POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Spectral embeddings for directed graph clustering

Tesi di Laurea Magistrale in
High Performance Computing Engineering

Author: **Jacopo Palumbo**

Student ID: 251528
Advisor: Prof. Gianluca Palermo
Academic Year: 2024-25

Abstract

Many real networks are inherently directed: messages, goods and money flow in specific directions, and this orientation carries crucial structural information. This thesis revisits five adjacency–based spectral clustering algorithms for directed graphs: Adjacency Spectral Embedding (ASE), Hermitian (HERM), Skew–symmetric (SKEW), Block–Cyclic (BCS) and Block–Acyclic (BAS). We provide a study on a modification of these algorithms: before k-means, the spectral coordinates are reduced to two dimensions using t-SNE, leading to an improvement of the final clustering. The spectral core of each method is left unchanged.

The analysis covers the considered algorithms, implementation details, and complexity analyses. In the dense regime all spectral variants are dominated by the $\mathcal{O}(n^3)$ cost of the eigen/SVD step with $\mathcal{O}(n^2)$ memory, whereas the added Barnes–Hut t-SNE contributes only $\mathcal{O}(n \log n)$ and does not alter the asymptotic bottleneck. Performance is evaluated on synthetic and real graphs using both label–based metrics (NMI, F-Score, ARI) and either directionality-aware, or flow-based criteria (Cut–Imbalance and Trade–Flow). Directed Stochastic Block Models (DSBMs) show that the refinement consistently reduces the errors in the classification across a wide range of community counts. LFR benchmarks indicate that the gain persists under substantial inter–community mixing. On real-world datasets emerging from e-mail communications and banking transactions, the refined pipelines yield more interpretable clusters and improved flow structure between them.

Overall, a geometric adjustment applied after the eigenspectrum analysis step reliably tightens same–community neighborhoods without changing the dominant computational cost. The result is a unified, implementation–ready toolkit for clustering directed graphs that improves empirical separability while preserving the theoretical and algorithmic foundations of the underlying spectral methods.

Keywords: spectral clustering, directed graph clustering

Abstract in lingua italiana

Molte reti reali sono intrinsecamente dirette: messaggi, beni e denaro fluiscono in direzioni specifiche e tale orientamento veicola informazioni strutturali cruciali. Questa tesi riconsidera cinque algoritmi di clustering spettrale basati sulla matrice di adiacenza per grafi diretti: Adjacency Spectral Embedding (ASE), Hermitian (HERM), Skew–symmetric (SKEW), Block–Cyclic (BCS) e Block–Acyclic (BAS). Studiamo una modifica di tali algoritmi: prima di applicare l'algoritmo k-means, le coordinate spettrali vengono ridotte a due dimensioni tramite t-SNE, con un miglioramento del clustering finale. Il nucleo spettrale di ciascun metodo rimane invariato.

L'analisi copre gli algoritmi considerati, i dettagli implementativi e le analisi di complessità. Utilizzando metodi diretti, tutte le varianti spettrali sono dominate dal costo $\mathcal{O}(n^3)$ dovuto al passaggio di calcolo degli autovettori o di calcolo del SVD con memoria $\mathcal{O}(n^2)$, mentre l'implementazione di t-SNE tramite Barnes–Hut aggiunge soltanto $\mathcal{O}(n \log n)$ e non altera il collo di bottiglia asintotico. Le prestazioni vengono valutate su grafi sintetici e reali utilizzando sia metriche basate sulle etichette (NMI, F-Score, ARI) sia criteri sensibili alla direzionalità o al flusso (Cut-Imbalance e Trade-Flow). I Directed Stochastic Block Models (DSBM) mostrano che il raffinamento riduce sistematicamente gli errori di classificazione su un ampio intervallo di numerosità delle comunità. I benchmark LFR indicano che il miglioramento persiste anche sotto forte mescolamento inter-comunità. Su insiemi di dati reali provenienti da comunicazioni e-mail e transazioni bancarie, le pipeline affinate producono cluster più interpretabili e una struttura di flusso tra essi più coerente.

Nel complesso, un aggiustamento geometrico applicato dopo il passo di analisi dello spettro restringe in modo affidabile i nodi intra-comunità senza modificare il costo computazionale dominante. Il risultato è un toolkit unificato e pronto all'uso per il clustering di grafi diretti, che migliora la separabilità empirica preservando i fondamenti teorici e algoritmici dei metodi spettrali sottostanti.

Parole chiave: clustering spettrale, clustering di grafi diretti

Acknowledgements

I would like to express my sincere gratitude to Prof. Gianluca Palermo for accepting to be my advisor for this thesis and for his guidance during the review. I am also deeply grateful to Prof. Olaf Schenk and Dr. Dimosthenis Pasadakis (USI - Università della Svizzera Italiana) for proposing this interesting topic to me during my third semester.

I kindly ask the reader to forgive me for the following words, which will be written in Italian, in order to acknowledge the unwavering support I have received from my family and friends.

Grazie ai miei genitori, che ci sono sempre stati e mi hanno sempre supportato. Grazie a mio padre Alessandro, che mi ha dimostrato come il duro lavoro ripaghi sempre. Grazie a mia madre Alice, che mi ha guidato verso il raggiungimento dei miei sogni e mi ha insegnato a coltivare la mente. Non sempre ve l'ho dimostrato, ma vi sono davvero grato per tutto.

Grazie a mia nonna Franca, che è sempre così fiera di me e ha sempre buone parole da condividere con gli altri sulla mia persona e sul mio percorso. Grazie anche ai nonni che c'erano e che non ci sono più: sono sicuro che avrebbero condiviso con gioia questo mio traguardo.

Grazie ai miei parenti, che sempre mi hanno sostenuto e apprezzato. In particolare, voglio rivolgere un ringraziamento a mio zio Francis: sei stato una grande fonte di ispirazione durante la mia infanzia e adolescenza.

Grazie ad Alice: la tua presenza è stata per me fondamentale in questi due anni, sia nei momenti difficili sia in quelli felici. Hai reso questo percorso molto più bello, dandomi un motivo in più per tornare a casa ogni volta che mi è stato possibile.

Grazie agli amici e colleghi che non ho citato per nome, ma che mi hanno accompagnato in questi anni: avete reso questo cammino molto più gratificante di quanto sarebbe stato altrimenti.

Infine, grazie a me stesso per non essermi mai arreso e per aver iniziato a realizzare i sogni che avevo da bambino.

Contents

List of Figures

List of Tables

List of Algorithms

1 | Introduction

Directed graphs are a natural language for many real systems in which interactions have a direction: information propagates in e-mail networks, goods and payments flow through supply chains and banking systems, biomass travels up a food web, migrants relocate from origin to destination. In all these settings, orientation matters: treating a directed edge as undirected erases the structure we wish to recover. This thesis studies spectral embeddings tailored to directed graphs and investigates a subsequent non-linear dimensionality-reduction step that preserves local neighborhoods by minimizing a Kullback–Leibler divergence between high and low-dimensional similarity distributions, yielding embeddings in which clusters are easier to separate and interpret.

We start by presenting a family of adjacency-based spectral methods that keep the directionality in the model rather than symmetrizing the data. We revisit the Adjacency Spectral Embedding (ASE) derived from the Random Dot Product Graph framework [2], the Hermitian representation of directed graphs (HERM) [3], the skew-symmetric approach (SKEW) [4], and the block-structured methods designed for cyclic or acyclic flow between groups (BCS and BAS) [5]. In all methods, we keep the spectral construction unchanged and pair it with a non-linear dimensionality reduction of the resulting embeddings to two dimensions using t-SNE [6, 7], after which clustering is performed with k-means. This modification is minimal in complexity and keeps the computational bottleneck unchanged (the spectrum extraction), but it amplifies the separation between groups in practice.

We demonstrate the effectiveness of our approach with numerical experiments in synthetic and real-world data. First, we adopt stochastic generative models to provide test benchmarks and theoretical insights. We use the Directed Stochastic Block Model (DSBM) and its RDPG parametrization to justify adjacency-based embeddings and to generate graphs with known ground truth [2]. We also employ the LFR benchmark [8], which matches power-law degree and community-size distributions typical of real networks and allows us to test the presented methods with a more established framework. Second, we quantify performance with both label based metrics (NMI, F-Score, ARI) and flow-based criteria

(Cut-Imbalance variants and Trade-Flow), so that improvements are visible whether or not a reference partition is available.

Beyond definitions and algorithms, we devote a chapter to t-SNE itself and to the analysis of the embeddings obtained by the presented methods. We recall the t-SNE probabilistic view (high and low-dimensional similarities), the KL objective, and the optimization heuristics, such as early exaggeration, momentum, adaptive learning rate, which are important for stable maps [6].

At the end, we provide some experiments and their results based both on synthetic and real-world networks. We begin with synthetic DSBMs, varying the number of communities to show that the t-SNE stage consistently improves the recovery of memberships over the baseline spectral methods. We then move to LFR graphs and increase the mixing parameter to test robustness under perturbations. Finally, we consider two real-world scenarios: an e-mail communication network across European research institutions [9–11] and transaction graphs produced by the AMLSim simulator for anti–money laundering, where ground truth is absent and flow-based metrics reveal cluster structure [12–14].

Organization Chapter 2 provides a background on directed graphs, the graph models considered and the metrics used to evaluate the results. Chapter 3 surveys the methods for directed spectral clustering. Chapter 4 analyses t-SNE and visualizes the resulting embeddings. Chapter 5 introduces the adapted pipelines and their complexity. Chapter 6 reports results on synthetic and real-world graphs. We conclude with a short summary and possible directions for future work.

1.1. Notation

Here we provide the notation that we use in the next chapters. A matrix is denoted by a capital letter, like A. It's transpose is $A^\top$. The Hermitian transpose or conjugate transpose is denoted by A^H. The i-th row of a matrix is denoted as A_i, the j-th column of a matrix is denoted by $A_j^\top$. The value of a matrix A in the i-th row and j-th column is denoted by A_{ij}. A vector is denoted by a bold lowercase letter (e.g. $\mathbf{v}$).

A set is defined using the curly brackets $\{\}$. The real set is denoted as $\mathbb{R}$, while the complex set as $\mathbb{C}$. The dimension of a vector is defined by the exponent of the set it belongs to (e.g. a column vector of real values is denoted by $\mathbf{v} \in \mathbb{R}^{n\times 1}$).

1.2. Reproducibility

All code and scripts to reproduce the experiments and figures in this thesis are publicly available at the following repository: `https://github.com/jacopopalumbo01/spectral-embeddings-for-directed-graph-clustering`.

2 | Background on directed graphs

This chapter consolidates the notation and concepts used throughout the thesis. We first formalize directed graphs, basic properties (e.g., in/out degrees) and the objective of flow-based community detection. We then introduce two stochastic models: the Random Dot Product Graph and the Directed Stochastic Block Model, used both as generators of synthetic data and as reference frameworks with established theory. The chapter concludes with the procedures used to generate synthetic graphs and the evaluation criteria adopted for assessing the quality of the obtained partitions.

2.1. Directed graphs

Let us formally define a directed graph as an ordered pair $G = (V, E)$, where:

- V is the set of vertices or nodes.
- E is the set of edges (u, v).

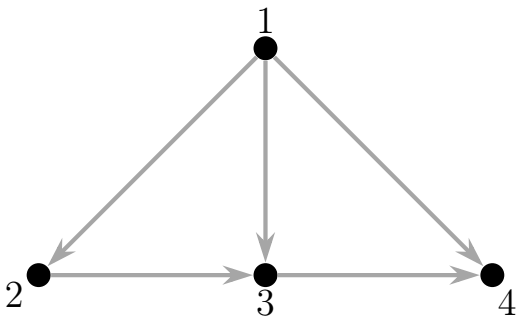
Given the set of edges, we define the adjacency matrix $A \in \{0, 1\}^{n \times n}$ as follows:

$$A_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

In some cases we may be interested in taking into account also weights W, where w_{ij} is associated with the edge (i, j). In these cases, the adjacency matrix A will become:

$$A_{ij} = \begin{cases} w_{ij} & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

When the graph is constructed from feature vectors (rather than observed edges), a common choice is to assign weights with a Gaussian kernel of pairwise distances. An alternative is to learn the weights or, more generally, the underlying graph/Laplacian structure, by maximizing a Gaussian likelihood subject to sparsity or M-matrix constraints (e.g.



$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2.1: Plot of a directed graph and its adjacency matrix. The graph has 4 nodes and 5 edges.

the sparse quadratic approximation framework for graph learning [15]).

We only consider graphs without weights, unless explicitly stated otherwise. An example of a directed graph is pictured in Fig. 2.1.

The degree of a node is given by the number of edges starting or ending from itself. In particular, we can define the out-degree as the number of edges starting from the node (i.e. $d_u^{out} = |D|, D = \{(u, v) \in E | v \neq u\}$) and the in-degree as the number of edges ending in the node (i.e. $d_u^{in} = |D|, D = \{(v, u) \in E | v \neq u\}$).

2.1.1. Cut of a graph

A cut of a graph is a bipartition of its vertex set into two disjoint subsets. From each cut we can derive the cut-set, formed by the edges starting from one partition and ending in the other. Formally, a cut $C = (B, C)$ is a partition of the set of nodes V into two subsets B and C. The cut-set is then defined as $\{(u, v) \in E | u \in B \wedge v \in C\}$. In the following, we mainly consider cuts where one subset is a single node $u \in V$ and the other is composed by all the remaining nodes $V - \{u\}$. The cut-set is particular useful for some definitions and metrics. Throughout the thesis we refer to the outgoing cut and incoming cut of a node u. The former refers to all the edges $\{(u, v) \in E | v \in V - \{u\}\}$, while the latter refers to all the edges $\{(v, u) \in E | v \in V - \{u\}\}$.

2.1.2. Flow-based clustering

We are interested in the flow-based clustering of directed graphs. We follow the definition given by [4]: the edges belonging to the cut-set between different clusters are mainly oriented in the same direction. This results in large imbalanced cuts in the directed graphs. This particular type of pattern can be found in may real world scenarios, such as migration

networks [3], email exchanges [1], trophic networks [5] and bank debt interchange networks [16]. Recently, researchers have used spectral techniques based on the adjacency matrix in order to find flow-based clusters in directed graphs.

2.1.3. Random Dot Product Graph (RDPG)

Hoff et al. [17] proposed a latent space model associated with random graphs. Young et al. proposed a particular latent space model, named Random Dot Product Graph (RDPG) [18].

In the RDPG framework, each node i in a graph is associated with a latent vector $X_i \in \mathbb{R}^d$, where d is the dimension of the latent space. The probability of having an edge between nodes i and j is given by the dot product of their latent vectors:

$$\mathbb{P}(A_{ij} = 1) = (X_i)^\top X_j, \tag{2.3}$$

where it is required that $(X_i)^\top X_j \in [0, 1] \ \ \forall i, j$. Given these probabilities, the adjacency matrix $A \in \{0, 1\}^{n \times n}$ of the graph is generated by a Bernoulli distribution:

$$A_{ij} \sim \text{Bernoulli}((X_i)^\top X_j), \quad \text{for } i < j. \tag{2.4}$$

An important feature of the RDPG is its invariance under orthogonal transformations. For any orthogonal matrix $Q \in \mathbb{R}^{d \times d}$, the latent positions X_i and $X_i Q$ yield the same edge probability matrix. This motivates the use of spectral embedding techniques, such as the adjacency spectral embedding (ASE) [2], for estimating latent positions up to an orthogonal transformation [19].

RDPGs are particular useful in the study of the models we are going to present, thanks to their mathematical properties.

2.1.4. Directed Stochastic Blockmodel (DSBM)

Most of the graphs we are employing in the following methods are based on stochastic blockmodels [20]. In particular, we consider the Directed Stochastic Blockmodels (DSBMs) presented by Wang et al. [21].

In the DSBM, each node i is assigned to a block $z_i \in \{1, \ldots, K\}$, where K is the total number of blocks. The key idea is that the probability of a directed edge from node i to node j depends only on the block memberships of i and j. Specifically, the model

is parametrized by $P \in [0,1]^{K\times K}$ and $\boldsymbol{\rho} \in [0,1]^K$, where $\sum_{i=1}^{K} \boldsymbol{\rho} = 1$. P is a matrix of connection probabilities, where P_{ij} denotes the probability of a directed edge from a node in block i to a node in block j. $\boldsymbol{\rho}$ is a vector representing the block membership probability, i.e. a node i will belong to block z with a probability ρ_z.

The generative model proceeds as follows:

- Assign each node i to a block $z_i \sim \text{Categorical}(\rho)$.
- For each pair of nodes (i, j), draw an edge $A_{ij} \sim \text{Bernoulli}(P_{z_i z_j})$ independently.

Unlike the undirected SBM, the matrix P is not constrained to be symmetric, allowing for asymmetric relationships between blocks. This feature enables the model to represent patterns such as hierarchical structure, which commonly occurs in directed systems.

Reconduct the DSBM to a RDPG

Using another parametrization [2], the DSBM can be reconducted to a RDPG. This will be particular useful to show that the following spectral methods are well suited for a DSBM, exploiting the RDPG theory. Let us consider a DSBM with $rank(P) = d$. Then, we can find $\nu, \mu \in \mathbb{R}^{K\times d}$ such that $P = \nu\mu^T$ and, by definition $P_{ij} = \nu_i \cdot \mu_j$. Let $\tau : \{1, \dots, n\} \to \{1, \dots, K\}$ be the block membership function. Let $X, Y \in \mathbb{R}^{n\times d}$ have row u given by $X_u = \nu_{\tau(u)}, Y_u = \nu_{\tau(u)} \ \ \forall u$. Then, we have:

$$\mathbb{P}[A_{ij} = 1] = P_{\tau(i),\tau(j)} = \nu_{\tau(i)} \cdot \mu_{\tau(j)} = X_i \cdot Y_j. \tag{2.5}$$

In this way, a DSBM can be parametrized by ρ, ν, μ, provided that $(\nu\mu^T)_{ij} \in [0,1]\ \forall i, j \in \{1, \dots, K\}$.

2.2. k-means clustering algorithm

The k-means algorithm is a widely used distance-based clustering method that finds a partition of n data points $\{x_1, \dots, x_n\} \subset \mathbb{R}^d$ into k disjoint clusters. It is mandatory for us to present k-means because, in all spectral methods considered, it is the final post-processing step that converts the spectral embedding into a discrete partition: specifically, k-means is applied to the rows of the embedding matrix Z, treating each row as a feature vector that captures the graph's latent structure, and the resulting assignments yield the community labels.

The aim of k-means is to minimize the within–cluster sum of squared distances [22].

Formally, letting $\tau : \{1, \dots, n\} \to \{1, \dots, k\}$ denote the cluster assignment function and $\mu_c \in \mathbb{R}^d$ the centroid of cluster c, the objective is

$$\min_{\tau, \{\mu_c\}_{c=1}^k} \sum_{i=1}^{n} \left\| x_i - \mu_{\tau(i)} \right\|_2^2. \tag{2.6}$$

A standard iterative method, called Lloyd's algorithm [23], alternates between assignment (updating τ by mapping each point to its nearest centroid) and update (recomputing each μ_c as the mean of the points assigned to cluster c) until convergence. Although solving Eq. 2.6 exactly is NP-hard in the worst case, Lloyd's heuristic typically converges in a few iterations and has per-iteration cost $\mathcal{O}(nkd)$.

