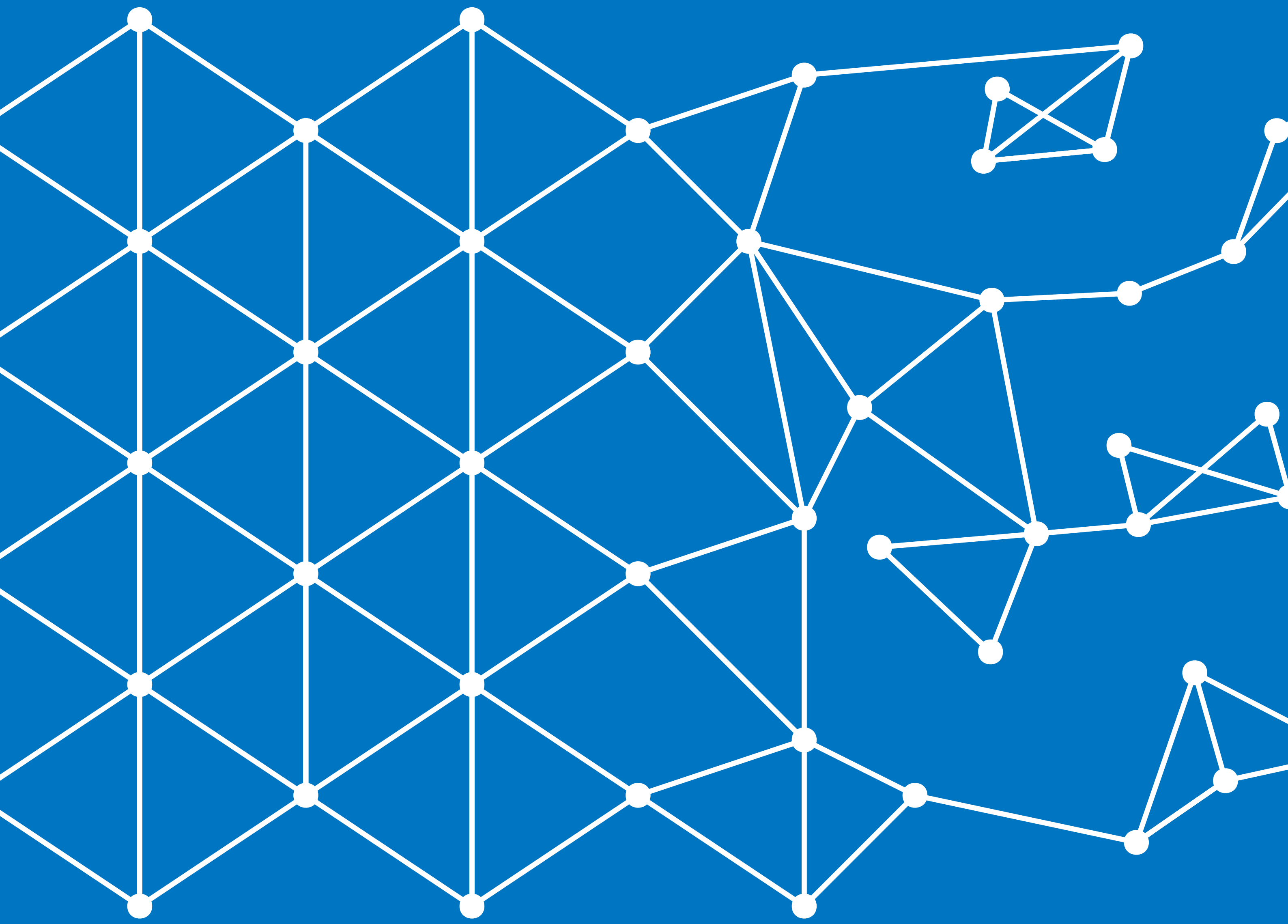
On Whole-Graph Embeddings from Node Feature Distributions

Triangle Count reveals Communities and improves Graph Neural Networks

Lourens Touwen



On Whole-Graph Embeddings from Node Feature Distributions

Triangle Count reveals Communities and improves Graph Neural Networks

by

Lourens Touwen

Student Number
5840848

MSc Thesis

Supervisor:	Dr. Júlia Komjáthy (Delft University of Technology)
Co-supervisor:	Prof. Paweł Prałat (Toronto Metropolitan University)
Date of defense:	July 2, 2024
Department:	Delft Institute of Applied Mathematics

Abstract

We consider three topics motivated by the **Ne**twork **Ex**ploration **T**oolkit (NEExT) for building unsupervised graph embeddings. NEExT vectorizes the graphs in a graph collection using the Wasserstein (optimal transport) distance between the distributions of node features of each graph. We inspect the effect of sampling only a proportion of the nodes of each graph, and show that even if the sampling fraction tends to zero (sufficiently slowly), so does the Wasserstein distance. NEExT relies on the assumption that local node features are informative to global characteristics of graphs. With this in mind, we consider triangle count: triangles are a local feature that correlate with the strength of a graph's community structure. We give asymptotic results for the number of triangles in a 2-community stochastic block model setting and the ABCD random graph model in both a scale-free and finite variance degree regime. Lastly, we show by experiment that augmenting the node features of graphs with triangle count improves a Graph Neural Network (GNN) in its ability to pick up on community structure and local clustering. For this, we use a random graph dataset with varying community structure, a random graph dataset with varying clustering coefficient, and a real-life dataset. We position random graph models as an effective tool for benchmarking the expressiveness of GNNs.

Acknowledgements

During the writing of this thesis, I had the great pleasure to spend five months at Toronto Metropolitan University. My time in Toronto has been rewarding, both academically and personally. I thank Paweł Prałat for his academic supervision, his efforts in financial arrangements, and him and the department as a whole for the warm and stimulating environment. Moreover, I would like to thank Nelly Litvak and Remco van der Hofstad for making the first connection and their recommendation.

Júlia Komjáthy has at all times been a constructive supervisor, and I am happy to have enjoyed her mentorship and her willingness to overcome timezone differences. I appreciate Ash Dehghan and François Théberge for allowing me to contribute to NEExT.

Many a time have I enjoyed the support and advice of the people around me. These include in Toronto, Austin Eide, Mateusz Zawisza, Meghan Koo and Emily Jin, over the phone, Timo Homma, Ece Kilinç, in the Netherlands, my housemates Niels Goedegebure, Karim Guettache, Pietro Pezzoli, and above all, my parents.

This project and my time abroad was financially supported by the Mitacs Globalink Research Award, the Nuffic NL Scholarship and the ASML Technology Scholarship.

Notation

Throughout this thesis, we adopt the following notation, unless specified otherwise.

For a set A we denote its volume by $\mathcal{A} = \sum_{a \in A} a$, and $\sum_{x,y \in A}$ denotes the *ordered* summation over all distinct elements $x, y \in A$, i.e. $\sum_{x,y \in A} \equiv \sum_{x \in A} \sum_{y \in A, y \neq x}$. We use $[n]$ as the shorthand notation for the set of indices $\{1, \ldots, n\}$.

For a random variable X, a fixed sample from the random variable is denoted by the lower case x. When two random variables are i.i.d. we mean them to be *independent and identically distributed.* A vector of i.i.d. samples is denoted by $\mathbf{x}$. Furthermore, we denote by $\overset{d}{=}$ equality in distribution.

The results in this thesis are asymptotic in nature, i.e. when the number of nodes $n \to \infty$. We let $\overset{\mathbb{P}}{\to}$ denote convergence in probability, i.e. $X_n \overset{\mathbb{P}}{\to} X$ means that for all $\varepsilon > 0$, $\lim_{n\to\infty} \mathbb{P}\left(|X_n - X| > \varepsilon\right) = 0$. For an event that happens *asymptotically almost surely (a.a.s)*, it holds that as $n \to \infty$, the probability of the event tends to 1. We say that a function $f(n) = \mathcal{O}(g(n))$ if there exists a constant c such that for all n it holds that $|f(n)| \leq c|g(n)|$. Similarly $f(n) = \Omega(g(n))$ denotes $g(n) = \mathcal{O}(f(n))$ and $f(n) = \Theta(g(n))$ if both $f(n) = \mathcal{O}(g(n))$ and $f(n) = \Omega(g(n))$. We write $f(n) = o(g(n))$ if $\lim_{n\to\infty} f(n)/g(n) = 0$.

Contents

1 Introduction

When describing the world, modelling interactions between entities are unavoidable. A webpage exists in the world wide web, an atom in a molecule, a city in a highway network. In the field of *graph theory*, a network of interactions is described by a *graph*: a collection of elements, called the *vertices* or *nodes*, and connections between them, called the *edges*. The vertices and edges can represent a wide range of things: papers and citations, countries and alliances, proteins and protein-binding, bank accounts and transactions. The graph structure can reveal properties of the systems they describe. For example, a network of fraudulent transactions may have a different structure from a network of legitimate activity. This brings us to the task of *graph classification*, which has the aim of predicting the whole-graph label of a given graph. Applications include molecular property prediction [87, 7], recommender systems [19] and online article classification [51].

A graph is at its core a high-dimensional object, as a graph with n vertices has on the order of n^2 possible edges. Because of this, a common approach in graph data mining is to consider *graph embeddings*. A graph embedding encodes a graph in a low-dimensional vector, with the aim of preserving or extracting characteristics of the graph. These embeddings can then be used for a downstream task such as visualization, clustering or graph classification. An embedding is necessarily lossy, meaning that they lose information, and what embedding methodology to use depends on the application and dataset. The **NE**twork **Ex**ploration **T**oolkit (NEExT [18] is a framework for generating unsupervised graph embeddings of collections of graphs, based on a powerful technique recently introduced in the literature [46, 66]. In NEExT, manually chosen node features (such as degree centrality, local clustering coefficient, or given node attributes) are used to generate the embeddings. This allows the user to manually select which node features to use, based on domain expertise or experimentation.

In this thesis, we are motivated by applications of NEExT. For large graphs, computing all node features can be computationally infeasible. NEExT offers the functionality to create graph embeddings using only a subset of the vertices. We investigate how this sampling impacts the accuracy of the framework. With NEExT in mind, we are interested in how the macrostructure of the graph influences the local (node) features, as NEExT uses the node feature distribution to generate the graph embedding. To this end, we investigate for two random graph models how the the number of triangles in a graph corresponds to the community structure. Lastly, we show with an exploratory experiment how augmenting nodes with their triangle count improves the power of Graph Neural Networks, a popular deep learning technique for graph classification. We will use the remainder of the introduction to place the topics of this thesis in the context of recent research about graph embeddings, sampling convergence, triangle count in random graph models and graph neural networks. We close this section by giving an overview of the structure of the thesis.

1.1 Graph Embeddings

Graph embeddings, also called *graph representations* [39], aim to encode high-dimensional graph-structured data in low-dimensional vectors. The term *graph embedding* is used in the literature to refer both to the vectorization of entire graphs as well as the individual vectorization of each node in a graph (see [43]). In this thesis, we call the latter *node embeddings*. We can broadly categorize graph embedding methods into statistical methods, graph kernels, and graph neural network based methods.

A straight-forward embedding method is to describe a graph by a vector of calculated metrics, such as the assortativity coefficient, the Pearson's correlation coefficient between degrees of two vertices, or statistics on the distribution of vertex features [3, 10, 58, 67]. If labels are available, *graph boosting*, an iterative procedure for counting relevant subgraphs [71, 60], can be used. These methods are highly interpretable, but have trouble distinguishing graphs from the same domain that are very similar.

In a Graph Neural Network (GNN), first introduced by in [31], nodes pass along their features with their neighbors in convolution-like layers to find learned node embeddings, as we will see later in the introduction in Section 1.4. For graph-level tasks, such as graph classification, the node embeddings are aggregated with a pooling operation to obtain a graph embedding. These methods have been applied extensively [82], for example with great in success in molecular property prediction [81]. In GNNs, graph embeddings are obtained as an intermediate step in a machine learning task, and are thus task-specific. Transfer learning, using the graph embeddings from one task on a different task, remains nascent in the graph data domain [75]. In the case of unlabelled graphs, recently self-supervised representation learning methods, such as graph contrastative learning [86], have gained popularity [50, 74, 77, 84]. Although effective in supervised tasks, these algorithms are "black box" and their learned graph embeddings are thus hard to interpret. On top of that, they require a labelled dataset, which makes them not suited for unsupervised tasks or data exploration.

Intimately related to graph embeddings are graph kernel methods, a popular approach for graph-level tasks. As the name suggests, the methodology relies on the formulation of a graph kernel function that measures the similarity of two graphs, used in a kernel machine such as a support vector machine. Many choices for this kernel function have been suggested in the literature [47], all with strengths and weaknesses, suitable for different tasks. In most cases, graph kernel methods do not generate explicit graph embeddings, although those can be obtained from the similarity matrix. Although NEExT shares characteristics with the graph kernel methods—as we will see, the Wasserstein distance between the graph's node feature distributions is a measure of graph similarity—it is unsupervised and agnostic to the downstream task. Quick iteration over different node features makes the framework suitable for data exploration, and interpretable results.

1.2 Sampling Node Features

For calculating graph embeddings, the NEExT framework relies on the Wasserstein distance between the distributions of node features on the different graphs in a collection, see Section 2. These distributions are discrete, with at most n point masses in $\mathbb{R}^d$, for a graph with n node and d node features. Calculating the node features for all nodes in large graphs may be computationally expensive. Therefore, we would like to consider only a sample of s nodes. We call this *proportion sampling.* We are then interested in how the Wasserstein distance $\mathcal{W}_2(\mu^{(n)}, \nu^{(m)})$ between the full distributions $\mu^{(n)}$ and $\nu^{(m)}$, corresponds to graphs of size n and m, differs from the Wasserstein distance $\mathcal{W}_2(\mu_s^{(n)}, \nu_s^{(m)})$ of the sampled distributions $\mu_s^{(n)}$ and $\nu_s^{(m)}$.

The version of this problem where the sample of s nodes is taken *without replacement* is directly considered by Fatras et al. [28] in the discussion of the Wasserstein convergence of minibatch sampling from two large empirical distributions. This can be seen as a generalization of proportion sampling by taking k not necessarily disjoint subsets of the point masses. They show asymptotic consistency, but use the trivial transport plan and no rate of convergence.

In sampling *with replacement* the point masses in $\mu^{(n)}$ might be sampled more than once. If we view $\mu^{(n)}$ as the empirical distribution of some underlying "true' feature distribution, then $\mu_s^{(n)}$ is the *bootstrap* (i.e. resampled) approximation (see [72]). The naive bootstrap of resampling $s = n$ is not consistent in the Wasserstein sense, see [23] and more recently [27]. However, when $s < n$, the bootstrap is consistent. Sommerfeld et al. [64] show that when $s/n \to 0$, then $\mathcal{W}_p(\mu_s^{(n)}, \nu_s^{(m)})$ is a consistent estimator for the true Wasserstein distance $\mathcal{W}_p(\mu^{(n)}, \nu^{(m)})$. Furthermore, proportion sampling, either with or without replacement, is a specific case of the exchangeable bootstrap, for which it can be shown that a large deviation principle holds [69].

More generally, one can consider the Wasserstein convergence of the empirical measure μ_s to a general μ, not necessarily finitely supported. If μ is absolutely continuous, then $\mathbb{E}[\mathcal{W}_2(\mu, \mu_s)] \leq s^{-1/d}$ [22]. This fact is called the *curse of dimensionality*: as the dimension increases, the rate of convergence decreases. When μ is not absolutely continuous, the Wasserstein convergence can be sharply bound by using the notion of the *inherent dimension* $d^*(\mu)$ of the measure μ (see [80]), such that $\mathbb{E}[\mathcal{W}_p(\mu, \mu_s)] < s^{-1/d^*(\mu)}$. In fact, if μ is supported on finitely many points, as is the case in for the probability measure over the node features, then the convergence is independent of the ambient dimension d. In our setting however, we care to find the rate of convergence as $s/n \to 0$. In other words, it is not trivial to see how $d^*(\mu^{(n)})$ increases as n grows, i.e. when considering a sequence of measures $(\mu^{(n)})_{n \geq 1}$. With this in mind, we show that for $s := n^\sigma, \sigma \in (0, 1)$ then $\mathcal{W}_p(\mu^{(n)}, \mu_s^{(n)})$ can be bounded from above by $\mathcal{O}(n^{-\sigma/(d+3p)+\varepsilon})$, for arbitrarily small $\varepsilon > 0$.

1.3 Quantifying Community Structure with Triangle Count

In Section 4, we give asymptotic results for the number of triangles (also called three-cycles or K_3 cliques) in two random graph models with community structure: the stochastic block model [35] and the ABCD model [41]. The stochastic block model is a simple model for random graphs with different communities (blocks). The ABCD model is more involved, but crucially has the capacity to generate *scale-free* graphs with a community structure. In a scale-free network, the degree distribution of the graph follows a power law, such that the proportion of vertices with k or more degrees p_k follows $p_k \sim k^{-\gamma}$, for some $2 < \gamma < 3$. It is a common feature of real-life networks and much studied [4]. Both models have parameters that influence the strength of the community structure. A popular way in the literature of quantifying the strength of the community structure is through *modularity* [54], where high modularity means that a high proportion of edges fall within communities. Strictly, consider a partition of the vertices, and calculate the fraction of edges within parts minus the expected number of edges in a part, if the edges were distributed at random. Modularity is then defined as the maximum of this quantity over all partitions. Considering all partitions is computationally expensive, so the heuristic approach is to use a community detection algorithm, such as the Louvain or Leiden algorithm [68], to find a lower bound on the modularity of a graph. The parameters of the models that influence the modularity can be viewed as *global* features of a given random graph: they influence the macro structure. Calculating the modularity also requires a global approach that considers the whole graph, and even for the heuristic approach this leads to computational challenges for large graphs.

Motivated by the NEExT methodology, we care about local (vertex) features and their relation to global characteristics. Using local features to create whole-graph embeddings relies on the assumption that local features are meaningful to global characteristics. To support this, we analyze triangles. Triangles in graphs are a simple motif that occur when two vertices in a vertex's neighborhood connect. Triangles are a building block for many network statistics that aim to quantify the local clustering in a network, such as the local clustering coefficient, given by the ratio of realized to unrealized triangles in a vertex's neighborhood [36]. Moreover, triangle-count can be used for community detection [56]. In a graph with a strong community structure, we expect to find more triangles. Triangles are a small subgraph. The study of the distribution of subgraphs in random graphs has a rich history, starting in the Erdős–Rényi random graph [26], and has since evolved into a large field [30, 65, 11, 73]. We contribute to the literature by providing results for the recently introduced ABCD model [41].

1.4 Triangle Augmentation for Graph Neural Networks

After analyzing the triangle count as a quantification for the strength of the community structure, we consider how the triangle count can improve Graph Neural Networks expressiveness. Graph Neural Networks (GNNs) [82] have achieved state of the art performance in many graph data mining tasks. However, it can be shown that their expressiveness is at most that of the 1-Dimensional Weisfeiler-Lehman test for graph isomorphism [53]. As a consequence, GNNs cannot detect clustering coefficients [85], count any subgraph of three or more nodes [12], or recover community structure [70]. We confirm this with an experiment on two synthetic graph datasets, one with varying strenght of community structure and one with varying clustering coefficient. Recently, the literature has suggested augmenting graphs with subgraph counts improves the GNNs expressiveness [33, 8, 15]. We show that the triangle count is sufficient for our datasets, and improves performance on a real-life dataset.

1.5 Structure of thesis

In Section 2, the NEExT framework for creating unsupervised graph embeddings is explained, including the motivation and details of the choice of the distance distribution: the approximate Wasserstein distance. To show the expressiveness of the NEExT embeddings, we do a simulation experiment with graphs generated using graphons.

In Section 3, we inspect the effect of considering only a proportion of nodes on the Wasserstein distance between the node feature distributions. We show that even if the sampling fraction tends to zero (sufficiently slowly), so does the Wasserstein distance.

In Section 4, we focus on counting triangles in two random graph models. We are motivated by NEExT's underlying assumption that node features are meaningful for analyzing graph-level features. Triangles are a local feature that correlate with the strength of a graph's community structure. We introduce and give asymptotic results for the number of triangles in a 2-community stochastic block model setting and the ABCD model in both a scale-free and finite variance degree regime.

Lastly, in Section 5 we show that augmenting graphs with local triangle counts improves the popular class of Graph Neural Network algorithms in their ability to pick up on community structure and local clustering. We show this for two random graph models, one with varying community structure and one with varying clustering coefficient, and a real-life dataset.

2 Unsupervised Graph Embeddings with NEExT

In this section we describe the NEExT framework for building unsupervised graph embeddings based on node features [18]. The implementation[1] makes the steps for generating graph embeddings quick and easy. The steps are as follows.

1. Consider a collection of graphs $G_1, G_2, \ldots, G_m$, with G_i having n_i vertices, for $i \in [m]$. Let $\mathcal{G}$ denote the space of all possible graphs.

2. Choose some node embedding function $h : \mathcal{G} \mapsto \mathbb{R}^{n_i \times k}$ with node embedding dimension k. The choice of h is flexible, see Section 2.1.

3. Let μ_i be the empirical probability measure on the node embedding space induced by the rows of $h(G_i)$.

4. Consider a notion of distribution distance between (μ_i, μ_j) for $i, j \in [m]$, relying on a projection $\phi(\mu_i)$ of μ_i to a reference distribution μ_0, such that $\|\phi(\mu_i) - \phi(\mu_j)\|_2$ approximates the Wasserstein distance between μ_i and μ_j, see Section 2.2.

5. Then, let the similarity matrix $S \in \mathbb{R}^{m \times m}$ be given $S_{i,j} = \|\phi(\mu_i) - \phi(\mu_j)\|_2$.

6. Finally, obtain a d-dimensional ($d \leq m$) embedding by applying reduced-rank singular value decomposition (SVD) to the similarity matrix S.

2.1 Node embeddings

One of the strengths of NEExT is that it is flexible with respect to the choice of the node embedding function h. How h vectorizes the nodes can be adapted for the application, based on domain expertise or experimentation. If a graph is augmented with inherent node attributes, these are an obvious choice for the vectorization. If the attribute is categorical, the feature can be one-hot encoded. NEExT also offers a range of structural features, such as PageRank [9], community-aware features [42], or simply the degree. For further granularity, every node feature can be *expanded* over the egonet of the node. Consider some node feature x_i for $i \in V$, and denote by $\bar{x}(N)$ the average value of x over some set of vertices N, i.e $\bar{x}(N) = \frac{1}{|N|}\sum_{i \in N} x_i$. Then the r-th expansion $E_r^x(v)$ of feature x on vertex v is then

$$E_r^x(v) = \left(x_v, \bar{x}(N_1), \ldots, \bar{x}(N_r)\right). \tag{1}$$

where N_k denote the sets of vertices that are exactly k steps away from v.

One important requirement for the node embedding is that the embedding space is *comparable* between different graphs. This excludes learned representation methods such as the popular node2vec [32] or struct2vec [57].

