Attributed Labeled BTER-Based Generative Model for Benchmarking of Graph Neural Networks

Andreeva Polina
andreeva.polina06@gmail.com
National Center for Cognitive Research, ITMO University
Saint-Petersburg, Russian Federation

Bochenina Klavdiya
National Center for Cognitive Research, ITMO University
Saint-Petersburg, Russian Federation
k.bochenina@gmail.com

Shikov Egor
National Center for Cognitive Research, ITMO University
Saint-Petersburg, Russian Federation
egorshikov@itmo.ru

ABSTRACT

Graph Neural Networks (GNNs) have become increasingly popular for tasks such as link prediction, node classification, and graph generation. However, a number of models show weak performance on graphs with low assortativity measure. At the same time, other graph characteristics may also influence GNN quality. Therefore, it is extremely important for benchmark datasets to cover a wide range of different graph properties, which can not be provided by real-world sources. In this paper, we present a generative model for attributed graphs based on Block Two-Level Erdős-Rényi model. Our model allows one to vary larger number of graph structural characteristics (namely, clustering coefficient, average degree, average shortest paths length, label and attribute assortativity) in a wider range. Our attribute generative method can be applied to any other non-attributed graph generative model with community structure and allows to control attribute assortativity corresponding to structure of graph. The experimental study shows that AL-BTER outperforms ADC-SBM and GenCAT under the assumption of equal importance of desired graph characteristics and provides wider ranges for attribute assortativity and average shortest paths and outperforms LFR in terms of clustering coefficient. GNN performance analysis confirms the sensitivity of the results to all topological properties except average degree and shows that benchmark graphs provided by AL-BTER are useful to discover new regimes of performance of graph convolutional networks.

CCS CONCEPTS

• **Mathematics of computing** → *Random graphs*; • **Computing methodologies** → **Machine learning**; • **Networks** → **Network performance evaluation**.

KEYWORDS

benchmark, datasets, graph generator, synthetic generator,graph neural networks

ACM Reference Format:
Andreeva Polina, Bochenina Klavdiya, and Shikov Egor. 2022. Attributed Labeled BTER-Based Generative Model for Benchmarking of Graph Neural Networks. In *17th International Workshop on Mining and Learning with Graphs (Workshop'17)* , 8 pages.

Workshop'17, August 2022, Washington, DC, USA

1 INTRODUCTION

Nowdays, graph neural networks has become popular deep learning architecture for solving tasks in many domains, where data could be presented as interconnected objects. There are many examples of such data and tasks on them in genetics (classification of molecules and prediction of protein interactions [25]), social life (multi-label classification and friendships prediction [23]), research communities (collaboration prediction [18]), and so on.

Despite the fact that recent studies show state of the art performance of graph neural networks for classification and regression tasks on graph-structured data, some drawbacks of the graph neural networks has revealed [21]. The main shortcoming is in the architecture of message passing algorithm, which lies at the core of graph convolutional network (the most popular architecture for graph neural networks). Since it involves aggregation of messages from neighbors, the performance of convolutional networks increases on graphs with high homophily (by homophily or assortativity we mean a tendency to form relations with similar nodes). Consequently, convolutional networks show relatively poor performance on graphs, where neighbors have tangibly different labels [12, 19, 21]. Most of the modern benchmark graphs [9, 31] has high measure of homophily. Moreover, other graph structural characteristics may also influence the performance of GNN as will be shown in Section 2.1, and they should be taken into account while developing new benchmarks for GNN comparison.

Traditionally, the quality of graph neural networks has been measured on real-world networks [9, 11, 31]. However, the ranges of topological characteristics and attribute/label assortativities for them are significantly constrained. In turn, the approach based on the generation of synthetic graphs allows generating sets of graphs with a predetermined distribution of parameters. There exists a number of generators with certain inductive biases and a set of tunable graph properties. However, such generators allow to tune only small number of parameters. In this study, we are aimed at developing more flexible generator with larger number of tunable properties to support more comprehensive and systematic GNN comparison.

The main contributions of this work are as follows:

(1) we review and summarize graph characteristics which may influence the performance of graph convolutional networks;
(2) we propose a new Attributed Labeled BTER generator, which allows to vary all these characteristics;

(3) we propose an attribute generation method with two parameters which allows to vary attribute assortativity over any graphs with community structure;
(4) we show that our generator allows to create graphs with broader ranges of selected graph characteristics than other considered generators;
(5) we show that proposed generator is able to create sets of graph instances resulting in poor performance of convolutional networks, thus allowing to find the borders of applicability of a given neural network architecture.

Notations. Let $G = (V, E)$ be an undirected graph. $V = \{v_i\}_{i=1}^{n}$ is the set of nodes, where n is the number of nodes. $E = \{e_l\}_{l=1}^{m} \subset V \times V$ is the set of edges, where m is the number of edges and (v_i, v_j) is an edge between vertices v_i and v_j. $X \in \mathbb{R}^{n \times d}$ is a matrix of attributes of nodes, where d is the dimension of attribute vector and $x_i \in \mathbb{R}^d$ indicates the attribute vector of node v_i. Each node v_i has a label l_i, list L indicates list of labels and K is the number of distinct labels. $\mathcal{N}(v_i)$ is the set of neighbors of node v_i.