2.3. Synthetic graphs generation

During our experiments, we will use both real and synthetic datasets. Here we report the procedure followed for the generation of the latters. In particular, we employ two methods of generation: the Directed Stochastic Blockmodel (DSBM), which is the theoretical model on which all presented methods build their foundations, and the well-established LFR Benchmark Graphs [8], which is better in modeling real graphs.

2.3.1. Directed Stochastic Blockmodel (DSBM)

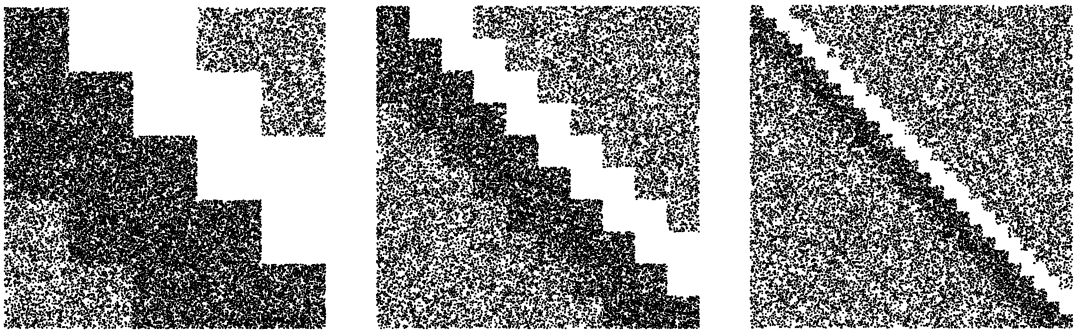
This type of graphs is based on the work of Wang et al. [21]. Here, we choose the parameters following the proposal of Cucuringu et al. and Hayashi et al. [3, 4]. In particular, we focus on the Directed Acyclic DSBM proposed by [4].

The parameters used for the generation are the following: k the number of clusters, p the intra-cluster edge probability, q the inter-cluster edge probability, $\mathbf{c} \in \mathbb{N}^k$, which specifies the number of nodes in each cluster and $F \in [0, 1]^{k \times k}$, which encodes cluster-level orientation probabilities.

Each node u is assigned to a cluster via the true membership function $\tau(u) : \{1, \dots, n\} \to \{1, \dots, k\}$. For any pair of nodes u, v, we have:

$$\mathbb{P}[A_{uv} = 1] = \begin{cases} p \cdot F_{\tau(u),\tau(v)} & \text{if } \tau(u) = \tau(v) \\ q \cdot F_{\tau(u),\tau(v)} & \text{if } \tau(u) \neq \tau(v) \end{cases}. \tag{2.7}$$

Note that F itself resembles a graph, in particular a meta-graph representing the directionality inside the generated network. In the Directed Acyclic DSBM proposed by



(a) DSBM with 5 clusters. (b) DSBM with 10 clusters. (c) DSBM with 25 clusters.

Figure 2.2: Synthetic graphs generated using the DSBM procedure with $5, 10, 25$ clusters and 2500 nodes.

Hayashi et al. [4], the meta-graph is a Directed Acyclic Graph (DAG) and it is built as follows:

$$F_{uv} = \begin{cases} \mu & \text{if } v = u+1 \vee v = u+2 \\ 1-\mu & \text{if } v = u-1 \vee v = u-2\,, \\ \frac{1}{2} & \text{otherwise} \end{cases} \tag{2.8}$$

where $\mu \in [0, 0.3]$ is a noise parameter. As $\mu \to 0.5$, the edge directionalities becomes increasingly random, making cluster recovery more difficult.

Regarding the other parameters, we set $p = q = 0.008$ and we evenly distribute the nodes to each cluster (e.g. if $n = 800$ and $k = 8$, we set $\mathbf{c}_i = 100\ \forall i \in [1, 8]$).

We report the sparsity pattern of three graphs generated using the presented procedure in Fig. 2.2. This type of graph comes quite useful for the testing of methods based on the RDPG theory, however they are quite far from the real-world scenarios. Indeed, in the following we are going to always prefer synthetic directed graphs generated using the LFR benchmark graphs model[8].

2.3.2. LFR benchmark graphs

Lancichinetti et al. [8] provided a framework for the generation of synthetic networks with realistic community structures, featuring power-law degree and community size distributions. They allow control over inter-community mixing and support overlapping, making them ideal for testing community detection algorithms.

LFR improves on the earlier Girvan–Newman benchmark [24] by reproducing two empirical regularities commonly observed in real networks:

- a power-law degree distribution with exponent γ.
- a power-law community-size distribution with exponent β.

Generation procedure. Given the target number of vertices n and the desired average in-degree $\langle k \rangle$, the generator:

1. Samples an integer degree sequence $\{k_i\}$ from a power law $p(k) \propto k^{-\gamma}$. The extremes of the distribution k_{min}, k_{max} are chosen in such a way that the average in-degree is $\langle k \rangle$.
2. A mixing parameter μ is introduced. μ is used to determine the edges the node is going to have: a fraction $1 - \mu$ of its edges are connected with nodes of the same community, while a fraction μ with the nodes of other communities.
3. Samples community sizes $\{c_j\}$ from $p(c) \propto c^{-\tau_2}$ until $\sum_j c_j = n$. Each community has a minimum and a maximum size s_{min}, s_{max}, which are chosen in such a way to respect the relations: $s_{min} > k_{min}$ and $s_{max} > k_{max}$. This is done to guarantee that each node can belong to at least one community.
4. Assigns each vertex to a community, respecting the sampled sizes. This process is done through multiple iterations. A node remains homeless until the sampled sizes are not respected. If a node respects the sampled sizes but the community is full, a random node of the community is expelled from the community and becomes homeless, while the initial node is inserted in the community. This continues until convergence, i.e. there is not any node which is homeless.
5. Connects nodes at random while satisfying the internal/external constraints; self-loops and multi-edges are discarded or rewired.

Key parameters. Table 2.1 summarizes the parameters we used during the generation of synthetic graphs using the LFR benchmark. Using these parameters, we always obtained networks with 29 communities. In Fig. 2.3 we report both a plot of the graph generated using these parameters and its adjacency matrix sparsity pattern.

2.4. Graph metrics

We now detail the criteria used to evaluate how well the nodes are assigned to clusters. We distinguish between two different type of metrics:

Label-based metrics compare the inferred labels against a known ground truth. They

Table 2.1: Main parameters of the LFR benchmark

Symbol	Description	Value
n	number of nodes	10^3
$\langle k \rangle$	average in-degree	15
$k_{\max}$	maximum in-degree	50
γ	degree exponent	2 (default)
β	community-size exponent	1 (default)
$c_{\min}$, $c_{\max}$	min / max community size	$20, 50$
μ	mixing parameter	$[0.1, 0.4]$

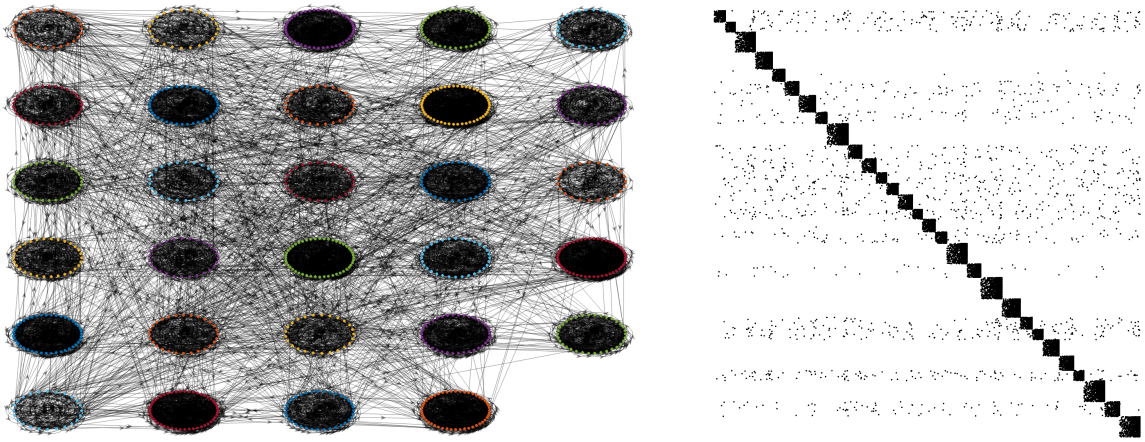


Figure 2.3: Plot of a directed graph generated using the LFR benchmark procedure and the sparsity pattern of its matrix. The mixing parameter used is $\mu = 0.1$
.

are meaningful only when that truth is available.

Cluster-based metrics rely solely on the graph structure and can therefore be computed even when no reference partition is known. They depict how well the methods are recovering a flow-based structure in the clusters.

2.4.1. Label-based metrics

Normalized Mutual Information (NMI)

The Normalized Mutual Information (NMI) [25] is the normalized version of the Mutual Information (MI), firstly introduced by Shannon in his work of information theory [26].

Let X be the random variable of ground-truth labels and Y the variable of predicted

labels. The (symmetric) normalized mutual information is

$$\mathrm{NMI}(X,Y) = \frac{2\,I(X;Y)}{H(X)+H(Y)} \quad \in [0,1], \tag{2.9}$$

where $I(X;Y) = H(X) - H(X|Y)$ is the mutual information and $H(X) = -\sum_{x\in X} p(x)\log p(x)$ is the Shannon entropy [26].

NMI $= 1$ indicates perfect agreement, while NMI $= 0$ means statistical independence between X and Y.

F-Score

The F-score or F-measure has been introduced during the Fourth Message Understanding Conference (MUC-4) [27]. It is a well-established metric used to evaluate prediction performances.

F-score is defined as follows:

$$\text{F-Score} = 2\frac{P\cdot R}{P+R} = \frac{2TP}{2TP+FP+FN}, \tag{2.10}$$

where $P = TP/(TP+FP)$ is the precision and $R = TP/(TP+FN)$ is the recall. TP, FP and FN are counted over the set of inferred labels and their ground-truth counterparts:

- A node counts as a true positive (TP) if the cluster label inferred by the algorithm matches its ground-truth label.
- A node counts as a false positive (FP) with respect to a cluster C if the algorithm assigns the node to C while its ground-truth label belongs to a different cluster.
- A node counts as a false negative (FN) with respect to a cluster C if the node truly belongs to C but the algorithm assigns it to some other cluster.

F-score combines precision and recall in a single score in $[0,1]$. The highest value for F-score ($= 1$) indicates perfect precision and recall, which are associated with perfect cluster recovery. On the other hand, the lowest value for F-score ($= 0$) indicates that either precision or recall is equal to zero, which is associated with non-existent cluster recovery.

Adjusted Rand Index (ARI)

Adjusted Rand Index has been introduced by Gates and Ahn [28] and it is an improvement on the well-known Rand Index [29].

Let $X = \{x_1, x_2, \ldots, x_k\}$ be the set of the inferred clusters and $Y = \{y_1, y_2, \ldots, y_k\}$ be the set of ground-truth clusters. Given n the total number of nodes, we define: $n_{x_i} = |x_i|$ and $n_{x_i,y_j} = |x_i \cap y_i|$. Then, ARI is defined as:

$$\mathrm{ARI}(X,Y) = \frac{\sum_{ij}\binom{n_{x_i,y_i}}{2} - \frac{\sum_i \binom{n_{x_i}}{2}\sum_j \binom{n_{y_i}}{2}}{\binom{n}{2}}}{\frac{1}{2}\sum_i \binom{n_{x_i}}{2} + \sum_j \binom{n_{y_j}}{2} - \frac{\sum_i \binom{n_{x_i}}{2}\sum_j \binom{n_{y_i}}{2}}{\binom{n}{2}}}. \tag{2.11}$$

ARI ranges from -1 to 1 . We expect to get results near to -1 when the labeling is completely random and equals to 1 when the inferred clusters and ground truth clusters are identical.

2.4.2. Cluster-based metrics

Let $G = (V, E)$ be a directed graph with adjacency matrix A and a partition $\mathcal{C} = \{C_1, \ldots, C_k\}$.

Cut-Imbalance (CI)

We follow the definition of Cut-Imbalance provided by [3]. For two clusters C_p and C_q denote

$$w_{pq} = \sum_{i \in C_p}\sum_{j \in C_q} A_{ij}, \qquad w_{qp} = \sum_{i \in C_q}\sum_{j \in C_p} A_{ij}. \tag{2.12}$$

The Cut-Imbalance between the two clusters C_p, C_q is

$$\mathrm{CI}(C_p, C_q) = \frac{1}{2}\left|\frac{w_{pq} - w_{qp}}{w_{pq} + w_{qp}}\right| = \left|\frac{w_{pq}}{w_{pq} + w_{qp}} - \frac{1}{2}\right|. \tag{2.13}$$

The Cut-Imbalance can range from 0 to 1/2 ($\mathrm{CI}(C_p, C_q) \in [0, 1/2]\ \forall C_p, C_q \in \mathcal{C}$). The direction of edges between C_p and C_q are completely balanced if $CI(C_p, C_q) = 0$, while they are completely unbalanced when $CI(C_p, C_q) = 1/2$. Since we are interested in flow-based clustering, which maintains an unique directionality all over the graph, we consider good a partitioning the one with high Cut-Imbalance values.

Size-normalized Cut-Imbalance To account for different community sizes we normalize using the cluster's size:

$$\mathrm{CI}^{\mathrm{size}}(C_p, C_q) = \mathrm{CI}(C_p, C_q) \cdot \min\{|C_p|, |C_q|\}. \tag{2.14}$$

Volume-normalized Cut-Imbalance Alternatively, one can normalize by the volume $\mathrm{vol}(C_p) = \sum_{i \in C_p} d_i^{\mathrm{out}} + d_i^{\mathrm{in}}$:

$$\mathrm{CI}^{\mathrm{vol}}(C_p, C_q) = \mathrm{CI}(C_p, C_q) \cdot \min\{\mathrm{vol}(C_p), \mathrm{vol}(C_q)\}. \tag{2.15}$$

The scaling used in the normalized version of the Cut-Imbalance is done with the aim of penalizing small clusters, following the idea used for the normalized cut value by Shi et al. [30]. During the evaluation of our experiments, we are going to use a extension of the Cut-Imbalance, called Top-CI, first proposed by Hayashi et al. [4].

Top Cut-Imbalance We define the Top-CIs as:

$$\mathrm{TopCI}^{\mathrm{size}}(C_1, \ldots, C_k) = \sum_{t=1}^{c} \mathrm{CI}^{\mathrm{size}}(C_{j_t}, C_{i_t}), \quad \mathrm{TopCI}^{\mathrm{vol}}(C_1, \ldots, C_k) = \sum_{t=1}^{c} \mathrm{CI}^{\mathrm{vol}}(C_{j_t}, C_{i_t}), \tag{2.16}$$

where $\mathrm{CI}^{\mathrm{size}}(C_{j_t}, C_{i_t}), \mathrm{CI}^{\mathrm{vol}}(C_{j_t}, C_{i_t})$ are the t-th largest $\mathrm{CI}^{\mathrm{size}}, \mathrm{CI}^{\mathrm{vol}}$ pair of clusters. The value of c is always set heuristically to $c = 2(k-1)$.

Trade-Flow (TF)

Another metric we are going to use it the Trade-Flow (TF), proposed by Laenen [31]. Trade-Flow is similar to CI: with high TF values the directionality is mostly oriented from one cluster to another (i.e. it is unbalanced). It is defined as:

$$\mathrm{TF}(C_p, C_q) = |w_{pq} - w_{qp}|. \tag{2.17}$$

Hayashi et al. [4] proposed an extension of the TF as for the CI, called Top-TF.

Top Trade-Flow We define the Top-TF as:

$$\mathrm{TopTF}(C_1, \ldots, C_k) = \sum_{t=1}^{c} \mathrm{TF}(C_{j_t}, C_{i_t}), \quad j_t \geq i_t, \tag{2.18}$$

where, as for the CI case, $\mathrm{TF}(C_{j_t}, C_{i_t})$ is the t-th largest TF pair of clusters. As before, $c = 2(k-1)$.

3 | Spectral methods for the clustering of directed graphs

This chapter presents the spectral approaches used to partition directed graphs while preserving edge orientation. Each method follows a common template: 1. construct an operator that encodes directionality (e.g., the adjacency matrix itself, a Hermitian or skew–symmetric transform, or a transition matrix); 2. compute a small set of eigenpairs or singular triplets; 3. form a spectral embedding by stacking the selected eigenvectors (and, when appropriate, their real/imaginary parts or singular-value scalings); 4. obtain discrete communities by applying k–means to the rows of the embedding. The models introduced in the background chapter (RDPG/DSBM) motivate these constructions and clarify when block structure is recoverable from spectral coordinates.

The sections that follow report this template for five algorithms commonly used with directed networks: Adjacency Spectral Embedding (ASE), a Hermitian representation, a skew–symmetric formulation, and block–structured variants designed for cyclic (BCS) and acyclic (BAS) graphs.

Beyond these pipelines, recent work investigates spectral partitioning schemes tailored to directed graphs that explicitly account for orientation in the choice of spectral coordinates and objective functions, like the symmetry-breaking analysis in [32].

3.1. Adjacency Spectral Embedding (ASE)

Proposed by Sussman et al. [2] the adjacency spectral embedding uses an embedding procedure motivated by the structure of random graphs. In particular, the embedding proposed by the authors is based on the Random Dot Product Graph (RDPG) [18] latent space model. ASE exploits the Singular Value Decomposition (SVD) of the adjacency matrix. The obtained embedding is similar to the one obtained by the well known spectral clustering [33], however it is directly computed from the adjacency matrix, without the need of building the graph Laplacian.

Given an adjacency matrix $A \in \{0,1\}^{n\times n}$ of a graph generated from a RDPG, ASE aims to recover latent positions $X_1, \dots, X_n \in \mathbb{R}^d$ such that the dot product $X_i^\top X_j$ approximates the true edge probability between vertices i and j. The embedding of a RDPG with adjacency matrix A is:

$$(\tilde{X}, \tilde{Y}) = \underset{(X^\dagger, Y^\dagger)\in\mathbb{R}^{n\times d}\times\mathbb{R}^{n\times d}}{\operatorname{argmin}} \|A - X^\dagger Y^{\dagger\top}\|_F, \tag{3.1}$$

where d is the target dimensionality of the embedding and $\|\cdot\|_F$ is the Frobenius norm. The solution to Eq. 3.1 is given by [34]. Given the singular value decomposition of A:

$$A = \hat{U}'\hat{\Sigma}'\hat{V}'^T, \tag{3.2}$$

from which the first d truncated singular values are extracted (i.e. the first d rows of each matrix): $\hat{\Sigma} \in \mathbb{R}^{d\times d}$ and $\hat{U} \in \mathbb{R}^{n\times d}$. The solution to Eq. 3.1 is given by $\tilde{X} = \hat{U}\hat{\Sigma}^{1/2}$ and $\tilde{Y} = \hat{V}\hat{\Sigma}^{1/2}$. $(\tilde{X}, \tilde{Y})$ is called scaled adjacency spectral embedding, while $(\hat{U}, \hat{V})$ is called unscaled adjacency spectral embedding.