[1] https://pypi.org/project/NEExT/

2.2 Approximate Wasserstein distance

As mentioned above, the NEExT framework relies on the notion of a distance between distributions. In this section we explain and motivate the approximate Wasserstein distance. Various other distribution distances exist, such as the *total variation distance* [34, eq. 2.2.4]. For probability measures μ and ν on $(\Omega, \mathcal{A})$, the total variation distance is given by

$$TV(\mu, \nu) = \sup_{A \in \mathcal{A}} \left| \mu(A) - \nu(A) \right| .$$

Although powerful for many theoretical applications, it has practical limitations. It only considers the maximum disagreement between the measures, and does not take into account the underlying metric space of the measures. Take, for example, the probability measures μ, ν, ω on $\mathbb{R}$ given by $\mu = \delta(1), \nu = \delta(2)$ and $\omega = \delta(1000)$. Here δ denotes the Dirac measure. Although μ and ν place the mass close together, whereas ω places it far away, the total variation distance $TV(\mu, \nu)$ is equal to $TV(\mu, \omega)$.

Another notion of distribution distance is Kullback-Leibler divergence, or relative entropy [48]. For μ absolutely continuous with respect to ν, the KL-divergence is given by

$$KL(\mu, \nu) = \int \log \frac{d\mu}{d\nu} d\mu.$$

However, KL-divergence is not symmetric and does not satisfy the triangle inequality. Although it encapsulates a notion of "distance", it is not a metric. In addition, like total variation, it does not take the underlying metric space of the measures into account. Instead, NEExT uses an approximation of the *Wasserstein distance*. Also called the Monge-Kantorovich-Rubinstein distance, Kantorovich distance, Mallows distance, earth-mover's distance, or optimal transport (OT) distance [64], the Wasserstein distance relies on the concept of the Optimal Transport plan between two probability measures.

Definition 2.1 (Transport plan). *Let μ be a probability measure on a measurable space $\mathcal{Z}$ and ν be a probability measure on $\mathcal{Z}'$. The probability measure γ on $\mathcal{Z} \times \mathcal{Z}$ is a transport plan when its marginals are μ and ν, i.e. for every $A \subset \mathcal{Z}$, $\gamma(A \times \mathcal{Z}') = \mu(A)$ and for every $B \subset \mathcal{Z}'$, $\gamma(\mathcal{Z} \times B) = \nu(B)$.*

The Wasserstein distance between two probability measures is then defined as the "cost" of the optimal transport plan, i.e. the transport plan with the least cost over all possible transport plans. An intuitive explanation is that of the earth mover's distance [59]: given two piles of the same amount of dirt (the probability measures) the Wasserstein distance is the minimal amount of dirt you have to move on one of the piles to make the piles identical. For example, in Figure 1, the mass on $(4, 4)$ in (a) needs to be moved to $(3, 4)$ in (b), i.e. a distance of 1. Between (a) and (c), the distribution distance is larger, as mass has to be moved from $(4, 4)$ to $(1, 1)$, a distance of $\sqrt{18}$. In full generality, this can be formalized for any metric space. Here, we consider the Wasserstein distance for probability measures on $\mathbb{R}^d$, equipped with the Euclidean distance.

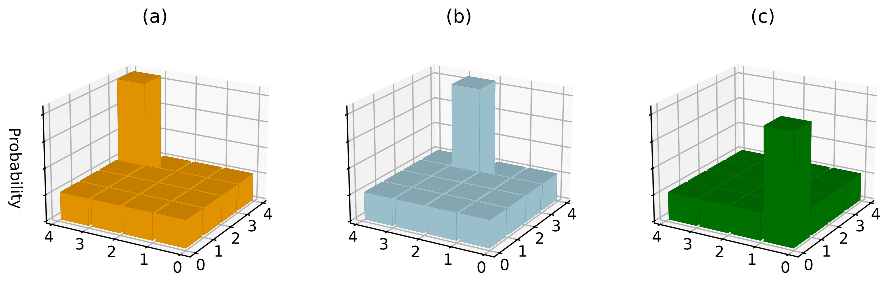


Figure 1 – The Wasserstein distance between (a) and (b) is less than the Wasserstein distance between (a) and (c).

Definition 2.2 (p-Wasserstein distance)**.** *Let μ be a probability measure on $\mathcal{Z}$ and ν be a probability measure on $\mathcal{Z}'$, with $\mathcal{Z}, \mathcal{Z}' \subset \mathbb{R}^d$. Let $\Gamma(\mu, \nu)$ be the set of all transport plans between μ and ν. For $p \in [1, \infty]$ the p-Wasserstein distance is given by*

$$\mathcal{W}_p(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu,\nu)} \left(\int_{\mathcal{Z} \times \mathcal{Z}'} \|z - z'\|^p d\gamma(z, z') \right)^{1/p}.$$

The p-Wasserstein distance is symmetric and can be shown to satisfy the triangle inequality [14]. In the NEExT framework, the 2-Wasserstein distance is used. Finding to optimal transport plan to calculate the Wasserstein distance between two discrete distributions can be formulated as a linear program, and solved in polynomial time by the Simplex algorithm [17]. Still, as the graphs and the number of graphs the graph collection grow large, calculating the pairwise distance becomes computationally prohibitive. With that in mind, we consider the approximate Wasserstein distance between the distributions, by using a reference distribution. This reduces the number of optimal transport calculations from $m(m-1)$ to m (recall that m is the number of graphs in the collections). A reference probability measure μ_0, defined on $\mathcal{Z} \subset \mathbb{R}^d$, can be obtained by taking k-means clustering on $\bigcup_{i=1}^m h(G_i)$ with $N = \lfloor \frac{1}{m} \sum_{i=1}^m n_i \rfloor$ centroids [46], or the Wasserstein barycenter (see [16]). We then define

$$\phi(\mu_i) := (f_i - id)\sqrt{p_0}, \tag{2}$$

where $id(z) = z$ is the identity, $p_0(z)$ is the probability density function of μ_0 such that $d\mu_0(z) = p_0(z)dz$ and $p_0(z) > 0, \forall z \in \mathcal{Z}$, and $f_i(z)$ is the optimal transport plan between μ_i and μ_0 (also called Monge map when μ_i and μ_0 are discrete), i.e.

$$f_i = \operatorname*{argmin}_{f \in \Gamma(\mu_0, \mu_i)} \int_{\mathcal{Z}} \|z - f(z)\|^2 d\mu_0(z). \tag{3}$$

This provides an approximately *isometric embedding* for $\{\mu_i\}_{i=1}^m$, meaning that ϕ is approximately distance-preserving: $\|\phi(\mu_i) - \phi(\mu_j)\|_2 \approx \mathcal{W}_2(\mu_i, \mu_j)$. See Figure 2 for an

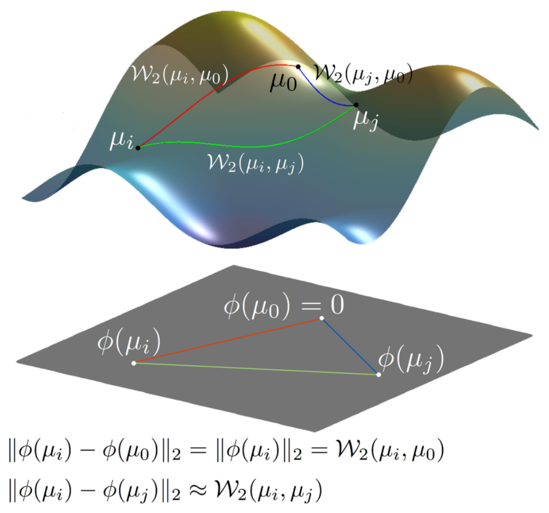

$\mathcal{W}_2(\mu_i, \mu_0)$
μ_0
$\mathcal{W}_2(\mu_j, \mu_0)$
μ_j
μ_i
$\mathcal{W}_2(\mu_i, \mu_j)$
$\phi(\mu_0) = 0$
$\phi(\mu_i)$
$\phi(\mu_j)$
$\|\phi(\mu_i) - \phi(\mu_0)\|_2 = \|\phi(\mu_i)\|_2 = \mathcal{W}_2(\mu_i, \mu_0)$
$\|\phi(\mu_i) - \phi(\mu_j)\|_2 \approx \mathcal{W}_2(\mu_i, \mu_j)$

Figure 2 – Illustration of approximate Wasserstein, adapted from [46].

illustration of this concept. Since the optimal transport f_i of μ_0 to μ_0 is the identity, we have $\phi(\mu_0) = (z - z)\sqrt{p_0(z)} = 0$. Also, we can see that

$$\begin{aligned}
\|\phi(\mu_i)\|_2 &= \|(f_i - z)\sqrt{p_0}\|_2 \\
&= \left(\int_{\mathcal{Z}} \|(f_i(z) - z)\sqrt{p_0(z)}\|^2 dz \right)^{\frac{1}{2}} \\
&= \left(\int_{\mathcal{Z}} \|(f_i(z) - z)\|^2 p_0(z) dz \right)^{\frac{1}{2}} \\
&= \left(\int_{\mathcal{Z}} \|(f_i(z) - z)\|^2 d\mu_0(z) \right)^{\frac{1}{2}} = \mathcal{W}_2(\mu_i, \mu_0).
\end{aligned}$$

By an appropriate choice of the reference distribution, it indeed follows that

$$\begin{aligned}
\|\phi(\mu_i) - \phi(\mu_j)\|_2 &= \left(\int_{\mathcal{Z}} \|(f_i(z) - z)\sqrt{p_0} - (f_j(z) - z)\sqrt{p_0}\|^2 dz \right)^{\frac{1}{2}} \\
&= \left(\int_{\mathcal{Z}} \|f_i(z) - f_j(z)\|^2 d\mu_0(z) \right)^{\frac{1}{2}} \approx \mathcal{W}_2(\mu_i, \mu_j).
\end{aligned}$$

NEExT uses the implementation of the approximate Wasserstein from the python VECTORIZERS package[2].

[2]https://pypi.org/project/vectorizers/

2.3 Graphon Simulation Experiment

To demonstrate the NEExT framework, we do a simulation experiment with graphons. Graphons are the natural limit object of a sequence of exchangeable random graph models [21]. A random graph model is exchangeable if its probability distribution is invariant under permutations of the vertices. A graphon is defined as the symmetric function $W : [0,1]^2 \mapsto [0,1]$. The graphon defines exchangable random graph models by assigning each vertex i an i.i.d value $u_i \sim \mathit{Unif}[0,1]$, and including each edge (i,j) independently with probability $W(u_i,u_j)$. For graph sequences, W can be seen as the probabilistic limit of the adjacency matrix. A graphon offers a natural way of simulating graph sequences $G_1,\dots,G_n$. To generate G_n with n vertices, simply sample i.i.d $u_1,\dots u_n$, such that $u_i \sim \mathit{Unif}[0,1]$, and include the edge with probability $W(u_i,u_j)$. The graph sequence $G_1,\dots,G_n$ then consists of independent graphs that for large n represent the graphon limit.

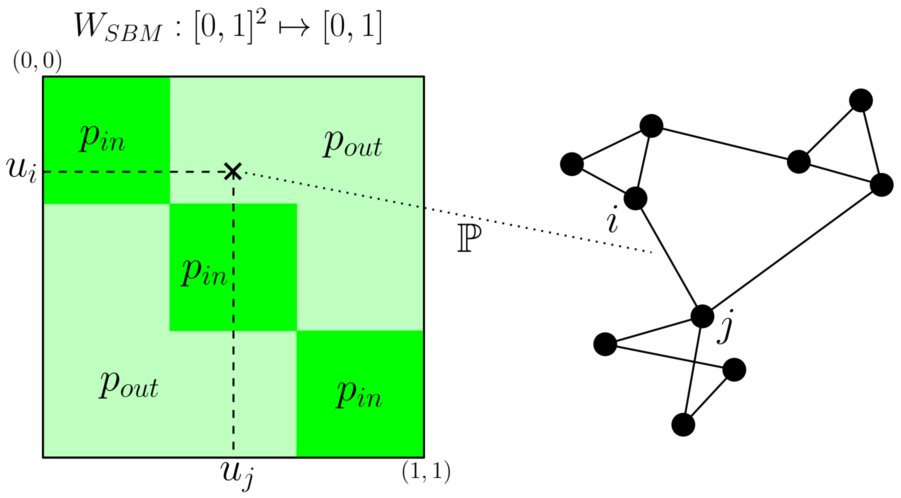


Figure 3 – Schematic illustration of how the Stochastic Block Model graphon W_{SBM} gives edge probabilities.

In this experiment, we aim to evaluate NEExT's graph embeddings with graphon generated graph sequences. As the underlying graph sequences converge to the graphon, we expect informative graph embeddings to converge too. We consider two different graphons, corresponding to the Erdős–Rényi random graph and the stochastic block model. In the Erdős–Rényi random graph, each edge exists independently with probability p. The corresponding graphon is the constant function $W_{ER}(x,y) = p$. Here we use $p = 0.2$. The stochastic block model is a generalization of the Erdős–Rényi random graph, with edge probabilities depending on a node community partition. For more details see Section 4.1. Here, we consider the stochastic block model with a partition of three blocks, with $p_{in} = 0.2$ and $p_{out} = 0.05$. The corresponding graphon, visualized in

Figure 3, is given by

$$W_{SBM}(x,y) = \begin{cases} & x \in [0,0.3) \wedge y \in [0,0.3), \\ 0.2, & x \in [0.3,0.6) \wedge y \in [0.3,0.6), \\ & x \in [0.6,1] \wedge y \in [0.6,1], \\ 0.05, & \text{otherwise.} \end{cases} \tag{4}$$

For both W_{ER} and W_{SBM} we generate a graph sequence with graphs of increasing sizes $5, 10, \ldots, 245, 250$. We find graph embeddings for the graphs in these sequences using NEExT and the node feature *eigenvector centrality*, a measure of a node's connectedness within the network. For a graph G of size n with adjacency matrix A, let λ denote the largest eigenvalue of A. The eigenvalue centrality $\mathbf{x} = (x_1, \ldots x_n)^T$ is defined by $A\mathbf{x} = \lambda\mathbf{x}$. In other words, the eigenvalue centrality is the eigenvector corresponding to the largest eigenvalue of the adjacency matrix. We take the 2-expansion, following (1), and using the approximate Wasserstein distance, calculate graph embeddings in $\mathbb{R}^2$ with NEExT as described at the start of this section. As visualized in Figure 4, we see indeed that the graph embeddings do converge as expected. We also note that there is no interpretation to the embeddings dimension. For more extensive random graph experiments with NEExT, we refer to [18].

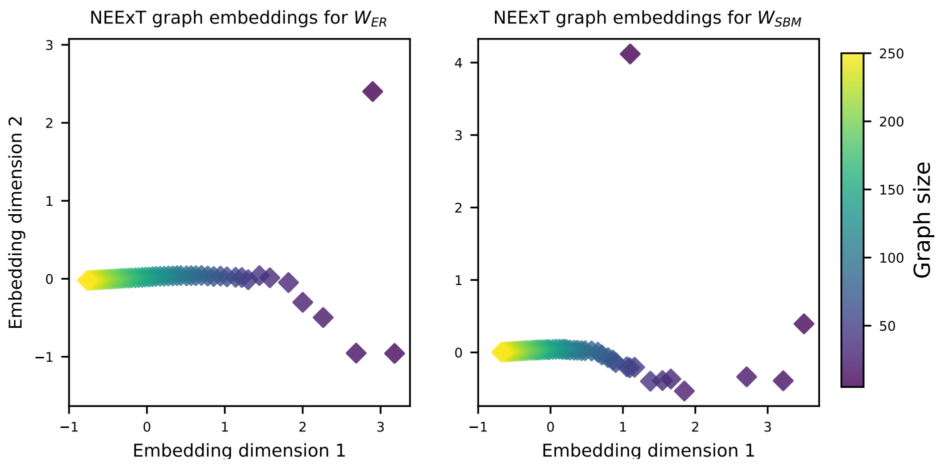


Figure 4 – NEExT embeddings from graphon-generated graph sequences, using eigenvalue centrality with 2-expansion as node feature, standardized to have expectation 0 and standard deviation 1. Both plots show the embeddings for a collection of 50 graphs of sizes $5, 10, \ldots, 245, 250$. The color indicates the graph size. In the plot on the left, the graphs have been generated from the graphon $W_{ER}(x,y) = 0.2$. On the right, the graphs have been generated from the graph W_{SBM} as given in (4). We also note that there is no interpretation to the embeddings dimension.

3 Sampling Node Features

As the graphs in a graph collection grow large, it becomes computationally expensive to calculate structural node features for all nodes in a graph. This can be addressed by only calculating the features for a sample of the nodes in a graph, at the risk of losing accuracy. In this section, we are interested in quantifying the effect of what we call *proportion sampling*: sampling only $s(n) < n$ of the n points in the population. In our result, if the *sampling fraction* $s(n)/n \to 0$ sufficiently slowly, so does the Wasserstein distance.

3.1 Preliminaries

A standard probabilistic tool we use is the Chernoff bound, for which we state an adapted form for binomial random variables. The Chernoff bound on X can be seen as application of Markov's inequality on e^{tX}.

Theorem 3.1 (Markov's inequality)**.** *For an integrable non-negative random variable X it holds that*

$$\mathbb{P}\left(X \geq a\right) \leq \frac{\mathbb{E}[X]}{a}.$$

Proof. Let $f(x)$ denote the density of X. Then

$$a\mathbb{P}\left(X \geq a\right) = a\mathbb{E}\big[\mathbb{I}_{X\geq a}\big] \leq \mathbb{E}\big[X\mathbb{I}_{X\geq a}\big] \leq \mathbb{E}[X]. \qquad \square$$

We now state the Chernoff bound for binomial random variables.

Theorem 3.2 (Chernoff bound for binomials)**.** *Let $X \sim \operatorname{Bin}(n,p)$ be a binomial random variable, and write $\mu = \mathbb{E}[X] = np$. Then*

$$\begin{aligned}
\mathbb{P}(X \geq \mu + t) &\leq \exp\left(-\tfrac{t^2}{2(\mu+t/3)}\right), \quad && t \geq 0,\\
\mathbb{P}(X \leq \mu - t) &\leq \exp\left(-\tfrac{t^2}{2\mu}\right), && t \geq 0.
\end{aligned}$$

Proof. We give a short proof, for more details, see Hofstad [34, Theorem 2.21] for a slightly more general case of independent but not identically distributed Bernoulli's. Note first that $\mathbb{P}\left(X \geq \mu + t\right) = 0$ for $t > n - \mu$, and $\mathbb{P}\left(X \leq \mu - t\right) = 0$ for $t > \mu$. For $0 \leq t \leq n - \mu$, and any $a > 0$, we see by applying the Markov inequality (Theorem 3.1), and the moment generating function of a binomial random variable, that

$$\mathbb{P}\left(X \geq \mu + t\right) = \mathbb{P}\left(e^{aX} \geq e^{a(\mu+t)}\right) \leq e^{-a(\mu+t)}\mathbb{E}\Big[e^{aX}\Big] = e^{-a(\mu+t)}(1 + (e^a - 1)p)^n.$$

This term takes a minimum over a for $e^a = ((\mu + t)(1 - p)/(n - \mu - t)p)$, which gives

$$\mathbb{P}\left(X \geq \mu + t\right) \leq \left(\frac{\mu}{\mu+t}\right)^{\mu+t}\left(\frac{n-\mu}{n-\mu-t}\right)^{n-\mu-t} \leq \exp\left(-\mu\phi\left(\frac{t}{\mu}\right)\right),$$

for $\phi(x) = (1 + x)\log(1 + x) + x$. It is not hard to show that if $x \geq 0$ then $\phi(x) \geq x^2/(2(1 + x/3))$. This shows

$$\mathbb{P}\left(X \leq \mu + t\right) \leq \exp\left(-\mu \frac{t^2}{2\mu^2(1 + \frac{t}{3\mu})}\right) = \exp\left(-\frac{t^2}{2(\mu + t/3)}\right).$$

We can find the lower bound by replacing X with $X - n$, and using that for $x \in [-1, 0]$ it holds that $\phi(x) \geq x^2/2$. So we have

$$\mathbb{P}\left(X \leq \mu - t\right) \leq \exp\left(-\mu\phi\left(-\frac{t}{\mu}\right)\right) \leq \exp\left(-\frac{t^2}{2\mu}\right),$$

which completes the proof. □

Another common tool is the union bound, also known as Boole's inequality, which is a direct consequence of the subadditivity of a probability measure.

Lemma 3.3 (Union Bound)**.** *In the probability space* $(\Omega, \Sigma, \mathbb{P})$*, let* $A_1, \ldots, A_n$ *be events in* Σ*. Then it holds that*

$$\mathbb{P}\left(\bigcup_{i=1}^{n} A_i\right) \leq \sum_{i=1}^{n} \mathbb{P}\left(A_i\right).$$

3.2 Proportion Sampling

Consider a graph G_n of size n, and some node vectorization function, as described in Section 2.1. We say that each node $i \in [n]$ is mapped to a d-dimensional vector $x_i \in \mathbb{R}^d$. We can assume that the collection of node embeddings $\mathbf{X}^{(n)} = \{x_1, \ldots, x_n\}$ is normalized[3], so that for each dimension the minimum is 0 and the maximum is 1. In other words, $\mathbf{X}^{(n)} \subset [0, 1]^d$. The set of points $\mathbf{X}^{(n)}$ *induces* a probability measure $\mu^{(n)}$ taking support $[0, 1]^d$.

Definition 3.1 (Induced Probability Measure)**.** *For a finite set* $Y \subset [0, 1]^d$ *with cardinality* $|Y| = n$*, the probability measure* $\nu : \mathcal{B}([0, 1]^d) \mapsto [0, 1]$ *induced by* Y *is given by*

$$\nu = \sum_{y \in Y} \frac{1}{n}\delta_y, \tag{5}$$

where δ_y *is the Dirac measure, and* $\mathcal{B}([0, 1]^d)$ *denotes the Borel* σ*-algebra of* $[0, 1]^d$.

We are interested in seeing how the probability measure $\mu^{(n)}$ differs in terms of the Wasserstein distance when we consider only a random subset of $\mathbf{X}^{(n)}$. Specifically, we

[3]It should be noted that in the NEExT framework, features are *standardized* to have mean 0 and standard deviation 1, a standard preprocessing step that is improves algorithmic stability and performance with multiple features.

care what happens when $n \to \infty$. For this, we want to consider a sequence of sets with increasing cardinality.

Definition 3.2. *Let Ξ_n be the set of all finite subsets of the hypercube $[0,1]^d$ of size n, i.e for $n \in \mathbb{N}^+$,*

$$\Xi_n = \left\{ Y \subset [0,1]^d \mid |Y| = n \right\}.$$

With this, we can state the following theorem.