2 RELATED WORKS

2.1 Graph Properties

Recently, Pei et al. [21] has shown the failure of some graph convolutions on datasets with low assortativity because of the neighborhood aggregation mechanism. To tackle this problem, Geom-GCN was suggested by Pei et al. [21]. It aggregates not only adjacent nodes but also those nodes that in latent space are not far from the considered vertex. Since then, there have been developed some other methods [4, 12, 13] for tackling disassortative graphs.

However, assortativity is not the only parameter which influences the performance of graph convolutional networks. Some other graph properties can also be decisive. For instance, for self-supervised SuperGAT [13] algorithm, it is shown that the result depends on the assortativity and average degree of the graph, simultaneously. Klicpera et al. [15] conclude that the length of the average shortest path of a graph may influence the performance in the manner as assortativity does — if the average shortest path is long, then few layers of graph convolutional network cannot cover all nodes, which may be important.

Attribute assortativity also should be considered as features in many methods act as messages in message passing procedure. In addition, the clustering coefficient should be addressed, as some similarity measures of unsupervised loss functions are based on it, and it can vary for different graphs.

So, properties which should be considered while benchmarking graph convolutional networks are: *average degree* $\mathbf{d_{avg}}$, *average clustering coefficient* **c**, *average shortest paths length* **asp**, *label assortativity* **la**, *attribute assortativity* **aa**.

We use the definition of label assortativity proposed in [21]:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|\{v_j \in \mathcal{N}(v_i) : l_i == l_j\}|}{|\mathcal{N}(v_i)|}. \tag{1}$$

We define attribute assortativity *aa* similarly to label assortativity:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|\{v_j \in \mathcal{N}(v_i) : cos(x_i, x_j) < q\}|}{|\mathcal{N}(v_i)|}, \tag{2}$$

where q is a threshold, we set $q = 0.5$, but it can be potentially chosen to any other number if the problem requires that.

2.2 Parametric Generators

In this paper we consider methods (which we call parametric generators), which imply building a graph with specific characteristics, which act as input parameters, for instance, by specifying degree sequence. Other existing typologies of synthetic graph generators could be found in [5, 29].

There are two types of parametric generators depending on whether a generator provides node attribute generation (e. g. [17, 22]) or not [2, 8, 28]. Although we are interested in attributed graphs, we do not limit our review only to attributed generators as attributes could be applied over any graph-structured data like in [26].

By types of supported degree distributions, generators could be divided into the following classes: providing graphs with power-law degree distribution [2, 16, 17], providing graphs with any predetermined distribution, and without specification of the parameters of the degree distribution [8, 28].

Another important graph structure property that should be considered is the community structure, which means presence of densely connected sub-regions. Label assortativity depends on label homogeneity within these regions.

So, for the comparison we select some existing generators which support:

(1) possibility to tune a community structure;
(2) possibility to specify desired degree distribution and especially power-law degree distribution;
(3) possibility to control properties of graph with input parameters.

We have chosen three generators to compare with: LFR [16], ADC-SBM [26], GenCAT [20] as they satisfy all requirements and significantly differ from each other in the mechanism of edge assignment. Further, we will consider only the case of undirected graphs. Let us briefly describe selected parametric generators.

LFR. This generator assigns every randomly chosen node to a randomly chosen community if the community size exceeds the internal degree of the node, otherwise, different community is chosen. Internal degree of a node is defined by μ. Sizes of communities are chosen from power law with different exponents. The limitation of this method is that it is not possible to choose sizes or even the number of communities. More details of this generator could be found in [16].

ADC-SBM. This generator draws an edge between two nodes from Poisson distribution with the parameter equal to the expected number of edges between communities of nodes, multiplied by degrees of these two nodes normalized for each group. As long as we are interested in simple graphs, we remove multi-edges and self-loops what changes the actual degree distribution. Every node is supposed to belong to one of feature groups, which are somehow connected to labels. So features are correlated with labels. All details about ADC-SBM could be found in [26].

GenCAT. GenCAT generator [20] constructs latent factors based on predefined matrices by minimizing special loss functions and generates graphs from these latent factors. Class preference mean,

Table 1: Functional Properties Of Generators. "Degree dist." indicates whether generator supports certain degree distribution, "Tunable K" – the ability to predetermine number of classes, "low label assortativity", "low attribute assortativity" – the ability of a generator to build graphs with low label or attribute assortativity, "Attributes" column indicates whether attributes are provided by the generator, "Framework" – the presence of framework of a generator in free access

Generator	Properties Of Generator					
	Degree dist.	***Tunable K***	***low label assortativity***	***Attributes***	***low attribute assortativity***	***Framework***
LFR	power law	×	✓	×	×	✓
ADC-SBM	any	✓	✓	✓	×	✓
GenCAT	any	✓	✓	✓	×	✓
AL-BTER	**any**	✓	✓	✓	✓	✓

class preference deviation, and attribute class correlation matrix are defined as input parameters. Edges of a graph are constructed according to probabilities obtained by multiplication of latent factors. Attributes are generated by multiplying other latent factors. As long as features here are constructed correlated to labels, they do not satisfy our need of attribute assortativity, defined in Section 2.1.

There is a variety of other synthetic graph generation methods (see e. g. comprehensive overview given by Bonifati et al. [5]) but, to the best of our knowledge, they provide comparable or less functionality and flexibility against LFR, ADC-SBM, and GenCAT.