This procedure embeds each node into a d-dimensional Euclidean space, where geometric proximity among embeddings reflects structural similarity in the original graph. Under the DSBM (a special case of the RDPG model), each node has a latent position corresponding to its block. In particular, Sussman et al. [2] proved that the following clustering criterion can be used for clustering a graph distributed according to a DSBM. Let $Z \in \mathbb{R}^{n\times 2d}$ be the obtained embedding. The following clustering criterion can be used for clustering the rows of Z into K blocks:

$$(\hat{\boldsymbol{\psi}}, \hat{\tau}) = \underset{\boldsymbol{\psi},\tau}{\operatorname{argmin}} \sum_{u=1}^{n} \|Z_u - \psi_{\tau(u)}\|_2^2, \tag{3.3}$$

where $\hat{\boldsymbol{\psi}} \in \mathbb{R}^{K\times 2d}$, $\hat{\boldsymbol{\psi}}_i \in \mathbb{R}^{2d}$ gives the centroid of block i and $\hat{\tau} : \{1, \dots, n\} \to \{1, \dots, K\}$ is the block membership function. We can approximate the solution of Eq. 3.3 by using k-means on the rows of the embedding matrix Z.

3.1.1. Theoretical Guarantees for RDPG Clustering

The consistency of ASE for RDPGs is grounded in several key theorems. Most notably, Sussman et al. [2] showed that under suitable conditions, the number of misassigned nodes is bounded.

Let us consider a DSBM parametrized as a RDPG. We can define the following constants

not depending on the number of nodes n:

- $\alpha > 0$ such that all the eigenvalues of $\nu^\top \nu$ and $\mu^\top \mu$ are greater than α.
- $\beta > 0$ such that $\beta < \|\nu_i - \nu_j\|$ or $\beta < \|\mu_i - \mu_j\|$ for all $i \neq j$.
- $\gamma > 0$ such that $\gamma < \rho_i$ for all $i \in \{1, \dots, K\}$.

Theorem 3.1 (Sussman et al.). *Suppose that the number of blocks K and the latent vector dimension d are known. Let $\hat{\tau} : V \to \{1, \dots, K\}$ be the block membership function according to Eq. 3.3. Let S_k be the set of permutations on $\{1, \dots, K\}$. It almost always holds that:*

$$\min_{\pi \in S_k} |\{u \in V | \tau(u) \neq \pi(\hat{\tau}(u))\}| \leq \frac{2^3 3^2 6}{\alpha^5 \beta^2 \gamma^5} \log n, \tag{3.4}$$

where π is a permutation belonging to S_k and an event occurs "almost always" if with probability 1 the event occurs for all but finitely many $n \in \{1, 2, \dots\}$.

3.1.2. ASE algorithm

The complete algorithm proposed by Sussman et al. [2] reads:

Algorithm 3.1 Adjacency Spectral Embedding (ASE) or SVD

Require: Adjacency matrix $A \in \mathbb{R}^{n \times n}$, embedding dimension d

1: Compute the singular value decomposition:

$$A \approx \widetilde{U} \widetilde{\Sigma} \widetilde{V}^\top.$$

2: Select $\widetilde{U}_d, \widetilde{V}_d \in \mathbb{R}^{n \times d}$ the top d singular vectors, and $\widetilde{\Sigma}_d \in \mathbb{R}^{d \times d}$ the top d singular values.

3: Form the scaled latent position matrix:

$$Z = \left[\widetilde{U}_d \widetilde{\Sigma}_d^{1/2} \;\middle|\; \widetilde{V}_d \widetilde{\Sigma}_d^{1/2} \right] \in \mathbb{R}^{n \times 2d}.$$

4: *(Optional)*: For the unscaled variant, use only $\widetilde{U}_d$ and/or $\widetilde{V}_d$, omitting $\widetilde{\Sigma}_d^{1/2}$.

5: The embedding is $E = Z$.

6: Apply k-means to the rows of the embedding E.

Ensure: Cluster assignments for all nodes

3.2. Hermitian Spectral Clustering (HERM)

In the context of clustering directed graphs, the inherent asymmetry of the adjacency matrix poses a challenge for spectral methods, which traditionally rely on symmetric matrix decompositions. Cucuringu et al. [3] introduced a Hermitian matrix representation designed specifically for directed graphs, enabling the application of spectral techniques while retaining directional information.

Given a directed graph with adjacency matrix $A \in \{0,1\}^{n\times n}$, the authors define a complex-valued Hermitian matrix $H \in \mathbb{C}^{n\times n}$ as follows:

$$H = i(A - A^\top), \tag{3.5}$$

where $i(A - A^\top)$ is a scaled skew-symmetric matrix encoding directionality.

Because H is Hermitian, it admits a spectral decomposition with real eigenvalues and orthonormal eigenvectors. This allows for embedding nodes in a Euclidean space using the eigenvectors corresponding to the top eigenvalues in magnitude. Letting $H = U\Lambda U^H$ be the eigen-decomposition of H, where U is the square $n \times n$ matrix whose i-th column is the i-th eigenvector of U, and Λ is a diagonal matrix whose entries are the corresponding eigenvalues.

A d-dimensional embedding is constructed by selecting the top-d eigenvectors $U_d \in \mathbb{C}^{n\times d}$, followed by a projection step, which yields a projection matrix on the subspace spanned by the selected eigenvectors:

$$P = \sum_{j=1}^{d} g_i g_j^H, \tag{3.6}$$

where g_i, g_j are respectively the i-th and j-th top eigenvectors. The rows of P provide real-valued feature vectors that are suitable for standard clustering algorithms such as k-means.

The use of Hermitian matrices in this setting offers several key insights. First, it avoids the loss of information associated with graph symmetrization. Second, it enables the detection of direction-sensitive community structures, such as flow-based clusters.

3.2.1. Herm algorithm

The complete algorithm proposed by Cucuringu et al. [3] reads:

Algorithm 3.2 Hermitian Spectral Clustering

Require: Directed adjacency matrix $A \in \mathbb{R}^{n \times n}$, number of clusters k, threshold $\varepsilon > 0$

1: Construct the Hermitian adjacency matrix:

$$H = i(A - A^\top)$$

2: Compute the eigenpairs $\{(\lambda_j, \mathbf{g}_j)\}_{j=1}^{\ell}$ of H with $|\lambda_j| > \varepsilon$

3: Form the projection matrix:

$$P = \sum_{j=1}^{\ell} \mathbf{g}_j \mathbf{g}_j^\dagger$$

4: The embedding is $E = P$.

5: Apply k-means to the rows of the embedding E.

Ensure: Cluster assignments for all nodes

In general, it must be $l \leq k$ and for practical purposed, Cucuringu et al. propose to set $l = k$. However, in our implementation, we followed the approach proposed by Hayashi et al. [4] (which is based on the same initial Hermitian matrix) and set $l = k$ when k is even, $l = k - 1$ otherwise.

3.3. Skew-Symmetric Spectral Clustering (SKEW)

Recent literature introduced various complex-valued adjacency matrices to better study directed graphs, employing also directionality. In particular, remarkable are the works from Liu et al. [35] and Guo et al. [36]. Hayashi et al.[4] proposed a novel approach that leverages the skew-symmetric part of the adjacency matrix to uncover directional cluster structures. This method is based on the insight that the asymmetry in the graph encodes meaningful information that can be exploited to detect asymmetric community patterns, such as hierarchical or flow-like structures.

Hayashi et al. focus on the Hermitian matrix first proposed by Cucuringu et al. [3], which takes into account the directionality of a directed graph. Let $A \in \mathbb{R}^{n \times n}$ be the adjacency

matrix of a directed graph. This matrix $H \in \mathbb{C}^{n\times n}$ is given by:

$$H_{ij} = \begin{cases} i \cdot a_{ij} & \text{if } i \to j \\ -i \cdot a_{ij} & \text{if } i \leftarrow j \\ 0 & \text{otherwise} \end{cases}. \tag{3.7}$$

This matrix is both Hermitian and skew-symmetric and can also be written as $H = iK = i(A - A^T)$, where $K = A - A^T$ is real and skew-symmetric.

Given that H is Hermitian, it admits a complete set of orthonormal eigenvectors and has purely real eigenvalues. Moreover, since $H = iK$, the eigenvectors of H and K coincide and share the same information needed during the clustering task. This observation allows us to equivalently perform spectral analysis on K instead of the complex-valued matrix H, avoiding the need for complex arithmetic.

Let us denote the eigen-decomposition of the Hermitian matrix H as:

$$H = U\Lambda U^H, \tag{3.8}$$

where $U \in \mathbb{C}^{n\times n}$ is a unitary matrix whose columns are the eigenvectors, and $\Lambda \in \mathbb{R}^{n\times n}$ is a diagonal matrix containing the real eigenvalues. This is equivalent to considering the singular value decomposition of the real skew-symmetric matrix K, given by:

$$K = A - A^T = U\hat{\Sigma}V^T. \tag{3.9}$$

Hayashi et al. show that the final embedding Z can be obtained from the columns of both U and V.

3.3.1. SKEW algorithm

The complete algorithm proposed by Hayashi et al. [4] reads:

Algorithm 3.3 Skew-Symmetric

Require: Directed adjacency matrix $A \in \mathbb{R}^{n \times n}$, desired number of clusters k.

1: Construct the skew-symmetric matrix:

$$K = A - A^\top$$

2: Compute the rank-d eigenvalue decomposition of S:

$$S = \tilde{U}\tilde{\Sigma}\tilde{V}^\top,$$

where $\{\tilde{U}, \tilde{\Sigma}, \tilde{V}\} \in \mathbb{R}^{n \times l}$. Let $l = k$ if k is even, $l = k - 1$ id k is odd.

3: The embedding is $E = \tilde{U}$.

4: Apply k-means to the rows of the embedding E.

Ensure: Cluster assignments for all nodes

3.4. Block-Cyclic Spectral Clustering (BCS)

We next outline an alternative method derived from the spectral clustering framework for directed graphs [33]. We summarize the main assumptions and derivation steps underlying the algorithm presented in [5].

3.4.1. Block-Cyclic Graphs

A block-cycle is a directed graph where nodes can be partitioned into nonempty blocks with a cyclic pattern of connections between blocks. It can be formally defined as follows [5].

Definition 3.2 (Block-Cycle). *Given V a set of nodes, $E \in V \times V$ the set of directed edges, and $W \in \mathbb{R}^{n \times n}$ a matrix of positive edge weights (i.e. the adjacency matrix).*

A directed graph $G = (V, E, W)$ is a block-cycle of k blocks if it contains at least one directed cycle of length k and if there exists a function $\tau : V \to \{1, \dots, k\}$ partitioning the nodes of V into k non-empty subsets, such that

$$E \subseteq \{(u, v) : (\tau(u), \tau(v)) \in \mathcal{C}\} \tag{3.10}$$

where $\mathcal{C} = \{(1, 2), (2, 3), \dots, (k - 1, k), (k, 1)\}$.

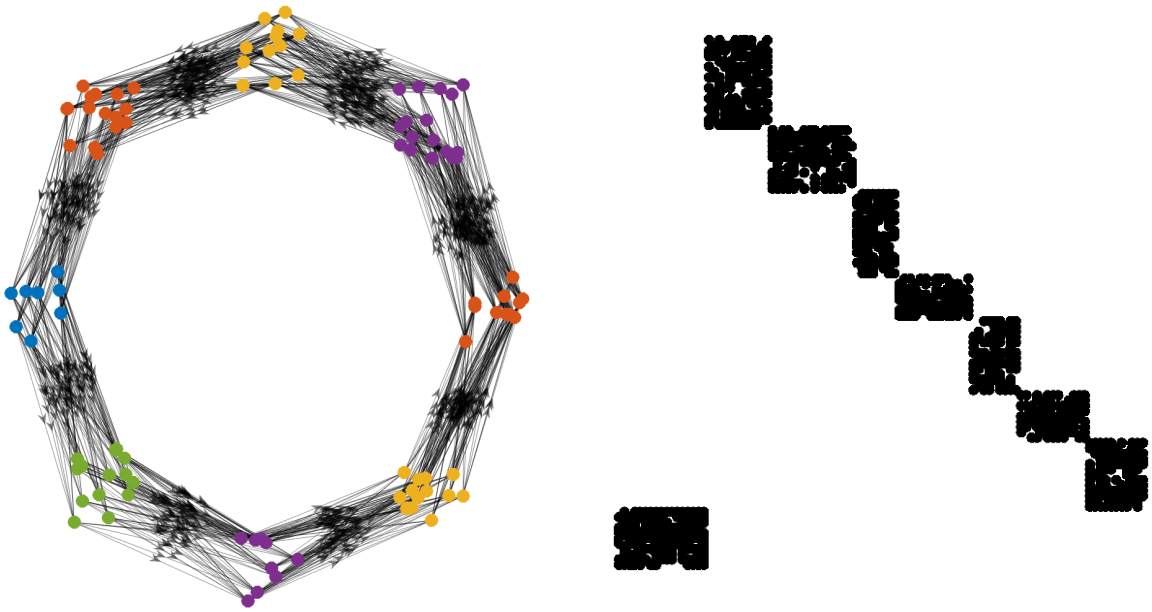


Figure 3.1: Plot of a block-cyclic graph with 8 blocks and 100 nodes and the sparsity pattern of its adjacency matrix.

So, each block is identified by a label and the nodes belonging to a block should only be connected to the nodes belonging to the consequent block. The consequent to the last block is the first one, which guarantees the cyclic pattern.

An example of a block-cyclic graph is displayed in Fig. 3.1.

To identify the communities of a block-cyclic graph, in [5] the authors propose to compute the so-called transition matrix:

$$P = D_{\text{out}}^{-1} W, \tag{3.11}$$

where D_{out}^{-1} is the matrix containing the out-degrees for each node. It can be rewritten as:

$$P_{ij} = \begin{cases} \dfrac{1}{d_i^{\text{out}}} & \text{if } d_i^{\text{out}} > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{3.12}$$

where d_i^{out} is the out-degree of node i. A spectrum analysis is then conducted on the transition matrix, yielding to an embedding on which k-means can recover the original communities.

3.4.2. BCS algorithm

Here we report the slightly modified Block-Cyclic Spectral clustering algorithm from [5].

Algorithm 3.4 Block-Cyclic Spectral (BCS) Clustering

Require: Directed adjacency matrix $A \in \mathbb{R}^{n \times n}$, number of clusters k

1: Compute the transition matrix:

$$P = D_{\text{out}}^{-1} W,$$

where $D_{\text{out}} = \text{diag}(d_1^{\text{out}}, \ldots, d_2^{\text{out}})$.

2: Find the k eigenvalues of P with the largest modulus (cycle eigenvalues) which satisfy the condition $\{\lambda \in \mathbb{C} : \mathcal{R}(\lambda) < 1 \ \wedge \ \mathcal{I}(\lambda) \geq 0\}$. Store the corresponding eigenvectors (cycle eigenvectors) as columns of a matrix $\Gamma \in \mathbb{C}^{n \times k}$.

3: The embedding is $E = [\mathcal{R}(\Gamma) \mid \mathcal{I}(\Gamma)]$.

4: Apply k-means to the rows of the embedding E.

Ensure: Cluster assignments for all nodes

3.5. Block-Acyclic Spectral Clustering (BAS)

Many real-world directed networks display hierarchical (acyclic) structure between groups of vertices. Classical spectral clustering extensions for directed graphs either ignore orientation or assume strong regularity that rarely holds in practice. Van Lierde et al. [5] proposed the BAS algorithm to better detecting this type of partitions. Its core insight is to convert the acyclic digraph into a closely related nested block-cycle [5], recover blocks spectrally, and map results back.

3.5.1. Nested Block-Cycles Graphs

Definition 3.3. *Nested Block-Cycle* *A Nested Block-Cycle of k blocks is a directed graph $G = (V, E, W)$ whose nodes V are partitioned into k non-empty blocks based on the membership function $\tau : V \to \{1, \ldots, k\}$. It must satisfy the following condition:*

$$E \subseteq \{(u, v) : \tau(u) < \tau(v) \vee \tau(u) = k\} \tag{3.13}$$

In such a graph, each block can be connected to every consecutive block, except for the last block, which can be connected to every block (even itself).

It can be observed [5] that when a nested block-cycle of k blocks contains a block-cycle of k blocks as a subgraph then:

1. there are k outlying eigenvalues in the spectrum of the transition matrix of G, which are distinguishable from the other eigenvalues thanks to their significantly

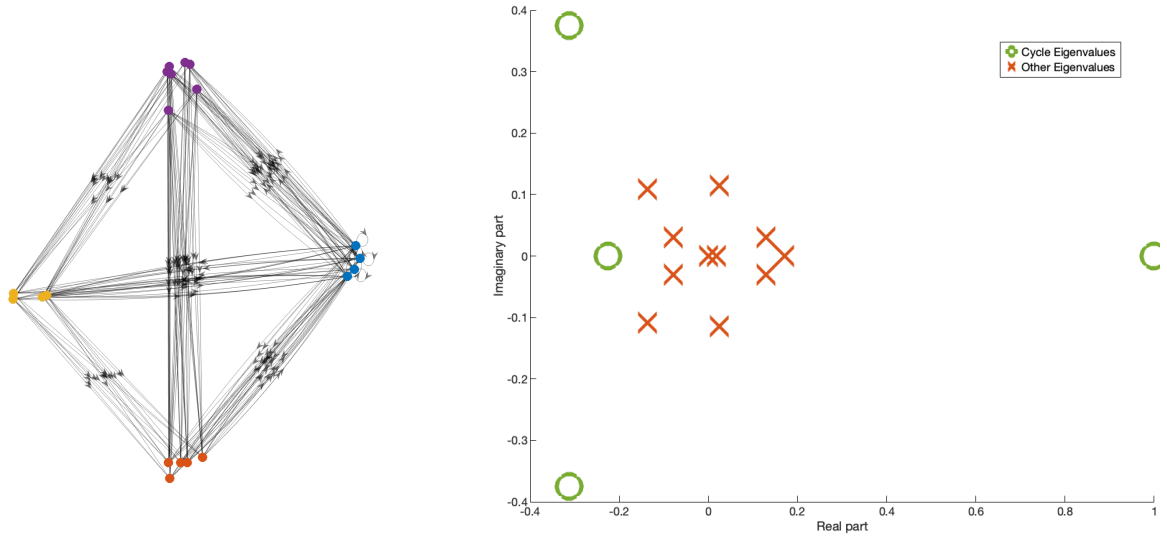


Figure 3.2: Nested Block-Cycle of 4 blocks and 20 nodes and its spectrum.

larger modulus;

2. the corresponding right eigenvectors contains the same information properties as the right cycle eigenvectors of a block-cycle (i.e. they can be used to perform the clustering).

In Fig. 3.2 is displayed an example of a Nested Block-Cycle graph and its spectrum.

3.5.2. Block-Acyclic Graphs

A block-acyclic graph is a directed graph whose vertices can be partitioned into blocks with an acyclic pattern of connections between the blocks. Here follows the formal definition:

Definition 3.4. *Block-acyclic graph* *A directed graph $G = (V, E, W)$ is a block-acyclic graph of k blocks if there exists a function $\tau : V \to \{1, \dots, k\}$ which partitions the nodes of V into k non-empty blocks, such that*

$$E \subseteq \{(u, v) : \tau(u) < \tau(v)\} \tag{3.14}$$

The block membership function τ is basically a ranking function built in such a way that each edge of the graph starts in a block with lower rank than the ending block. This property can be found in various real world scenarios (e.g. trophic networks).