Theorem 3.4 (Proportion sampling $\mathcal{W}$-Convergence)**.** *Let $(\mathbf{X}^{(n)})_{n\in\mathbb{N}^+}$ be any sequence of sets such that for all n, $\mathbf{X}^{(n)} \in \Xi_n$. Let $\mathbf{X}_s^{(n)}$ be a collection of $s := s(n)$ randomly sampled (with or without replacement) points from $\mathbf{X}^{(n)}$. Let $\mu^{(n)}$ be the probability measure induced by $\mathbf{X}^{(n)}$, and let $\mu_s^{(n)}$ be the probability measure induced by $\mathbf{X}_s^{(n)}$. Let $s(n) = \lfloor n^\sigma \rfloor$ for some $\sigma \in (0,1)$ and fix $\varepsilon > 0$. Then, for $p \geq 1$,*

$$\mathcal{W}_p(\mu^{(n)}, \mu_s^{(n)}) = \mathcal{O}\left(\frac{1}{n^q}\right),$$

where

$$q = \frac{\sigma}{d + 3p} - \varepsilon.$$

This inverse relation between d and the rate of convergence is also called the *curse of dimensionality.* Moreover, the results says that as $n \to \infty$ even as the sampling fraction s/n tends to zero sufficiently slowly, so does the Wasserstein distance.

Proof of Theorem 3.4. Our goal is to give a transport plan, not necessarily optimal, between $\mu_s^{(n)}$ and $\mu^{(n)}$. The cost of this transport plan will then provide an upper bound on $\mathcal{W}_p(\mu^{(n)}, \mu_s^{(n)})$. To give a transport plan and calculate its cost, our approach is to partition the hypercube up in a number of boxes, and consider the boxes "heavy" when they contain many points. Then we show 1) that $\mu_s^{(n)}$ places enough mass in these boxes to match with the points from $\mu^{(n)}$ in that box, and 2) that almost all of the points of $\mu^{(n)}$ are in these heavy boxes. The transport plan then consists of moving the mass within these heavy boxes (at most the box-diameter) and moving the mass to the few points that are not in heavy boxes.

We partition the hypercube into $k := k(n)$ rectangular boxes $B_1, \ldots, B_k$. Then each box has sides of length $k^{-1/d}$ and diameter $\sqrt{d} \cdot k^{-1/d}$. For $i \in [k]$, we call n_i be the number of points from $\mathbf{X}^{(n)}$ that fall in box B_i, and call s_i the number of points $\mathbf{X}_s^{(n)}$ in box B_i.

The probability measure $\mu_s^{(n)}$ places a mass of $1/s$ on s points in $\mathbf{X}_s^{(n)}$, which have been randomly sampled either with or without replacement from $\mathbf{X}^{(n)}$. So, $s_i \sim \mathit{Bin}\left(s, n_i/n\right)$ if we sample with replacement, or $s_i \sim \mathit{Hypergeometric}\left(n, n_i, s\right)$ if sampled without replacement.

Now, we call a box *heavy* if $\mu^{(n)}$ places "a lot" of mass on it. The precise requirement is that $n_i s/n \geq g$ for some $g := g(n)$, which we specify later. We let $H := \{i : n_i s/n \geq g, i \in [k]\}$ denote the set of heavy boxes. To ensure that there is at least one heavy box, we employ the Pigeon Hole principle: we know that there exists at least one box with $n_i \geq \lceil \frac{n}{k} \rceil$, so if we ensure that

$$g(n) \leq s(n)/k(n), \tag{6}$$

it is guaranteed that there is at least one heavy box. Our aim is now to show that under asymptotically almost all realisations of $\mu_s^{(n)}$, the heavy boxes receive the same mass as under $\mu^{(n)}$, except for a decreasing fraction. To find the optimal fraction, we introduce $h(n)$ that we will specify later, such that

$$g(n) \gg h(n), \quad h(n) \gg \log k(n). \tag{7}$$

We now show that asymptotically almost surely all heavy boxes have enough mass, i.e that

$$\mathbb{P}\left(\bigcup_{i\in F} s_i \leq \frac{n_i s}{n}\left(1 - \sqrt{\frac{h}{g}}\right)\right) \to 0. \tag{8}$$

To see this, apply first the Union bound (Lemma 3.3).

$$\mathbb{P}\left(\bigcup_{i\in H} s_i \leq \frac{n_i s}{n}\left(1 - \sqrt{\frac{h}{g}}\right)\right) \leq \sum_{i\in H}\mathbb{P}\left(s_i \leq \frac{n_i s}{n}\left(1 - \sqrt{\frac{h}{g}}\right)\right) \tag{9}$$

$$\leq \sum_{i\in H}\mathbb{P}\left(s_i \leq \frac{n_i s}{n}(1-\epsilon_i)\right). \tag{10}$$

Here $\epsilon_i = \sqrt{h/\frac{n_i s}{n}}$, and the inequality in (10) holds, as in heavy boxes $\frac{n_i s}{n} \geq g \gg h$. Furthermore $\epsilon_i < 1$, so we can apply the Chernoff bound (Theorem 3.2) for binomials. To see that this also holds when sampling without replacement, note that the hypergeometric distribution is more concentrated around its expectation than the binomial distribution, and a strictly sharper bound as in Theorem 3.2 holds. For more details, see [62].

$$\sum_{i\in H}\mathbb{P}\left(s_i \leq \frac{n_i s}{n}(1-\epsilon_i)\right) \leq \sum_{i\in H}\exp\left(-\frac{\epsilon_i^2 n_i s}{2n}\right) \tag{11}$$

$$= \sum_{i\in H}\exp\left(-\frac{h}{2}\right) \leq k\exp\left(-\frac{h}{2}\right) \to 0. \tag{12}$$

The last step follows from $h \gg \log k$, as assumed in (7). So this shows that a.a.s.

$$s_i \geq \frac{n_i s}{n}\left(1 - \sqrt{\frac{h}{g}}\right), \quad \forall i \in H.$$

As every point in $\mu_s^{(n)}$ has $1/s$ of mass, this means that a heavy box has a.a.s enough mass to match $\mu^{(n)}$, minus a fraction $\sqrt{h/g}$.

We call a box *light* if it is not heavy. Then $\frac{n_i s}{n} < g$, so we can upperbound the number of points in light boxes by

$$\sum_{i\in[k]\setminus H} n_i < \sum_{i\in[k]\setminus H} \frac{ng}{s} < k\cdot\frac{ng}{s} = \frac{kg}{s}\cdot n, \tag{13}$$

so at most a fraction of kg/s of the n points are in light boxes.

We now present a transport plan from $\mu_s^{(n)}$ to $\mu^{(n)}$. In Section 2.2 we gave the measure theoretic definition of the transport plan and the "earth mover" interpretation. We use the latter here. Consider that $\mu_s^{(n)}$ distributes 1 unit of mass across $[0,1]^d$. We want to move this mass to obtain $\mu^{(n)}$. Our approach is the following: first we move mass between boxes B_i, so that each box has $\mu^{(n)}(B_i)$ mass. Then we move the mass within each box to concentrate on the points of $\mathbf{X}$.

Transport plan. It is useful to introduce some notion of what mass we have moved and what is yet left untouched. To this end, we introduce the auxiliary boxes $(B'_1,\ldots,B'_k)$ of $(B_1,\ldots B_k)$, which have the same size and position in the hypercube. After the transport, the mass in boxes $(B_1,\ldots,B_k)$ are empty, and the all auxiliary boxes B'_i have exactly $\mu^{(n)}(B'_i)$ mass.

Step 1. For every $i\in H$, we know that a.a.s $\mu_s^{(n)}(B_i)\geq (n_i/n)(1-\sqrt{h/g})$, so we can assign this mass to B'_i. After this assignment, the mass left in the B_i's is

$$\bigcup_{i\in[k]} B_i = 1-\sum_{i\in H}\frac{n_i}{n}\left(1-\sqrt{\frac{h}{g}}\right).$$

Step 2. For each $i\in H$, we now assign $(n_i/n)\sqrt{h/g}$ mass from anywhere in $(B_1,\ldots B_k)$ to B'_i, so that all heavy boxes $B'_i,\ i\in H$ have a total of $n_i/n=\mu^{(n)}(B'_i)$ mass. Now the mass left in the B_i's is

$$\bigcup_{i\in[k]} B_i = 1-\sum_{i\in H}\frac{n_i}{n}\left(1-\sqrt{\frac{h}{g}}\right)-\sum_{i\in H}\frac{n_i}{n}\sqrt{\frac{h}{g}} = 1-\sum_{i\in H}\frac{n_i}{n} = \sum_{i\in[k]\setminus H}\frac{n_i}{n}.$$

Step 3. This is exactly the mass required for the light boxes $B'_i,\ i\in[k]\setminus H$, and this is also assigned in some arbitrary way.

Step 4. Now, every box B'_i has the required mass n_i/n as under $\mu^{(n)}$. Each box B'_i contains n_i of the n total points in $\mathbf{X}$. Each of these points requires $1/n$ mass, so the mass within each box can be moved accordingly.

Cost of transport plan. With this transport plan, we now find an upper bound for the cost of each step. In the p-Wasserstein distance, the cost $c(z)$ of moving one unit of mass a distance z is $c(z) = z^p$. Note that the auxiliary boxes $B'_1, \dots, B'_k$ are in the same position in the hypercube as $B_1, \dots, B_k$.

i For Step 1, we assign mass from boxes B_i to their auxiliary B'_i. The distance moved is 0, as the auxiliary boxes are in the same position as the original, and thus there is no cost associated.

ii For Step 2, we move $\sum_{i \in H}(n_i/n)\sqrt{h/g}$ mass from anywhere to the heavy boxes. That means each unit of mass is moved *at most* the diameter of the hypercube, $d^{1/2}$. So the cost associated with this step is at most $\sum_{i \in H}(n_i/n)\sqrt{h/g} \cdot d^{p/2}$.

iii For Step 3, $\sum_{i \notin H} n_i/n$ units of mass is moved from anywhere to the light boxes, so the cost associated is again at most $\sum_{i \notin H}(n_i/n) \cdot d^{p/2}$.

iv Lastly, in Step 4 all mass is moved within a box to concentrate on the points $\mathbf{X}$, so 1 unit of mass is moved at most the diameter of a box, which is $d^{1/2} \cdot k^{-1/d}$. So the associated cost is at most $d^{p/2}k^{-p/d}$.

With these upper bounds on the cost of the transport plan, we can give an upper bound for the Wasserstein distance,

$$\mathcal{W}_p\Big(\mu^{(n)}, \mu_s^{(n)}\Big)^p \le d^{\frac{p}{2}} \sum_{i \in H} \frac{n_i}{n}\sqrt{\frac{h}{g}} + d^{\frac{p}{2}} \sum_{i \notin H} \frac{n_i}{n} + d^{\frac{p}{2}} k^{-\frac{p}{d}} \tag{14}$$

$$= d^{\frac{p}{2}} \left(\sqrt{\frac{h}{g}} \sum_{i \in H} \frac{n_i}{n} + \sum_{i \notin H} \frac{n_i}{n} + k^{-\frac{p}{d}} \right). \tag{15}$$

Using (13) to upper bound the number of points in light boxes,

$$\begin{aligned}\mathcal{W}_p\Big(\mu^{(n)}, \mu_s^{(n)}\Big)^p &\le d^{\frac{p}{2}} \left(\sqrt{\frac{h}{g}} + \frac{kg}{s} + k^{-\frac{p}{d}} \right) \\ &= \mathcal{O}\left(\sqrt{\frac{h}{g}}\right) + \mathcal{O}\left(\frac{kg}{s}\right) + \mathcal{O}\left(k^{-\frac{p}{d}}\right).\end{aligned}$$

From the statement of the theorem, we have that $s(n) = \lfloor n^\sigma \rfloor$ for some $\sigma \in (0,1)$. We now need to pick k, g, h, all functions of n, and all with the constraint to be smaller than n. Furthermore, collecting the assumptions in (6) and (7), we need to have $g \le s/k$, $g > h$, and $h \gg \log k$, The goal is to maximize the rate of convergence of the upper bound. We take k, g, h of the form $k = n^\lambda, g = n^\gamma, h = n^\rho$, with $\lambda, \sigma, \gamma, \rho \in (0,1)$. Then for some q^*, we have

$$\mathcal{W}_p\Big(\mu^{(n)}, \mu_s^{(n)}\Big)^p = \mathcal{O}\Big(n^{-\frac{p\lambda}{d}}\Big) + \mathcal{O}\Big(n^{\frac{1}{2}(\rho-\gamma)}\Big) + \mathcal{O}\Big(n^{\gamma+\lambda-\sigma}\Big) = \mathcal{O}\Big(n^{-q^*}\Big).$$

Finding q^* is then equivalent to the following optimization problem.

$$\begin{aligned} q^* = \max_{\lambda,\gamma,\rho\in(0,1)} &\min\left(\frac{p\lambda}{d}, \frac{\gamma-\rho}{2}, \sigma-\gamma-\lambda\right), \\ \text{subject to} \quad &\gamma > \rho, \\ &\gamma < \sigma - \lambda. \end{aligned}$$

Because of the max min objective, this optimization problem has no simple closed form expression of q^* in terms of σ, d and p. We can simplify the parameter space by fixing $\lambda := c_1\sigma$ and $\gamma := c_2\sigma$. Also note that we can pick any $\rho > 0$ arbitrarily small. With this in place, we get for arbitrary small $\varepsilon > 0$,

$$\begin{aligned} q^* = \max_{c_1,c_2\in(0,1)} &\min\left(\frac{pc_1\sigma}{d}, \frac{c_2\sigma}{2} - \varepsilon, \sigma - c_1\sigma - c_2\sigma\right), \\ &c_2\sigma < \sigma - c_1\sigma. \end{aligned}$$

Or equivalently

$$\begin{aligned} q^*/\sigma = \max_{c_1,c_2\in(0,1)} &\min\left(\frac{pc_1}{d}, \frac{c_2}{2} - \varepsilon, 1 - c_1 - c_2\right), \\ &c_2 < 1 - c_1. \end{aligned}$$

These planes intersect for $c_2 = 2p/(d+3p)$ for $c_1 = d/(d+3p)$. Note also that

$$c_2 = \frac{2}{(d/p)+3} = 1 - \frac{(d/p)-1}{(d/p)+3} < 1 - \frac{d/p}{(d/p)+3} = 1 - c_1,$$

so the constraint is satisfied. Then we have that $\gamma = 3p\sigma/(d+3p)$ and $\lambda = d\sigma/(d+3p)$, which gives $q^* = p\sigma/(d+3p) - \varepsilon$. So then, since $\mathcal{W}_p\left(\mu^{(n)}, \mu_s^{(n)}\right)^p = \mathcal{O}(n^{-p\sigma/(d+3p)})$, then $\mathcal{W}_p\left(\mu^{(n)}, \mu_s^{(n)}\right) = \mathcal{O}(n^{-\sigma/(d+3p)})$. □

3.3 Implications for NEExT

With the result from Theorem 3.4, we can say the following in the context of the NEExT framework. Given two graphs of size n, with associated node embedding distributions $\mu^{(n)}, \nu^{(n)}$ on $[0,1]^d$, and we sample $s = \lfloor n^\sigma \rfloor$ points from each, giving $\mu_s^{(n)}, \nu_s^{(n)}$. Then, by the triangle inequality, we find that the loss in accuracy in the 2-Wasserstein distance between them is

$$\begin{aligned} \mathcal{W}_2(\mu^{(n)}, \nu^{(n)}) &\leq \mathcal{W}_2(\mu^{(n)}, \mu_s^{(n)}) + \mathcal{W}_2(\mu_s^{(n)}, \nu^{(n)}) \\ &\leq \mathcal{W}_2(\mu^{(n)}, \mu_s^{(n)}) + \mathcal{W}_2(\mu_s^{(n)}, \nu_s^{(n)}) + \mathcal{W}_2(\nu^{(n)}, \nu_s^{(n)}) \\ &= \mathcal{W}_2(\mu_s^{(n)}, \nu_s^{(n)}) + \mathcal{O}\left(n^{-\sigma/(d+6)+\varepsilon}\right). \end{aligned}$$

If we have m graphs, all of size n, with node embedding distributions $\mu^{(n),1}, \ldots, \mu^{(n),m}$, and consider the similarity matrix $S \in \mathbb{R}^{m \times m}$. Recall the matrix is defined by $S_{i,j} = \mathcal{W}_2(\mu^{(n),i}, \mu^{(n),j})$. We now have a bound on the perturbations from sampling on the similarity matrix, $\hat{S} = S + E$, where element-wise $E = \mathcal{O}(n^{-\sigma/(d+6)+\varepsilon})$. Although out of the scope of this thesis, this can be used to bound the difference in eigenvalues and eigenvectors of respectively S and $\hat{S}$, and consequently their singular values. See for example [25]. With this, one could analyze the effect of the inaccuracy introduced by sampling on the generated graph embeddings.

4 Triangle Count in Community Structure Graphs

In this section, we give asymptotic results for the number of triangles (three cycles) in two random graph models with community structure: the stochastic block model and the ABCD model, both defined later in this section. The formal definition of a triangle is straight-forward.

Definition 4.1 (Triangle)**.** *In the graph $G = (V, E)$, a triangle is an unordered set of three vertices $\{u, v, w\} \subset V$ such that $(u, v), (v, w), (w, u) \in E$.*

To avoid ambiguity, we define the total number of triangles in a graph.

Definition 4.2. *$T(G)$ denotes the number of triangles in graph $G = (V, E)$, i.e.*

$$T(G) = \left|\{\{u, v, w\} \mid u, v, w \in V, \{u, v, w\} \text{ is a triangle}\}\right|.$$

Note that the triangle is unordered, meaning that we count a triangle consisting of three particular vertices once, instead of counting it three times as seen from each vertex.

4.1 Triangles in the Stochastic Block Model

The Stochastic Block Model, first formulated by Holland et al. [35], is an extension of the Erdős–Rényi random graph to include a community structure. The model takes n vertices, partitioned into k communities $C_1, \ldots, C_k$, and an edge probability matrix $P \in \mathbb{R}^{k\times k}$. The edge between $u \in C_i$ and $v \in C_j$ is then sampled to be present with probability $P_{i,j}$. In this section, we show a concentration on the number of triangles in the 2-community setting, and how a local count of triangles can be used to estimate the global parameters. We then show by an experiment that graph embeddings based on triangle count, built by NEExT, distinguish different parameter choices. We finish by highlighting the complexity of more general cases.

Theorem 4.1. *Let the graph G_n be the SBM with two communities of m vertices, for a total of $n = 2m$ vertices. The edge probabilities are given by*

$$P = \begin{bmatrix} p & q \\ q & p \end{bmatrix},$$

so that the intra-community edge probability is p and the inter-community edge probability is q. Then, as $n \to \infty$,

$$\mathbb{E}\big[T(G_n)\big] \to \frac{n^3}{24}\left(p^3 + 3pq^2\right) - \frac{n^2}{4}\left(p^3 + pq^2\right) + \frac{n}{3}p^3,$$

and

$$\frac{T(G_n)}{n^3} \xrightarrow{\mathbb{P}} \frac{1}{24}\left(p^3 + 3pq^2\right).$$

Proof. For a vertex v, we call the number of triangles it is part of

$$T_v(G_n) = \big|\{\{v,u,w\} \mid u,w \in V, \{v,u,w\} \text{ is a triangle}\}\big| .$$

A given triangle within a community occurs with probability p^3, and a given triangle with one vertex in the other community occurs with probability pq^2. Then, by case distinction, we find

$$\begin{aligned}
\mathbb{E}\big[T_v(G_n)\big] &= \binom{m-1}{2} p^3 && \text{Neighbors in same community as } v \\
&+ m\,(m-1)\,pq^2 && \text{One neighbor in same community as } v \\
&+ \binom{m}{2} pq^2 && \text{Both neighbors in the other community} \\
&= \frac{m^2}{2}\left(p^3 + 3pq^2\right) - \frac{3m}{2}\left(p^3 + pq^2\right) + p^3 .
\end{aligned}$$

We obtain the expected total number of triangles by summing over the expected triangles per vertex. Although these events are not independent, this is allowed by the linearity of the expectation. Since this way every triangle is counted three times, we divide the total by three, to obtain that

$$\begin{aligned}
\mathbb{E}\big[T(G_n)\big] = \frac{1}{3}\sum_{v\in[n]} \mathbb{E}\big[T_v(G_n)\big] &= \frac{m^3}{3}\left(p^3 + 3pq^2\right) - m^2\left(p^3 + pq^2\right) + \frac{2m}{3}p^3 \\
&= \frac{n^3}{24}\left(p^3 + 3pq^2\right) - \frac{n^2}{4}\left(p^3 + pq^2\right) + \frac{n}{3}p^3 . \qquad (16)
\end{aligned}$$

To show that $T(G_n)$ converges in probability to its expectation, we first show that $\mathbb{V}\mathrm{ar}\left[T(G_n)\right] = \mathcal{O}(n^4)$. To do this, we sum over all triples of vertices and consider whether they form a triangle. Let $\Delta = \big\{\{v_1, v_2, v_3\} \mid \{v_1, v_2, v_3\} \subset V, \{v_1, v_2, v_3\} \text{ is a triangle}\big\}$ denote the set of triples that form a triangle. Let $\sum_{\Delta_1\in[n]}$ denote $\sum_{\{v_1,v_2,v_3\}\in[n]}$, the summation over all possible triples. Then we can write $T(G_n)$ as a sum of indicators, so that

$$T(G_n) = \sum_{v_1,v_2,v_3\in[n]} \mathbb{I}_{\{v_1,v_2,v_3\}\in\Delta} .$$

We can then express the variance as a sum of covariances

$$\mathbb{V}\mathrm{ar}\left[T(G_n)\right] = \sum_{\Delta_1\in[n]}\sum_{\Delta_2\in[n]} \mathrm{Cov}\left(\mathbb{I}_{\Delta_1\in\Delta}, \mathbb{I}_{\Delta_2\in\Delta}\right) .$$

If two triples share none or only one vertex, the event that the triples are triangles are independent, and their covariance is zero. The covariance is not zero when 1) the triples are the same or 2) the triples share two vertices (i.e. one edge).

$$\begin{aligned}
&\mathbb{V}\mathrm{ar}\left[T(G_n)\right] = \\
&\quad \sum_{u,v,w\in[n]} \mathrm{Cov}\left(\mathbb{I}_{\{u,v,w\}\in\Delta}, \mathbb{I}_{\{u,v,w\}\in\Delta}\right) + \sum_{u_1,u_2,v,w\in[n]} \mathrm{Cov}\left(\mathbb{I}_{\{u_1,u_2,v\}\in\Delta}, \mathbb{I}_{\{u_1,u_2,w\}\in\Delta}\right) .
\end{aligned}$$

We use the following inequality on the covariance for events A and B.