All functional abilities of selected generators and our proposed model, AL-BTER, are listed in Table 1. As we can see, none of considered generators except suggested AL-BTER are able to provide all desired characteristics.

2.3 Graph Convolutional Neural Networks

As long as state of the art methods of solving tasks on graphs are graph convolutional neural networks, we will test some of them on graphs produced by the proposed model. The general framework of one convolutional layer can be seen in equation:

$$h_i^{(k)} = F(W^{(k)} \cdot AGG(h_j^{(k)} : \{v_j \in \mathcal{N}(v_i) \cup v_i\})), \qquad (3)$$

where $h_i^{(k)}$ is a representation of a node v_i, $W^{(k)}$ is a weight matrix of a $k-th$ layer and F is a non-linear function.

We selected six different convolution networks, which are implemented in PyTorch Geometric library:

(1) GCN [14] is the simplest version of graph convolution. Sum or Mean can serve as AGG function and sigmoid as non-linear function.
(2) SAGE [10] generalizes the previous convolution by introducing different weight matrices for the selected vertex and for its neighbors.
(3) GAT [27] proposes learnable attention for every neighbor according to the importance of this neighbor for a considered node.
(4) GIN [30] adds the message from the node itself with the learnable parameter $1+\epsilon$ to the sum of messages from neighbors of the node.
(5) APPNP [15]: node representations are transformed according to personalized PageRank with teleport probability α.
(6) FA [4] sums neighbours' messages with attention coefficients. Attention coefficients have meaning of proportion of low-frequency and high-frequency signals. This convolution is supposed to perform good results on low-assortative graphs.

3 AL-BTER MODEL

Our generator is based on the BTER model [24], as long as BTER is built to match the real world networks: its clustering controlling method provides the correspondence to an assumption, stating that in real world graphs low degree nodes have much higher clustering coefficient than higher degree ones. Besides that, it generates graphs with degree distribution which properly match input distribution. Our proposed generator, AL-BTER, is an extension of an original BTER algorithm.

You can find all detailed information on BTER model in original paper [24], but we highlight some basic ideas of this method here, and in Section 3.1 we provide the details on proposed AL-BTER algorithm.

BTER suggests to group nodes according to their degrees. Then, edges are inserted in each group with Erdős-Rényi model with properly defined probability to provide different clustering coefficients. Moreover, two parameters η, ρ, controlling this probability, influence the clustering coefficient of a whole graph. One-degree nodes are handled separately of the other nodes to balance possibilities of two-degree nodes become one-degree due to stochastic nature of generating edges. Edges between groups are chosen using Chung-Lu model [7] according to remaining excesses degrees in expectation.

3.1 Description Of AL-BTER

We propose a method, which uses BTER instances as building blocks to support tuning of all graph properties from Section 2.1. Pseudo-code of the described generator is presented by Algorithm 1, and general scheme can be seen in Figure 1. The source code is available at: https://anonymous.4open.science/r/AL-BTER.

Step 1. Firstly, we generate the sequence of degrees of all nodes according to any law. Then, we divide all nodes for predetermined number of communities uniformly or according to the ratios of sizes. After that, for every node we divide its degree on in-degree (number of connections inside one group) and out-degree (number of connections to groups of different label) according to the

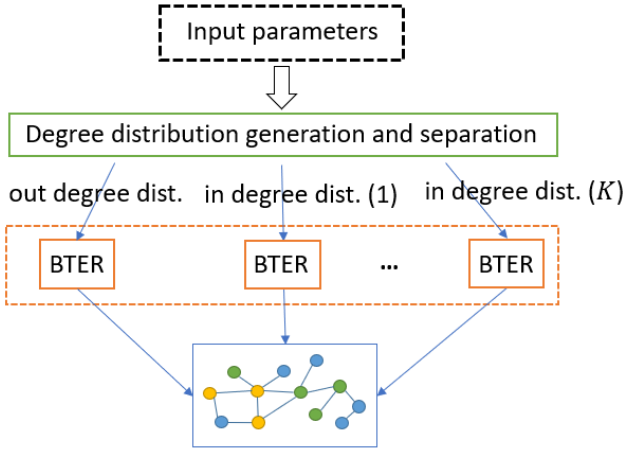


Figure 1: General scheme of AL-BTER

parameter μ: $d_{in} = \mu d$, $d_{out} = (1-\mu)d$. If we use power law degree distribution, there are many low-degree nodes. For nodes with degree less than $k = \lceil \frac{1}{\mu} \rceil$, where $\lceil ... \rceil$ means rounded up number, their in-degree will be one or zero while out-degree will be $k-1$ or k. This leads to many connected components and biased label assortativity. To overcome this issue, we say for some part of k-degree nodes that their in-degree is k and out-degree is 0 and vice versa for remaining of nodes: in-degree is 0 and out-degree is k. To keep the overall assortativity close to μ, the expectancy of number of nodes for first part is μ and for another part is $(1-\mu)$.

Step 2. For every cluster and its in-degree nodes we construct graph with BTER model. Then we construct BTER model using out-degree of nodes as input degree distribution. BTER is applied as in original paper, from preprocessing step till phase 2 inclusive..