It should also be noticed that this type of graph unlikely arises from a stochastic block-

model, since the definition does not include any regularity assumption on the degrees of nodes between blocks.

The main idea proposed by Van Lierde et al. is to append some edges to the block-acyclic graph on which we want to perform the clustering, in such a manner that we end up with a nested block-cycle graph, with the same number of nodes and membership function. This is straightforward since the edge sets from which the graphs are sampled only differ from the contribution given by $\tau(u) = k$ in the nested block-cycle case (see Def. 3.3 and Def. 3.4). In the block-acyclic graph, the nodes belonging to the last block do not have any outgoing edge by definition (these are all nodes with zero out-degree). On the other hand, in the nested block-cycle graph, the nodes belonging to the last block may have outgoing edges connecting them to any other node of the graph. After this consideration, we can derive the transformation we need: if a node is out-isolated (it has a zero-out degree), add out-edges connecting itself to all the other nodes of the graph.

We can now perform the BCS clustering algorithm on the obtained nested block-cycle graph to find the clusters associated with the original block-acyclic graph.

Formally, the transformation can be simply applied by using a slightly modified transition matrix P:

$$P_{ij} = \begin{cases} \frac{1}{d_i^{out}} & \text{if } d_i^{out} > 0 \\ \frac{1}{n} & \text{otherwise} \end{cases} \tag{3.15}$$

The obtained matrix is the transpose of the Google matrix with zero damping factor [37].

3.5.3. BAS algorithm

The complete algorithm proposed in [5] reads:

Algorithm 3.5 Block-Acyclic Spectral (BAS) Clustering

Require: Directed adjacency matrix $A \in \mathbb{R}^{n \times n}$, number of clusters k

1: Compute the transition matrix:

$$P_{ij} = \begin{cases} \frac{1}{d_i} & \text{if } d_i > 0 \\ \frac{1}{n} & \text{otherwise} \end{cases}$$

2: Find the k eigenvalues of P with the largest modulus (cycle eigenvalues) which satisfy the condition $\{\lambda \in \mathbb{C} : \mathcal{R}(\lambda) < 1 \ \wedge \ \mathcal{I}(\lambda) \geq 0\}$. Store the corresponding eigenvectors (cycle eigenvectors) as columns of a matrix $\Gamma \in \mathbb{C}^{n \times k}$.

3: The embedding is $E = [\mathcal{R}(\Gamma) \mid \mathcal{I}(\Gamma)]$.

4: Apply k-means to the rows of the embedding E.

Ensure: Cluster assignments for all nodes

4 | Embedding analysis

A common step in all the state-of-the-art methods we presented can be found: after obtaining an embedding through different theoretical approaches, k-means is used to map the rows of the embedding to the original latent space of the graph. In this chapter we provide an analysis on synthetic graphs, focusing mainly on the LFR benchmark graphs since they better represent common structures in a real-world scenarios. We firstly report a method for lowering the dimensionality of a set of points which will become the focus of our research. Next, we provide two experiments: the former is done with the aim of visualizing the obtained embedding both in an high-dimensional space and in a low-dimensional space, the latter aims to show how effectively the state-of-the-arts methods are recovering the information regarding the latent space encoded in the embedding.

4.1. t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) was first proposed by van der Maaten and Hinton [6] as an improvement of the previous Stochastic Neighbor Embedding (SNE) presented by Hinton and Roweis [38]. It is a non-linear dimensionality reduction technique designed for visualizing high-dimensional data in two or three dimensions. It models the pairwise similarities between data points using probability distributions with the aim of preserving the local structure of the data. Its main goal is to place similar data points in the high-dimensional space close together in the low-dimensional space and dissimilar points in the high-dimensional space far apart in the low-dimensional space.

At a high level the algorithm:

1. Converts pairwise distances in the input space into probabilistic similarities.
2. Defines a similar probability distribution in the low-dimensional space.
3. Learns the embedding by minimizing the Kullback–Leibler (KL) divergence [39] between these two distributions.

4.1.1. High-dimensional similarities

For each point $\mathbf{x}_i \in \mathbb{R}^D$ the conditional probability of selecting $\mathbf{x}_j$ as its neighbour is modeled with a Gaussian kernel

$$p_{j|i} = \frac{\exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2\right)}, \tag{4.1}$$

where the variance σ_i is chosen through a binary search to obtain a P_i with a fixed perplexity, where P_i is the conditional probability distribution over all other datapoints given the data point x_i. The perplexity is defined as:

$$\text{Perp}(P_i) = 2^{H(P_i)}, \tag{4.2}$$

where $H(P_i)$ is the Shannon entropy [26] measured in bits:

$$H(P_i) = -\sum_j p_{j|i}\log_2 p_{j|i}. \tag{4.3}$$

In this scenario, the perplexity may be interpreted as an estimation of the number of neighbors. It is user-defined and typical values ranges from 5 to 50. Since we are interested only in the similarity of pairwise points, we set $p_{i|i} = 0$.

We can now define the joint, symmetric high-dimensional similarity:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}. \tag{4.4}$$

4.1.2. Low-dimensional similarities

Let $\mathbf{y}_i \in \mathbb{R}^d$, with $d = 2$ or 3, denote the low-dimensional embedding we are searching for. In the low-dimensional space, t-SNE replaces the Gaussian used in SNE [38] by a heavy-tailed Student t-distribution [40] with one degree of freedom to mitigate the crowding problem (e.g. when working with an high-dimensional curved manifold, such as the well-known "Swiss-roll", the pairwise distances in a two-dimensional map fail to correctly model the distances between the datapoints of the high-dimensional manifold):

$$q_{ij} = \frac{\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2\right)^{-1}}{\sum_{k\neq l}\left(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2\right)^{-1}}. \tag{4.5}$$

The authors provided both a map justification (i.e. this particular distribution accounts for the crowding problem), a theoretical justification (i.e. the Student t-distribution is related to the Gaussian distribution since it can be seen as an infinite mixture of Gaussians) and a computational justification (i.e. it is faster to evaluate a Student t-distribution since it does not involve the exponential, opposite to the Gaussian). As in the high-dimensional case, we set $q_{ii} = 0$.

4.1.3. Cost function and gradient

As said before, t-SNE aims to find a low-dimensional representation which minimizes the mismatch between the high-dimensional similarities and the low-dimensional ones. The authors utilize the same measure proposed by Hinton [38], the Kullback-Leibler divergence [39]. The minimization step is done through a gradient descent technique and the cost function reads:

$$C = \mathrm{KL}(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{4.6}$$

where $P = [p_{ij}]$ and $Q = [q_{ij}]$.

The gradient of the proposed cost function is:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2\right)^{-1} (\mathbf{y}_i - \mathbf{y}_j). \tag{4.7}$$

4.1.4. Optimization strategy

The authors propose to minimize the cost function by momentum gradient descent. Various strategies are employed to optimize the algorithm:

- Momentum update. The momentum term is kept small until the map points start to become organized. In [6] the momentum $\alpha^{(t)}$ is increased after the first 250 iterations following the scheme:

$$\alpha^{(t)} = \begin{cases} 0.5 & t < 250 \\ 0.8 & t \geq 250 \end{cases}$$

- Adaptive learning rate. The adaptive learning rated proposed by Jacobs [41] is employed. The learning rate is increased following the directions in which the gradient is stable. The initial learning rate η is set to 100 and it is gradually increased following the Jacobs' scheme.

- Early exaggeration. The idea is to multiply p_{ij} by a factor during the first T_{ex} iterations to create larger between-cluster gaps. This implies that most of the q_{ij} values, although they still sum to 1, are far too small to match their corresponding p_{ij}. The optimizer therefore concentrates on reproducing the relatively large p_{ij} probabilities with likewise large q_{ij}. In turn, the true data groups collapse into compact, well-separated islands in the embedding. The map ends up with a lot of empty space, allowing those islands to slide freely with respect to one another and making it easier to discover an overall layout that reflects the global structure. In their experiments, the authors used an exaggeration term equals to 4 for the first $T_{\text{ex}} = 50$ iterations.

The update performed by gradient descent reads:

$$Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial Y} + \alpha^{(t)}(Y^{(t-1)} - Y^{(t-2)}) \tag{4.8}$$

4.1.5. t-SNE algorithm

The complete algorithm proposed by van der Maaten and Hinton [6] reads:

Algorithm 4.1 t-SNE

Require: Data matrix $\mathbf{X} \in \mathbb{R}^{n \times D}$, target dimension d=2, perplexity Perp, learning rate η

1: Compute $P = [p_{ij}]$ via Eqs. 4.1–4.4.
2: Initialise $\mathbf{Y}^{(0)} \in \mathbb{R}^{n \times d}$ from $\mathcal{N}(0, 10^{-4})$.
3: **for** $t = 1$ **to** T **do**
4: Obtain $Q = [q_{ij}]$ from Eq. 4.5.
5: Compute gradient $\partial C / \partial \mathbf{Y}$ using Eq. 4.7.
6: Update $\mathbf{Y}^{(t)}$ with Eq. 4.8.
7: **if** $t = T_{\text{ex}}$ **then**
8: remove early exaggeration on P.
9: **end if**
10: **end for**

Ensure: 2-D embedding $\mathbf{Y}^{(T)}$

4.1.6. MATLAB® implementation details

MATLAB® provides a high-level routine `tsne` in the Statistics and Machine Learning toolbox. It is based on a computationally efficient implementation of t-SNE provided by

van der Maaten [7], which is based on the Barnes-Hut algorithm [42] for obtaining feasible and efficient approximations. This particular approach is based on the observation that the t-SNE gradient can be seen as the sum of attractive and repulsive forces (given by the similarities) between the different data points. This observation let us tackle the t-SNE optimization step as an n-body problem, on which the Barnes-Hut algorithm is well suited. The interface mirrors the algorithmic steps outlined before while exposing several engineering shortcuts:

- Syntax. `Y = tsne(X,'Algorithm','barneshut', ...)` embeds the $n \times D$ matrix `X` into a $n \times 2$ array `Y`. Setting `'Algorithm','exact'` enforces the quadratic variant. The default value for `'Algorithm` is `'barneshut`, which performs an approximate optimization that is faster and uses less memory when the number of data rows is large.
- Automatic pre-processing. If $D > 50$ the function reduces the input to the leading 50 PCA components (`'NumPCAComponents',50`) and then z-scores every column (`'Standardize',true`). The defult value is `0`, meaning that `tsne` does not use PCA but directly performs t-SNE on the data.
- Perplexity guard. As defined in the t-SNE algorithm. It is used to define a measure of the effective number of local neighbors for each point. In the Barnes-Hut algorithm, `tsne` uses `min(3*Perplexity,N-1)`.
- Reproducibility. The embedding is sensitive to the random initialization; fixing the global RNG state (`rng(42)`) guarantees identical mappings across runs.

4.2. Visualization of the embedding in both high-dimensional and low-dimensional spaces

The aim of this experiment is to observe how the obtained embedding appears in both the original high-dimensional space and in the low-dimensional one. After conducting various experiments we found that the method with better performances is the Adjacency Spectral Embedding (ASE) method [2]. With this in mind, we focused on the embeddings produced by such method (we used the more computationally efficient unscaled version) and we are going to report visualizations accordingly.

4.2.1. Experiment setting

The experiments we conducted followed the steps:

- Produce a synthetic graph with a well defined number of clusters k and number of nodes n.
- Compute the embedding following the ASE approach [2]. We end up with an embedding $E \in \mathbb{R}^{n \times 2k}$.
- Plot the high-dimensional embedding. This step is obviously not straightforward since there is not a standard way for visualizing high-dimensional spaces. In our approach we plotted each component (with a total of n components) of each column (with a total of $2k$ columns) of the embedding. We used the x-axis to represent each component, different coloring to represent each column, and the y-axis to represent the value of each component for each column. The aim of this visualization is to show how well different components are separated from each other, performing a simple linear mapping from the high-dimensional space to the two-dimensional space.
- Compute the map to the two-dimensional space using tSNE and plot the mapped data using the ground truth labels for coloring.

The idea behind this approach is to see both how the high-dimensional embedding appears and if it provides enough valid information to effectively visualize the clusters associated with it in the low-dimensional space. Since we want to have direct control on the number of cluster for each synthetic graph, we use the DSBM routing for their generation. The parameters we use are the following: $k \in [2, 4]$, $n = 5000$, $p = q = 0.008$ and $\mu = 0$.

4.2.2. Experiment results

The obtained plots for the DSBM with 2, 3 and 4 communities are pictured in Figs. 4.1, 4.2 and 4.3, respectively. We can observe that in the high-dimensional embedding, all the components belong to a small interval in values (this may be a problem for k-means, as we will see next). However it codifies meaningful information regarding the latent graph, since in the two-dimensional plot we can see a separation between the different clusters colored based on their ground truth.

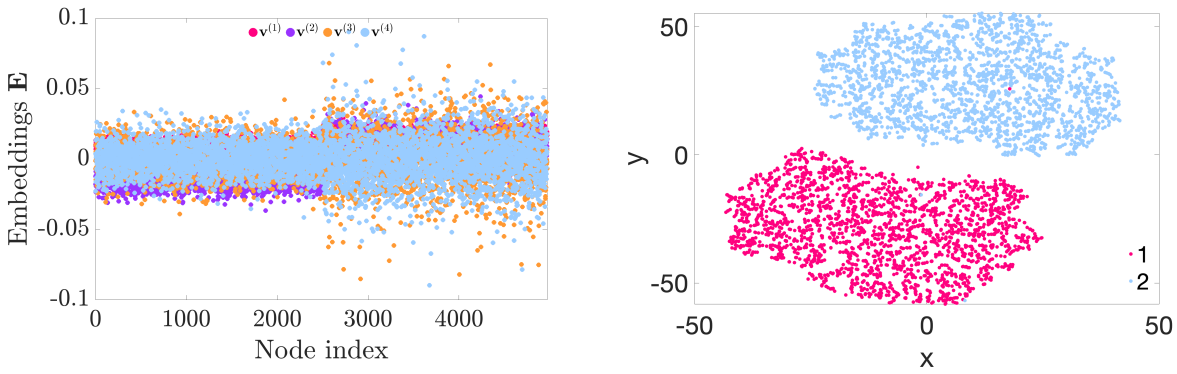


Figure 4.1: High-dimensional and two-dimensional visualization of the embedding obtained through ASE for a DSBM with 2 communities.

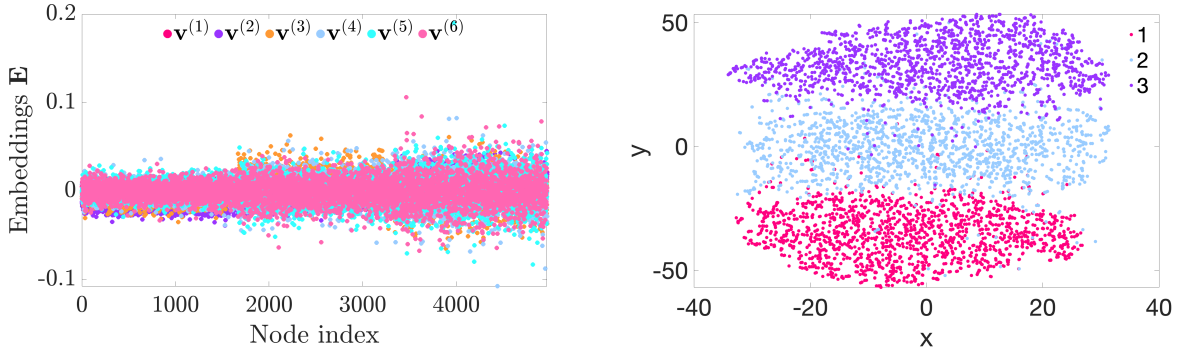


Figure 4.2: High-dimensional and two-dimensional visualization of the embedding obtained through ASE for a DSBM with 3 communities.

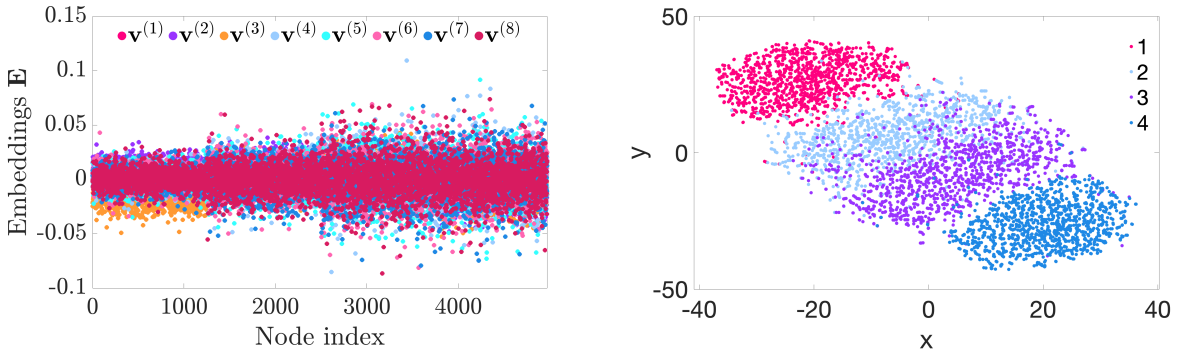


Figure 4.3: High-dimensional and two-dimensional visualization of the embedding obtained through ASE for a DSBM with 4 communities.

4.3. Comparison between the inferred labels and the ground truth

The aim of this experiment is to compare the inferred labels with the ground truth. We are interested in observing how well the cluster assignment algorithm (i.e. k-means) is recovering the latent graph information given the embedding. As for the previous experiment, we are going to use the ASE routine [2] (in this case the scaled version, loosing in computational performances but increasing in cluster recognition). The main idea is to see when the best performing method struggles and try to understand why.

4.3.1. Experiment setting

The experiments we conducted followed the steps:

- Produce a synthetic graph with n nodes, k communities and a well defined membership ground truth $\tau : \{1, \dots, n\} \to \{1, \dots, k\}$.
- Compute the inferred membership $\tau' : \{1, \dots, n\} \to \{1, \dots, k\}$ and the embedding $E \in \mathbb{R}^{n \times 2k}$ following the ASE approach [2].
- Compute the map to the two-dimensional space using tSNE.
- Plot two scatterplots of the two-dimensional embedding: the former colored based on the ground truth membership τ and the latter colored based on the inferred membership τ'.

We work with both DSBM graphs and a LFR Benchmark graph. Regarding the DSBM the parameters we use are the following: $k \in \{3, 4, 5, 10\}$, $n = 5000$, $p = q = 0.008$ and $\mu = 0$.

We use 3, 4 and 5 clusters since it is easier to observe mismatches in the membership with less clusters. We report also the results for the 10 clusters graph to show how it becomes difficult to observe mismatches since the clusters separation becomes harder to identify. This behavior is strictly present in the DSBM since it is denser and cluster recovery becomes difficult. Indeed, we also provide the results for a LFR Benchmark graph, on which it is easier to identify the clusters in the low-dimensional embedding. Regarding the LFR Benchmark graph the parameters we used are the following: $n = 1000$, $\langle k \rangle = 15$, $k_{\max} = 50$, $c_{\min} = 20, c_{\max} = 50$ and $\mu = 0.1$.