$$\mathrm{Cov}(\mathbb{I}_A, \mathbb{I}_B) = \mathbb{E}[\mathbb{I}_A \mathbb{I}_B] - \mathbb{E}[\mathbb{I}_A]\mathbb{E}[\mathbb{I}_B] = \mathbb{P}(A, B) - \mathbb{P}(A)\,\mathbb{P}(B) \leq \mathbb{P}(A, B)\,.$$

So we obtain

$$\begin{aligned} &\mathbb{V}\mathrm{ar}\left[T(G_n)\right] \leq \\ &\qquad \sum_{u,v,w\in[n]} \mathbb{P}\left(\{u,v,w\} \in \Delta\right) + \sum_{u_1,u_2,v,w\in[n]} \mathbb{P}\left(\{u_1,u_2,v\},\{u_1,u_2,w\} \in \Delta\right). \end{aligned}$$

Observe that the first term is exactly the expected number of triangles, which we found in (16) to be $\mathcal{O}(n^3)$. We look at the triples that share two vertices, which form motifs of four vertices in total. Say the shared vertices are u_1 and u_2, and v and w are the not-shared vertices. Consider then the 5-case distinction as visualized in Figure 5. This gives us

$$\begin{aligned} &\sum_{u_1,u_2,v,w\in[n]} \mathbb{P}\left(\{u_1,u_2,v\},\{u_1,u_2,w\} \in \Delta\right) = \\ &\quad 2\binom{m}{2}\binom{m-2}{2}p^5 && \textcircled{1} \\ &\quad + 2\binom{m}{2} m\,(m-2)\,q^2p^3 && \textcircled{2} \\ &\quad + 2\binom{m}{2}^2 pq^4 && \textcircled{3} \\ &\quad + 2m^2\binom{m-1}{2}p^2q^3 && \textcircled{4} \\ &\quad + m^2\,(m-1)^2\,q^3p^2 && \textcircled{5} \\ &\quad = \left(\frac{p^5}{2^5} + \frac{q^2p^3}{2^4} + \frac{pq^4}{2^5} + \frac{q^3p^2}{2^3}\right)n^4 + \mathcal{O}\left(n^3\right). \end{aligned}$$

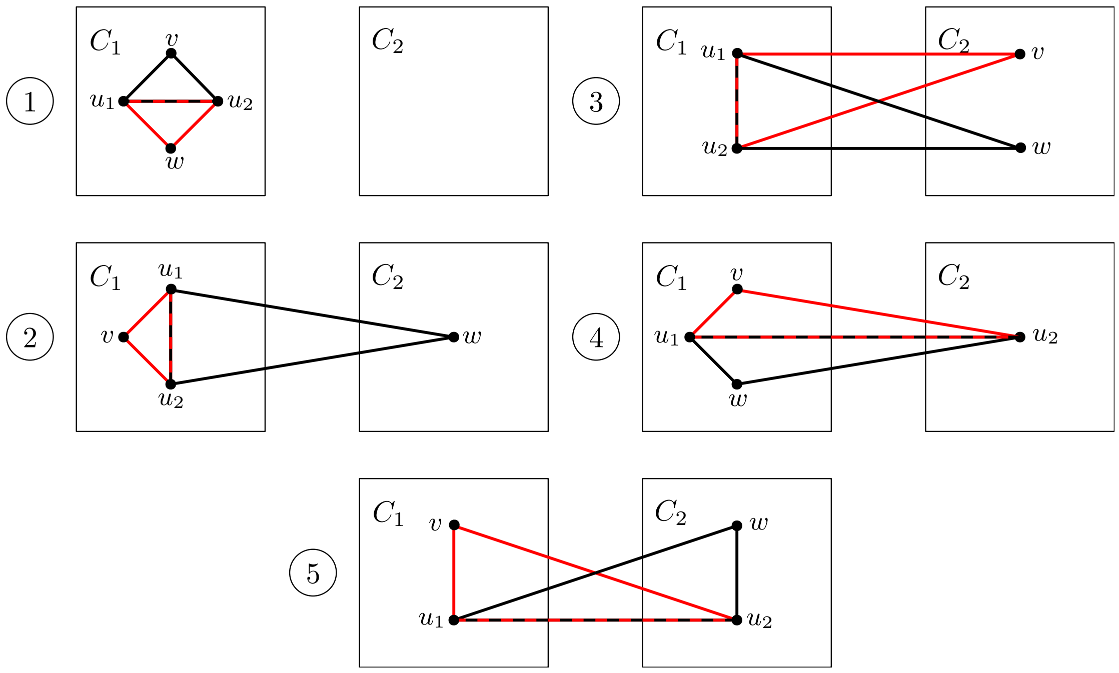


Figure 5 – Overlapping triangles

This shows that $\mathbb{V}\mathrm{ar}\,[X] = \mathcal{O}(n^4)$, so by the second moment method we have that

$$\mathbb{P}\left(|T(G_n) - \mathbb{E}[T(G_n)]| \geq \varepsilon\mathbb{E}[T(G_n)]\right) \leq \frac{\mathbb{V}\mathrm{ar}\,[T(G_n)]}{\varepsilon^2\mathbb{E}[T(G_n)]^2} = \mathcal{O}\left(\frac{1}{n^2}\right) \to 0.$$

From this it follows that, as $n \to \infty$, the number of triangles converges in probability to its expectation. $\square$

We return to the question about estimating global parameters from local features. In this setting, we care about estimating the global parameters p and q from the number of triangles. For the ER graph $G(n,p)$, it is common to consider the sparse parametrization where $p := \lambda/n$ for $\lambda > 0$, and it is well-known that in this setting the asymptotic number of triangles is Poisson distributed with expectation $\lambda^3/6$ [6]. In this 2-community SBM setting, the analogous parametrization is to set $p := \lambda/m$ and $q := \kappa/m$. Asymptotically then $T(G_n) = (1/24)(\lambda^3 + \kappa\lambda^2)$. Combine this with the convergence of the average degree $\bar{d}(G_n) \to \lambda + \kappa$, and we can solve these two equations for λ and κ. Both the degree and number of triangles are local features, and this gives an intuition how NEExT embeddings capture graph-level features using the distribution of local features. Figure 6 displays the NEExT graph embeddings using the 2-expansion of the triangle count as node embedding. The graph collection is generated for two parameter choices of λ and κ.

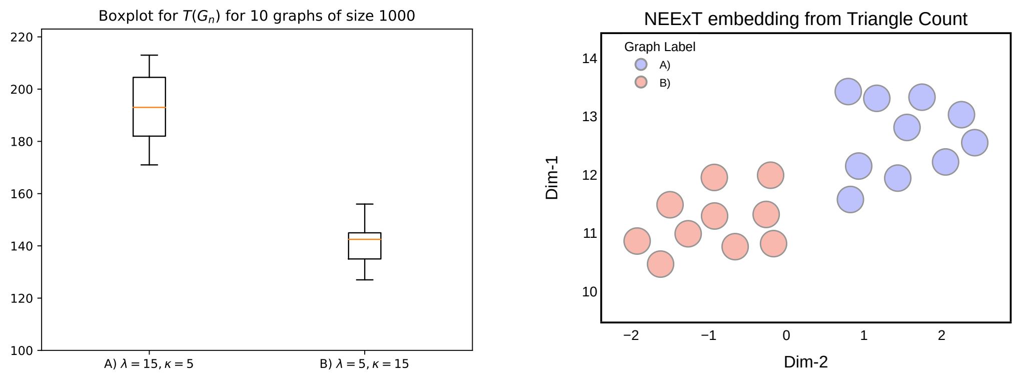


Figure 6 – On 10 graphs with $n = 1000$ vertices, generated by the 2-community SBM model, for two choices of $p = \lambda/m$ and $q = \kappa/m$. The NEExT graph embeddings using the 2-expansion triangle count separate the two settings well.

For large graphs, calculating all triangles becomes computationally prohibitive. In the spirit of Section 3, only a sample of the nodes could be considered. For the 2-community setting, this has little practical use. In the sparse parametrization, there is an expected constant number of triangles in the infinite graph. Then for any finite subset of nodes, the expected number of triangles including these nodes is zero. The finite number of triangles is a consequence of the fact that the size of both communities goes to infinity. Even within a community, connecting to a common neighbor becomes increasingly rare. The natural next step is then to consider a setting in which the community size is finite and the number of communities grows. Take, for example, the general setting where we have k communities $C_1, \dots, C_k$, with C_i having s_i vertices, so that $n := \sum_{j=1}^{k} s_j$, and again having intra-community edge probability p and inter-community edge probability q. Then let the number of triangles of which vertex v in community l is a part of be $T_v^l(G_n)$. Then the expected number of triangles can be expressed as sums over the different communities.

$$\begin{aligned}
\mathbb{E}[T_v^l(G_n)] &= \frac{1}{2}\left(s_l-1\right)\left(s_j-2\right)p^3 && \text{Neighbors in } C_l \\
&+ \sum_{\substack{j=1,\\ j\neq l}}^{k} \frac{1}{2}s_j\left(s_l-1\right)pq^2 && \text{One neighbor in } C_l\text{, other in other community} \\
&+ \sum_{\substack{j=1,\\ j\neq l}}^{k} \frac{1}{2}s_j\left(s_j-1\right)pq^2 && \text{Both neighbors in } C_j, j\neq l \\
&+ \sum_{\substack{j=1,\\ j\neq l}}^{k} \frac{1}{2}s_j\left(n-s_l-s_j\right)q^3. && \text{All vertices in different communities}
\end{aligned}$$

Without making additional assumptions on the distribution of the community sizes s_i, we cannot say much more. In the next sections, we will introduce the ABCD model, which although not directly analogous to the stochastic block model, does have a community structure, and does make assumptions over the distribution over the community sizes. In this setting, we will see that we can say more about the expected number of triangles. We first introduce the configuration model, on top of which the ABCD model builds forth.

4.2 Configuration model

For the algorithm of the ABCD model, which will be introduced in the Section 4.3, the configuration model is an important building block, and it is a key tool for its analysis. In this section, we introduce the model and explain its usefulness in analysis.

For a degree sequence $\mathbf{d} = (d_1, \ldots, d_n)$, we call its sum $\ell_n = \sum_{i\in[n]} d_i$. We assume that ℓ_n is even, as otherwise no graph exists with degree sequence $\mathbf{d}$. We now introduce the notion of the *half-edge*: the "stump" of an edge at a vertex. Connecting two half-edges creates one edge, so vertex i has d_i half-edges. Now, we consider the set of all half-edges, which has ℓ_n elements, and we sort them in an arbitrary order. To create an edge, we match the first half-edge with equal probability to any one of the $\ell_n - 1$ remaining half-edges, and remove both from the set. We continue this process until the set of unmatched half-edges is empty.

This adapted process of randomly pairing half-edges generates a uniformly random choice among the $(\ell_n - 1)!!$ possible pairings. Here !! denotes the semi-factorial given for even k by $k!! = k(k-2)\ldots\cdot 4\cdot 2$ and for odd k by $k!! = k(k-2)\ldots\cdot\cdot 3\cdot 1$. To see that there exist $(\ell_n-1)!!$ pairings, consider that all pairings can be seen as a permutation of a list of ℓ_n, where adjacent half-edges are paired. There are $\ell_n!$ permutations, but many will lead to the same pairing. So then we get that the total number of pairings is

$$\frac{\ell_n!!}{2^{\ell_n/2}(\ell_n/2)!} = \frac{\ell_n(\ell_n-1)\cdot\ldots\cdot 2\cdot 1}{2^{\ell_n/2}\ell_n/2(\ell_n/2-1)\cdot\ldots\cdot 2\cdot 1} = \frac{\ell_n(\ell_n-1)\cdot\ldots\cdot 2\cdot 1}{\ell_n(\ell_n-2)\cdot\ldots\cdot 4\cdot 2} = (\ell_n-1)!!.$$

The process generates a multigraph, meaning that self-loops and multi-edges may exist. However, it can be shown that, conditional on simplicity, the configuration model selects a graph uniformly from the set of graphs with that particular degree sequence. The configuration model algorithm also offers a way of analyzing graphs. By the random matching of half-edges we can use combinatorics to calculate the probability of an edge existing. For example, the expected number of edges $X_{u,v}$ between that vertex u and v is

$$\mathbb{E}\big[X_{u,v}\big] = d_v \cdot \frac{d_u}{\ell_n - 1},$$

as each of the d_v half-edges has probability of $d_u/(\ell_n-1)$ of being paired with one of the half-edges at u. Note that the existence of two different edges are not independent, but we can still say things about global properties of the graph through the linearity of expectation, a trick we often use in this thesis. Take for example the expected number of self-loops $\mathcal{S}(G)$ in multigraph $G=(V,E)$ generated by the configuration model on degree sequence $\mathbf{d}$.

$$\mathbb{E}\big[\mathcal{S}(G)\big] = \mathbb{E}\left[\sum_{v\in V}\mathbb{I}_{(v,v)\in V}\right] = \sum_{v\in V}\mathbb{E}\Big[\mathbb{I}_{(v,v)\in V}\Big] = \sum_{v\in V}\mathbb{P}\left((v,v)\in V\right) = \sum_{v\in V}\frac{d_v(d_v-1)}{\ell_n-1}.$$

For more details on the configuration model, see a textbook such as Hofstad [34].

4.3 ABCD Model

The Artificial Benchmark for Community Detection (ABCD)[41] is a random graph model that generates graphs with a community structure. It improves the LFR graph model [49] in its computational efficiency, suitability for theoretical analysis and the interpretability of community mixing parameter. The ABCD model is a much more general model than the Stochasic Block Model, and can model the characteristics of real-life networks, such as scale-free behavior. Furthermore, for simulation purposes, there exist efficient implementations for generating graphs with the ABCD model.[4]

4.3.1 Parameters

A general parametrization of the ABCD model consists of the size of the graph n, the degree distribution D_n, the community size distribution S_n, and a noise parameter ξ. The noise determines the proportion of edges that cross different communities. The default choices for the D_n and S_n, as given by [41], rely on the truncated power-law distribution.

[4]https://github.com/bkamins/ABCDGraphGenerator.jl

Definition 4.3 (Truncated Discrete Power law)**.** *Let $\gamma \in (2,3)$. Let $a, b \in \mathbb{N}$ with $a < b$. Then X is a truncated power-law random variable with exponent γ between $[a, b]$ if for $k \in \{a, \dots b\}$,*

$$\mathbb{P}(X = k) = \frac{\int_k^{k+1} x^{-\gamma} dx}{\int_a^{b+1} x^{-\gamma} dx}, \tag{17}$$

and 0 otherwise, and we denote $X \sim \mathcal{P}(\gamma, a, b)$.

With this, the parameters of the ABCD model are summarized in Table 1.

Parameter	Description	Default
$n \in \mathbb{N}^+$	Size of network	
D_n	Degree distribution	$D_n \sim \mathcal{P}(\gamma, \delta, n^\zeta)$ Exponent $\gamma \in (2,3)$ Minimum degree $\delta \in \mathbb{N}^+$ Maximum degree n^ζ $\zeta \in (0, \frac{1}{\gamma-1})$
S_n	Community Size distribution	$S_n \sim \mathcal{P}(\beta, s, n^\tau)$ Exponent $\beta \in (1,2)$ Minimum size $s \in \mathbb{N}^+$ Maximum size n^τ $\tau \in (\zeta, 1)$
$\xi \in (0,1)$	Level of noise	

Table 1 – Parameters of ABCD model

The range for $\zeta \in (0, 1/(\gamma - 1))$ is motivated by the fact that for any $\omega(n)$, with $\omega(n) \to \infty$ as $n \to \infty$, w.h.p the maximum degree will be $n^{1/(\gamma-1)}\omega(n)$, see [40, Lemma 5.1]. So for two choices $\zeta_1, \zeta_2 \in (1/(\gamma - 1), 1)$, the ABCD model may be coupled such that w.h.p it will generate the same graph. Then the range for $\tau \in (\zeta, 1)$ is chosen such that the maximum sampled community size is higher than the maximum sampled degree. When we refer to a graph generated by the ABCD model, unless otherwise mentioned, we implicitly fix these parameters. In this thesis, we focus on the default case for the community size distribution $S_n = \mathcal{P}(\beta, s, n^\tau)$, with $\beta \in (1,2)$. In the limit, the community size follows a pure power law truncated only from below, i.e.

$$\lim_{n\to\infty} S_n \overset{d}{=} \mathcal{P}(\beta, s, \infty) = \mathcal{P}(\beta, s).$$

For the choice D_n, we analyze two scenarios.

1. In Section 4.4, we consider an arbitrary choice of D_n, with the assumption that $\mathbb{E}[D_n^2] < \infty$ for all n.

2. In Section 4.5, we fix the degree distribution as its default $D_n = \mathcal{P}(\gamma, \delta, n^\zeta)$ with $\gamma \in (2,3)$. In this scenario $\lim_{n\to\infty} \mathbb{E}[D_n^2] = \infty$.

4.3.2 Algorithm

The algorithm for generating a graph from the ABCD model consists of four steps, explained in this section. In short, we first generate a degree sequence from D_n and a community size sequence from S_n. Then, the degree sequence is split to inner-community and outer-community edges according to the noise parameter ξ. In the third step, the vertices are uniformly assigned to communities based on the inner-community degree sequence. Lastly, the communities and the outer-community edges are matched independently following the configuration model, and then merged.

Step 1. Generating the community size sequence and degree sequence. The first step of generating an ABCD graph consists of generating an admissible community size sequence $\mathbf{s} = (s_1, \dots, s_k)$ from S_n so that $\sum_{l=1}^{k} s_l = n$, and an admissible degree sequence $\mathbf{d} = (d_1, \dots, d_n)$ from D_n so that$\sum_{i=1}^{n} d_i$ is even. To do this, sample values $s_1, \dots$ i.i.d. from S_n, until after some k values for the first time $\sum_{i=1}^{k} s_i \geq n$. Then decrease s_k by $n - \sum_{i=1}^{k} s_i$. For $\mathbf{d}$, sample n values $d_1, \dots, d_n$ i.i.d. from D_n. Then if $\sum_{i=1}^{n} d_i$ is not even, decrease the maximum of $\mathbf{d}$ by 1. Not all combinations of degree and community sequences will be admissible for generating an ABCD graph. However, if S_n and D_n are chosen nicely, such as in the default specification, the probability of admissible sequences is sufficiently large.

Step 2. Splitting degrees. We split the degree sequence into within-community degrees $\mathbf{b}$ and between-community degrees $\mathbf{r}$.

$$\mathbf{b} = (1-\xi)\mathbf{d},$$
$$\mathbf{r} = \xi\mathbf{d}.$$

To do this exactly, we use the following random splitting.

Definition 4.4. *For $x \in \mathbb{Z}$ and $y \in [0, 1)$ define the random variable $\lfloor x + y \rceil$ as*

$$\lfloor x + y \rceil = \begin{cases} x, & \text{with probability } 1 - y, \\ x + 1, & \text{with probability } y. \end{cases} \tag{18}$$

With this definition, $\mathbb{E}[\lfloor x+y \rceil] = x+y$. Now define $b_i := \lfloor (1-\xi)d_i \rceil$ and $r_i := d_i - b_i$. So then $\mathbb{E}[b_i] = (1-\xi)d_i$ and $\mathbb{E}[r_i] = \xi d_i$.

Step 3. Forming communities. In this step, the goal is to find a uniformly random chosen assignment from the set of $[n]$ vertices to $C_1, \dots, C_k$ communities of sizes $\mathbf{s}$, in such a way that $\mathbf{b}$ and $\mathbf{r}$ are graphic degree sequences. This means that each vertex with intra-community degree b_i is in a community of at least size b_i. To do this, we first define an upper bound on b_i,

$$b_i \leq x_i := \lceil (1 - \xi\phi)d_i \rceil, \quad \phi = 1 - \sum_{l \in [k]} \left(\frac{s_l}{n}\right)^2.$$

Vertex i will only be allowed in community C_l if $x_i \leq s_l - 1$. Assume now that the vertices $i \in [n]$ are sorted in descending order by x_i, so that $x_1 \geq x_2 \geq \ldots \geq x_n$. We then sort communities $l \in [k]$ by descending size s_l, so that $s_1 \geq s_2 \geq \ldots \geq s_n$. Let $s_{\leq l}$ for $l \in [k]$ denote the cumulative size of $\mathbf{s}$, i.e. $s_{\leq l} := \sum_{l=1}^{k} s_l$. Consider the mapping $f : [n] \mapsto [k]$ with $f(i) = l$ if and only if $s_{\leq l-1} \leq i \leq s_{\leq l}$. This mapping assigns the first s_1 vertices, with highest x_i, to the largest community C_1, and the following s_2 vertices to the second largest community C_2, and so on. This admissible assignment is not uniformly random, so we want to consider the set all permutations of the $[n]$ vertices, such that f still maps each vertex to a large enough communities. This set is given by

$$\mathcal{A} := \left\{ \sigma : [n] \to [n] : x_i \leq s_{f(\sigma(i))} - 1 \text{ for all } i \in [n] \right\}.$$

Our goal is to sample uniformly from this set. One way would be to sample permutations of $[n]$ randomly until one is in $\mathcal{A}$. Depending on the size of $\mathcal{A}$, the rejection of a random permutation might be too high. Fortunately, the following straight-forward algorithm generates permutations from $\mathcal{A}$ uniformly random. Let q_l for $l \in [k]$ denote the number of free spots in community C_l. So, at initialization of the algorithm, $q_l = s_l$ for all $l \in [k]$. Now, for $i \in [n]$, in order of $x_1 \geq x_2 \geq \ldots \geq x_n$, assign i to community C_l chosen uniformly from the set $\{C_l : l \in [k], x_i \leq s_l - 1, q_l > 0\}$, communities that are large enough and have a free spot. Then, decrease q_l, so that $q_l := q_l - 1$. Following this algorithm, we get a partition $\{C_1, \ldots C_k\}$ of the n vertices. This algorithm always terminates, as long as $\mathcal{A} \neq \emptyset$.

Step 4. Creating subgraphs G_j and merging. The final step for generating the ABCD graph relies on the configuration model as described in Section 4.2. An alternative formulation of the ABCD model relies on the Chung-Lu model instead [13], which generates graphs such that vertices follow the given degree sequence in expectation.

First, the background graph G_0 is created following the configuration model with degree sequence $\mathbf{r}$. The edges in the background graph contain edges between communities, although also possibly edges within communities. Then, independently from each other, the community graphs G_l for $l \in [k]$ are created with degree sequence $\{b_i : i \in C_l, i \in [n]\}$, while keeping track of vertex labels as in G_0. As a last step, the edge sets of G_0 and $G_1, \ldots, G_k$ are combined to obtain the final graph G.

This formulation does not guarantee a simple graph. Self-loops may appear in the background graph G_0 or the community graphs $G_1, \ldots, G_k$. Multi-edges may appear both in the $G_0, \ldots, G_k$, and in their union G. If the a simple graph is required, the ABCD algorithm relies on a *switching* procedure that rewires self-loops and multi-edges. (for more details, see [41]). In the theoretical analysis of the model, we do not consider this step and focus on the multigraph instead.