Modification of BTER. To keep degree sequence satisying to the correct distribution in BTER model, excess degrees of 75% of one-degree nodes are set to zero, and degrees of the remaining one-degree nodes are slightly raised to 1.1. A first part of these nodes is connected with other nodes. As original BTER builds graphs with many connected components, and it is not very convenient to run graph convolutional networks on such graphs, we handle one-degree nodes a little differently. We suggest to set excesses of the second part of the one-degree nodes as 0.1 and let all one-degree nodes to form edges with the rest of the nodes including the second part of the one-degree nodes. This guarantees that all one-degree nodes will have at least one edge, but two-degree nodes still may however lower their actual degree due to stochastic manner of the generator.

3.2 Attribute Generation

We generate attributes such that attribute assortativity could be controllable for any graph with community structure. For that purpose we firstly use any clustering algorithm (e.g., Louvain's algorithm) on the generated structure to support tuning of intra/inter cluster connectivity. To each cluster we assign vector f_c sampled from normal distribution with zero mean and $\sigma_{cluster}$ deviation. Then, for each node its attribute is a summation of the vector of the cluster and the noise vector, sampled from normal distribution with zero mean and σ_{node} deviation. The value of attribute assortativity is controlled by these two deviations. If $\sigma_{cluster}$ is large and σ_{node} is small, then all nodes within one cluster would be very similar while tangibly different from nodes in other clusters. The general trend is as follows: as σ_{node} is bigger, the attribute assortativity is lower, and as $\sigma_{cluster}$ is lower, the attribute assortativity is also lower.

Algorithm 1 AL-BTER

Input :
$\{ \mathbf{d_i} \}$ List of nodes' degrees
$\{ \mathbf{l_k} \}$ List of nodes' labels
Parameters :
$\mu, \eta, \rho, \sigma_{cluster}, \sigma_{node}, K$

procedure BTERSETUP($\{ \mathbf{d_i} \}, \{ \mathbf{l_i} \}, \mu, \eta, \rho, \sigma_{cluster}, \sigma_{node}$)
  $G \leftarrow \emptyset$
  $\{ \mathbf{d_i^{in}} \}, \{ \mathbf{d_i^{out}} \} \leftarrow$ SEPARATION($\mu, \{ \mathbf{d_i} \}$)
  for $l \in \{ \mathbf{l_k} \}$ **do**
    $G \leftarrow G \cup$ BTER($\{ d_i^{in} \mid l_i = l \}, \eta, \rho$)
  end for
  $G \leftarrow G \cup$ BTER($\{ d_i^{out} \}, \eta, \rho$)
  $X \leftarrow$ ATTRIBUTE_GENERATOR($G, \sigma_{cluster}, \sigma_{node}$)
  return G, X

procedure SEPARATION($\mu, \{ \mathbf{d_i} \}$)
  for $d \in \{ \mathbf{d_i} \}$ **do**
    if $d > \lceil \frac{1}{\mu} \rceil$ **then**
      $\{ d_i^{in} \} \leftarrow \{ d_i^{in} \} \cup \{ round(d \cdot \mu) \}$
      $\{ d_i^{out} \} \leftarrow \{ d_i^{out} \} \cup \{ d - round(d \cdot \mu) \}$
    else
      $p \sim Bernoulli(\mu)$
      if $p = 1$ **then**
        $\{ d_i^{in} \} \leftarrow \{ d_i^{in} \} \cup \{ d \}$
        $\{ d_i^{out} \} \leftarrow \{ d_i^{out} \} \cup \{ 0 \}$
      else
        $\{ d_i^{in} \} \leftarrow \{ d_i^{in} \} \cup \{ 0 \}$
        $\{ d_i^{out} \} \leftarrow \{ d_i^{out} \} \cup \{ d \}$
      end if
    end if
  end for
  return $\{ \mathbf{d_i^{in}} \}, \{ \mathbf{d_i^{out}} \}$

procedure ATTRIBUTE_GENERATOR($G, \sigma_{cluster}, \sigma_{node}$)
  $clusters \leftarrow clustering(G)$
  for cl $\in$ clusters **do**
    $f_c \sim N(0, \sigma_{cluster})$
    for node $\in$ cl **do**
      $X[node] \leftarrow f_c + N(0, \sigma_{node})$
    end for
  end for
  return X

4 EXPERIMENTAL STUDY

Our model is compared with three other generators, chosen in Section 2.2 to investigate the abilities and limitations. Section 4.1 shows our generator supports comparable or broader ranges of considered properties than other models. Section 4.2 confirms the sensitivity of GNN performance to different topological characteristics.

4.1 Comparative Study of Synthetic Graph Properties

Graphs are generated with changing of each generator input parameters, and then the range of obtained graph characteristics is analysed.

4.1.1 Experimental Setup. We used the same or comparable input parameters for all generators. In each generator, the power law degree distribution is constructed with selected power α, minimum d_{min}, and maximum d_{max} degree of the graph α. We constructed attributes for LFR with our attribute generator for more fair comparison. We constructed power law degree distribution for all parameters. For the uniformity, we decided to vary number of communities, but not sizes of them. A size of a graph was set to 1000 for each generator, what seems to be a reasonable trade-off between the statistical significance of graph properties and the time spent on constructing a large number of graphs.

In GenCAT class preference mean was constructed as symmetric matrix where all elements on diagonal are the same and equals μ, all elements out of the diagonal also the same. Class preference deviation was generated randomly. To estimate the influence of deviation on results, we multiplied every element of this matrix on factor *dev*. Attribute class correlation matrix was constructed randomly, but we filled only p random elements of the matrix while varying p. In ADC-SBM we considered nested and grouped match types of feature groups as only they and random match type were implemented in the framework.