These parameters lead to a graph with 29 communities.

Graph case	F-Score
DSBM 3 communities	0.926797
DSBM 4 communities	0.915184
DSBM 5 communities	0.944211
DSBM 10 communities	0.353127
LFR 29 communities	0.912698

Table 4.1: F-Scores obtained using the SVD scaled algorithm for each graph case analyzed.

4.3.2. Experiment results

The obtained plots for the DSBM with 3, 4, 5 and 10 communities are pictured in Figs. 4.4, 4.5, 4.6 and 4.7, respectively. It is interesting to notice that most of the misclassified nodes are on the clusters boundaries. However, from this particular case we cannot conclude anything regarding the information fetched by k-means from the high-dimensional embedding, since also in the two-dimensional one the errors are strictly on the boundaries of the clusters. Definitely more interesting is what we found in the LFR Benchmark graph, depicted in Fig. 4.8. In particular, in Fig. 4.8 we highlighted where the most insightful misclassifications are. There we can see that in the low-dimensional embedding the clusters are well separated, however the k-means classification on the high-dimensional embedding produces errors that would not be produced in the two-dimensional mapping. We also report all the accuracies (F-Scores) associated with the inferred labels for each graph case in Tab. 4.1. Let us remember how big the high-dimensional embedding is. Since we are using the ASE method [2], the final embedding is going to be $E \in \mathbb{R}^{n\times 2k}$, where n is the number of nodes and k is the number of communities. Since the latent graph is generated with 29 communities, we end up with an embedding in a $29 \times 2 = 58$ dimension Euclidean space. A first hypothesis is that we are observing the well-known curse of dimensionality [43]. Indeed, in the two-dimensional map the clusters are well-separated and we could identify the different clusters by eye. Now the following question arises: is the nonlinear dimensionality reduction technique we are using (i.e. t-SNE) capable of better recognize the clusters in the high-dimensional space than k-means? If this is true, it could be possible to determine the final clusters by using k-means directly on the two-dimensional map. In the next chapter we are going to investigate and confirm this hypothesis, providing a new approach for the spectral based clustering of directed graphs which outperforms the state-of-the-art methods.

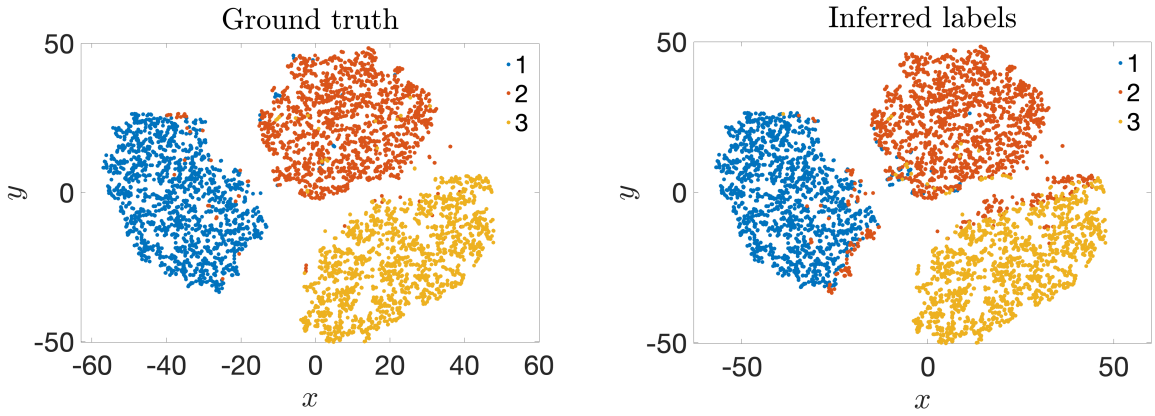


Figure 4.4: Ground truth and inferred membership colored visualization of the embedding obtained through ASE for a DSBM with 3 communities. The accuracy (F-Score) associated with the inferred labels is 0.926797.

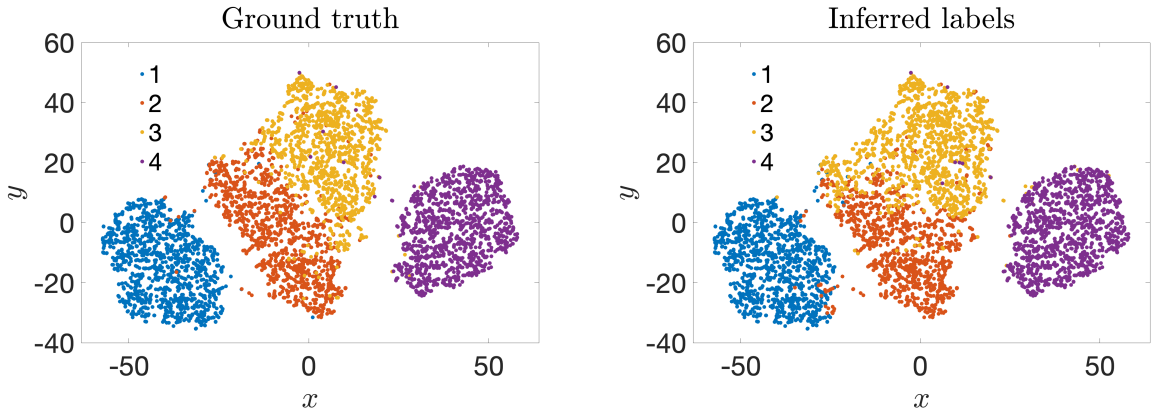


Figure 4.5: Ground truth and inferred membership colored visualization of the embedding obtained through ASE for a DSBM with 4 communities. The accuracy (F-Score) associated with the inferred labels is 0.915184.

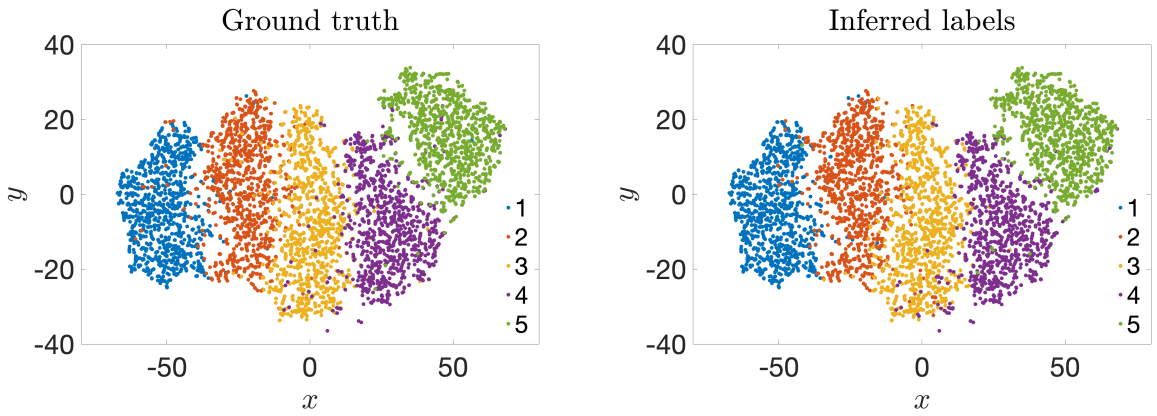


Figure 4.6: Ground truth and inferred membership colored visualization of the embedding obtained through ASE for a DSBM with 5 communities. The accuracy (F-Score) associated with the inferred labels is 0.944211.

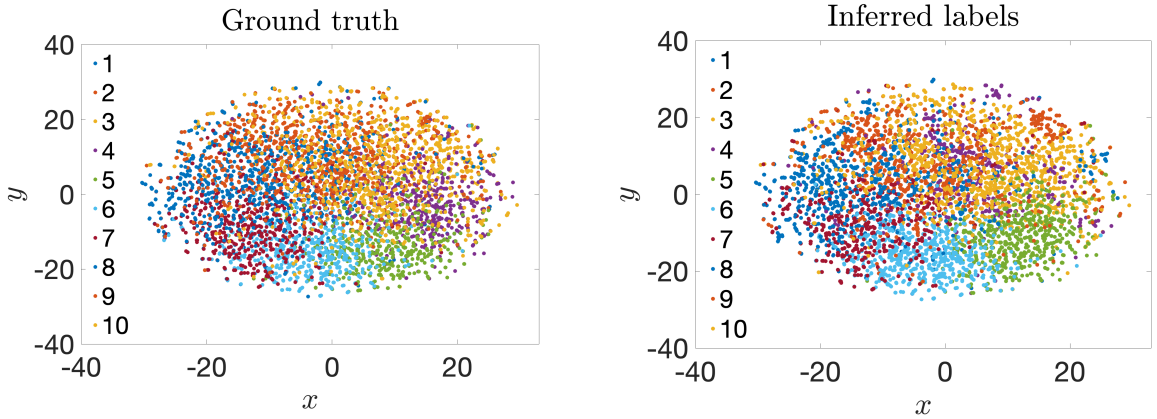


Figure 4.7: Ground truth and inferred membership colored visualization of the embedding obtained through ASE for a DSBM with 10 communities. The accuracy (F-Score) associated with the inferred labels is 0.353127.

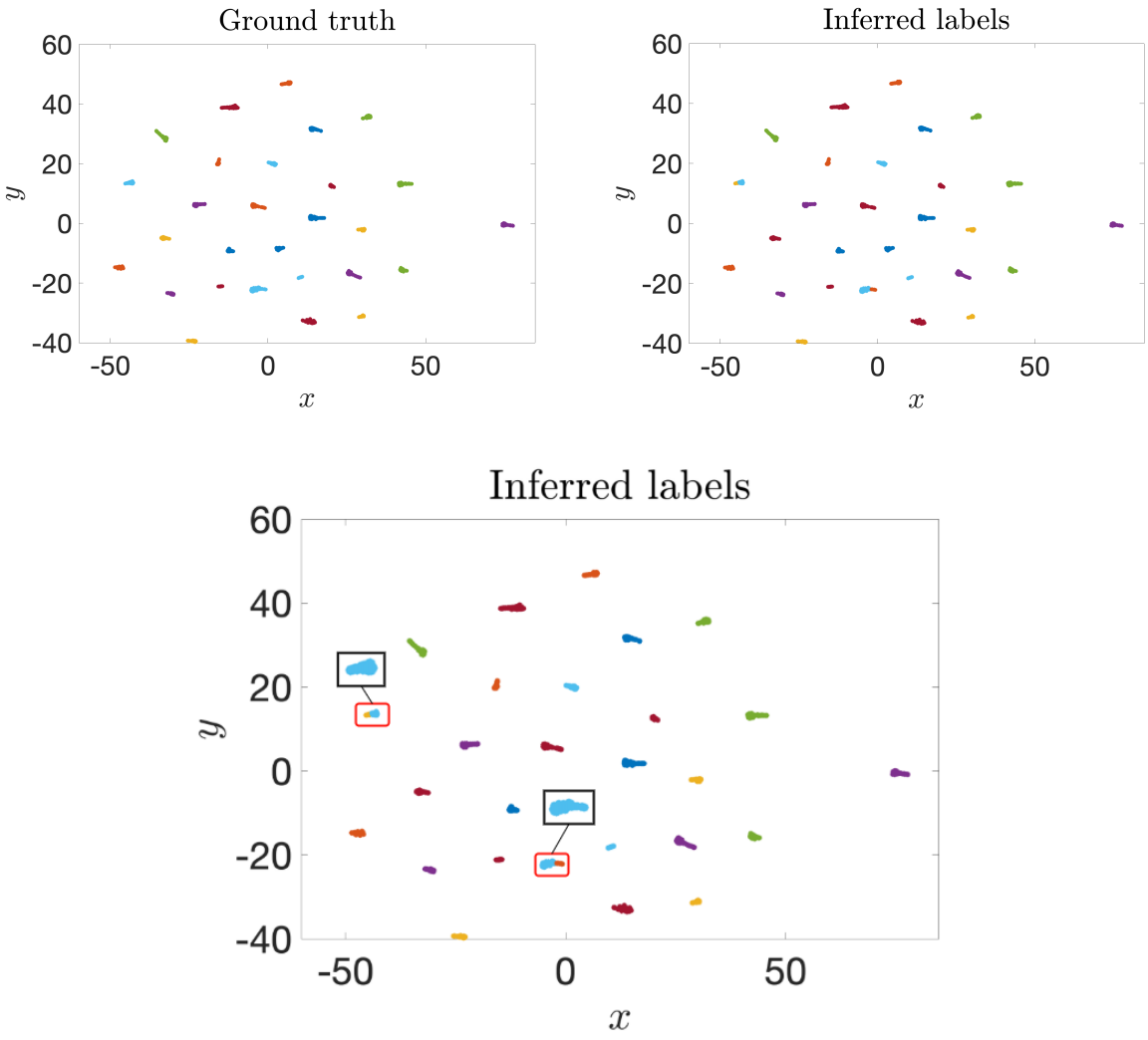


Figure 4.8: Ground truth and inferred membership colored visualization of the embedding obtained through ASE for a LFR Benchmark graph with 29 communities. Comparison of the most visible assignment errors. The accuracy (F-Score) associated with the inferred labels is 0.912698.

5 | Reducing the dimensionality of directed spectral embeddings

In this chapter we investigate the curse of dimensionality that arises when using k-means on the high-dimensional embedding obtained from the spectrum based methods. First, we perform an experiment on DSBMs with increasing number of communities with the aim of visualizing the loss of information provided by the L_2 norm in the high-dimensional embedding space. In particular, we show how, with increasing number of communities (i.e. increasing dimensions in the embedding space), the distance metric used by the k-means algorithm converges to the same value, making it ineffective for the clustering task. Subsequently, we are going to perform the same experiment, but instead of applying k-means directly on the high-dimensional embedding, we first perform a dimensionality reduction through tSNE and after we apply k-means on the two-dimensional map we obtained. In particular, we are going to show how the L_2 norm in the two-dimensional space is able to depict the distances between the clusters. Next, we provide a theoretical background that justifies the usage of tSNE to better recover the latent structure of the original graph and we provide the novel algorithm for all the spectral methods presented before. Finally, we perform a complexity study of the state-of-the-art methods and our novel approach.

5.1. Visualizing the curse of dimensionality

Recent literature addressed the usage of dimensionality reduction techniques before k-means. We cite the work of Baligodugula and Amsaad [44] as a comprehensive study on the improvements obtained by using various dimensionality reduction techniques, such as t-SNE, before the usage k-means for determining the original clusters on different datasets. Others worked on constructed (algorithmically derived) feature spaces for k-means. In particular, Drineas et al. [45] proposed an approach based on the Singular Value Decomposition (SVD), Johnson and Lindenstrauss [46] provided a method based on random projections and finally, Boutsidis et al. [47] proposed a provably accurate

feature selection method for k-means clustering based on the previous SVD and random projections articles.

Here, we want to experimentally show the curse of dimensionality in the specific case of spectral based methods for the clustering of directed graphs. In particular, we consider the embedding on which k-means infers the label assignments and show how the Euclidean based distance lacks of information when approaching high-dimensions spaces.

5.1.1. Average distance ratio (ADR)

Let us define a metric that will be useful in understanding how well the clusters are separated in the high-dimensional space: the Average Distance Ratio (ADR). Given E the embedding and n the number of nodes, let $d_{\min}(i) = \underset{j\in\{1,\dots,n\},\, j\neq i}{\operatorname{argmin}} \|E_i - E_j\|$ be the distance of a point i from it nearest neighbor and $d_{\text{avg}}(i) = \frac{1}{n-1}\sum_{j\in\{1,\dots,n\},\, j\neq i}\|E_i - E_j\|$ be the average distance of i from all the other points. The Average Distance Ratio reads:

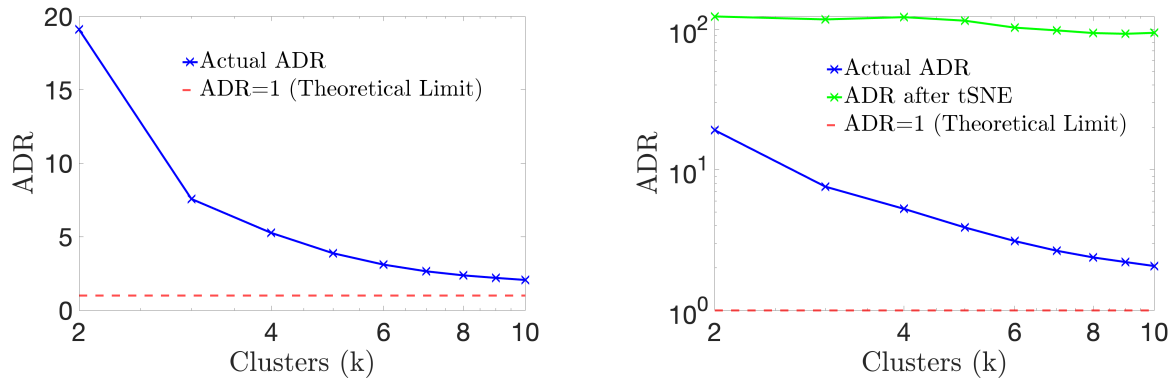
$$\text{ADR} = \frac{\text{Avg. distance to nearest neighbor}}{\text{Avg. distance to random point}} = \frac{\frac{1}{n}\sum_{i\in\{1,\dots,n\}} d_{\min}(i)}{\frac{1}{n}\sum_{i\in\{1,\dots,n\}} d_{\text{avg}}(i)} \tag{5.1}$$

When $R \to 1$ (where 1 is a theoretical limit) the distance from the nearest neighbor approaches the distance to all the other neighbors, making it impossible to use any distance-based clustering criterion (e.g. k-means).

5.1.2. ADR for the high-dimensional embeddings

In our experiment, we considered the unscaled ASE embedding [2] for each graph and we computed the ADR. We performed tests on DSBMs with the following parameters: $k \in \{1, \dots, 10\}$, $n = 5000$, $p = q = 0.008$ and $\mu = 0$.

To confirm the arise of the curse of dimensionality in high-dimensionality embeddings we expect $R \to 1$ with increasing number of clusters (the embedding dimension for ASE is dependent on the number of clusters since $E \in \mathbb{R}^{n\times 2k}$). In Fig 5.1a we report the results. As expected, the ADR decreases with increasing number of clusters, making it difficult for k-means to recover the structure of the latent graph.



(a) High-dimensional embedding. (b) High and two-dimensional embeddings.

Figure 5.1: ADR for DSBMs with increasing number of communities calculated from the original high-dimensional embedding and from the two-dimensional map obtained through t-SNE.

5.1.3. ADR for the two-dimensional embeddings

We now want to show how the mapping to a lower-dimensional space let us recover more information through the Euclidean distance. We expect to obtain higher values of ADR, meaning better cluster separation and recovery by k-means. To show that, we compute the ADR for the two-dimensional mapping of the original embedding obtained through t-SNE. After that, we compare both the ADR of the high-dimensional embedding and the ADR of the two-dimensional embedding. We report the obtained results in Fig. 5.1b. The difference is really clear, with 2 orders of magnitude difference between the ADR of the two-dimensional embedding and the one of the high-dimensional embedding. This numerical result confirms that t-SNE preserves local structure, making the Euclidean distances bigger between different clusters. This should lead to better cluster recovery for k-means.