4.3.3 Known result for ABCD model

From Kaminski et al. [40, Corollary 5.5] we have the following result for the asymptotic number of communities.

Theorem 4.2. *Let $\beta \in (1,2)$, the minimum communtiy size $s \in \mathbb{N}$, and $\tau \in (\zeta, 1)$. Then the number of communities k is equal to*

$$k = k(n) = \left(1 + \mathcal{O}\left((\log n)^{-1}\right)\right) \hat{c} n^{1-\tau(2-\beta)} = \Theta(n^{1-\tau(2-\beta)}),$$

where

$$\hat{c} = \frac{2-\beta}{(\beta-1)s^{\beta-1}}.$$

4.4 Triangles in ABCD Model: Finite Variance Degree Distribution

We first consider a setting where the degree distribution has finite variance. We show the following result.

Theorem 4.3. *Consider a graph sequence $(G_n)_{n\in\mathbb{N}^+}$, where G_n is a graph of n vertices generated by the ABCD model with $S_n \sim \mathcal{P}(\beta, s, n^\tau)$ and some D_n with $\mathbb{E}[D_n^2] < \infty$ for all n, and corresponding admissible sequences $\mathbf{s}^{(n)}, \mathbf{b}^{(n)}, \mathbf{r}^{(n)}$. Then, conditional on admissible degree and community sequences, and the condition that for all n it holds that*

$$\mathbb{P}(b_i^{(n)} \geq 2) = \mathbb{P}\left(D_n \geq 2/(1-\xi)\right) > 0, \tag{19}$$

there exists some constant c such that, as $n \to \infty$,

$$\frac{\mathbb{E}\left[T(G_n)\right]}{n^{(1-\tau(2-\beta))}} \to c. \tag{20}$$

Put differently, the number of triangles growth linearly with the number of communities, which grows with rate $n^{1-\tau(2-\beta)}$ as by Theorem 4.2. The constant c depends on the parameters of the ABCD model, importantly on the noise parameter ξ. The condition (19) ensures that there is a non-zero probability of vertices having intra-community degree of 2 or more, a necessary condition so that triangles within communities can occur. Note that for convenience we will often drop the subscript from $\mathbf{s}^{(n)}, \mathbf{b}^{(n)}, \mathbf{r}^{(n)}$ and simply write $\mathbf{s}, \mathbf{b}, \mathbf{r}$ instead when the index is unambiguous.

Proof of Theorem 4.3. For simplicity of the theoretical analysis, we consider here graphs that might have multi-edges and/or have self-loops. The ABCD algorithm has a rewiring step that ensures a simple graph that is known to be very close to uniformly random. [37] It should be noted that self-loops and multi-edges can occur in two places: when wiring the communities and background graph as CM, and when merging the background graph and communities.

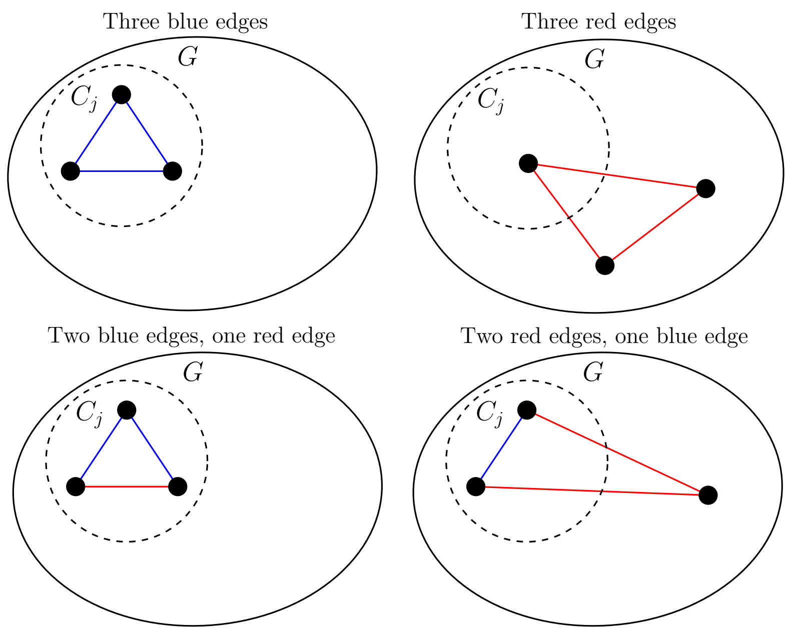


Figure 7 – The four types of triangles in the ABCD model

We adopt the following terminology: we say an edge is *red* if it is formed in the background graph, and thus likely will be between communities. An edge is *blue* if it is within a community. As introduced before, the red degree of vertex v we denote by r_v, and we note that by r_v and b_v are distributed as by Definition 4.4. Then we have the following types of triangles, as visualized in Figure 7.

1. Three blue edges, denoted by $T^{3b}(G_n)$.
2. Three red edges: $T^{3r}(G_n)$.
3. Two red edges, one blue edge: $T^{2r,1b}(G_n)$.
4. Two blue edges, one red edge: $T^{2b,1r}(G_n)$.

To show the result, we show that $\mathbb{E}[T^{3b}(G_n)]$ gives a $\Theta(k)$ contribution, and that the contributions of $\mathbb{E}[T^{3r}(G_n)], \mathbb{E}[T^{2r,1b}(G_n)]$ and $\mathbb{E}[T^{2b,1r}(G_n)]$ are negligible. We start by looking at $\mathbb{E}[T^{3b}(G_n)]$.

All blue edge triangles. We first work conditional on the red and blue degree sequences and the allocation to communities, denoted by $\mathbb{E}[\cdot \mid \mathbf{s}, \mathbf{b}, \mathbf{r}]$. Recall that C_l denotes the set of vertices in the l-th community, and we call the volume of that community $\mathcal{C}_l := \sum_{v \in C_l} b_v$, and that k denotes the (random) number of communities. Note that because we are ordering over labelled triples in each community, we have to count each triangle three times, so we have to divide by 3.

$$\mathbb{E}\Big[T^{3b}(G_n)\;\Big|\;\mathbf{s},\mathbf{b},\mathbf{r}\Big] =$$

$$\mathbb{E}\left[\frac{1}{3}\sum_{l\in[k]}\sum_{u,v,w\in C_l}\sum_{h_1^b,h_2^b\sim u}\sum_{h_3^b,h_4^b\sim v}\sum_{h_5^b,h_6^b\sim w}\mathbb{I}_{h_1^b\leftrightarrow h_3^b,h_4^b\leftrightarrow h_5^b,h_6^b\leftrightarrow h_2^b}\;\middle|\;\mathbf{s},\mathbf{b},\mathbf{r}\right],$$

where $\sum_{h_1^b,h_2^b\sim v}$ denote the summation of two distinct blue half-edges at vertex v. Because of the linearity of expectation and using that $\mathbb{E}[\mathbb{I}_A] = \mathbb{P}(A)$, we can use the configuration model's pairing algorithm to see that

$$\mathbb{E}\Big[T^{3b}(G_n)\;\Big|\;\mathbf{s},\mathbf{b},\mathbf{r}\Big] = \frac{1}{3}\sum_{l\in[k]}\sum_{u,v,w\in C_l}\frac{b_u b_v(b_v-1)b_w(b_w-1)(b_u-1)}{(\mathcal{C}_l-1)(\mathcal{C}_l-3)(\mathcal{C}_l-5)}. \tag{21}$$

For the next step, we introduce the following quantity, for $j \in \mathbb{N}, j \geq s$,

$$\mathbb{E}[Z_j] := \mathbb{E}\left[0 \vee \sum_{u,v,w\in C_l}\frac{b_u b_v(b_v-1)b_w(b_w-1)(b_u-1)}{(\mathcal{C}_l-1)(\mathcal{C}_l-3)(\mathcal{C}_l-5)}\;\middle|\;|C_l| = j\right]. \tag{22}$$

The quantity $\mathbb{E}[Z_j]$ denotes the expected number of triangles in a community of size j. It is not easy to see how $\mathbb{E}[Z_j]$ behaves for different j. Communities with more vertices have more possible triangles, but the probability of two neighbors connection becomes smaller. We can say, however, that $\mathbb{E}[Z_j]$ converges: as $b_u \leq d_u$, and $d_u \sim D_n$ has finite variance, the Law of Large Numbers imples that $\mathbb{E}[Z_j] \to \mathbb{E}[Z]$ for some Z, as visualized by a simulation in Figure 8.

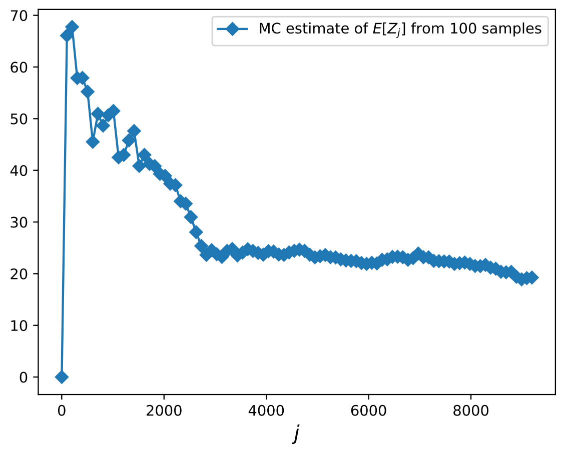


Figure 8 – A Monte Carlo simulation of $\mathbb{E}[Z_j]$ for $D \sim \mathcal{P}(3.5)$

Let the empirical density of the community size be $p_j^{(k)} = \sum_{l\in[k]}\mathbb{I}_{|C_l|=j}/k$. We now write the expectation over the degree sequence and community allocations as the sum over the different community sizes. Although the number of triangles between communities is not independent, the expectation is linear, so this not a problem.

$$\mathbb{E}\Big[\mathbb{E}\Big[T^{3b}(G_n)\ \Big|\ \mathbf{s},\mathbf{b},\mathbf{r}\Big]\Big] = \frac{k}{3}\sum_{j=s}^{n^\tau} p_j^{(k)}\mathbb{E}\big[Z_j\big]. \tag{23}$$

Then note that by the convergence of the empirical CDF, we have $p_j^{(k)} \to p_j := \mathbb{P}\left(S = j\right)$, where S is the limiting distribution of the community size, given by $S := \lim_{n\to\infty} S_n \overset{d}{=} \mathcal{P}(\beta, s, \infty)$. By Theorem 4.2 that

$$\mathbb{E}\Big[T^{3b}(G_n)\Big] = \Theta\left(n^{1-\tau(2-\beta)}\right)\sum_{j=s}^{\infty} p_j\mathbb{E}\big[Z_j\big] = \Theta(n^{1-\tau(2-\beta)}).$$

Note that the sum converges as $p_j \to 0, \sum_j p_j = 1$ and $\mathbb{E}[Z_j] \to \mathbb{E}[Z]$, and the sum is strictly larger as $p_j > 0$ and for finite j we have definitely that $\mathbb{E}[Z_j] > 0$. To see this, note that by the condition (19), there is for every j a non-zero probability that both the denominator and numerator in Z_j are positive, and by the fact that $Z_j \geq 0$, it follows that $\mathbb{E}[Z_j] > 0$.

All red edge triangles We now move on to the triangles formed in the background graph.

$$\mathbb{E}\Big[T^{3r}(G_n)\ \Big|\ \mathbf{s},\mathbf{b},\mathbf{r}\Big] = \\ \mathbb{E}\left[\frac{1}{3}\sum_{u,v,w\in[n]}\ \sum_{h_1^r,h_2^r\sim u}\ \sum_{h_3^r,h_4^r\sim v}\ \sum_{h_5^r,h_6^r\sim w} \mathbb{I}_{h_1^r\leftrightarrow h_3^r, h_4^r\leftrightarrow h_5^r, h_6^r\leftrightarrow h_2^r}\ \middle|\ \mathbf{s},\mathbf{b},\mathbf{r}\right].$$

Let $\ell_n = \sum_{v\in[n]} r_v$. Using the same trick of linearity of expectation we see that

$$= \frac{1}{3}\sum_{u,v,w\in[n]} \frac{r_u r_v(r_v-1)r_w(r_w-1)(r_u-1)}{(\ell_n-1)(\ell_n-3)(\ell_n-5)}$$
$$= \frac{1}{3}\frac{1}{(\ell_n-1)(\ell_n-3)(\ell_n-5)}\sum_{u\in[n]} r_u(r_u-1)\sum_{v\in[n]} r_v(r_v-1)\sum_{w\in[n]} r_w(r_w-1)) + o(1),$$

where the $o(1)$ term that accounts for double counting of vertices. So we find then

$$\mathbb{E}\Big[T^{3r}(G_n)\ \Big|\ \mathbf{s},\mathbf{b},\mathbf{r}\Big] = \frac{1}{3}\frac{1}{(\ell_n-1)(\ell_n-3)(\ell_n-5)}\left(\sum_{u\in[n]} r_u(r_u-1)\right)^3 + o(1).$$

We can see that

$$(\ell_n-1)(\ell_n-3)(\ell_n-5) = \ell_n^3 + \mathcal{O}(\ell_n^2) = \mathcal{O}\Big(n^3\Big),$$

so we can follow the intuition that the cube over a sum over n, divided by the cube of n, converges to a constant. In fact, we can find this constant exactly. We write

$$\mathbb{E}[T^{3r}(G_n) \mid \mathbf{s}, \mathbf{b}, \mathbf{r}] = \frac{1}{3}\left(\frac{n}{\ell_n + o(1)}\right)^3 \left(\frac{\sum_{u\in[n]} r_u(r_u-1)}{n}\right)^3 + o(1).$$

Recall that $\mathbb{E}[r_u] = \xi d_u$, so that

$$\mathbb{E}\big[r_u(r_u-1)\big] = \mathbb{E}[r_u^2] - \mathbb{E}[r_u] = \xi(1-\xi)d_u + \xi^2 d_u^2 - \xi d_u = \xi^2 d_u^2 - \xi^2 d_u = \xi^2 d_u(d_u-1).$$

Let $\lim_{n\to\infty} D_n \overset{d}{=} D$. Then by applying the LLN for both terms, using $\ell_n/n \to \xi\mathbb{E}[D]$, we see

$$\mathbb{E}\Big[T^{3r}(G_n)\Big] \to \frac{1}{3}\frac{1}{\xi^3\mathbb{E}[D]^3}\left(\xi^2\mathbb{E}\big[D(D-1)\big]\right)^3 = \frac{\xi^3}{3}\left(\frac{\mathbb{E}\big[D(D-1)\big]}{\mathbb{E}[D]}\right)^3 = \mathcal{O}(1).$$

Two red edges, one blue edge triangles. We proceed in similar vein for the triangles formed by two red edges and one blue edge. These triangles can occur for each vertex u combined with all pairs v, w, where v and w are in the same community. Note that the red edges may also form within a community. We then see that

$$\begin{aligned}&\mathbb{E}\Big[T^{2r,1b}(G_n) \;\Big|\; \mathbf{s}, \mathbf{b}, \mathbf{r}\Big] \\ &\quad= \mathbb{E}\left[\frac{1}{2}\sum_{u\in[n]}\sum_{l\in[k]}\sum_{v,w\in C_l}\sum_{h_1^r,h_2^r\sim u}\sum_{h_3^r,h_4^b\sim v}\sum_{h_5^r,h_6^b\sim w}\mathbb{I}_{h_1^r\leftrightarrow h_3^r,h_4^b\leftrightarrow h_6^b,h_5^r\leftrightarrow h_2^r}\;\middle|\;\mathbf{s},\mathbf{b},\mathbf{r}\right].\end{aligned}$$

From which we get that

$$\mathbb{E}\Big[T^{2r,1b}(G_n) \;\Big|\; \mathbf{s}, \mathbf{b}, \mathbf{r}\Big] = \frac{1}{2}\sum_{u\in[n]}\sum_{l\in[k]}\sum_{v,w\in C_l}\frac{r_u(r_u-1)r_v r_w b_v b_w}{(\ell_n-1)(\ell_n-3)(\mathcal{C}_l-1)} \tag{24}$$

$$= \frac{1}{2}\frac{\sum_{u\in[n]} r_u(r_u-1)}{(\ell_n-1)(\ell_n-3)}\sum_{l\in[k]}\frac{\sum_{v\in C_l}\sum_{w\in C_l} r_v r_w b_v b_w}{\mathcal{C}_l - 1} + o(1), \tag{25}$$

where $o(1)$ accounts the double counting of vertices. We continue to see that

$$\begin{aligned}\mathbb{E}\Big[T^{2r,1b}(G_n) \;\Big|\; \mathbf{s}, \mathbf{b}, \mathbf{r}\Big] &\le \frac{1}{(\ell_n-1)(\ell_n-3)}\sum_{u\in[n]} r_u^2 \sum_{l\in[k]}\frac{1}{\mathcal{C}_l-1}\sum_{v\in C_l} r_v b_v \sum_{w\in C_l} r_w b_w \\ &\le \frac{1}{(\ell_n-1)(\ell_n-3)}\sum_{u\in[n]} r_u^2 \sum_{l\in[k]}(\mathcal{C}_l-1)\left(\frac{\sum_{v\in C_l} r_v b_v}{\mathcal{C}_l-1}\right)^2.\end{aligned}$$

Define $\mathbb{E}[Z_j^*]$ similarly to (22), for $j \in \mathbb{N}, j \geq s$,

$$\mathbb{E}\Big[Z_j^*\Big] := \mathbb{E}\left[0 \vee \left(\frac{\sum_{v\in C_l} r_v b_v}{\mathcal{C}_l - 1}\right)^2 \,\middle|\, |C_l| = j\right], \quad \lim_{j\to\infty} \mathbb{E}\Big[Z_j^*\Big] = \mathbb{E}\big[Z^*\big]. \tag{26}$$

We write, as in (23), the conditional sum over community size.

$$\begin{aligned}\mathbb{E}\Big[T^{2r,1b}(G_n) \,\Big|\, \mathbf{s},\mathbf{b},\mathbf{r}\Big] &\leq \frac{\sum_{u\in[n]} r_u^2}{\ell_n - 1} k \frac{1}{\ell_n - 3} \sum_{j=1}^{n^\tau} (j-1) p_j^{(k)} \mathbb{E}\Big[Z_j^*\Big] \\ &\leq \frac{\sum_{u\in[n]} r_u^2}{\ell_n - 1} k \frac{n^\tau}{\ell_n - 3} \sum_{j=1}^{n^\tau} p_j^{(k)} \mathbb{E}\Big[Z_j^*\Big].\end{aligned}$$

For the last inequality, observe that $j-1 \leq n^\tau$, as $j-1$ sums from $1 \ldots n^\tau$. The first fraction converges by the Law of Large Numbers. The sum converges as $n \to \infty$, following the reasoning in the all blue edge case. Then, as $\ell_n - 3 = \Theta(n)$ we end up with a rate $\Theta(kn^{\tau-1})$. Since $\tau < 1$, we have as required that

$$\mathbb{E}\Big[T^{2r,1b}(G_n)\Big] = o(k).$$

Two blue edges, one red edge triangles. Lastly, we count all triangles consisting of two blue and one red edge. These occur when a triple of vertices, all in the same community, form one red edge in the background graph, and two blue edges in the community graph. We see that

$$\mathbb{E}\Big[T^{2b,1r}(G_n) \,\Big|\, \mathbf{s},\mathbf{b},\mathbf{r}\Big] =$$
$$\frac{1}{2}\mathbb{E}\left[\sum_{l\in[k]} \sum_{u,v,w\in C_l} \sum_{h_1^r, h_2^b \sim u} \sum_{h_3^r, h_4^b \sim v} \sum_{h_5^b, h_6^b \sim w} \mathbb{I}_{h_1^r \leftrightarrow h_3^r, h_4^b \leftrightarrow h_6^b, h_5^b \leftrightarrow h_2^b} \,\middle|\, \mathbf{s},\mathbf{b},\mathbf{r}\right].$$

We then find, again by the configuration model's matching that

$$\begin{aligned}\mathbb{E}\Big[T^{2b,1r}(G_n) \,\Big|\, \mathbf{s},\mathbf{b},\mathbf{r}\Big] &= \frac{1}{2} \sum_{l\in[k]} \sum_{u,v,w\in C_l} \frac{r_u r_v}{\ell_n - 1} \frac{b_v b_w (b_w - 1) b_u}{(\mathcal{C}_l - 1)(\mathcal{C}_l - 3)} \\ &\leq k \frac{1}{\ell_n - 1} \sum_{l\in[k]} (\mathcal{C}_l - 1) \left(\frac{\sum_{u\in C_l} r_u b_u}{\mathcal{C}_l - 1}\right)^2 \frac{\sum_{v\in C_l} b_v (b_v - 1)}{\mathcal{C}_l - 3}.\end{aligned} \tag{27}$$

Define $\mathbb{E}[Z_j^{**}]$ equivalently to (22), for $j \in \mathbb{N}, j \geq s$,

$$\mathbb{E}\Big[Z_j^*\Big] := \mathbb{E}\left[0 \vee \left(\frac{\sum_{u\in C_l} r_u b_u}{\mathcal{C}_l - 1}\right)^2 \frac{\sum_{v\in C_l} b_v (b_v - 1)}{\mathcal{C}_l - 3} \,\middle|\, |C_l| = j\right], \quad \lim_{j\to\infty} \mathbb{E}\Big[Z_j^*\Big] = \mathbb{E}\big[Z^*\big]. \tag{28}$$

We write, as in (23), the conditional sum over community size,

$$\begin{aligned}\mathbb{E}\Big[T^{2b,1r}(G_n)\;\Big|\;\mathbf{s},\mathbf{b},\mathbf{r}\Big] &= k\frac{1}{\ell_n-1}\sum_{j=1}^{n^\tau}(j-1)p_j^{(k)}\mathbb{E}\Big[Z_j^{**}\Big]\\ &\leq k\frac{n^\tau}{\ell_n-1}\sum_{j=1}^{n^\tau}p_j^{(k)}\mathbb{E}\Big[Z_j^{**}\Big],\end{aligned}$$

Then again we see that this term is $\Theta(kn^{\tau-1})=o(k)$.