To quantitatively evaluate generation performance of AL-BTER and comparison models, we compute graph properties listed in section 2.1: *average degree, average clustering coefficient, average shortest paths length, label assortativity, attribute assortativity*. However multiple metrics makes it difficult to compare the generators directly. That's why we proposed a hypervolume metric which shows within a single value how wide are the ranges of the parameters of graphs for different generators. The higher this value is, the more expressive is a generator in terms of the diversity of the properties of the synthetic graphs.

The methodology for hypervolume calculation is based on [3], and consists of the following stages: (i) kernel density estimation by overlaying hyperbox kernels around each observation, (ii) Monte Carlo importance sampling the space using these boxes and performing range tests on random points using a recursive partitioning tree, (iii) calculating the effective number of points using the kernel density, (iv) calculating the hypervolume value as the ratio of the density of points to the effective number.

All the listed parameters, except for the average path length, are normalized to a segment from 0 to 1. To normalize the latter, we used the fact that the maximum value for the fixed number of vertices will be reached when the graph is a chain. In this case average path length can be calculated analytically and is equal to (n+1)/3.

4.1.2 Results. Ranges for all characteristics of the generated graphs are presented in Table 2. To compare the ranges, we calculate two metrics: the difference between maximum and minimum value, and the difference between ninth and first deciles (to mitigate the effect of outliers). Then we calculate ranks of different generators according to these two metrics (rank 1 means that a generator provides the most broad range of a characteristic). One may observe that AL-BTER provide the lowest values for the sum of ranks, namely, 9, for max-min metric. Even if it shows worse result for Q9-Q1 metric than LFR, the latter model does not provide attributes itself. The other disadvantage is that it reaches the fourth rank on clustering coefficient while AL-BTER never reaches the last rank. This

Table 2: Ranges Of Output Parameters

method	decile	characteristics					rank sum
		c	d_{avg}	asp	la	aa	
LFR	max	0.54	44.49	5.37	0.98	0.97	
	Q-9	0.25	34.21	3.10	0.81	0.70	
	mean	0.14	17.55	2.26	0.40	0.31	
	Q-1	0.04	3.31	1.31	0.04	0.03	
	min	0.00	2.13	0.75	0.01	0.007	
	rank max-min	3	3	2	1	1	10
	rank Q9-Q1	4	1	1	1	1	8
ADC-SBM	max	0.50	29.82	2.48	0.49	0.78	
	Q-9	0.30	27.60	2.40	0.43	0.26	
	mean	0.17	26.61	2.33	0.28	0.16	
	Q-1	0.006	1.86	2.26	0.08	0.04	
	min	0.00	1.53	2.15	0.03	0.00	
	rank max-min	4	4	4	4	3(4)	19(20)
	rank Q9-Q1	3	4	4	3	3	17
GenCAT	max	0.87	79.62	2.62	0.84	0.78	
	Q-9	0.77	57.51	2.31	0.37	0.03	
	mean	0.55	40.04	2.14	0.21	0.03	
	Q-1	0.36	27.66	2.02	0.09	0.00	
	min	0.23	19.93	1.97	0.04	0.00	
	rank max-min	1	1	3	3	3(4)	11(12)
	rank Q9-Q1	1	2	3	4	4	14
AL-BTER	max	0.61	48.72	7.60	0.98	0.96	
	Q-9	0.37	29.72	3.58	0.87	0.49	
	mean	0.22	15.75	2.55	0.53	0.34	
	Q-1	0.05	2.82	2.13	0.21	0.20	
	min	0.0	1.82	0.0	0.11	0.03	
	rank max-min	2	2	1	2	2	9
	rank Q9-Q1	2	3	2	2	2	11

Table 3: Hypervolume Z Values For Two Sets Of Output Parameters.

Model	$Z(d_{avg}, asp, c, la, aa) \cdot 10^{-4}$	$Z(asp, c, la, aa) \cdot 10^{-3}$
ADC-SBM	0.01	0.09
GenCAT	0.05	0.14
LFR	**1.43**	4.86
AL-BTER	1.18	**5.97**

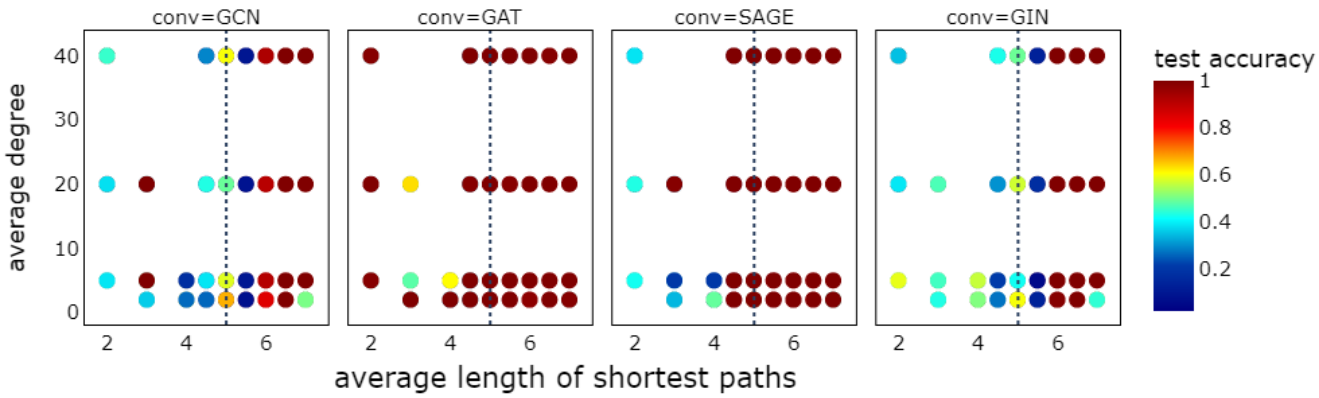


Figure 2: Weighted average F1-score of classification in different ranges of average degree vs average shortest paths