5.2. Reducing the dimensions of the spectral embeddings

The calculations below are not intended as a formal proof. Rather, they provide an analytical motivation for applying a Kullback–Leibler (KL) divergence minimization step to obtain low–dimensional maps of spectral coordinates prior to k-means. We show heuristically how Euclidean distances in high–dimensional spectral embeddings concentrate and why a KL objective, instantiated by t-SNE, can preserve neighborhood relations that are

informative for clustering. We note that, 1. an asymptotic relation between the number of nodes n and the number of communities k is not established, and hence between n and the embedding dimension d, and that 2. proving that the KL minimization circumvents the curse of dimensionality in a general setting is left for future work. We demonstrate the effectiveness of this step in Chapter 6 with experiments on synthetic and real-world directed graphs.

5.2.1. Adjancecy spectral embedding (ASE)

For a RDPG, the ASE embedding $\hat{X}$ is obtained by truncating the SVD of the adjacency matrix W to rank d [2]. Then, for a fixed index i and any vector $\mathbf{z} \in \mathbb{R}^d$, we have the central limit theorem for rows of ASE [19]:

$$\lim_{n\to\infty} \Pr[n^{1/2}(\hat{X}_n W_n - X_n)_i \leq \mathbf{z}] = \int_{suppF} \Phi(\mathbf{z}, \Sigma(\mathbf{x}))dF(\mathbf{x}), \tag{5.2}$$

where:

- $\hat{X}$ is the $n \times d$ matrix of ASE embeddings.
- W_n is a $d \times d$ orthogonal matrix (which aligns $\hat{X}$ to X).
- $\Phi(\mathbf{z}, \Sigma(\mathbf{x}))$ is the cdf of a multivariate normal distribution with mean zero and covariance matrix $\Sigma(\mathbf{x})$, evaluated at $\mathbf{z}$.

This theorem says that the distribution of $n^{1/2}(\hat{X}_n W_n - X_n)_i$ converges to a mixture of normals.

We can write:

$$n^{1/2}(\hat{X}_n W_n - X_n)_i \xrightarrow{d} \eta_i, \quad \eta_i \sim \int \mathcal{N}(0, \Sigma(\mathbf{x}))dF(\mathbf{x}), \tag{5.3}$$

which means that the lhs converges in distribution to a random vector η_i whose distribution is a weighted average of normals.

Thanks to Skorokhod's representation theorem [48] we can construct random variables such that convergence happens almost surely:

$$n^{1/2}(\hat{X}_n W_n - X_n)_i = \eta_i + \mathbf{r}_n, \tag{5.4}$$

where $\mathbf{r}_n = o_p(1)$ (vanishes in probability as $n \to \infty$). Let us now rearrange equation 5.4:

$$\begin{aligned}(\hat{X}_n W_n)_i &= X_i + n^{-1/2}\eta_i + n^{-1/2}\mathbf{r}_n \\ &= X_i + n^{-1/2}\eta_i + o_p(n^{-1/2})\end{aligned} \tag{5.5}$$

Let $\varepsilon_i = n^{1/2}\eta_i$, a distribution with mean $\mathbb{E}[\varepsilon_i] = 0$ and covariance $Cov(\varepsilon_i) = Cov(n^{-1/2}\eta_i) = n^{-1}\Sigma(\mathbf{x}_i)$ (remember $Cov(aX) = a^2 Cov(X)$. We can rewrite:

$$(\hat{X}_n W_n)_i = X_i + \varepsilon_i + o_p(n^{-1/2}) \tag{5.6}$$

Since the matrix W_n is orthogonal, for distance calculations we have:

$$||(\hat{X}W_n)_i - (\hat{X}W_n)_j||_2 = ||\hat{X}_i - \hat{X}_j||_2 \tag{5.7}$$

because rotations preserve Euclidean distances. Thus, we can drop $\mathbf{W}_n$, obtaining:

$$\hat{\mathbf{x}}_i = \mathbf{x}_i + \varepsilon_i + o_p(n^{-1/2}) \tag{5.8}$$

In ASE (and many directed spectral embeddings) the working dimension d scales with the target number of clusters (e.g., $d = 2k$ for the concatenated ASE). When we analyze the regime $d \to \infty$ below, we use it as a proxy for increasing k. Our analysis is based on the strong assumption that the number communities has an asymptotical connection with the number of nodes. In particular, for a MLE approach on a SBM [20] is proven that MLE is consistent for $O(n^{1/2})$ [49]. A valid value for our approach requires further investigation, however in the following we make the assumption that $d = 2k = O(n^{1/2})$.

5.2.2. k-means

Let us now consider two nodes i, j. We have:

$$\begin{aligned}||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_2^2 &= ||(\mathbf{x}_i - \mathbf{x}_j) + (\varepsilon_i - \varepsilon_j) + o_p(n^{-1/2})||_2^2 \\ &= ||(\mathbf{x}_i - \mathbf{x}_j)||_2^2 + 2(\mathbf{x}_i - \mathbf{x}_j)^T(\varepsilon_i - \varepsilon_j) + ||\varepsilon_i - \varepsilon_j||_2^2 + o_p(n^{-1})\end{aligned} \tag{5.9}$$

We are interested in increasing dimensions. Let us perform an aymptotic analysis for $d \to \infty$.

- $A = ||(\mathbf{x}_i - \mathbf{x}_j)||_2^2$ is the Euclidean distance between node i and j in the original

latent space. It is independent of d and we assume it is bounded. $A = O(1)$.

- $B = 2(\mathbf{x}_i - \mathbf{x}_j)^T(\varepsilon_i - \varepsilon_j) = 2\sum_{k=1}^{d}(x_{i,k} - x_{j,k})(\varepsilon_{i,k} - \varepsilon_{j,k})$. For the Cauchy-Schwarz inequality: $||B||_2 \leq 2||\mathbf{x}_i - \mathbf{x}_j||_2^2 \cdot ||\varepsilon_i - \varepsilon_j||_2 = O(1) \cdot ||\varepsilon_i - \varepsilon_j||_2$. We have $Var(\varepsilon_{i,k}) = \mathbb{E}[\varepsilon_{i,k}^2] - (\mathbb{E}[\varepsilon_{i,k}])^2 = \mathbb{E}[\varepsilon_{i,k}^2] - 0$, from which $\mathbb{E}[\varepsilon_{i,k}] = Var(\varepsilon_{i,k}) = Var(n^{-1/2}\eta_{i,k}) = n^{-1}Var(\eta_{i,k}) = n^{-1}[\Sigma_i]_{kk}$. Σ_i is bounded and does not depend on d, so $\mathbb{E}[\varepsilon_{i,k}^2] = n^{-1}O_p(1) = O_p(n^{-1})$. We now have:

$$||\varepsilon_i||_2^2 = \sum_d \varepsilon_{i,d}^2 = dO_p(n^{-1}) = O_p(dn^{-1}) \rightarrow ||\varepsilon_i||_2 = O_p(\sqrt{dn^{-1}}). \tag{5.10}$$

  To sum up, we obtain: $B = O(1) \cdot O_p(\sqrt{dn^{-1}}) = O_p(\sqrt{dn^{-1}})$.

- $C = ||\varepsilon_i - \varepsilon_j||_2^2$. In the previous point we already obtained: $C = O_p(dn^{-1})$

Wrapping all up, we have:

$$||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_2^2 = O(1) + O_p(\sqrt{dn^{-1}}) + O_p(dn^{-1}) + o_p(n^{-1}) \tag{5.11}$$

Now we can normalize by the dimension d:

$$\frac{||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_2^2}{d} = \frac{O(1)}{d} + \frac{O_p(\sqrt{dn^{-1}})}{d} + \frac{O_p(dn^{-1})}{d} + \frac{o_p(n^{-1})}{d} \tag{5.12}$$

For increasing number of dimension we get:

$$\lim_{d\to\infty} \frac{||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_2^2}{d} = 0 + 0 + O_p(n^{-1}) + 0 \tag{5.13}$$

When $d \to \infty$, the only term that gives the distance is $||\varepsilon_i - \varepsilon_j||_2^2$. We are now interested in its expectation value:

$$\begin{aligned} \mathbb{E}(||\varepsilon_i - \varepsilon_j||_2^2) = \mathbb{E}\left[\sum_{k=1}^{d}(\varepsilon_{i,k} - \varepsilon_{j,k})^2\right] &= \sum_{k=1}^{d}\mathbb{E}[(\varepsilon_{i,k}^2 - \varepsilon_{j,k}^2)^2] \\ &= \sum_{k=1}^{d}\left(\mathbb{E}[\varepsilon_{i,k}^2] + \mathbb{E}[\varepsilon_{j,k}^2] - 2\mathbb{E}[\varepsilon_{i,k}\varepsilon_{j,k}]\right) \end{aligned} \tag{5.14}$$

- $\mathbb{E}[\varepsilon_{i,k}\varepsilon_{j,k}] = 0$ because $\varepsilon_i, \varepsilon_j$ are asymptotically independent.
- $\mathbb{E}[\varepsilon_{i,k}^2] = n^{-1}[\Sigma(\mathbf{x}_i)]_{kk}$.

Thus,

$$\sum_{k=1}^{d} n^{-1}[\Sigma(\mathbf{x}_i)]_{kk} + n^{-1}[\Sigma(\mathbf{x}_j)]_{kk} = n^{-1}\left[tr(\Sigma(\mathbf{x}_i)) + tr(\Sigma(\mathbf{x}_j)\right] \tag{5.15}$$

We can now conclude that, as $d \to \infty$

$$\frac{||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_2^2}{d} \xrightarrow{P} \sigma_\varepsilon, \tag{5.16}$$

where $\sigma_\varepsilon = \lim_{d\to\infty} \frac{n^{-1}}{d}\left[tr(\Sigma(\mathbf{x}_i)) + tr(\Sigma(\mathbf{x}_j)\right]$. This means that distances converge to the same constant and curse of dimensionality arises, making k-means fail. This concentration suggests replacing a purely Euclidean objective on the raw spectral coordinates with a similarity-preserving criterion on the same coordinates: specifically, minimizing a KL divergence between high and low-dimensional neighborhood distributions before applying k-means.

5.2.3. t-SNE

t-SNE [6] minimizes Kullback–Leibler (KL) between high-dimensionality (P) and low-dimensionality (Q) similarity distributions:

$$D_{KL}(P||Q) = \sum_{i\neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{5.17}$$

where p_{ij} and q_{ij} are symmetrized similarities:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}, \quad p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-\|x_i - x_k\|^2/2\sigma_i^2\right)} \tag{5.18}$$

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k\neq l}(1 + \|y_k - y_l\|^2)^{-1}} \tag{5.19}$$

Let us now consider two cases:

- x_i, x_j belong to the same cluster. Since the points are similar, the similarity p_{ij} is high. Minimizing D_{KL} forces q_{ij} to increase when p_{ij} is high. The t-distribution in q_{ij} ensures that:

  $$q_{ij} \sim (1 + ||y_i - y_j||_2^2)^{-1} \xrightarrow{q_{ij}\uparrow} ||y_i - y_j||_2^2 \downarrow \tag{5.20}$$

  Thus, the optimization encourages pairs with large high-dimensional similarities p_{ij} to also have large low-dimensional similarities q_{ij}. Points that are neighbors in the original space remain neighbors in the embedding.

- x_i, x_j belong to the different clusters. Asymptotically, we have $\frac{||\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j||_2^2}{d} \xrightarrow{P} \sigma_\varepsilon$ from which: $p_{ij} \sim \exp\left(-\frac{d\sigma_\varepsilon}{2\sigma_i^2}\right) = e^{-O(d)} \xrightarrow{d\to\infty} 0$. Minimizing D_{KL} does not penalize $q_{ij} \to 0$ so, when q_{ij} decrease, $||y_i - y_j||_2^2 \to \infty$. Thus, between-cluster distances diverge after applying t-SNE.

In our setting, the high-dimensional inputs to t-SNE are the spectral coordinates of the graph. Thus, the KL objective is minimized on the spectral eigenvectors themselves, not on raw node attributes. This aligns the low-dimensional map with neighborhoods that are already enriched for community information by the spectral step, providing an empirically effective bridge between eigenvector embeddings and discrete partitions.

The derivations above indicate that Euclidean distances in high-dimensional spectral embeddings tend to concentrate, which undermines k-means. Minimizing a KL divergence between neighborhood distributions on the spectral coordinates offers a principled way to preserve local structure in a low-dimensional map prior to clustering. Thus, t-SNE acts as a nonlinear filter that mitigates high-dimensional noise and enhances cluster separability, making k-means effective in 2 dimensions.

We emphasize that these arguments are heuristic rather than a proof: we do not establish a scaling relation between n and k, nor do we provide a general theorem that KL minimization resolves distance concentration. Instead, this analysis motivates the empirical strategy evaluated in Chapter 6, where we observe consistent gains across synthetic and real directed graphs.

5.3. Proposed algorithms

After our analysis we can conclude that the t-SNE mapping provides an insightful embedding which lets k-means extrapolate a better information on the latent space with respect to the original high-dimensional embedding. Our approach focuses on a slight modification of the state-of-the-art spectral methods for the clustering of directed graphs we presented. In particular, we are going to map the high-dimensional embedding to a two-dimensional one through t-SNE before applying the k-means algorithm. In Alg. 5.1 we report our novel approach that applies to every spectral method we presented.

Algorithm 5.1 tSNE-enhanced spectral algorithms

Require: Adjacency matrix $A \in \mathbb{R}^{n \times n}$, number of communities k

1: Compute the high-dimensional embedding E using the procedure from the algorithm we want to enhance.
2: Compute the two-dimensional map $E' \in \mathbb{R}^{n \times 2}$ through t-SNE.
3: Apply k-means to the rows of the two-dimensional embedding E' to find the k communities.

Ensure: Cluster assignments for all nodes

5.4. Complexity analysis

We are now going to analyze the improved algorithms, in order to obtain the time complexity for each one. With respect to the original methods, the only modification is the usage of t-SNE, which is going to slightly reduce the final performances.

5.4.1. t-SNE complexity

Let n be the number of points and T the number of gradient-descent iterations (typically $T = 1000$–2000).

Exact t-SNE

The first implementation proposed by van der Maaten [6] computes the full pairwise affinity matrix $P \in \mathbb{R}^{n \times n}$ and, at every iteration, evaluates the repulsive forces in the gradient by summing over all pairs of points. Both steps cost $\mathcal{O}(n^2)$ time and $\mathcal{O}(n^2)$ memory, giving an overall $\mathcal{O}(Tn^2)$ algorithm.

Barnes–Hut t-SNE

To accelerate the repulsive-force calculation, van der Maaten introduced a Barnes–Hut approximation [7] that groups distant points into cells of a quadtree (2-D) or octree (3-D). Approximating the contribution of an entire cell by its centroid reduces the per-iteration cost to $\mathcal{O}(n \log n)$, while reducing the memory footprint to $\mathcal{O}(n)$ (for the tree and the sparse affinity matrix). This is the method employed in the MATLAB® `tsne` routine.

Table 5.1: Asymptotic runtimes and memory requirements of popular t-SNE variants (per iteration).

	Time complexity	Memory complexity
Exact t-SNE	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Barnes–Hut t-SNE	$\mathcal{O}(n \log n)$	$\mathcal{O}(n)$

5.4.2. k-means complexity

The k-means minimization problem is well-known to be NP-hard, however Lloyd's proposed an approximation approach [23] which is used nowadays. Assume we cluster a data matrix $X \in \mathbb{R}^{n \times d}$ into k centroids by Lloyd's algorithm and let J be the number of outer iterations until convergence (typically $J = 10$–30 in practice).

Lloyd's (standard) k–means. Each iteration performs two steps:

1. Assignment: compute all k distances for every point ($\mathcal{O}(nkd)$); mark the index of the nearest centroid.
2. Update: recompute every centroid as the arithmetic mean of the points assigned to it ($\mathcal{O}(nd)$).

Hence the total cost is

$$\mathcal{O}(J\,nkd), \quad \text{memory } \mathcal{O}(nd + kd).$$

Because the assignment step dominates, the algorithm is effectively linear in n but quadratic in k. In high-dimensional settings distance computations are the principal bottleneck.

k–means++ initialization. Arthur–Vassilvitskii seeding [50] chooses the first centroid uniformly at random and selects each subsequent centroid with probability proportional to the squared distance from the nearest existing centroid. This adds a one-off

$$\mathcal{O}((J+1)nkd) \quad \text{time}, \quad \mathcal{O}(nd + kd) \quad \text{memory},$$

where the "+1" accounts for the k–means++ pass. With k–means++ the number of Lloyd iterations J typically drops enough that the final algorithm is faster and yields more stable clusters than the ones obtained by random seeding. This is the method employed in the MATLAB® `kmeans` routine.

Table 5.2: Asymptotic costs for Lloyd's k–means with and without k–means++ seeding.

	Time complexity	Memory complexity
Lloyd (per iteration)	$\mathcal{O}(nkd)$	$\mathcal{O}(nd + kd)$
k–means++ seeding	$\mathcal{O}(nkd)$	$\mathcal{O}(nd + kd)$

5.4.3. ASE complexity

We split the Adjacency Spectral Embedding (ASE) of Sussman *et al.* [2] into three stages.

1. Singular–value decomposition of the adjacency matrix. A complete factorization $A = U\,\Sigma\,V^\top$ via a dense QR–based routine costs

$$\mathcal{O}(n^3) \text{ time}, \quad \mathcal{O}(n^2) \text{ memory},$$

   see Golub and Van Loan [51].

2. Compute the embedding. For the unscaled variant we simply pad $[\,U_d \,|\, V_d] \in \mathbb{R}^{n\times 2d}$. For the scaled variant we form $X = U_d\,\Sigma_d^{1/2}$ and $Y = V_d\,\Sigma_d^{1/2}$. Since $\Sigma_d^{1/2}$ is diagonal, both options require only

$$\mathcal{O}(nd) \text{ time}, \quad \text{no extra memory}.$$

3. k-means on the embedding. Clustering the n points in $\mathbb{R}^p$ ($p = 2d$) by Lloyd's algorithm costs

$$\mathcal{O}\big(Jnkp\big) \text{ time}, \qquad \mathcal{O}(np) \text{ memory}.$$

Table 5.3: Total asymptotic cost of ASE.

Time complexity	Memory complexity
$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$

5.4.4. HERM complexity

As for ASE, we split the Hermitian Spectral Clustering proposed by Cucuringu et al. [3] into three stages.

1. Eigen–decomposition of the Hermitian matrix. A dense QR routine on H requires

$$\mathcal{O}(n^3) \text{ time}, \quad \mathcal{O}(n^2) \text{ memory},$$

see Golub and Van Loan [51].

2. Compute the embedding. Let $G = [g_1 \dots g_d] \in \mathbb{C}^{n\times d}$ collect the d eigenvectors associated with the selected eigenvalues of H. HERM represents every vertex through the orthogonal projection onto the span of these eigenvectors,

$$P \;=\; GG^{\dagger} \;=\; \sum_{j=1}^{d} g_j g_j^{\dagger}.$$

The i-th vertex is embedded as the real vector given by the i-th row of P. Forming the full projection explicitly requires one dense matrix–matrix product of an $(n \times d)$ by a $(d \times n)$ matrix, which costs

$$\mathcal{O}(n^2 d) \text{ time}, \quad \mathcal{O}(n^2) \text{ memory}.$$

3. k-means on the embedding. Lloyd's algorithm on the n points in $\mathbb{R}^{n\times d}$ needs

$$\mathcal{O}(Jnkd) \text{ time}, \quad \mathcal{O}(nd) \text{ memory}.$$

Table 5.4: Total asymptotic cost of HERM.

