol\_sum}}$	0.08±0.02	0.07±0.01	0.07±0.01	0.07±0.01	0.07±0.02	0.07±0.02	**0.15±0.03**
$\mathcal{O}^{\text{sort}}_{\text{vol\_min}}$	0.15±0.04	**0.51±0.10**	0.48±0.09	0.47±0.10	**0.51±0.11**	0.50±0.10	0.47±0.09
$\mathcal{O}^{\text{sort}}_{\text{vol\_max}}$	0.08±0.02	0.07±0.01	0.06±0.01	0.06±0.01	0.07±0.01	0.07±0.01	**0.14±0.03**
$\mathcal{O}^{\text{sort}}_{\text{plain}}$	0.15±0.04	**0.66±0.09**	0.64±0.08	0.63±0.08	**0.66±0.09**	0.65±0.09	0.53±0.09
$\mathcal{O}^{\text{std}}_{\text{vol\_sum}}$	**0.23±0.06**	0.04±0.01	0.04±0.01	0.04±0.01	0.04±0.01	0.04±0.01	0.12±0.03
$\mathcal{O}^{\text{std}}_{\text{vol\_min}}$	**0.46±0.11**	0.27±0.04	0.27±0.04	0.25±0.04	0.27±0.03	0.27±0.03	0.38±0.07
$\mathcal{O}^{\text{std}}_{\text{vol\_max}}$	**0.23±0.05**	0.04±0.00	0.03±0.00	0.03±0.00	0.03±0.00	0.03±0.00	0.11±0.02
$\mathcal{O}^{\text{std}}_{\text{plain}}$	**0.46±0.11**	0.40±0.05	0.39±0.05	0.38±0.05	0.40±0.05	0.40±0.05	0.44±0.07
$\mathcal{O}^{\text{naive}}_{\text{vol\_sum}}$	**0.23±0.06**	0.03±0.01	0.03±0.01	0.03±0.01	0.03±0.01	0.03±0.01	0.08±0.04
$\mathcal{O}^{\text{naive}}_{\text{vol\_min}}$	**0.46±0.11**	0.20±0.05	0.19±0.05	0.18±0.05	0.19±0.04	0.19±0.04	0.26±0.10
$\mathcal{O}^{\text{naive}}_{\text{vol\_max}}$	**0.23±0.05**	0.03±0.01	0.02±0.01	0.02±0.01	0.02±0.00	0.02±0.00	0.08±0.03
$\mathcal{O}^{\text{naive}}_{\text{plain}}$	**0.46±0.11**	0.30±0.06	0.28±0.06	0.27±0.06	0.29±0.05	0.29±0.05	0.32±0.11
size ratio	124.530	3.67	**3.34**	3.900	4.110	3.880	8.070
size std	47.960	9.31	**9.14**	10.090	10.490	10.360	17.060



(a) Bi_sym

(b) DD_sym

(c) DISG_LR

(d) InfoMap

(e) Herm

(f) Herm_rw

(g) DIGRAC

Figure 15: US migration predicted clusters, along with the geographic locations of the counties as well as state boundaries (in black). The input digraph has extremely large entries; unlike in Figure 13, we do not employ here the normalization given by Eq. (9). Altogether, this demonstrates the robustness of DIGRAC to outliers in the data, which is not a characteristic of other state-of-the-art methods such as Herm and Herm_rw.

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.

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.

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.

Louis HY Chen, Larry Goldstein, and Qi-Man Shao. *Normal approximation by Stein's method*. Springer Science & Business Media, 2010.

Mihai Cucuringu, Vincent Blondel, and Paul Van Dooren. Extracting spatial information from networks with low-order eigenvectors. *Phys. Rev. E*, 87:032803, Mar 2013. doi: 10.1103/PhysRevE.87.032803. URL http://link.aps.org/doi/10.1103/PhysRevE.87.032803.

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, 2020.

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.

Matthias Fey, Jan E Lenssen, Frank Weichert, and Jure Leskovec. Gnnautoscale: Scalable and expressive graph neural networks via historical embeddings. *arXiv preprint arXiv:2106.05609*, 2021.

Gero Greiner and Riko Jacob. The I/O Complexity of Sparse Matrix Dense Matrix Multiplication", booktitle="LATIN 2010: Theoretical Informatics. pp. 143–156, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-12200-2.

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.

Adam P Harrison and Dileepan Joseph. High performance rearrangement and multiplication routines for sparse tensor arithmetic. *SIAM Journal on Scientific Computing*, 40(2):C258–C281, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

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.

Elan Sopher Markowitz, Keshav Balasubramanian, Mehrnoosh Mirtaheri, Sami Abu-El-Haija, Bryan Perozzi, Greg Ver Steeg, and Aram Galstyan. Graph traversal with tensor functionals: A meta-algorithm for scalable learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=6DOZ8XNNfGN.

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

Ryan L Phillips and Rita Ormsby. Industry classification schemes: An analysis and review. *Journal of Business & Finance Librarianship*, 21(1):1–25, 2016.

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.

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.

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