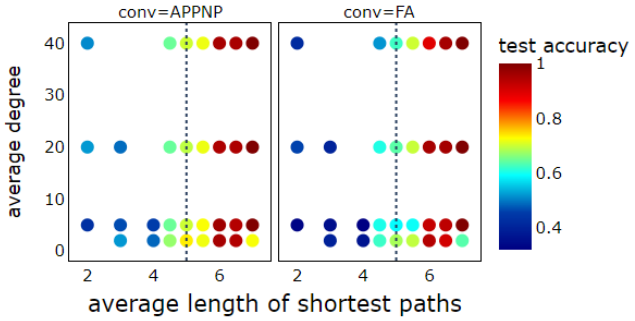


Figure 3: Weighted average F1-score of classification in different ranges of average degree vs average shortest paths

means that AL-BTER is beneficial, compared to other generators, for the case when all graph properties are equally important during benchmarking.

Our model has broader ranges of average shortest paths. Ranges of other properties are comparable to the competitors. The range of clustering coefficient is the best for GenCAT. However, GenCAT does not generate graphs with low and moderate clustering coefficients, while AL-BTER supports all values starting from 0 which is more suitable for realistic test cases. In other words, we observe that GenCAT generates only dense graphs which may be seen from c and d_{avg} values which restricts the applicability of this generator for benchmarking purposes.

The results of hypervolume analysis are presented in Table 3. They indicate that AL-BTER and LFR demonstrate close performance while GenCAT and ADC-SBM provide much lower values of hypervolume. Considering the whole set of graph properties, LFR has a larger hypervolume than AL-BTER. This advantage, is explained with better values for a single parameter d_{avg} which may be seen from the second column in Table 3. However, it can be caused by not following the power distribution. We tested the hypothesis if degree distribution of a generated graph follows power law according to Chi-square goodness-of-fit test corrected for sparse distributions [32]. This hypothesis is rejected with alpha-level of 5 % for ADC-SBM and LFR, and is not rejected for GenCAT and AL-BTER. So, we also provide values of hypervolume calculated for a subset of parameters without d_{avg}. In these conditions, AL-BTER has larger value than the competitors, showing that AL-BTER is beneficial for graphs with small to moderate densities.

4.2 Graph Neural Networks Performance

To identify the influence of values of graph characteristics on the performance of classifiers, we generated the dataset of synthetic graphs with evenly varying properties and then tested several graph convolutional networks. Hyper-parameters, including number of layers, size of hidden layers, learning rate, and dropout, were optimized by Optuna framework [1].

A weighted average F1-score is used as a quality metric since the classification is not binary. The results of node classification task with six graph convolutional networks mentioned in Section 2.3 (averaged over the rest of characteristics) are shown in Figures 2,3, 4. A vertical line separates the area of topological characteristics, which can and can not be achieved by any other generator, except AL-BTER. For example, in Figures 2,3 to the right of this line are values of length of average shortest paths, which can be generated only by AL-BTER.

In Figures 2,3 we may observe that huge influence on the quality of node classification by GNN in our experimental setting is imposed by length of average shortest paths. At the same time, there is no influence of average degree parameter. We may observe (i) gradual increase of weighted average F1-Score to 1 with an increase of average shortest path, (ii) a sharp decrease of performance from near 1 to 0.65-0.7 for $asp = 7$ and the minimum value of average degree. The second case is the case of chain-like graphs which, thus, should be considered as a special case for GNN testing.

Our study confirms that modern GNN works better on label-assortative data (see Figure 4). The results also confirm the sensitivity of the performance to the values of clustering coefficient (in general, the higher is better except for highly disassortative graphs where all methods provide constantly bad results). It is interesting that the absolute values of weighted average F1-Score are consistent between different GNNs with sligtly difference.