Time complexity	Memory complexity
$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$

5.4.5. SKEW time complexity

Hayashi et al. [4] base SKEW on the real skew-symmetric component of the adjacency matrix, $S = \frac{1}{2}(A - A^{\top}) \in \mathbb{R}^{n\times n}$ with $S^{\top} = -S$. Unlike Hermitian approaches, SKEW obtains its spectral coordinates from a singular-value decomposition (SVD) of S. The pipeline has three stages.

1. Singular-value decomposition of the skew matrix. A dense QR-based routine computes $S = U\,\Sigma\,V^{\top}$ in

$$\mathcal{O}(n^3) \text{ time}, \quad \mathcal{O}(n^2) \text{ memory},$$

see Golub and Van Loan [51].

2. Compute the embedding. SKEW uses only the left singular vectors. Each vertex is mapped to the d-dimensional row of $U_d \in \mathbb{R}^{n\times d}$. Because U_d is already produced by

the SVD, this step incurs

$$\mathcal{O}(1) \text{ additional time}, \quad \text{no additional memory.}$$

3. k-means on the embedding. Lloyd's iterations on the n points in $\mathbb{R}^{n\times d}$ require

$$\mathcal{O}(Jnkd) \text{ time}, \quad \mathcal{O}(nd) \text{ memory},$$

Table 5.5: Total asymptotic cost of SKEW.

Time complexity	Memory complexity
$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$

5.4.6. BCS time complexity

Block–Cyclic Spectral clustering (BCS) [5] operates on a directed graph in four computational stages:

1. Build the transition matrix. From a sparse adjacency list with $m = |E|$ directed edges, row-normalize to obtain $P \in \mathbb{R}^{n\times n}$.

$$\mathcal{O}(m) \text{ time}, \quad \mathcal{O}(m) \text{ memory}.$$

2. Eigen-decomposition of P. Using a dense QR routine to extract the k cycle eigenpairs requires

$$\mathcal{O}(n^3) \text{ time}, \quad \mathcal{O}(n^2) \text{ memory},$$

see Golub and Van Loan [51].

3. Build the embedding. Keep the first k right eigenvectors $U_k \in \mathbb{C}^{n\times k}$ (ordered by modulus) and form

$$E = [\mathcal{R}(U_k)|\mathcal{I}(U_k)] \in \mathbb{R}^{n\times 2k}.$$

This column concatenation requires

$$\mathcal{O}(nk) \text{ time}, \quad \text{no extra memory.}$$

4. k-means on the embedding. Lloyd iterations on the n points in $\mathbb{R}^k$ incur

$$\mathcal{O}(J\,nk^2) \text{ time}, \quad \mathcal{O}(nk) \text{ memory},$$

Table 5.6: Total asymptotic cost of BCS.

Time complexity	Memory complexity
$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$

5.4.7. BAS time complexity

Van Lierde et al. [5] extend cyclic spectral clustering to directed hierarchical graphs. BAS proceeds in four computational stages:

1. Build the augmented transition matrix. Starting from a sparse adjacency list with $m = |E|$ directed edges,
   (a) mark sinks,
   (b) add one uniform out–edge from every sink to every vertex, and
   (c) row-normalize to obtain $P \in \mathbb{R}^{n\times n}$.

$$\mathcal{O}(m) \text{ time}, \quad \mathcal{O}(m) \text{ memory}.$$

2. Eigen–decomposition of the Hermitian matrix. A dense QR routine on the transition matrix requires

$$\mathcal{O}(n^3) \text{ time}, \quad \mathcal{O}(n^2) \text{ memory},$$

   see Golub and Van Loan [51].

3. Build the embedding. Retain the first k right eigenvectors $U_k \in \mathbb{C}^{n\times k}$ (ordered by modulus of their associated cycle eigenvalues) and form

$$E = [\mathcal{R}(U_k)\big|\mathcal{I}(U_k)] \in \mathbb{R}^{n\times 2k}.$$

   The operation is a column-wise copy already accommodated in memory, so it costs

$$\mathcal{O}(nk) \text{ time}, \quad \text{no additional memory}.$$

4. k-means on the embedding. Lloyd's iterations on the n points in $\mathbb{R}^{n\times d}$ require

$$\mathcal{O}(Jnkd) \text{ time}, \quad \mathcal{O}(nd) \text{ memory},$$

Table 5.7: Total asymptotic cost of BAS.

Time complexity	Memory complexity
$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$

5.4.8. Overall impact of adding t-SNE

In every pipeline the t–SNE step operates on the spectral embedding ($n \times p$ with $p \leq 2d \ll n$). Even the faster Barnes–Hut variant requires at most $\mathcal{O}(T\, n \log n)$ work and $\mathcal{O}(n)$ memory, whereas each dense spectral routine (ASE, HERM, SKEW, BAS) expends $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ memory just to obtain the eigenpairs or singular vectors. t-SNE is therefore asymptotically negligible: the spectrum-extraction stage remains the clear bottleneck. In Table 5.8 we report all the complexities we found.

Table 5.8: Dominant worst-case costs for all methods discussed (n vertices, d embedding dimension, k clusters, T t-SNE iterations, J k-means iterations).

Algorithm / Step	Time complexity	Memory complexity
Exact t–SNE	$\mathcal{O}(T\, n^2)$	$\mathcal{O}(n^2)$
Barnes–Hut t–SNE	$\mathcal{O}(T\, n \log n)$	$\mathcal{O}(n)$
Lloyd k-means (per iter.)	$\mathcal{O}(nkd)$	$\mathcal{O}(nd + kd)$
k-means++ seeding	$\mathcal{O}(nkd)$	$\mathcal{O}(nd + kd)$
ASE	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
HERM	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
SKEW	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
BCS	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
BAS	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$

Adding t-SNE does not alter the overall complexity of the pipelines: the cubic cost of the dense spectral step dominates both runtime and memory, while the t-SNE stage is comparatively light, especially when the Barnes-Hut approach is employed.

6 | Results

In this chapter we provide results for both synthetic and real-world datasets. The objective is to demonstrate the effectiveness of the t-SNE stage, which actually improves the recovery of the original memberships for directed graphs. Regarding the synthetic graphs, we consider both the DSBM and the LFR benchmark graphs. The former is used to show how our approach actually improves the clustering depending on the number of communities in the latent graph. The latter is used to show how the recovery is maintained also when perturbation is added. In both cases, the number of communities in the latent graphs are known and not inferred. After that, we move to two real-world scenarios: a graph representing email exchanges across various european research institutions [9–11] and graphs representing synthetic banking transactions data together with a set of known money laundering patterns [12–14]. In the former, we take into account a bi-partitioning problem proposed by [1] for this specific dataset. In the latter, we explore various graph related metrics with different number of communities.

6.1. Experiments on the synthetic DSBMs

We conduct experiments on synthetically generated DSBMs [21] while varying the number of communities. The objective is to assess whether the additional dimensionality–reduction stage improves cluster recovery independently of the community count. In particular, as the number of clusters increases (thereby raising the dimensionality of the spectral embedding), we examine whether the proposed procedure continues to yield gains.

We evaluate all methods introduced earlier: SVD (scaled and unscaled), HERM, SKEW, BCS, BAS, and their corresponding variants with the additional dimensionality–reduction step (SVD–tSNE scaled and unscaled, HERM–tSNE, SKEW–tSNE, BCS–tSNE, BAS–tSNE). Since ground-truth labels are known, we report the F-Score only, focusing on how well the latent communities are recovered.

For each k we generate 10 independent graph instances using different random seeds

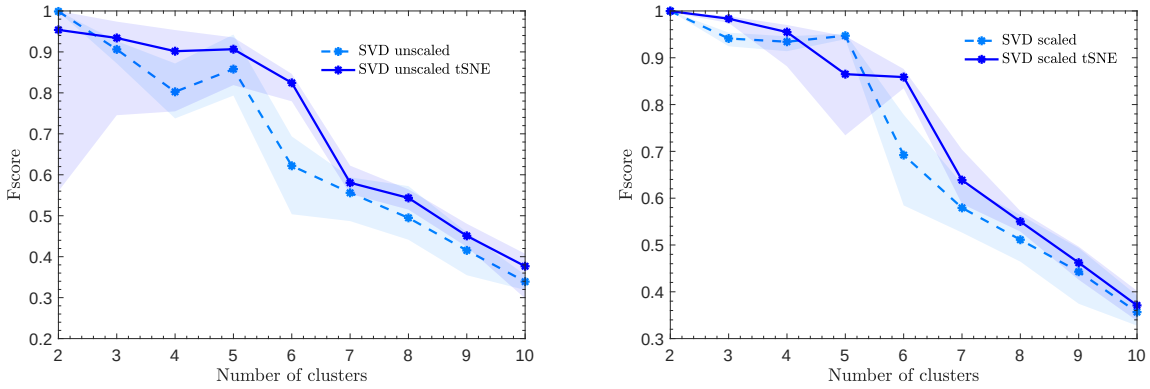


Figure 6.1: Results on DSBMs with different number of communities obtained by SVD scaled, unscaled and their t-SNE variant.

and report, for every point on the curves, the mean F-Score across seeds together with confidence intervals. The DSBM parameters are: $k \in \{2, \ldots, 10\}$, $n = 5000$, $p = q = 0.008$ and $\mu = 0$.

Figures 6.1, 6.2, and 6.3 display the resulting averages with confidence bands. Across all operators, the variants with the additional dimensionality–reduction stage achieve higher F-Scores, and the relative improvement tends to grow with k, consistent with the fact that the spectral embedding dimension scales with the number of clusters.

6.2. Experiments on the LFR benchmark graphs

We conduct a second set of experiments to test robustness under perturbations using the LFR benchmark [8], which emulates real-world degree and community-size heterogeneity. We evaluate the same suite of methods as before—SVD (scaled and unscaled), HERM, SKEW, BCS, BAS, and their variants with the additional dimensionality–reduction stage (SVD–tSNE scaled and unscaled, HERM–tSNE, SKEW–tSNE, BCS–tSNE, BAS–tSNE). Because ground-truth labels are available, we report the F-Score only.

For each μ we generate 10 independent networks with different random seeds and plot the mean F-Score together with the confidence intervals. The LFR parameters are: $n = 1000$, $\langle k \rangle = 15$, $k_{\max} = 50$, $c_{\min} = 20$, $c_{\max} = 50$ and $\mu \in [0.1, 0.4]$.

Figures 6.4, 6.5, and 6.6 display the resulting averages with confidence bands. Across all methods, the variants with the additional dimensionality–reduction stage achieve systematic improvements. In particular, the Adjacency Spectral Embedding [2] paired with the

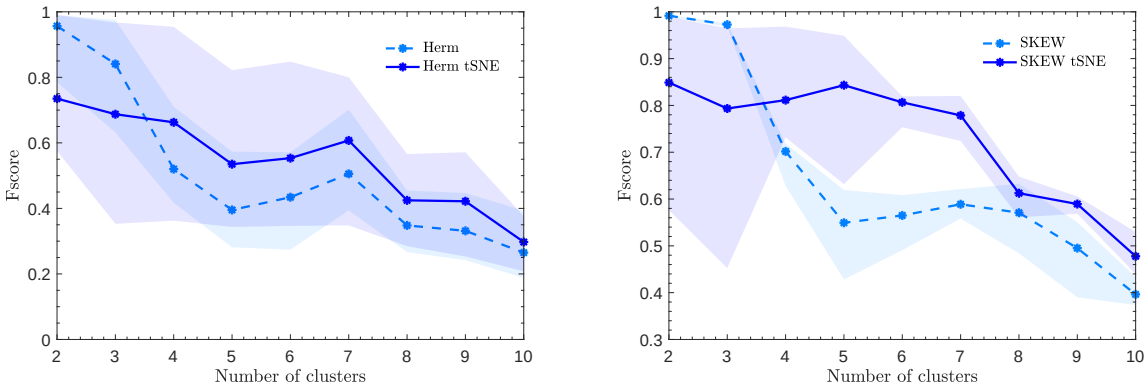


Figure 6.2: Results on DSBMs with different number of communities obtained by HERM, SKEW and their t-SNE variant.

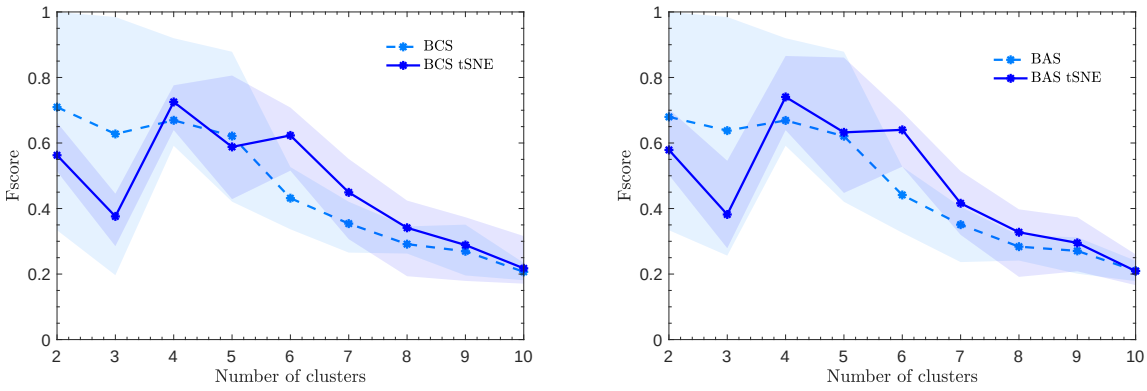


Figure 6.3: Results on DSBMs with different number of communities obtained by BCS, BAS and their t-SNE variant.

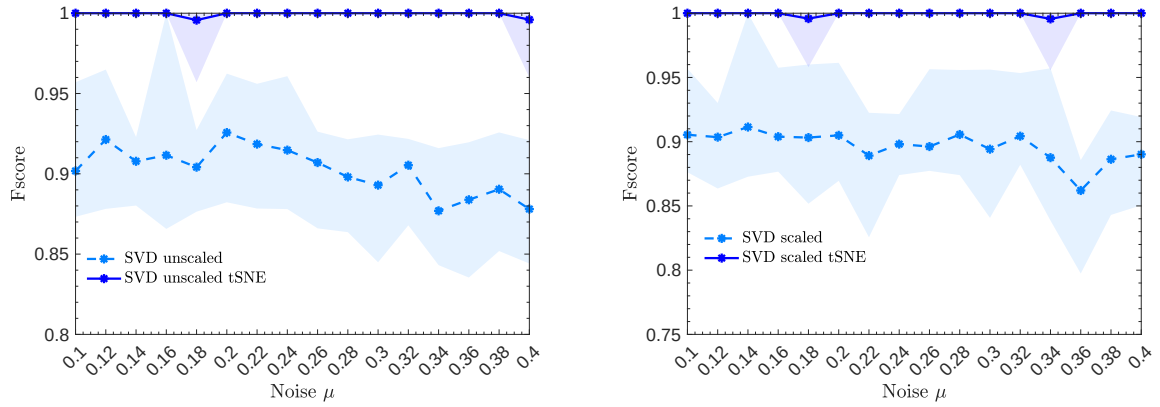


Figure 6.4: Results on LFR Benchmark Graphs with increasing perturbation obtained by SVD scaled, unscaled and their t-SNE variant.

reduction step emerges as the most reliable approach, attaining almost 100% community recovery across all noise levels considered.

6.3. Experiments on real world graphs

6.3.1. EmailEU networks

email-Eu-core [9–11] is a directed network representing email exchanges between individuals at European research institutions. An edge (u, v) indicates that person u sent at least one email to person v. The dataset includes 42 institutions, which define the ground-truth communities. Cucuringu et al [1] proposed to partition the full graph into two smaller graphs in order to perform a bi-partitioning task. We evaluate clustering methods on two-community subgraphs: *email-Eu-core12* (1st and 2nd largest communities) and *email-Eu-core23* (2nd and 3rd largest communities). We performed experiments on all the methods we presented: SVD scaled and unscaled, HERM, SKEW, BCS, BAS, SVD-tSNE scaled and unscaled, HERM-tSNE, SKEW-tSNE, BCS-tSNE, BAS-tSNE. We considered only the F-Score and ARI metrics since the ground truth labeling is known and we are interested in the observation of how well the latent communities are recovered. The obtained results are reported in Tab. 6.1. Also in this case, the Adjacency Spectral Embedding improved by t-SNE results as the best performing method, gaining a considerable improvement.

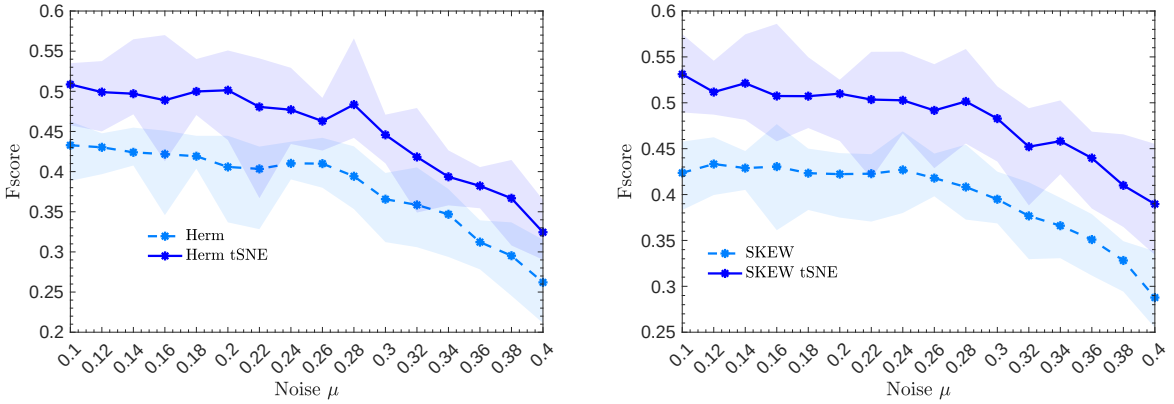


Figure 6.5: Results on LFR Benchmark Graphs with increasing perturbation obtained by HERM, SKEW and their t-SNE variant.

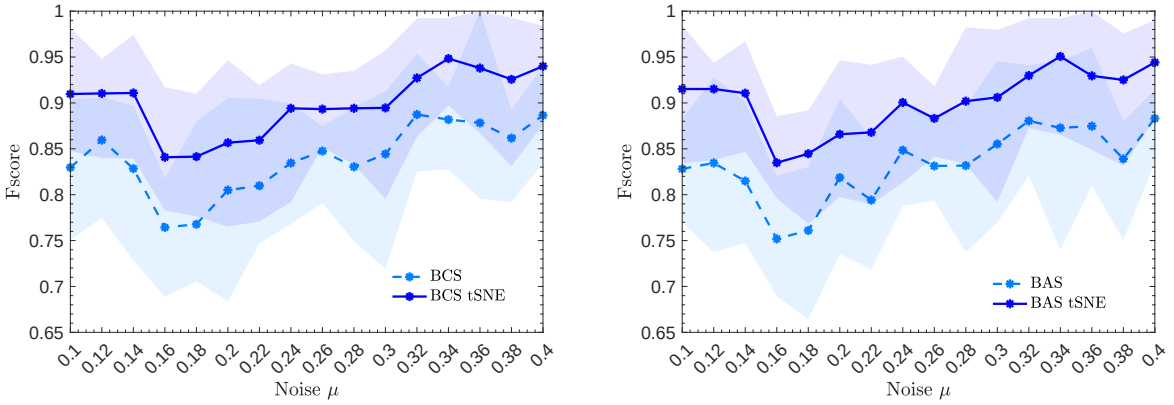


Figure 6.6: Results on LFR Benchmark Graphs with increasing perturbation obtained by BCS, BAS and their t-SNE variant.