Conclusion of proof. To see the result simply sum the different types of triangles to get the required result

$$\begin{aligned}\lim_{n\to\infty}\mathbb{E}\big[T(G_n)\big] &= \lim_{n\to\infty}\Big(\mathbb{E}\Big[T^{3r}(G_n)\Big]+\mathbb{E}\Big[T^{3b}(G_n)\Big]+\mathbb{E}\Big[T^{2r,1b}(G_n)\Big]+\mathbb{E}\Big[T^{2b,1r}(G_n)\Big]\Big)\\ &= \Theta(k)+\mathcal{O}(1)+o(k)+o(k)\\ &= \Theta(k).\end{aligned}$$

Then by Theorem 4.2, which states the rate of growth of the number of communities, we conclude that $\Theta(k)=\Theta(n^{1-\tau(2-\beta)})$. □

So the number of blue triangles is asymptotically proportional to the number of communities. This seemingly trivial fact is not clear when one considers that *most vertices fall in large communities.* So far, we have considered the setting where $\lim_{n\to\infty}\mathbb{E}[D_n^2]<\infty$. In the next section, D_n follows a power law with exponent $\gamma\in(2,3)$. Then $\lim_{n\to\infty}\mathbb{E}[D_n^2]=\infty$, and Theorem 4.3 does not apply. In this setting, the number of triangles in the large communities does not converge, i.e. $\lim_{j\to\infty}\mathbb{E}[Z_j]=\infty$ with $\mathbb{E}[Z_j]$ defined as in (22). This is the subject of the next section. We set out this section with the goal of quantifying the strength of the community structure by triangle count. For the ABCD model, this is still an open questions, as we have not shown how the constant c changes for different values of ξ. At the end of this section we do a simulation experiment, see Figure 10, which confirms the intuition that a higher ξ (weaker communtiy structure) decreases the number of triangles.

4.5 Triangles in ABCD Model: Power-Law Degree Distribution

So far, we have assumed that $\mathbb{E}[D_n^2]<\infty$ for all n. This simplifies some things, namely that the number of triangles within a community and the background graph are asymptotically constant. Now we want to consider a setting where $\lim n\to\infty\mathbb{E}[D_n^2]=\infty$. To this end, we consider the default formulation of the ABCD model as in Kaminski et al. [40]. In this model, D_n follows a truncated power-law distribution with exponent $\gamma\in(2,3)$, as in Definition 4.3. Our main result is that we find sufficient conditions on the parameters of the model such that the number of triangles is dominated by *blue* (i.e. within-community) triangles. However, unlike in the bounded variance setting in the previous setting, the number of triangles in a given community now grows with its size, instead of converging to a constant number. This means that the number of triangles grows faster than the number of communities. Before we state the main result, we introduce the notion of a power-law tail.

Condition 4.5 (Power-law tail degree sequence). *Let $F_n(j)$ be the empirical degree distribution of degree sequence $\mathbf{x} = (x_1, \ldots, x_n)$*

$$F_n(j) = \frac{1}{n} \sum_{i \in [n]} \mathbb{I}_{x_i \leq j}.$$

Then we say that $\mathbf{x}$ has a power-law tail with exponent $\gamma \in (2,3)$, if

(i) There exists a constant $K > 0$ such that for every $0 \leq j \leq \max(\mathbf{x})$,

$$1 - F_n(j) \leq K j^{1-\gamma}. \tag{29}$$

(ii) There exists a constant $C > 0$ such that, for all $j = \mathcal{O}(\sqrt{n})$,

$$1 - F_n(j) = C j^{1-\gamma}(1 + o(1)). \tag{30}$$

We now state the following lemma.

Lemma 4.4. *Consider degree sequences $\mathbf{b}$ and $\mathbf{r}$ obtained by random splitting as in Definition 4.4, i.e. for each $u \in [n]$ take*

$$b_u := \lfloor (1-\xi)\, d_u \rceil, \quad r_u := d_u - b_u,$$

with $\mathbf{d} = (d_1, \ldots d_n)$ and $d_i \overset{i.i.d.}{\sim} \mathcal{P}(\gamma, \delta, n^{\zeta})$ with $\gamma \in (2,3)$. Then $\mathbf{b}$ and $\mathbf{r}$ have a power-law tail, i.e. satisfy Condition 4.5 with exponent γ.

Proof of Lemma 4.4. Let $F_n^b(j)$ be the empirical degree distribution of $\mathbf{b}$. To see that Condition 4.5(i) holds, simply note that, since $b_i \leq d_i$ for all $i \in [n]$, then $F_n^b(j) \geq F_n^r(j)$ for all j. Then a.a.s

$$1 - F_n^b(j) \leq 1 - F_n^d(j) \leq K j^{1-\gamma},$$

because $\mathbf{d}$ is directly sampled from a power law. The same reasoning applies to $\mathbf{r}$. For Condition 4.5(ii), consider some $j = \mathcal{O}(\sqrt{n})$, then

$$F_n^d(\lfloor (1-\xi) j \rfloor) \leq F_n^b(j) \leq F_n^d(\lfloor (1-\xi) j \rfloor + 1).$$

Since $\mathbf{d}$ satisfies the condition and the fact that $\lfloor (1-\xi) j \rfloor = \mathcal{O}(\sqrt{n})$, we have that there exists some $C > 0$ such that

$$F_n^d(\lfloor (1-\xi) j \rfloor) = 1 - C(\lfloor (1-\xi) j \rfloor)^{1-\gamma}(1 + o(1)) = 1 - C(1-\xi)^{1-\gamma} j^{1-\gamma}(1 + o(1)),$$

and similarly $F_n^d(\lfloor (1-\xi) j \rfloor + 1) = 1 - C(1-\xi)^{1-\gamma} j^{1-\gamma}(1 + o(1))$. So then a.a.s. for constant $C' := C(1-\xi)^{1-\gamma}$ it holds that $1 - F_n^b(j) = C' j^{1-\gamma}(1 + o(1))$. □

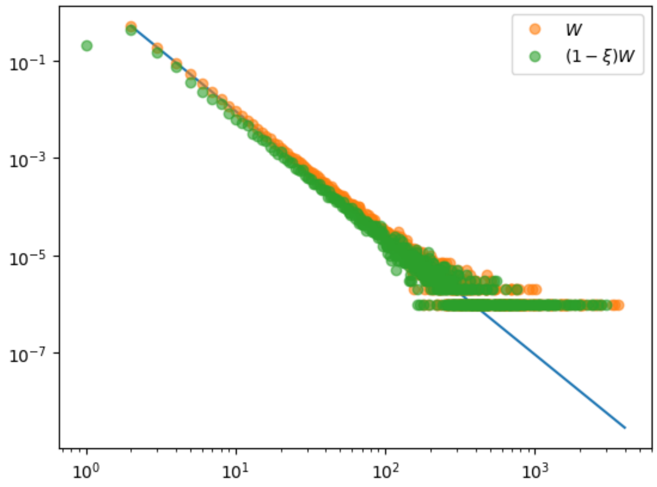


Figure 9 – Visualization of Lemma 4.4

An important ingredient for our result is the following theorem from Gao et al. [30], which states that the number of triangles in a uniform random graph with a power-law tail degree sequence satisfying Condition 4.5 is $\Theta(n^{\frac{3}{2}(3-\gamma)})$.

Theorem 4.5 (Theorem 1 from [30])**.** *Let $\gamma \in (2,3)$ and $\boldsymbol{d}_n$ be a degree sequence on n vertices that satisfies Condition 4.5 with exponent γ and constant C. Let G_n be a uniform random graph with degree sequence $\mathbf{d}_n$. Let μ be the mean of $\mathbf{d}_n$. Then $T(G_n)$, denoting the number of triangles in G_n, it holds that*

$$T\left(G_n\right) \xrightarrow{\mathbb{P}} -\frac{1}{12}\left(\frac{\pi C(\gamma-1)\mu^{-(\gamma-1)/2}}{\cos\left(\frac{\pi\gamma}{2}\right)}\right)^3 n^{\frac{3}{2}(3-\gamma)}.$$

Proof of Theorem 4.5. The proof consists of showing that most of the triangles are between vertices with degree of order $\sqrt{n}$, and any triangles with lower degree vertices are negligible. □

With this, we are ready to state sufficient conditions on the parameters to find the rate of growth on the number of triangles.

Theorem 4.6. *Consider a sequence of graphs $(G_n)_{n\in\mathbb{N}^+}$, each G_n generated by the ABCD model, with $S_n \sim \mathcal{P}(\beta, s, n^\tau)$ and $D_n \sim \mathcal{P}(\gamma, \delta, n^\zeta)$, and $\gamma \in (2,3)$. If it holds that*

$$\frac{2\tau(1-\beta)}{3(3-\gamma)} > -1,$$

and the maximum degree n^ζ is chosen such that

$$\zeta \leq \frac{1}{2}\tau + \frac{1-\tau}{3(3-\gamma)},$$

then for some constant c, as $n \to \infty$, it holds that

$$\frac{\mathbb{E}\big[T(G_n)\big]}{n^{(1-\tau+\frac{3}{2}\tau(3-\gamma))}} \to c.$$

We see that the number of triangles in this setting grows with rate $n^{1-\tau+(3\tau(3-\gamma)/2)}$. One interpretation is that the number of triangles is *decreasing* in the degree distribution's power-law exponent γ. This is as expected, as an increased power-law exponent leads to lighter tails in the distribution, so fewer highly connected degrees. The effect of the maximum community size n^τ is more involved. On the one hand, the number of triangles increases as community size is more restricted, as is reflected by the $1-\tau$ in the exponent. In larger communities, a vertex's neighbors are less likely to find each other. But by the exponent term $3\tau(3-\gamma)/2$, we see that a smaller τ also tampers the number of triangles. Notably, the power-law exponent of the community size β does **not** directly affect the asymptotic rate. The explanation, as we will see in the proof, is that the rate is the product of $n^{1-\tau(2-\beta)}$, the number of communities, and $n^{\tau(1-\beta)+(3\tau(3-\gamma)/2)}$, the typical number of triangles in the largest communities. The constants c depends on the value of ξ, as visualized in Figure 10(b), in that c is negatively correlated to the value of ξ.

Proof of Theorem 4.6. The proof follows the same approach to the proof of Theorem 4.3, where we considered the four types of triangles. We start with blue (fully within-community) triangles.

Three blue edges triangles. Consider $\mathbb{E}[Z_j]$ as defined in (22). We follow all steps analogous to the proof of Theorem 4.3 up to (23). Here, we can write the expected number of triangles as the conditional sum over the community sizes,

$$\mathbb{E}\Big[T^{3b}(G_n)\Big] = k\sum_{j=s}^{n^\tau} p_j \mathbb{E}\big[Z_j\big].$$

Now, pick some $\rho > 0$ such that

$$\rho < \min\left(\tau, \frac{2\tau(1-\beta)}{3(3-\gamma)} + 1\right). \tag{31}$$

We split the the sum into communities smaller than n^ρ and larger than n^ρ. We show that the first term is dominated by the second, i.e. the number of triangles in the smaller communities is insignificant compared to the triangles in the larger communities:

$$\mathbb{E}\Big[T^{3b}(G_n)\Big] = k\left(\sum_{j=s}^{n^\rho} p_j\mathbb{E}[Z_j] + \sum_{j=n^\rho}^{n^\tau} p_j^{(k)}\mathbb{E}[Z_j]\right). \tag{32}$$

For the first sum, we know by Theorem 4.5, the number of triangles is largest in the last term of the sum. Specifically, as the degree sequence b_u follows the power-law tail condition by Lemma 4.4. Then, by Theorem 4.5, we have that $\mathbb{E}[Z_{n^\rho}] = \Theta(n^{3\rho(3-\gamma)/2})$. So we write

$$\begin{aligned}\sum_{j=s}^{n^\rho} p_j\mathbb{E}[Z_j] &\le \sum_{j=s}^{n^\rho} p_j\mathbb{E}[Z_{n^\rho}]\\ &= \mathbb{P}\left(S \le n^\rho\right)\cdot\Theta\left(n^{\rho\frac{3}{2}(3-\gamma)}\right) = \mathcal{O}\Big(n^{\rho\frac{3}{2}(3-\gamma)}\Big).\end{aligned}$$

For the second term, as all terms are larger than n^ρ, they all follow the asymptotic behavior, such that

$$\sum_{j=n^\rho}^{n^\tau} p_j\mathbb{E}[Z_j] = \sum_{j=n^\rho}^{n^\tau} p_j\Theta(j^{\frac{3}{2}(3-\gamma)}).$$

We can approximate this sum by an integral, i.e. $\int_a^b f(x)dx + f(a) \le \sum_{i=a}^b f(i) \le \int_a^b f(x)dx + f(b)$. Here $a = n^\rho$ and $b = n^\tau$, and from $\rho < \tau$ and that f is increasing, we have $f(n^\rho) = \mathcal{O}(f(n^\tau))$. We can then write for some constant c' and c'' that

$$\begin{aligned}\sum_{j=n^\rho}^{n^\tau} p_j\Theta(j^{\frac{3}{2}(3-\gamma)}) &= c'\int_{n^\rho}^{n^\tau} x^{-\beta}x^{\frac{3}{2}(3-\gamma)}dx + \mathcal{O}\Big(n^{\tau(-\beta+\frac{3}{2}(3-\gamma))}\Big)\\ &= c''x^{1-\beta+\frac{3}{2}(3-\gamma)}\Bigg|_{n^\rho}^{n^\tau} + \mathcal{O}\Big(n^{\tau(-\beta+\frac{3}{2}(3-\gamma))}\Big)\\ &= \Theta\left(n^{\tau(1-\beta)+\frac{3}{2}\tau(3-\gamma)}\right) + \mathcal{O}\Big(n^{\tau(-\beta+\frac{3}{2}(3-\gamma))}\Big).\end{aligned}$$

Here, the last term is dominated by the first. Combining both the sums of (32), we then get

$$\mathbb{E}\Big[T^{3b}(G_n)\Big] = k\bigg(\mathcal{O}\Big(n^{\rho\frac{3}{2}(3-\gamma)}\Big) + \Theta\left(n^{\tau(1-\beta)+\frac{3}{2}\tau(3-\gamma)}\right)\bigg).$$

We see that the first term is dominated by the second, because of the choice of ρ from (31). Then again, using the asymptotic behavior of the number of communities k, as described by Theorem 4.2, we see

$$\begin{aligned}\mathbb{E}\Big[T^{3b}(G_n)\Big] &= \Theta\left(n^{1-\tau(2-\beta)}\right)\Theta\left(n^{\tau(1-\beta)+\frac{3}{2}\tau(3-\gamma)}\right)\\ &= \Theta\left(n^{1-\tau+\frac{3}{2}\tau(3-\gamma)}\right).\end{aligned}$$

Three red edges. To count the number of red triangles (triangles formed in the background graph), we simply note that the background graph has degree sequence r_u, which by Lemma 4.4 follows the power-law tail condition. Then we can directly apply Theorem 4.5 to see that

$$\mathbb{E}\Big[T^{3r}(G_n)\Big] = \Theta\left(n^{\frac{3}{2}(3-\gamma)}\right) = o(\mathbb{E}[T^{3b}(G_n)]).$$

Two red edges, one blue edge. We pick up for this case as in the proof for Theorem 4.3 in (24). We now take an upper bound using $b_u, d_u \leq d_u$ for all $u \in [n]$.

$$\begin{aligned}\mathbb{E}\Big[T^{2r,1b}(G_n)\;\Big|\;\mathbf{s},\mathbf{b},\mathbf{r}\Big] &= \frac{\sum_{u\in[n]} r_u(r_u-1)\sum_{l\in[k]}}{(\ell_n-1)(\ell_n-3)}\frac{\sum_{v\in C_l}\sum_{w\in C_l} r_v r_w b_v b_w}{\mathcal{C}_l - 1} + o(1)\\ &\leq \frac{1}{(\ell_n-1)(\ell_n-3)}\sum_{u\in[n]} d_u^2 \sum_{l\in[k]}\frac{1}{\mathcal{C}_l-1}\left(\sum_{v\in C_l} d_v^2\right)^2.\end{aligned}$$

Consider that, for some constant c', we can see, by using an integral upper bound of the sum, that

$$\sum_{u\in[n]} d_u^2 = n\sum_{j=\delta}^{n^\zeta}\mathbb{P}\left(d_u = j\right)j^2 = nc'\sum_{j=\delta}^{n^\zeta} j^{-\gamma}j^2 \leq nc'(n^\zeta)^{3-\gamma} = c'n^{1+\zeta(3-\gamma)}. \tag{33}$$

Also note that (for sufficiently large n), for $x = 1, 2$ or 3 and some constant c', we have $\ell_n - x \geq c'n$ and $\mathcal{C}_l - x \geq c''j$ if $|C_l| = j$. We also rewrite as the conditional sum over community sizes, similarly to (23). Combining this to see that for some other constant c' it holds that

$$\mathbb{E}\Big[T^{2r,1b}(G_n)\Big] \leq c'\frac{n^{1+\zeta(3-\gamma)}}{n^2}k\sum_{j=1}^{n^\tau}\frac{p_j}{j}\mathbb{E}[Z_j],$$

where now, for $j \in \mathbb{N}, j \geq s$,

$$\mathbb{E}\Big[Z_j^*\Big] := \mathbb{E}\left[\left(\sum_{v\in C_l} d_v^2\right)^2 \;\middle|\; |C_l| = j\right]. \tag{34}$$

Observe now that the $\mathbb{E}[Z_j^*]$ over j is increasing, and we find an upper bound for for the largest possible community analogously to (33), to see that for some constant c' it holds that.

$$\mathbb{E}\big[Z_{n^\tau}^*\big] \leq c'(n^\tau n^{\zeta(3-\gamma)})^2 = c'n^{2\tau+2\zeta(3-\gamma)}.$$

Then by using $p_j = j^{-\beta}$, we write

$$\begin{aligned}
\mathbb{E}\Big[T^{2r,1b}(G_n)\Big] &\le c' n^{\zeta(3-\gamma)-1} k \sum_{j=1}^{n^\tau} j^{-1-\beta} n^{2\tau+2\zeta(3-\gamma)} \\
&= c' k n^{\zeta(3-\gamma)-1+2\tau+2\zeta(3-\gamma)} \sum_{j=1}^{n^\tau} j^{-1-\beta} \\
&= \mathcal{O}\Big(k n^{\zeta(3-\gamma)-1+2\tau+2\zeta(3-\gamma)-\tau\beta}\Big) = \mathcal{O}\Big(k n^{3\zeta(3-\gamma)+\tau(2-\beta)-1}\Big).
\end{aligned}$$

To ensure this result is dominated by the number of blue triangles, i.e. $o(\mathbb{E}[T^{3b}(G_n)])$, we need to have that

$$\begin{aligned}
& 3\zeta(3-\gamma) + \tau(2-\beta) - 1 \le \tau(1-\beta) + \frac{3}{2}\tau(3-\gamma) \\
\iff & 3\zeta(3-\gamma) \le -\tau + 1 + \frac{3}{2}\tau(3-\gamma) \\
\iff & \zeta \le \frac{1}{2}\tau + \frac{1-\tau}{3(3-\gamma)},
\end{aligned}$$

which is true as per the assumption.

Two red edges, one blue edge. We pick up this case from (24), where we now take an upper bound using the degree d_i instead of the split degrees r_i and b_i. This gives

$$\begin{aligned}
\mathbb{E}\Big[T^{2b,1r}(G_n) \;\Big|\; \mathbf{s}, \mathbf{b}, \mathbf{r}\Big] &= \frac{1}{2} \sum_{l\in[k]} \sum_{u,v,w\in C_l} \frac{r_u r_v}{\ell_n - 1} \frac{b_v b_w (b_w - 1) b_u}{(\mathcal{C}_l - 1))(\mathcal{C}_l - 3)} \\
&\le \sum_{l\in[k]} \sum_{u,v,w\in C_l} \frac{d_v^2}{n} \frac{d_u^2 d_w^2}{(\mathcal{C}_l - 1))(\mathcal{C}_l - 3)} \\
&\le \frac{1}{n} \sum_{l\in[k]} \frac{(\sum_{v\in C_l} d_v^2)^3}{(\mathcal{C}_l - 1))(\mathcal{C}_l - 3)}.
\end{aligned}$$

Then we rewrite as the conditional sum over communtiy sizes

$$\mathbb{E}\Big[T^{2b,1r}(G_n)\Big] \le k \frac{1}{n} \sum_{j=1}^{n^\tau} j^{-\beta-2} \mathbb{E}\Big[Z_j^{**}\Big],$$

where now, for $j \in \mathbb{N}, j \ge s$,

$$\mathbb{E}\Big[Z_j^{**}\Big] := \mathbb{E}\left[\left(\sum_{v\in C_l} d_v^2\right)^3 \;\middle|\; |C_l| = j\right], \tag{35}$$

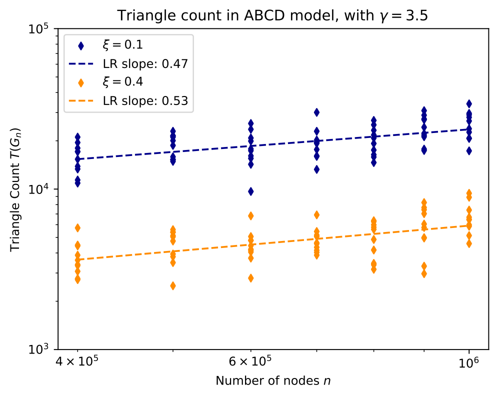
and like before $\mathbb{E}[Z^{**}_{n^\tau}] \leq n^{3\tau} n^{3\zeta(3-\gamma)}$. And then we get, using a bound as in (33),

$$\begin{aligned}\mathbb{E}\Big[T^{2b,1r}(G_n)\Big] &\leq \frac{k}{2} n^{3\tau+3\zeta(3-\gamma)-1} \sum_{j=1}^{n^\tau} j^{-\beta-2} \\ &\leq \mathcal{O}\Big(kn^{3\tau+3\zeta(3-\gamma)-1-\tau(1+\beta)}\Big) = \mathcal{O}\Big(kn^{3\zeta(3-\gamma)+\tau(2-\beta)-1}\Big).\end{aligned}$$

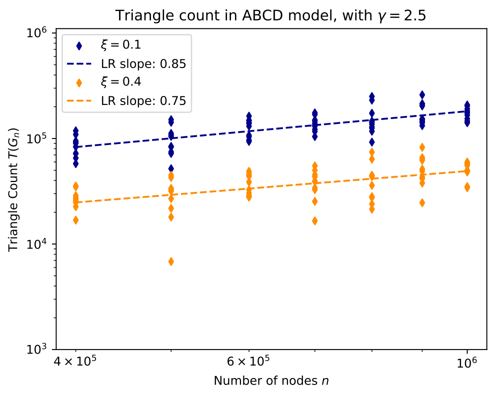
This is the same upper bound as for $\mathbb{E}[T^{2b,1r}(G_n)]$, which we have shown to be $o(\mathbb{E}[T^{3b}(G_n]))$.

Conclusion of proof. We have shown that all other types of triangles are dominated by the blue (within-community) triangles, and that the blue triangles grow by the required rate, which completes the proof. $\square$

Both Theorem 4.3 and Theorem 4.6 give the asymptotic rate of growth of the expected triangle count $\mathbb{E}[T(G_n)]$. They state two regimes in which the relation $\mathbb{E}[T(G_n)] = cn^r$ holds, for some unknown c depending on the parameters, with rate $r = (1-\tau(2-\beta))$ in Theorem 4.3 and $r = (1-\tau+3\tau(3-\gamma)/2)$ in Theorem 4.6. The goal of this section was to show that triangles can quantify the strength of the community structure in the ABCD model. As we have not shown how c depends on ξ, this is still an open question. We address this by a short simulation experiments, shown in Figure 10.