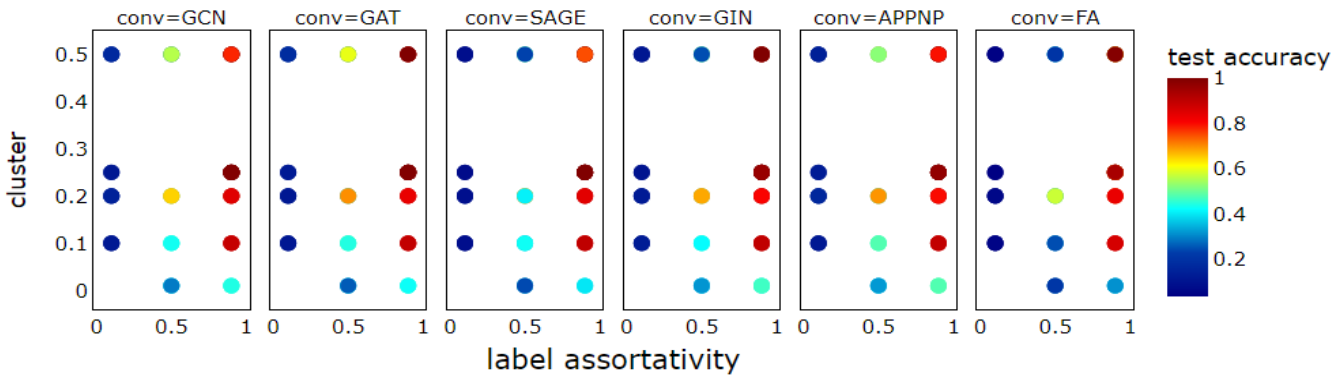


Figure 4: Weighted average F1-Score of classification in different ranges of label assortativity vs clustering coefficient

To check if the wider ranges of *asp* and *aa* provided with AL-BTER may lead to new insights about the applicability of GNNs, we analyze the dependence of performance on the length of average shortest paths for different values of attribute assortativity (Figure 5 is for GAT, Figure 6 is for all convolutions). One may see that for the range of *aa* (0.2-0.4) there exists quality transition for high *asp* converging to the values of weighted average F1-Score larger than 0.9. The behavior of the weighted average F1-Score for a varying attribute assortativity is also interesting because three regimes may be distinguished: (i) low relatively stable performance (0.4-0.55 on average) for border values of *aa*, namely, 0.1, 0.5; (ii) performance which is sensitive to the values of *asp* and depends on it in a non-linear way ($aa \in [0.2; 0.4]$; (iii) high stable performance for $aa \in [0.7; 0.8]$.

It is worth to mention that investigation of only single regime e.g. regime (iii) may lead to overly optimistic feeling about the performance of GNN. One may see also that wider range of *asp* of AL-BTER allows to observe qualitatively different behavior of GNN for different values of *aa*. Thus, we underline the sensitivity of GCN models to *asp* and *aa* characteristics with a possibility of non-linear quality dynamics. Thus, AL-BTER may be recommended as a choice if one is aimed to test the influence of attribute assortativity and average shortest paths on the results.

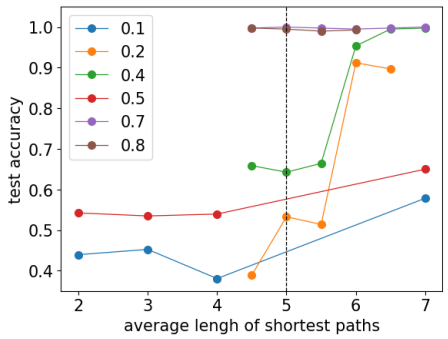


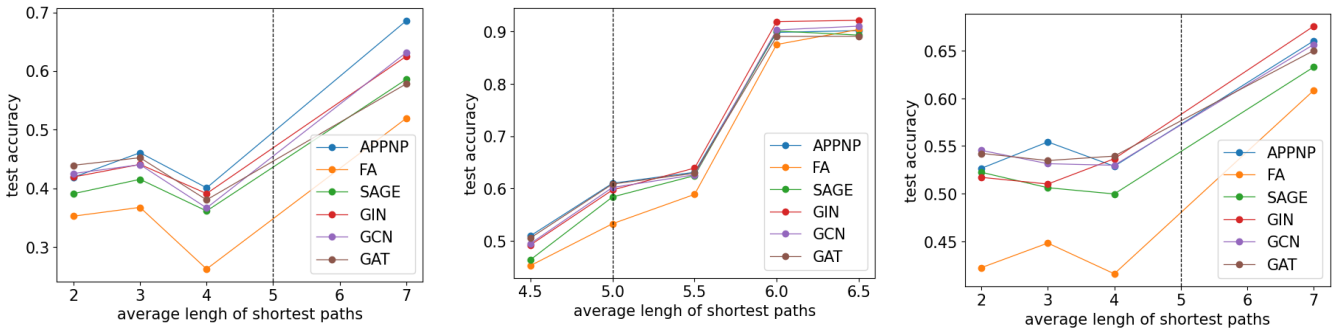
Figure 5: Weighted average F1-score of classification depending on average shortest paths for different attribute assortativity (GAT model)

5 CONCLUSION

In this study, we summarize graph properties which may influence the performance of graph neural network architectures, and report a lack of parametric generators which support tuning of all desired properties.

We introduced a novel graph generator, AL-BTER, consisting of blocks of nodes with the same label each constructed with BTER model. This method provides flexible generation of community structure and graphs with arbitrary degree distributions including power-law. In contrast to other parametric generators, we also focus on the attribute assortativity coefficient which represents the homophily property regardless of labels. In AL-BTER, an attribute generation method was introduced that allows to vary assortativity in a controlled way in graphs with community strcuture. In the experimental study, we have shown that our method outperforms the competitors for the case of equal importance of graph properties and provides wider ranges for average shortest paths while keeping ranges of other parameters comparable to other benchmark generators. Hypervolume analysis has shown that our method is more expressive in terms of the variety of generated graph properties than ADC-SBM and GenCAT, and that it is preferable to LFR for small to moderate graph densities.