Method	email-Eu-core12 F-Score	email-Eu-core12 ARI	email-Eu-core23 F-Score	email-Eu-core23 ARI
SVD Unscaled	0.8364	0.4754	0.7174	0.2679
SVD Unscaled tSNE	0.9596	0.8464	**0.9292**	**0.7377**
SVD Scaled	0.8307	0.4617	0.7895	0.3308
SVD Scaled tSNE	**0.9647**	**0.8649**	**0.9292**	**0.7377**
SKEW	0.6035	0.0992	0.3695	0.0060
SKEW tSNE	0.6909	0.1424	0.7406	0.2356
HERM	0.6035	0.0992	0.3695	0.0088
HERM tSNE	0.6962	0.1502	0.7406	0.2356
BAS	0.3635	0.0018	0.5518	0.0854
BAS tSNE	0.8955	0.6239	0.8050	0.4232

Table 6.1: Obtained F-Score and ARI on the email-EU dataset using the bi-partitioning problem proposed by Cucuringu et al. [1].

6.3.2. AMLSIM graphs

AMLSim graphs [12–14] are synthetic transaction networks generated by IBM's open-source AMLSim simulator, a multi-agent system that produces banking activity with built-in money-laundering typologies for benchmarking machine-learning and graph algorithms. In the graph view, vertices represent bank accounts (and optionally related parties), directed edges are time-stamped transfers annotated with amounts and types, and "alert" subgraphs mark instances of specified patterns. The generator scales to large networks and has been used to create graphs on the order of one million nodes and nine million edges, making it a practical proxy when real banking data are unavailable.

We conducted our experiments on two graphs. The former has 100 nodes and 10^4 edges, the latter has 1000 nodes and 10^5 edges. The methods we presented are not suitable to detect money laundering patterns, however they are well-suited in the task of detecting groups of users with common behaviors, such as regular buyers from e-commerce platforms and regular buyers from local shops. Since we do not have a ground truth for this type of communities for the AMLSim graphs, we are going to consider only the Top Cut Imbalance (TopCI) normalized by size and volume and the TradeFLow. We performed experiments on all the methods we presented: SVD scaled and unscaled, HERM, SKEW, BCS, BAS, SVD-tSNE scaled and unscaled, HERM-tSNE, SKEW-tSNE, BCS-tSNE, BAS-tSNE. We explored the results for various number of communities, evaluating the metrics for $k \in [2, 10]$.

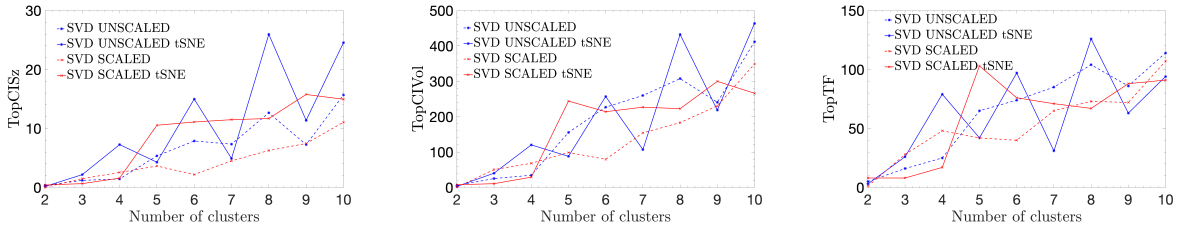


Figure 6.7: TopCI normalized by size and volume, TopTF obtained by clustering the AMLSim graph with 100 nodes with SVD unscaled, scaled and their t-SNE variants.

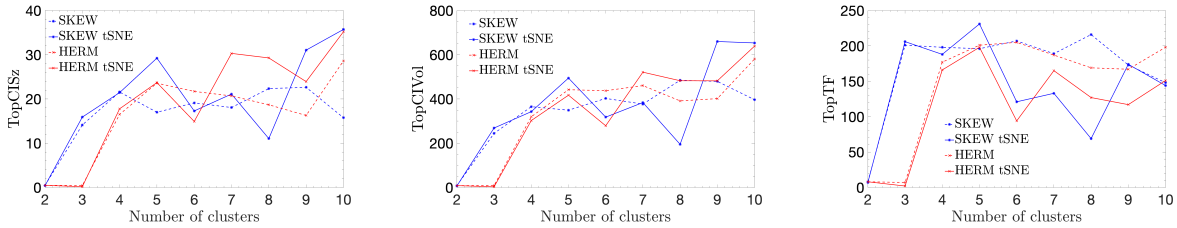


Figure 6.8: TopCI normalized by size and volume, TopTF obtained by clustering the AMLSim graph with 100 nodes with HERM, SKEW and their t-SNE variants.

The results are reported from Fig. 6.7 to Fig. 6.12. In the 100 nodes case the graph is still to small to really identify any improvement. However, when working with the graph with 1000 nodes, it is really noticeable how our approach is improving the metrics, leading to higher values (i.e. the desirable behavior for correctly detecting communities based on a flow-based clustering).

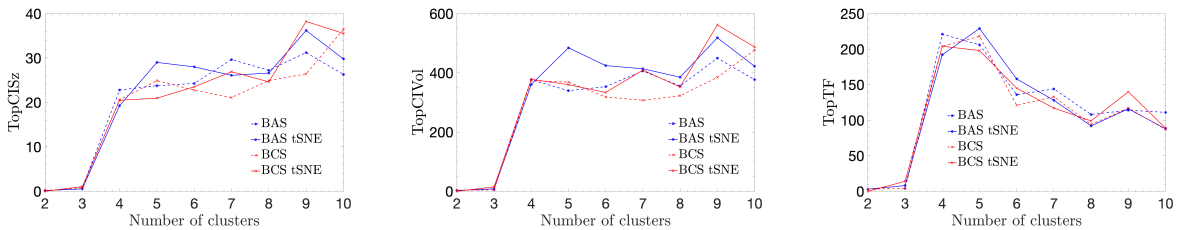


Figure 6.9: TopCI normalized by size and volume, TopTF obtained by clustering the AMLSim graph with 100 nodes with BCS, BAS and their t-SNE variants.

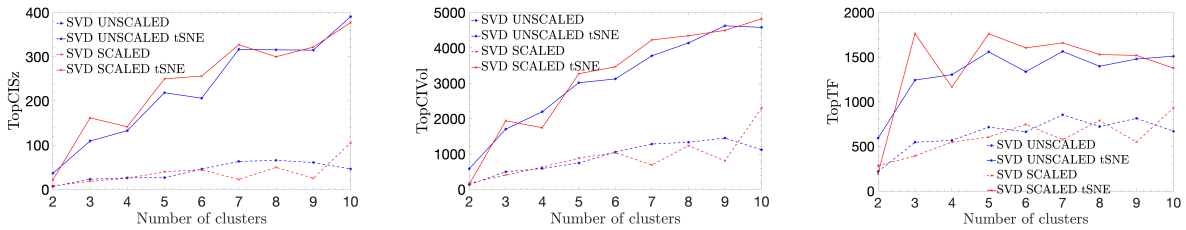


Figure 6.10: TopCI normalized by size and volume, TopTF obtained by clustering the AMLSim graph with 1000 nodes with SVD unscaled, scaled and their t-SNE variants.

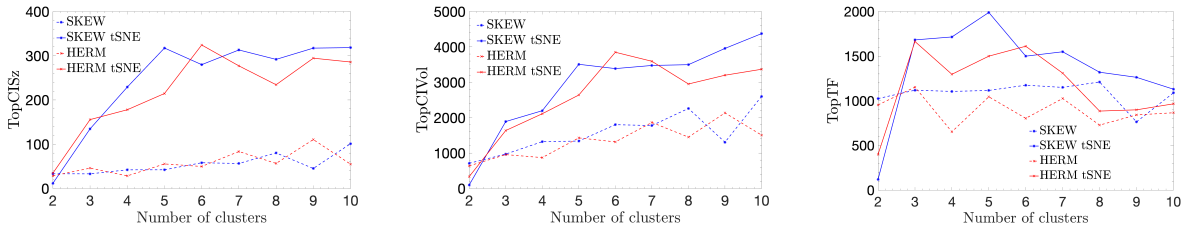


Figure 6.11: TopCI normalized by size and volume, TopTF obtained by clustering the AMLSim graph with 1000 nodes with HERM, SKEW and their t-SNE variants.

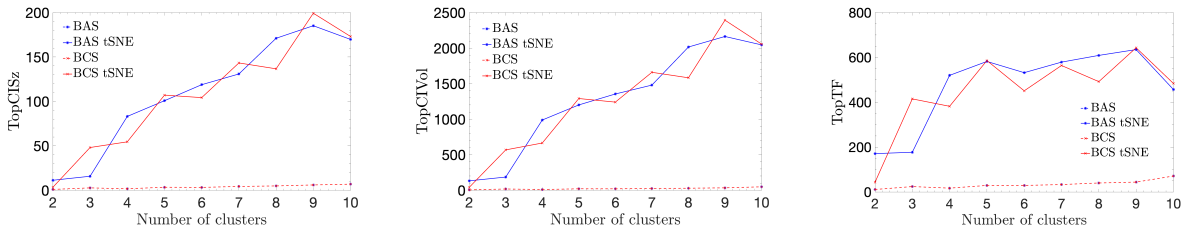


Figure 6.12: TopCI normalized by size and volume, TopTF obtained by clustering the AMLSim graph with 1000 nodes with BCS, BAS and their t-SNE variants.

7 | Conclusion

This thesis explores spectral clustering methods that respect the orientation of edges in directed graphs, and tests whether a light non-linear refinement based on t-SNE can make the resulting embeddings easier to separate in practice. The investigation considered five families of adjacency–based spectral pipelines (i.e. ASE, Hermitian, Skew–symmetric, BCS and BAS), kept their spectral cores unchanged and inserted the same refinement immediately before k-means. The central empirical finding is consistent across data regimes: mapping the spectral coordinates through t-SNE tightens same-community neighborhoods and increases the margin between groups, which in turn lowers errors in the classification on synthetic data and improves flow-based structure on real networks. This comes without changing the dominant computational cost of the pipelines, which is governed by the eigendecomposition/SVD step analyzed in Chapter 5. The additional $\mathcal{O}(\, n \log n)$ work of the Barnes–Hut variant is asymptotically negligible in dense settings.

On controlled benchmarks, DSBMs demonstrated that the refinement helps independently of the number of latent blocks given a sufficient number of communities, while LFR graphs showed that the gain persists under substantial inter-community mixing. On real-world data, the EmailEU networks confirmed that orientation-aware spectra, once refined, produce clearer bi-partitions aligned with communication flow, and AMLSim transaction graphs illustrated that tighter spectral neighborhoods translate into higher trade-flow and lower cut-imbalance for reasonable choices of k. Taken together, these results support a simple practical message: if a directed spectral method already captures the right local geometry, a compact two-dimensional map used as a staging area for k-means can turn a borderline separation into a robust one.

Computation was presented in the dense regime for clarity. While this matches the theoretical analysis, practical deployments on very large graphs will benefit from sparse storage and iterative solvers.

Future work A principled, fully automatic procedure for selecting k (e.g., eigengap) would make the pipelines ready to use. It would also be valuable to explore GPU-accelerated implementations of the proposed algorithms. Beyond these directions, two

alternative approaches which tackles the curse of dimensionality in the k-means algorithm deserve attention in our setting. in particular, we cite the Randomized Dimensionality Reduction approach in [47] and the complete review of k-means in [52]. Using directly this formulations could remove the t-SNE stage, maintaining only the modified k-means approach and leading to the same improvements we found with our approach.

In summary, the thesis contributes a unified, implementation-ready treatment of directed spectral clustering pipelines comprehensive with a minimal refinement that improves separability without altering asymptotic cost. The broader lesson is that small but carefully placed geometric adjustments can have a disproportionate impact on the final cluster recovery, provided the underlying spectral features faithfully encode the directionality of the network.

Bibliography

[1] Mihai Cucuringu, Xiaowen Dong, and Ning Zhang. Maximum likelihood estimation on stochastic blockmodels for directed graph clustering, 2024.

[2] Daniel L. Sussman, Minh Tang, Donniell E. Fishkind, and Carey E. Priebe and. A consistent adjacency spectral embedding for stochastic blockmodel graphs. *Journal of the American Statistical Association*, 107(499):1119–1128, 2012.

[3] Huan Li, Mihai Cucuringu, He Sun, and Luca Zanetti. Hermitian matrices for clustering directed graphs: insights and applications. *AISTATS*, 2020.

[4] Koby Hayashi, Sinan G. Aksoy, Haesun Park, and Haesun Park. Skew-symmetric adjacency matrices for clustering directed graphs. In *2022 IEEE International Conference on Big Data (Big Data)*. Pacific Northwest National Lab. (PNNL), Richland, WA (United States), 12 2022.

[5] Hadrien Van Lierde, Tommy W S Chow, and Jean-Charles Delvenne. Spectral clustering algorithms for the detection of clusters in block-cyclic and block-acyclic graphs. *Journal of Complex Networks*, 7(1):1–53, 05 2018.

[6] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[7] Laurens van der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, 15(93):3221–3245, 2014.

[8] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78:046110, Oct 2008.

[9] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 2018.

[10] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2017.

[11] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

[12] Toyotaro Suzumura and Hiroki Kanezashi. Anti-Money Laundering Datasets: InPlusLab anti-money laundering datadatasets. `http://github.com/IBM/AMLSim/`, 2021.

[13] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:5363–5370, 04 2020.

[14] Mark Weber, Jie Chen, Toyotaro Suzumura, Aldo Pareja, Tengfei Ma, Hiroki Kanezashi, Tim Kaler, Charles Leiserson, and Tao Schardl. Scalable graph learning for anti-money laundering: A first look, 11 2018.

[15] Dimosthenis Pasadakis, Matthias Bollhöfer, and Olaf Schenk. Sparse quadratic approximation for graph learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):11256–11269, 2023.

[16] Daron Acemoglu, Asuman Ozdaglar, and Alireza Tahbaz-Salehi. Systemic risk and stability in financial networks. Working Paper 18727, National Bureau of Economic Research, January 2013.

[17] Peter Hoff, Adrian Raftery, and Mark Handcock. Latent space approaches to social network analysis peter d. ho, adrian e. raftery, and mark s. handcock. *Journal of the American Statistical Association*, page 1090–1098, 12 2001.

[18] Stephen J. Young and Edward R. Scheinerman. Random dot product graph models for social networks. In *Proceedings of the 5th International Conference on Algorithms and Models for the Web-Graph*, WAW'07, page 138–149, Berlin, Heidelberg, 2007. Springer-Verlag.

[19] Avanti Athreya, Donniell E. Fishkind, Minh Tang, Carey E. Priebe, Youngser Park, Joshua T. Vogelstein, Keith Levin, Vince Lyzinski, and Yichen Qin. Statistical inference on random dot product graphs: a survey. *J. Mach. Learn. Res.*, 18(1):8393–8484, January 2017.

[20] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[21] Yuchung J. Wang and George Y. Wong. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82(397):8–19, 1987.

[22] James B McQueen. Some methods of classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symposium on Math. Stat. and Prob.*, pages 281–297, 1967.

[23] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[24] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[25] Ana L.N. Fred and Anil K. Jain. Robust data clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:II/128–II/133, 2003. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2003 ; Conference date: 18-06-2003 Through 20-06-2003.

[26] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[27] Nancy Chinchor. Muc-4 evaluation metrics. In *Proceedings of the 4th Conference on Message Understanding*, MUC4 '92, page 22–29, USA, 1992. Association for Computational Linguistics.

[28] Alexander J. Gates and Yong-Yeol Ahn. The impact of random models on clustering similarity. *bioRxiv*, 2017.

[29] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[30] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[31] Steinar Laenen. Directed graph clustering using hermitian laplacians. *Master's thesis, University of Edinburgh*, 2019.

[32] Dimosthenis Pasadakis, Raphael Steiner, Pál András Papp, Toni Böhnlein, and Albert-Jan N. Yzelman. Symmetry-breaking symmetry in directed spectral partitioning. *arXiv preprint arXiv:2508.16173*, 2025.

[33] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec 2007.

[34] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, Sep 1936.

[35] Jianxi Liu and Xueliang Li. Hermitian-adjacency matrices and hermitian energies of mixed graphs. *Linear Algebra and its Applications*, 466:182–207, 2015.

[36] Krystal Guo and Bojan Mohar. Hermitian adjacency matrix of digraphs and mixed graphs. *Journal of Graph Theory*, 489, 05 2015.

[37] Sergey Brin and Lawrence Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 56(18):3825–3833, 2012. The WEB we live in.

[38] Geoffrey E Hinton and Sam Roweis. Stochastic neighbor embedding. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002.

[39] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[40] Student. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908.

[41] Robert A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1(4):295–307, 1988.

[42] Josh Barnes and Piet Hut. A hierarchical o(n log n) force-calculation algorithm. *Nature*, 324(6096):446–449, Dec 1986.

[43] Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In Joan Cabestany, Alberto Prieto, and Francisco Sandoval, editors, *Computational Intelligence and Bioinspired Systems*, pages 758–770, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[44] Vishnu Vardhan Baligodugula and Fathi Amsaad. Unsupervised learning: Comparative analysis of clustering techniques on high-dimensional data. *arXiv preprint arXiv:2503.23215*, 2025.

[45] P. Drineas, Alan Frieze, Ravi Kannan, Santosh Vempala, and V. Vinay. Clustering in large graphs and matrices. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '99, page 291–299, USA, 1999. Society for Industrial and Applied Mathematics.

[46] William Johnson and Joram Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, 26:189–206, 01 1984.

[47] Christos Boutsidis, Anastasios Zouzias, Michael W. Mahoney, and Petros Drineas. Randomized dimensionality reduction for k -means clustering. *IEEE Transactions on Information Theory*, 61(2):1045–1062, 2015.

[48] Patrick Billingsley. *Convergence of probability measures*. Wiley Series in Probability and Statistics: Probability and Statistics. John Wiley & Sons Inc., New York, second edition, 1999. A Wiley-Interscience Publication.

[49] D. S. Choi, P. J. Wolfe, and E. M. Airoldi. Stochastic blockmodels with a growing number of classes. *Biometrika*, 99(2):273–284, 2012.

[50] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[51] Gene H. Golub and Charles F. Van Loan. *Matrix Computations - 4th Edition*. Johns Hopkins University Press, Philadelphia, PA, 2013.

[52] S.W. Qaiyumi and D.T. Mirikitani. An extension of the k-mean algorithm for dealing with dimensionality curse. In *2006 IEEE International Conference on Engineering of Intelligent Systems*, pages 1–5, 2006.