(a) In the regime of Theorem 4.3



(b) In the regime of Theorem 4.6

Figure 10 – Simulation results for the number of triangles. Each marker in the scatter plot corresponds to an independently generated ABCD graph, with $D_n \sim \mathcal{P}(\gamma, 3, n^{0.3}), S_n \sim \mathcal{P}(1.5, 10, n^{0.9})$ and $\gamma = 3.5$ in (a) and $\gamma = 2.5$ in (b). The dashed lines are the linear regression fit of $\log T(G_n) = \hat{c} + \hat{r} \log n$. Then $\hat{c}, \hat{r}$ are estimators for c, r in the relation $\mathbb{E}[T(G_n)] = cn^r$. Following Theorem 4.3 we know that in (a), the slope r is 0.55. The simulations estimates are 0.47 and 0.53, for intercept c the estimate is 3.63 for $\xi = 0.1$ and 1.33 for $\xi = 0.4$. From Theorem 4.6, we know that in (b) the slope is 0.775. The estimates are 0.85 and 0.75, for intercept c is given by 0.32 for $\xi = 0.1$ and 0.48 for $\xi = 0.4$.

In the loglog plot, we see the linear relation $\log \mathbb{E}[T(G_n)] = c + r \log n$. Since the slope of the relation appears the same between the choices of ξ, this supports our results that r does not depend on ξ. The separation of the point clouds confirms the conclusion that a lower ξ (a stronger community structure) corresponds to a higher value for c. This simulation is not without limitations. Our theoretic results are asymptotic, but at what point the asymptotic rate dominates is not clear. This simulation considers graphs of between $400,000$ and $1,000,000$ nodes, as computational restrictions forbid us from considering larger graphs, or more realizations.[5] As a consequence, the results deviate in some aspects from the theoretical expectations. For example, in Figure 10(b), the estimates for c with a lower ξ are lower than with a higher ξ, despite the fact that the point clouds are clearly separated.

[5]These restrictions are merely a consequence of the scope of this thesis. More performant implementations or more powerful hardware could fairly easily generate graphs of orders of magnitude larger.

5 Triangle Augmentation for Graph Neural Networks

In the previous section we have seen that the triangle count can be used to quantify the community structure of a graph. With this in mind, we turn to a popular class of models for graph data mining. In recent years, Graph Neural Networks (GNNs) have been used with great success in supervised tasks on graph-structured data [82]. Consider $G = (V, E, H^0)$, an n-node graph with a corresponding label y, where $H^0 \in \mathbb{R}^{n \times d_0}$ contains the d_0-dimensional node features. The goal of a GNN is to learn the graph function that maps G to y, i.e a GNN *predicts* y from G. The node feature H^0 may consist of inherent node attributes, but if those are absent, it is customary to simply let H^0 be the degree of each node, or an all-ones vector. In this section we inspect the benefit of augmenting H^0 with the triangle count of each node. As we will see, there is a good reason to do so, as GNNs have trouble detecting local clustering and community structure. Before we investigate this, we explain the general workings of a GNN.

On a high level, the GNN architecture for supervised graph-level tasks (such as graph classification), consists of three steps: convolution, readout and prediction, as visualized in Figure 11. We explain these steps in detail.

1. **Graph convolution layers**. In this first step, also called the *message passing* or *neighborhood aggregation* step, learnable convolution-like layers find node embeddings by aggregating the features of neighboring nodes. For each layer l in some k layers, for each vertex $v \in [n]$, with $\mathcal{N}(v)$ denoting the neighborhood of vertex v, the graph convolution layer is given in [53] by

$$H_v^l = \sigma\left(H_v^{l-1}W_1^l + \sum_{w \in \mathcal{N}(v)} H_w^{l-1}W_2^l\right).$$

   Here $\sigma : \mathbb{R}^{n \times d_l} \mapsto \mathbb{R}^{n \times d_l}$ is some dimension-wise differentiable activation function, and $W_1^l, W_2^l \in \mathbb{R}^{d_{l-1} \times d_l}$ are the parameter (*weight*) matrices of the layer, where d_l is referred to as the "hidden dimension" of layer l.

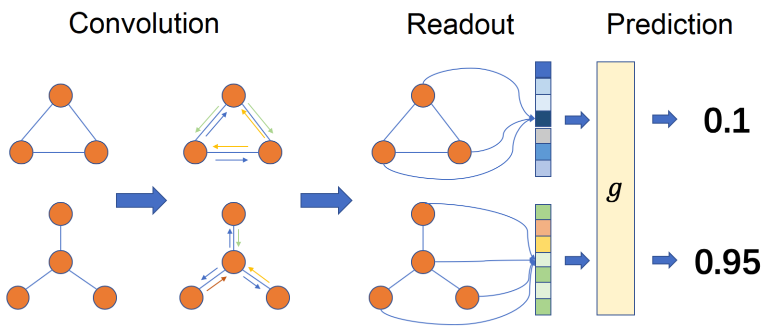


Figure 11 – Illustration of GNN architecture (Adapted from [20])

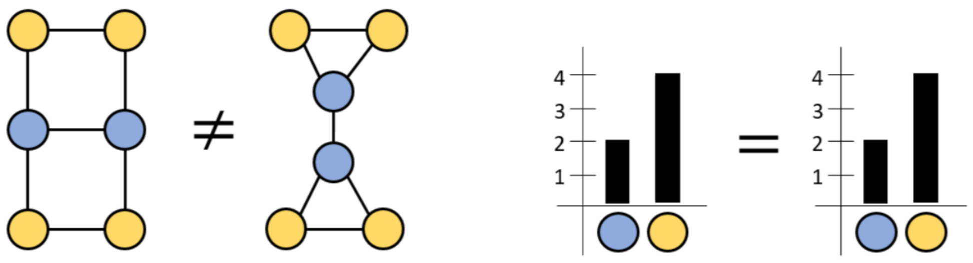
2. **Readout function**: After k consecutive graph convolution layers, the node embeddings are combined by the readout function to obtain one graph embedding. To ensure that the readout function is well-defined for any graph, this operation has to be *invariant* under the permutation of the node embeddings (as a graph after a permutation of the the nodes defines the same graph) and *flexible* to the number of nodes n:
$$H_g = f_{\text{readout}}(H^k),$$
where H^k is the output of the last convolution layer, $f_{\text{readout}} : \mathbb{R}^{n\times d_k} \mapsto \mathbb{R}^{d_g}$ any dimension-wise differentiable function that is invariant and flexible, and $H_g \in \mathbb{R}^{d_g}$ is the graph embedding of dimension d_g. Common choices for the readout function are the dimension-wise mean or maximum.

3. **Prediction function**: Then, the graph embedding H_g is used to do classification or regression of the graph's label, through some learnable prediction function,
$$\hat{y} = f_{\text{pred}}(H_g \mid W_p),$$
where H_g is the graph embedding obtained from the readout function, f_{p} a differentiable function with weights W_p. Generally, f_{pred} consists of 1 or more fully connected layers. Then, $\hat{y}$ is the prediction of the model. In the regression setting, $\hat{y}$ is simply a single real number. In classification, $\hat{y}$ is a vector of the length of the number of classes. This is called soft classification, and each entry represents the predicted class probability.

For a GNN with k convolution layers, the network has the learnable weights $(W_1^1, W_2^1)\dots(W_1^k, W_2^k), W_p$. These weights are optimized in an end-to-end fashion using a stochastic gradient descent variant, on a training set of pairs of graphs and their labels. The performance of the model is then measured on a validation set of unseen examples.

5.1 The 1-dimensional Weisfeiler-Lehman Test

Although GNNs are the preferred choice for many real world applications, it can be shown that they are at most as expressive as the 1-dimensional Weisfeiler-Lehman algorithm (1-WL), a heuristic test for isomorphism of graphs [83]. Here, we aim to explain what the 1-WL test is, and why it shows up in the analysis of GNNs. The graph isomorphism problem is to determine if, given two graphs, there exists a permutation of the nodes such that the graphs are identical. The 1-WL algorithm gives a heuristic test for this problem. It relies on a version of *color refinement*, an iterative procedure: start with an initial labeling (i.e. coloring) of the nodes of both graphs, for example their degree. In each iteration, two nodes with the same label get different labels if the number of neighbors with a particular label is not equal. This step is similar to the graph convolution layers: the representation of two nodes at layer $l + 1$ is different only if the representations at layer l of the nodes in their neighborhoods are different. The 1-WL algorithm terminates when refinement does not change the node labels. In this view, a GNN with k layers can be viewed as k refinement steps of the 1-WL algorithm on each node. By running the algorithm in parallel for two graphs, the 1-WL test concludes the graphs are non-isomorphic if their distributions of the labels are different. The conclusion in the opposite direction is not true: identical color histograms does not necessarily mean the graphs are isomorphic. The 1-WL test is generally expressive enough for practical applications, but it fails in many simple cases. For example, it cannot distinguish different d-regular graphs, graphs with different triangle counts or, in general, cyclic information [2]. See, for example, Figure 12. The label distribution for two clearly non-isomorphic graphs, one with triangles and one without triangles, is identical.



(a) Two non-isomorphic graphs **(b)** 1-WL label distribution

Figure 12 – 1-WL and GNNs cannot distinguish these two graphs. Source [89]

So, the graph convolution step of a GNN is at most as powerful as the 1-WL test, and can even be formulated to be exactly as powerful [83, 53]. It is clear that from identical node embeddings (i.e. the label distribution such as in Figure 12(b)), the readout step will lead to the same graph embedding, so that the two graphs cannot be distinguished in the prediction step.

Before we move on to our experiment, let us consider why GNNs are designed this way, and why improving their expressiveness is difficult. Fundamentally, this is because a GNN has to be permutation *invariant*: a permutation of the vertices should not change the output, as the graph does not change under permutations of the vertices either. GNNs ensure invariance by applying the same weights in the convolution layers to all nodes. However, this leads to a lack of identifiability between the nodes [24], and as a consequence GNNs cannot predict clustering coefficients [85], count any subgraph of three or more nodes [12] or recover community structure [70].

The expressiveness of GNNs is an open question in research, see Morris et al. [52] for a recent comprehensive summary. To address this issues, many variations have been suggested, such as higher-order architectures [53, 29], unique node identifiers [76], dropout layers [55], or subgraph augmentation [33, 8, 15]. Here, we consider an experiment based on the last suggestion. Inspired by the content of the earlier sections of this thesis, we consider the simplest subgraph with three or more nodes: the triangle. We augment the node features of the graphs with the number of triangles each node is part of, and show that this allows a minimal GNN architecture to detect community structure, learn to

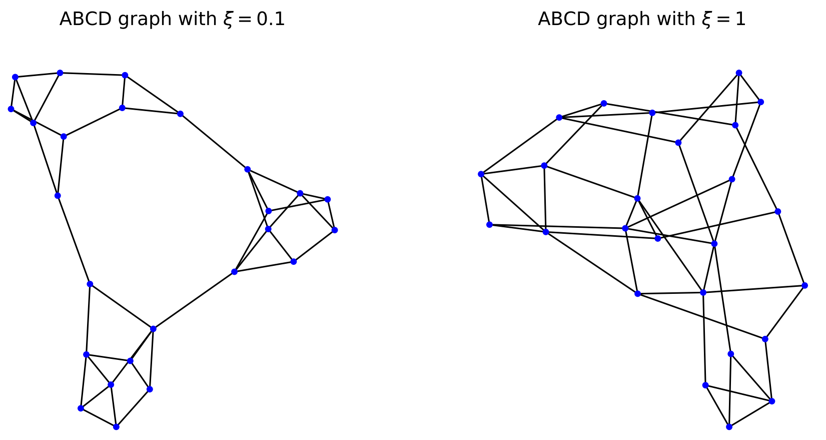


Figure 13 – Two graphs from the two classes in the ABCD dataset.

calculate local clustering and improve graph classification performance on real-life data.

5.2 Experiment

In this experiment, we use GNN with 3 graph convolution layers, a hidden representation dimension of 10, the mean readout function and a fully connected layer of dimension 10 for the final prediction (single value for regression and a class probability for soft classification). For more details on the model, training procedure and loss curves, we refer to Section A in the Appendix. This GNN is small (between 251 and 272 weights, see Table 3 in the Appendix.) and does not represent the state-of-the-art, but is sufficient for this experiment. We consider the following three datasets.

1. ABCD - Classification. The dataset consists of 400 graphs of the ABCD model, introduced in Section 4.3, with parameters $n = 25, \beta = 1.5, s = 7, \tau = 0.9, \gamma = 2.9, \delta = 3$ and $\zeta = 0.5$. Half of the graphs have $\xi = 1$ (no community structure) and half have $\xi = 0.1$, see a graph from each class in Figure 13. Only varying ξ means that the two classes have the same expected degree sequence, so that the community structure is the only distinguishing characteristic between the classes.

2. Watts-Strogatz - Regression. This dataset aims to test the GNN's ability to detect local clustering. We quantify this by the Global Clustering Coefficient (GCC) [36], a graph-level statistic given by the number of closed triplets over the total number of triplets (both open and closed). The Watts-Strogatz random graph model WS(n, K, p) on a ring lattice is generated in two steps. First, create a ring lattice with n nodes, and connect each node to its K closest neighbors in the ring, i.e. $K/2$ on both sides. Then for each node, take all edges to its neighbors on the right, and rewire them randomly to any

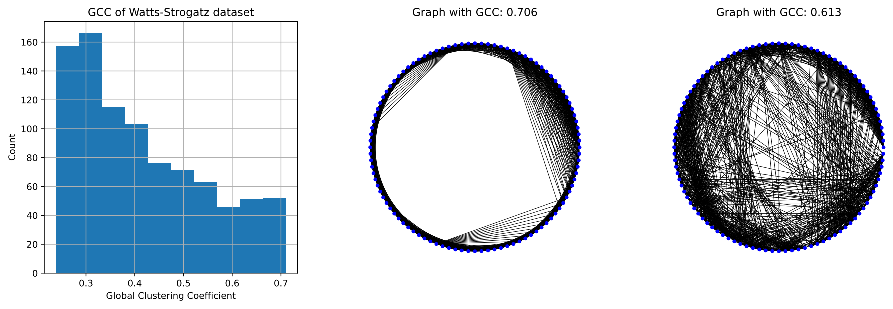


Figure 14 – The Watts-Strogatz dataset

other node, avoiding self-loops and multi-edges. For more details, see [79]. We use this model because the Global Clustering Coefficient decreases as the rewiring probability p increases. Using this relation, we generate a relatively balanced labelled regression dataset, as shown in the histogram in Figure 14. We generate 900 graphs, each with $n = 100$ and $K = 10$, varying p between 0 and 1. Note that in this dataset, each graph has $nK/2 = 500$ edges.

3. PROTEIN - Classification. The PROTEIN dataset [7] is a popular benchmark for graph classification. It consists of 1113 graphs representing proteins, labelled as enzymes (59%) or non-enzymes (41%) . Nodes represent the amino acids and two nodes are connected by an edge if the amino acids are less than 6 angstrom apart. On this dataset, the state of the art achieves a classification accuracy of 84.91% [88], which includes using the node attributes as node features. Here, we consider just the graph structure, and drop the node attributes. This means our results are not comparable to the state of the art. We do this, because we want to consider what signal the model can retrieve from just the graph structure, and how triangle augmentation improves this.

For each of the tree datasets, we train our GNN twice: once using only the degree as the node feature, and once using both the degree and the triangle count.

5.3 Results

The results are displayed in Table 2. For the loss curves of the training runs, see Figure 15 in the Appendix. We see that without triangle augmentation, the ABCD classification and Watts-Strogatz regression cannot be solved. The classification accuracy on the ABCD validation set is 45%, worse than random classification. With triangle augmentation, the classification is 100%. Similarly, we see that the GNN perfectly learns the graph function for the Global Clustering Coefficient on the Watts-Strogatz dataset. For the PROTEIN dataset, we see that with triangle augmentation, the accuracy improves from 59.1% to 64.13%. Note that the dataset is imbalanced, meaning that a constant "enzym" classification would lead to an accuracy of 59.1%. This means that without triangle augmentation, the graph structure is not informative to the classification. With

	Without triangle augmentation		With triangle augmentation	
	Loss	*On validation set*	*Loss*	*On validation set*
	Cross Entropy	**Accuracy**	**Cross Entropy**	**Accuracy**
ABCD	0.693	45%	0.102	100%
	MSE	**R2**	**MSE**	**R2**
Watts–Strogatz	0.017	0.00	1.02e−6	1.00
	Cross Entropy	**Accuracy**	**Cross Entropy**	**Accuracy**
PROTEIN	0.675	59.1%	0.632	64.13%

Table 2 – GNN prediction results on the three datasets, without triangle augmentation (just using degree as node feature) and with using triangle augmentation (using degree and triangle count as node features).

triangle augmentation, the signal a GNN can mine from real-life graph structures is improved.

In this experiment, we illustrate that GNNs perform very poorly in detecting community structure and local clustering. Those features are much present in real-life networks, and this presents a strong drawback to using the GNN architecture. We also show that the simple fix of triangles augmentation allows the GNN to pick up both community and local clustering, and that this improves performance on a real-life dataset. This shows, in another setting, how the triangle subgraph can quantify the community structure in graphs. We also notice a connection of NEExT as the unsupervised, interpretable alternative to GNNs, both relying on local features for graph-level tasks.

6 Conclusion and Discussion

In this thesis, we looked at two theoretical aspects of the NEExT framework for generating unsupervised graph embeddings, and we made a connection the Graph Neural Networks expressiveness. We give a result for proportional sampling, and show that even as the sampling fraction goes to zero, sufficiently slowly, the Wasserstein distance vanishes. Then, we inspect how the triangle count, a feature that can be calculated locally, can quantify the strength of a community structure. For this we looked at two random graph models. For the Stochastic Block Model, we show that asymptotic number of triangles in a 2-community setting. For the ABCD model, we find the rate of growth of the number of triangles when the degree distribution has finite variance. In the scale-free regime, with power-law degree distribution, we give the rate of growth of the triangle count for sufficient conditions on the parameters. Lastly, we see how the triangle count can address the inability of Graph Neural Networks in detecting community structure and clustering.

For sampling convergence, we show in Theorem 3.4 that the finitely supported measure $\mu_s^{(n)}$ on $[0,1]^d$, on a sample of $s := n^\sigma$ out of n points, is, in the p-Wasserstein sense, asymptotically $\mathcal{O}(n^{-\sigma/(d+3p)+\varepsilon})$ removed from the measure $\mu^{(n)}$ supported on all of the n points. This result is more general than the intended application of sampling node features requires, in the sense that we make no assumption on the distribution of the node features. Structural node features between adjacent nodes are not independent, and depending on the specific feature, the structure of the underlying graph may be exploited to sample nodes in a more optimal way. For example, many structural node features are correlated with degree. Consider random-friend sampling of the nodes: instead of uniformly drawing a node, pick uniformly at random a neighbor from a uniformly drawn node. This way, highly-connected nodes are more likely to be sampled, and we have *size-biased sampling*, which may lead to sharper convergence [61] depending on the degree distribution. If some community paritition of the graph's vertices is known, or if it is calculated for a community-aware feature [42], faster convergence may be obtained by sampling proportionally from each community. This leads to *stratified sampling* [63]. As structural features have been shown to vary between community [38], this could be another method of obtaining faster convergence. An approach for obtaining results for these directions could be to assume a random graph model, find the distribution of a node feature on that model, and then inspect the consequence of a sampling methodology. See, for example, how the connection between community and Personalized PageRank can be made explicit in the Stochastic Block Model [45]. A benefit of this random graph approach is that simulation experiments are straightforward.

We give asymptotic results on the triangle count in 2-community Stochastic Block Model (Theorem 4.1), the ABCD model with finite-variance degree distribution (Theorem 4.3) and the ABCD model with power-law degree distribution (4.6). The key insight for the different cases is that in the finite-variance degree setting, large communities have asymptotically constant triangles, while when degree distribution follows a power law with infinite variance, the number of triangles increases. As a consequence, in the first case the triangle count grows proportionally with the number of communities. In the second case, we find, under sufficient conditions on the parameters of the model, that the rate of growth is determined by large communities, corrected for their frequency of occurrence. The results for the ABCD model give a way of comparing the strength of their community structure of two graphs, possibly of different sizes, by triangle count. However, we find no explicit relation between the triangle count and the noise parameter of interest ξ. An extensive simulation experiment could be done to find the relation empirically. We conjecture that c, as in Theorem 4.3 and 4.6, is proportional to $(1-\xi)^3$, as this describes the probability of a given triangle within a community. Another direction is to quantify community structure using other subgraphs than the triangle. We have seen that the triangle count increases asymptotically, but this is not necessarily true for larger cycles. We formulate, by experiment, two random graph problems a GNN cannot solve without triangles, but can perfectly solve with triangle augmmentation. Then, we see that triangle augmentation also improves the performance of a GNN on a real-life dataset. Specifically, we see that both community structure and local clustering cannot be detected in graph-level tasks. The ABCD model appears as powerful model that can create graphs with community structures that GNNs by default cannot detect. The method of generating random graphs with a known structure presents itself as a powerful method for empirically testing the limitations of GNNs. Further research could focus on more extensive experiments, and augmentations with different subgraphs.

Lastly, we pose another use case for NEExT in connection to GNNs. Consider some GNN trained on some downstream full-graph task, for example graph classification, or molecule property prediction. The hidden layers of the GNN are learned representations/embeddings of the input data. Since a GNN is a deep learning “black box”, we do not know what properties of the graph these embeddings encode. An idea we would like to suggest is to use graph embeddings of NEExT to *probe* the learned representations, to understand what *node* features have been learned. Probing is a method that attempts to explain, analyze and interpret the learned representations in deep neural networks [5]. Take a deep model f trained for a specific task, let $f_l(x)$ be the neuron activation of the l-th layer for input x. This is the learned representation of x, which is then fed into a *probing classifier* g, which is trained to extract some property $z(x)$ from $f_l(x)$. If g performs well, we can conclude that the learned representation f_l encodes z. There exists some work on probing GNNs, in the setting of molecule property prediction [1] and Graph Self-Supervised Learning [78]. Using the graph embeddings from NEExT would allow probing of node features. This may lead to new insights about the correlation of local features on whole-graph properties.

References

[1] Mohammad Sadegh Akhondzadeh, Vijay Lingam, and Aleksandar Bojchevski. "Probing Graph Representations". In: *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. Ed. by Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent. Vol. 206. Proceedings of Machine Learning Research. PMLR, Apr. 2023, pp. 11630–11649. URL: `https://proceedings.mlr.press/v206/akhondzadeh23a.html`.

[2] Vikraman Arvind et al. "On the power of color refinement". In: *Fundamentals of Computation Theory: 20th International Symposium, FCT 2015, Gdańsk, Poland, August 17-19, 2015, Proceedings 20*. Springer. 2015, pp. 339–350.