The experimental study with benchmark graphs confirmed the dependence of performance of graph convolutional networks on all of the chosen properties except for the average degree. A number of conclusions can be made from these experiments: (i) chain-like topology may lead to sharp quality decrease, (ii) special methods for disassortative graphs are needed since low label assortativity leads to poor performace independent on the other characteristics, (iii) the performance for varying attribute assortativity may have different regimes including non-linear dependencies on the values of average shortest path of a graph. Thus, we underline the need for more systematic testing of GNNs in a more systematical way using more expressive benchmark datasets to avoid over-optimistic conclusions about their performance and the borders of their applicability. Further research is this area may include study of GNN extrapolation for a varying graph sizes and experimenting with graphs with arbitrary degree distributions (as not all domain graphs are necessarily scale-free, see e.g. [6]).



a) attribute assortativity=0.1 b) attribute assortativity=0.3 c) attribute assortativity=0.5

Figure 6: Weighted average F1-score of classification task for different GNNs on graphs with evenly varying average shortest paths while fixing attribute assortativity

ACKNOWLEDGMENTS

This work was supported by the Analytical Center for the Government of the Russian Federation (IGK 000000D730321P5Q0002), agreement №70-2021-00141.

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[2] Albert-László Barabási. 2013. Network science. Chapter 5. The Barabási-Albert Model. *Network Science* (2013), 1–45.

[3] Benjamin Blonder, Christine Lamanna, Cyrille Violle, and Brian J Enquist. 2014. The n-dimensional hypervolume. *Global Ecology and Biogeography* 23, 5 (2014), 595–609.

[4] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond Low-frequency Information in Graph Convolutional Networks. (2021). arXiv:2101.00797 http://arxiv.org/abs/2101.00797

[5] Angela Bonifati, Irena Holubová, Arnau Prat-Pérez, and Sherif Sakr. 2020. Graph Generators: State of the art and open challenges. *ACM Computing Surveys (CSUR)* 53, 2 (2020), 1–30.

[6] Anna D Broido and Aaron Clauset. 2019. Scale-free networks are rare. *Nature communications* 10, 1 (2019), 1–10.

[7] Fan Chung and Linyuan Lu. 2002. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences* 99, 25 (2002), 15879–15882.

[8] P. Erdős and A. Rényi. 2011. On the evolution of random graphs. *The Structure and Dynamics of Networks* 9781400841 (2011), 38–82. https://doi.org/10.1515/9781400841356.38

[9] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. 1998. CiteSeer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*. 89–98.

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

[11] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687* (2020).

[12] Wei Jin, Tyler Derr, Yiqi Wang, Yao Ma, Zitao Liu, and Jiliang Tang. 2021. Node similarity preserving graph convolutional networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 148–156.

[13] Dongkwan Kim and Alice Oh. 2020. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*.

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

[15] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997* (2018).

[16] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Physical review E* 78, 4 (2008), 046110.

[17] Christine Largeron, Pierre-Nicolas Mougel, Reihaneh Rabbany, and Osmar R Zaïane. 2015. Generating attributed networks with communities. *PloS one* 10, 4 (2015), e0122777.

[18] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2–es.

[19] Derek Lim, Xiuyu Li, Felix Hohne, and Ser-Nam Lim. 2021. *New Benchmarks for Learning on Non-Homophilous Graphs*. Vol. 1. Association for Computing Machinery. arXiv:2104.01404 http://arxiv.org/abs/2104.01404

[20] Seiji Maekawa, Yuya Sasaki, George Fletcher, and Makoto Onizuka. 2021. GenCAT: Generating Attributed Graphs with Controlled Relationships between Classes, Attributes, and Topology. *arXiv preprint arXiv:2109.04639* (2021).

[21] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287* (2020).

[22] Joseph J Pfeiffer III, Sebastian Moreno, Timothy La Fond, Jennifer Neville, and Brian Gallagher. 2014. Attributed graph models: Modeling network structure with correlated attributes. In *Proceedings of the 23rd international conference on World wide web*. 831–842.

[23] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2019. Multi-scale Attributed Node Embedding. arXiv:1909.13021 [cs.LG]

[24] Comandur Seshadhri, Tamara G Kolda, and Ali Pinar. 2012. Community structure and scale-free collections of Erdős-Rényi graphs. *Physical Review E* 85, 5 (2012), 056109.

[25] Damian Szklarczyk, John H Morris, Helen Cook, Michael Kuhn, Stefan Wyder, Milan Simonovic, Alberto Santos, Nadezhda T Doncheva, Alexander Roth, Peer Bork, et al. 2016. The STRING database in 2017: quality-controlled protein–protein association networks, made broadly accessible. *Nucleic acids research* (2016), gkw937.

[26] Anton Tsitsulin, Benedek Rozemberczki, John Palowitch, and Bryan Perozzi. 2021. Synthetic Graph Generation to Benchmark Graph Learning. (2021).

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[28] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *nature* 393, 6684 (1998), 440–442.

[29] Sheng Xiang, Dong Wen, Dawei Cheng, Ying Zhang, Lu Qin, Zhengping Qian, and Xuemin Lin. 2021. General graph generators: experiments, analyses, and improvements. *The VLDB Journal* (2021), 1–29.

[30] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv* (2018), 1–17. arXiv:1810.00826

[31] Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42, 1 (2015), 181–213.

[32] Daniel Zelterman. 1987. Goodness-of-fit tests for large sparse multinomial distributions. *J. Amer. Statist. Assoc.* 82, 398 (1987), 624–629.