[3] Niousha Attar and Sadegh Aliakbary. "Classification of complex networks based on similarity of topological network features". In: *Chaos: An interdisciplinary journal of nonlinear science* 27.9 (2017).

[4] Albert-László Barabási and Eric Bonabeau. "Scale-free networks". In: *Scientific american* 288.5 (2003), pp. 60–69.

[5] Yonatan Belinkov. "Probing Classifiers: Promises, Shortcomings, and Alternatives". In: *CoRR* abs/2102.12452 (2021). arXiv: `2102.12452`. URL: `https://arxiv.org/abs/2102.12452`.

[6] Béla Bollobás. "Threshold functions for small subgraphs". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 90. 2. Cambridge University Press. 1981, pp. 197–206.

[7] Karsten M. Borgwardt et al. "Protein function prediction via graph kernels". In: *Bioinformatics* 21 (June 2005), pp. i47–i56. ISSN: 1367-4803. DOI: `10.1093/bioinformatics/bti1007`. URL: `https://doi.org/10.1093/bioinformatics/bti1007`.

[8] Giorgos Bouritsas et al. "Improving graph neural network expressivity via subgraph isomorphism counting". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.1 (2022), pp. 657–668.

[9] Sergey Brin and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine". In: *Computer networks and ISDN systems* 30.1-7 (1998), pp. 107–117.

[10] James P Canning et al. "Predicting graph categories from structural properties". In: (2018).

[11] Suman Chakraborty, Remco van der Hofstad, and Frank den Hollander. "Sparse random graphs with many triangles". In: *arXiv preprint arXiv:2112.06526* (2021).

[12] Zhengdao Chen et al. *Can Graph Neural Networks Count Substructures?* 2020. arXiv: `2002.04025`.

[13] Fan Chung and Linyuan Lu. "Connected components in random graphs with given expected degree sequences". In: *Annals of combinatorics* 6.2 (2002), pp. 125–145.

[14] Philippe Clement and Wolfgang Desch. "An elementary proof of the triangle inequality for the Wasserstein metric". In: *Proceedings of the American Mathematical Society* 136.1 (2008), pp. 333–339.

[15] Hejie Cui et al. “On positional and structural node features for graph neural networks on non-attributed graphs”. In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022, pp. 3898–3902.

[16] Marco Cuturi and Arnaud Doucet. “Fast Computation of Wasserstein Barycenters”. In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, 2014, pp. 685–693. URL: `https://proceedings.mlr.press/v32/cuturi14.html`.

[17] George B Dantzig. “Application of the simplex method to a transportation problem”. In: *Activity analysis and production and allocation* (1951).

[18] Ashkan Dehghan, Paweł Prałat, and François Théberge. “Network Embedding Exploration Tool (NEExT)”. In: *Modelling and Mining Networks*. Ed. by Megan Dewar et al. Cham: Springer Nature Switzerland, 2024, pp. 65–79. ISBN: 978-3-031-59205-8.

[19] Yue Deng. “Recommender systems based on graph embedding techniques: A review”. In: *IEEE Access* 10 (2022), pp. 51587–51633.

[20] DGL. *Deep Learning Library 0.8x*. `https://docs.dgl.ai/en/0.8.x/guide/training-graph.html`. [Online; accessed 16-June-2024]. 2018.

[21] Persi Diaconis and Svante Janson. “Graph limits and exchangeable random graphs”. In: *arXiv preprint arXiv:0712.2749* (2007).

[22] Vladimir Dobric and Joseph E. Yukich. “Asymptotics for transportation cost in high dimensions”. In: *Journal of Theoretical Probability* 8 (1995), pp. 97–118. URL: `https://api.semanticscholar.org/CorpusID:119876550`.

[23] Lutz Dümbgen. “On nondifferentiable functions and the bootstrap”. In: *Probability Theory and Related Fields* 95 (1993), pp. 125–140.

[24] Vijay Prakash Dwivedi et al. “Graph neural networks with learnable structural and positional representations”. In: *arXiv preprint arXiv:2110.07875* (2021).

[25] Stanley C Eisenstat and Ilse CF Ipsen. “Relative perturbation techniques for singular value problems”. In: *SIAM Journal on Numerical Analysis* 32.6 (1995), pp. 1972–1988.

[26] Paul Erdős, Alfréd Rényi, et al. “On the evolution of random graphs”. In: *Publ. math. inst. hung. acad. sci* 5.1 (1960), pp. 17–60.

[27] Zheng Fang and Andres Santos. “Inference on directionally differentiable functions”. In: *arXiv preprint arXiv:1404.3763* (2014).

[28] Kilian Fatras et al. *Learning with minibatch Wasserstein : asymptotic and gradient properties*. 2021. arXiv: `1910.04091 [stat.ML]`.

[29] Jiarui Feng et al. “How powerful are k-hop message passing graph neural networks”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 4776–4790.

[30] Pu Gao et al. “Counting Triangles in Power-Law Uniform Random Graphs”. In: *The Electronic Journal of Combinatorics* 27 (3 Aug. 2020), P3.19–P3.19. ISSN: 1077-8926. DOI: `10.37236/9239`. URL: `https://www.combinatorics.org/ojs/index.php/eljc/article/view/v27i3p19`.

[31] Marco Gori, Gabriele Monfardini, and Franco Scarselli. "A new model for learning in graph domains". In: *Proceedings. 2005 IEEE international joint conference on neural networks, 2005.* Vol. 2. IEEE. 2005, pp. 729–734.

[32] Aditya Grover and Jure Leskovec. "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining.* 2016, pp. 855–864.

[33] Walid Guettala and László Gulyás. "On the Power of Graph Neural Networks and Feature Augmentation Strategies to Classify Social Networks". In: *arXiv preprint arXiv:2401.06048* (2024).

[34] Remco van der Hofstad. *Random graphs and complex networks.* Vol. 1. Cambridge University Press, Jan. 2016, pp. 1–338. ISBN: 9781316779422. DOI: `10.1017/9781316779422`.

[35] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. "Stochastic blockmodels: First steps". In: *Social networks* 5.2 (1983), pp. 109–137.

[36] Paul W Holland and Samuel Leinhardt. "Transitivity in structural models of small groups". In: *Comparative group studies* 2.2 (1971), pp. 107–124.

[37] Svante Janson. "Random graphs with given vertex degrees and switchings". In: *Random Structures & Algorithms* 57 (1 Aug. 2020), pp. 3–31. ISSN: 1098-2418. DOI: `10.1002/RSA.20911`. URL: `https://onlinelibrary.wiley.com/doi/full/10.1002/rsa.20911%20https://onlinelibrary.wiley.com/doi/abs/10.1002/rsa.20911%20https://onlinelibrary.wiley.com/doi/10.1002/rsa.20911`.

[38] Zhongzhou Jiang, Jing Liu, and Shuai Wang. "Traveling salesman problems with PageRank Distance on complex networks reveal community structure". In: *Physica A: Statistical Mechanics and its Applications* 463 (Dec. 2016), pp. 293–302. ISSN: 0378-4371. DOI: `10.1016/J.PHYSA.2016.07.050`.

[39] Wei Ju et al. "A Comprehensive Survey on Deep Graph Representation Learning". In: *Comprehensive Survey on Deep Graph Representation Learning. J. ACM* 1 (1 Apr. 2023), p. 85. URL: `https://arxiv.org/abs/2304.05055v2`.

[40] Bogumil Kaminski et al. *Modularity of the ABCD Random Graph Model with Community Structure.* 2022. arXiv: `2203.01480 [cs.SI]`.

[41] Bogumił Kamiński, Paweł Prałat, and François Théberge. "Artificial Benchmark for Community Detection (ABCD)—Fast random graph model with community structure". In: *Network Science* 9.2 (2021), pp. 153–178. DOI: `10.1017/nws.2020.45`.

[42] Bogumił Kamiński et al. *Predicting Properties of Nodes via Community-Aware Features.* 2024. arXiv: `2311.04730 [cs.SI]`.

[43] Shima Khoshraftar and Aijun An. "A survey on graph representation learning methods". In: *ACM Transactions on Intelligent Systems and Technology* 15.1 (2024), pp. 1–55.

[44] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[45] Isabel M Kloumann, Johan Ugander, and Jon Kleinberg. "Block models and personalized PageRank". In: *Proceedings of the National Academy of Sciences* 114.1 (2017), pp. 33–38.

[46] Soheil Kolouri et al. "Wasserstein Embedding for Graph Learning". In: *ICLR 2021 - 9th International Conference on Learning Representations* (June 2020). URL: https://arxiv.org/abs/2006.09430v2.

[47] Nils M Kriege, Fredrik D Johansson, and Christopher Morris. "A survey on graph kernels". In: *Applied Network Science* 5 (2020), pp. 1–42.

[48] Solomon Kullback and Richard A Leibler. "On information and sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[49] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. "Benchmark graphs for testing community detection algorithms". In: *Physical review E* 78.4 (2008), p. 046110.

[50] Yixin Liu et al. "Graph Self-Supervised Learning: A Survey". In: *IEEE Transactions on Knowledge and Data Engineering* (2022), pp. 1–1. ISSN: 2326-3865. DOI: 10.1109/tkde.2022.3172903. URL: http://dx.doi.org/10.1109/TKDE.2022.3172903.

[51] Yonghao Liu et al. "Deep attention diffusion graph neural networks for text classification". In: *Proceedings of the 2021 conference on empirical methods in natural language processing*. 2021, pp. 8142–8152.

[52] Christopher Morris et al. "Weisfeiler and leman go machine learning: The story so far". In: *The Journal of Machine Learning Research* 24.1 (2023), pp. 15865–15923.

[53] Christopher Morris et al. "Weisfeiler and leman go neural: Higher-order graph neural networks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 4602–4609.

[54] Mark EJ Newman and Michelle Girvan. "Finding and evaluating community structure in networks". In: *Physical review E* 69.2 (2004), p. 026113.

[55] P A Papp et al. "DropGNN: Random dropouts increase the expressiveness of graph neural networks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 21997–22009.

[56] Arnau Prat-Pérez et al. "Put three and three together: Triangle-driven community detection". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10.3 (2016), pp. 1–42.

[57] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. "struc2vec: Learning node representations from structural identity". In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2017, pp. 385–394.

[58] Ryan A Rossi and Nesreen K Ahmed. "Complex networks are structurally distinguishable by domain". In: *Social Network Analysis and Mining* 9 (2019), pp. 1–13.

[59] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. "A metric for distributions with applications to image databases". In: *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*. IEEE. 1998, pp. 59–66.

[60] Hiroto Saigo et al. “gBoost: a mathematical programming approach to graph classification and regression”. In: *Machine Learning* 75 (2009), pp. 69–89.

[61] R. L. Scheaffer. “Size-Biased Sampling”. In: *Technometrics* 14.3 (1972), pp. 635–644. ISSN: 00401706. URL: `http://www.jstor.org/stable/1267291` (visited on 06/17/2024).

[62] Robert J Serfling. “Probability inequalities for the sum in sampling without replacement”. In: *The Annals of Statistics* (1974), pp. 39–48.

[63] Ravindra Singh et al. “Stratified sampling”. In: *Elements of survey sampling* (1996), pp. 102–144.

[64] Max Sommerfeld and Axel Munk. “Inference for Empirical Wasserstein Distances on Finite Spaces”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 80.1 (May 2017), pp. 219–238. ISSN: 1369-7412. DOI: `10.1111/rssb.12236`. eprint: `https://academic.oup.com/jrsssb/article-pdf/80/1/219/49271410/jrsssb\_80\_1\_219.pdf`. URL: `https://doi.org/10.1111/rssb.12236`.

[65] Clara Stegehuis, Remco van der Hofstad, and Johan S H van Leeuwaarden. “Scale-free network clustering in hyperbolic and other random graphs”. In: *Journal of Physics A: Mathematical and Theoretical* 52.29 (June 2019), p. 295101. DOI: `10.1088/1751-8121/ab2269`. URL: `https://dx.doi.org/10.1088/1751-8121/ab2269`.

[66] Matteo Togninalli et al. “Wasserstein weisfeiler-lehman graph kernels”. In: *Advances in neural information processing systems* 32 (2019).

[67] Lourens Touwen et al. “Learning the mechanisms of network growth”. In: *Scientific Reports* 14.1 (2024), p. 11866.

[68] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. “From Louvain to Leiden: guaranteeing well-connected communities”. In: *Scientific reports* 9.1 (2019), p. 5233.

[69] José Trashorras and Olivier Wintenberger. “Large deviations for bootstrapped empirical measures”. In: *Bernoulli* 20.4 (2014), pp. 1845–1878. DOI: `10.3150/13-BEJ544`. URL: `https://doi.org/10.3150/13-BEJ544`.

[70] Anton Tsitsulin et al. “Graph Clustering with Graph Neural Networks”. In: *Journal of Machine Learning Research* 24.127 (2023), pp. 1–21. URL: `http://jmlr.org/papers/v24/20-998.html`.

[71] Koji Tsuda and Hiroto Saigo. “Graph Classification”. In: *Managing and Mining Graph Data*. Ed. by Charu C. Aggarwal and Haixun Wang. Boston, MA: Springer US, 2010, pp. 337–363. ISBN: 978-1-4419-6045-0. DOI: `10.1007/978-1-4419-6045-0_11`. URL: `https://doi.org/10.1007/978-1-4419-6045-0_11`.

[72] Aad W. van der Vaart and Jon A. Wellner. “Weak Convergence and Empirical Processes”. In: (1996). DOI: `10.1007/978-1-4757-2545-2`. URL: `http://link.springer.com/10.1007/978-1-4757-2545-2`.

[73] Johan SH Van Leeuwaarden and Clara Stegehuis. “Robust subgraph counting with distribution-free random graph analysis”. In: *Physical Review E* 104.4 (2021), p. 044313.

[74] Petar Veličković et al. *Deep Graph Infomax.* 2018. arXiv: `1809.10341 [stat.ML]`.

[75] Saurabh Verma and Zhi-Li Zhang. *Learning Universal Graph Neural Network Embeddings With Aid Of Transfer Learning.* 2019. arXiv: `1909.10086 [cs.LG]`.

[76] Clement Vignac, Andreas Loukas, and Pascal Frossard. "Building powerful and equivariant graph neural networks with structural message-passing". In: *Advances in neural information processing systems* 33 (2020), pp. 14143–14155.

[77] Chenguang Wang and Ziwen Liu. *Learning Graph Representation by Aggregating Subgraphs via Mutual Information Maximization.* 2021. arXiv: `2103.13125 [cs.LG]`.

[78] Hanchen Wang et al. *Evaluating Self-Supervised Learning for Molecular Graph Embeddings.* 2023. arXiv: `2206.08005 [cs.LG]`.

[79] Duncan J Watts and Steven H Strogatz. "Collective dynamics of 'small-world'networks". In: *nature* 393.6684 (1998), pp. 440–442.

[80] Jonathan Weed and Francis R. Bach. "Sharp asymptotic and finite-sample rates of convergence of empirical measures in Wasserstein distance". In: *Bernoulli* (2017). URL: `https://api.semanticscholar.org/CorpusID:51919254`.

[81] Oliver Wieder et al. "A compact review of molecular property prediction with graph neural networks". In: *Drug Discovery Today: Technologies* 37 (2020), pp. 1–12.

[82] Zonghan Wu et al. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[83] Keyulu Xu et al. "How powerful are graph neural networks?" In: *arXiv preprint arXiv:1810.00826* (2018).

[84] Minghao Xu et al. "Self-supervised Graph-level Representation Learning with Local and Global Structure". In: *Proceedings of the 38th International Conference on Machine Learning.* Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 11548–11558. URL: `https://proceedings.mlr.press/v139/xu21g.html`.

[85] Jiaxuan You et al. *Identity-aware Graph Neural Networks.* 2021. arXiv: `2101.10320`.

[86] Yuning You et al. "Graph Contrastive Learning with Augmentations". In: (NeurIPS Oct. 2020), pp. 1–12. URL: `https://arxiv.org/abs/2010.13902v3`.

[87] Zaixi Zhang et al. "Motif-based graph self-supervised learning for molecular property prediction". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 15870–15882.

[88] Zhen Zhang et al. "Hierarchical Graph Pooling with Structure Learning". In: (Nov. 2019). URL: `https://arxiv.org/abs/1911.05954v3`.

[89] Markus Zopf. "1-wl expressiveness is (almost) all you need". In: *2022 International Joint Conference on Neural Networks (IJCNN).* IEEE. 2022, pp. 1–8.

A Loss curves for Triangle Augmentation

The experiment was ran in Python 3.8 using the Deep Graph Library (DGL)[6], version 2.0.0, CUDA 121, which is built on top of PyTorch[7], version 2.1.1. The graph neural network consists of three Graph Convolutation layers (implemented in DGL as `GraphConv`). The graph convolution layer is given by [44]:

$$H_v^{(l+1)} = \sigma \left(b^{(l)} + \sum_{w \in \mathcal{N}(v)} \frac{1}{c_{vw}} H_w^{(l)} W^{(l)} \right)$$

for the neighborhood $\mathcal{N}(v)$ of vertex v, c_{vw} is a normalizing factor taken as the product of the square root of node degrees, and σ is some (non-linear) activation function.

We use hidden dimension 10 and the Rectified Linear Unit (ReLU, `torch.relu`) activation function, the mean pooling operation (implemented in DGL as `dgl.mean_nodes`), and fully connected layer (`torch.nn.Linear` in PyTorch) to one output variable (in the case of regression) or two output variables (in the case of soft classification with two classes). The loss function used for classification is Cross Entropy (`torch.nn.CrossEntropyLoss`) and for regression Mean Squared Error (`torch.nn.MSELoss`). The model is trained on a randomized $80 - 20$ stratified train-test split of each dataset. The model trains for 100 epochs with batch size 50. The Adam optimizer (`torch.optim.adam`) is used with learning rate 0.05. Every 25 epochs, the learning rate is divided by 10.

The number of weights in the model is a function of

1. The number of node features d_0. Without triangle augmentation when only the degree is used gives $d_0 = 1$ and $d_0 = 2$ when both degree and triangle-count are used.

2. The hidden dimension d_h. The hidden representation length of the three convolution layers and the linear prediction layer. We use $d_h = 10$.

3. The prediction label d_p, with $d_p = 1$ in the case of regression, and $d_p = 2$ in the case of soft classification.

[6]https://www.dgl.ai/
[7]https://pytorch.org/

Layer	in-dimension × out-dimension + bias	Calculation
GraphConv	$d_0 \cdot d_h + d_h$	$\{1,2\} \cdot 10 + 10$
GraphConv	$d_h \cdot d_h + d_h$	$10 \cdot 10 + 10$
GraphConv	$d_h \cdot d_h + d_h$	$10 \cdot 10 + 10$
Linear	$d_h \cdot d_p + d_p$	$10 \cdot \{1,2\} + \{1,2\}$
	Total weights	$= \{251, \ldots, 272\}$

Table 3 – Number of Weights in GNN

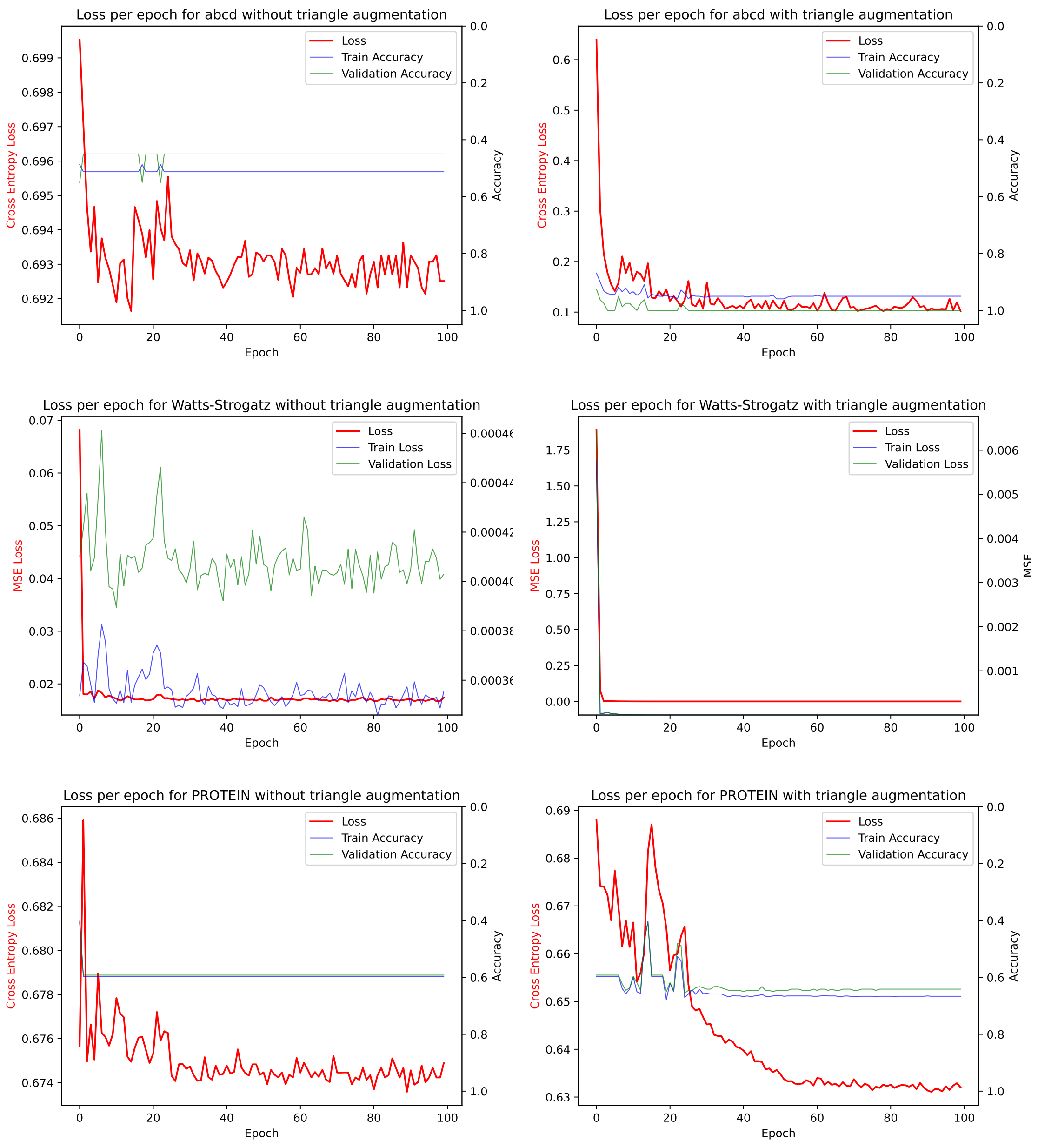


Figure 15 – Loss progression during training. The left y-axis gives the epoch's loss (Cross entropy for classification, MSE for regression), the right y-axis the evaluation on the train and test set (Cross entropy for classification, MSE for regression)