Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

A Feedback-Enhanced Graph Generative Model with Tunable Features

Takahiro Yokoyama[1], Yoshiki Sato[1], Sho Tsugawa[2], (Member, IEEE) and Kohei Watabe[3], (Member, IEEE)

[1]Graduate School of Engineering Nagaoka University of Technology, Nagaoka, Niigata, Japan
[2]Institute of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki, Japan (e-mail: s-tugawa@cs.tsukuba.ac.jp)
[3]Graduate School of Science and Engineering, Saitama University, Saitama, Saitama, Japan (e-mail: kwatabe@mail.saitama-u.ac.jp)

Corresponding author: Kohei Watabe (e-mail: kwatabe@mail.saitama-u.ac.jp).

This work was partly supported by JSPS KAKENHI Grant Number 23K28069.

ABSTRACT Graphs are data structures that represent abstract relationships between elements and are widely used across various domains. The ability to generate graphs with specific features is crucial for many applications. Recent advances in deep learning have led to progress in graph generation, enabling models to reproduce structural features observed in real-world graphs. A significant recent focus in this area is conditional graph generation, which allows models to tune desired features during the generation process. Among recent efforts in this direction, GraphTune was developed to support continuous tuning of graph features. Building on this approach, this paper introduces F-GraphTune, a model designed to improve the accuracy of feature specification in generated graphs. F-GraphTune incorporates a Feature Estimator (FE) into the GraphTune framework, which provides feedback on the feature values of the generated graphs. Experiments using a real-world graph dataset demonstrate that F-GraphTune outperforms existing models, including GraphTune, with up to a 53.7% improvement in accuracy of feature specification.

INDEX TERMS Graph generation, Conditional VAE, LSTM, Graph feature, Generative model.

I. INTRODUCTION

Graph generative models are becoming a key technology for supporting diverse graph data applications across multiple domains. In recent years, graph data applications have expanded in areas such as communication networks, social networks, databases, and cheminformatics. Representative examples include node classification, community detection, query acceleration, and drug discovery [1]. At the same time, the importance of graph-based simulations have increased for uncovering critical information in data mining, where repeated simulations serve as a fundamental approach [2]. However, researchers and professionals often face limited access to real-world graph data due to privacy concerns or insufficient measurements. Graph generative models address this by augmenting limited real-world graph data and expecting future or unknown graphs for applications such as predicting network growth or discovering new molecular structures.

Traditionally, stochastic models that generate graphs based on pre-defined edge and node probabilities have been developed, with a focus on a single structural aspect. Prominent examples include the Erdős-Rényi (ER) model [3], Watts-Strogatz (WS) model [4], and Barabási-Albert (BA) model [5]. These models accurately capture a specific structural feature, such as randomness [3], small-worldness [4], scale-free properties [5], or node clustering [6]. However, they are not adaptable to real-world graphs with multiple interacting features and cannot ensure the comprehensive reproduction of all graph features beyond the targeted one.

Machine learning-based generative models for graphs aim to learn features from graph data and comprehensively reproduce them, leveraging machine learning techniques [7]–[18]. In recent years, learning-based graph generation has attracted significant attention, and various methods have been investigated. While many models focus on generating small graphs, particularly for molecular design, recent work has expanded to relatively large graphs, including citation and social networks. These studies reproduce multiple features that capture the global properties of graphs, such as average shortest path length, clustering coefficient, and the power-law exponent of the degree distribution.

A recent trend in machine learning-based graph generative models is conditional graph generation, and various models, including our previously proposed GraphTune [19], [20], have been investigated. Most existing conditional generative models exhibit inherent issues, such as dependence on domain-specific knowledge in molecular chemistry [11],

[13], [15] and the inability to continuously tune global-level structural features [8], [12], [21], [22]. GraphTune, as proposed in our prior works, addresses these issues and enables the continuous tuning of global-level structural features across various graph domains. Although some studies have attempted to disentangle latent spaces under specific conditions [23], to the best of our knowledge, GraphTune remains the only graph generative model capable of continuous adjustment of global-level structural features applicable across diverse domains, including social networks and citation graphs.

While GraphTune was an innovative model enabling the continuous tuning of features in generated graphs, it faced challenges with tuning accuracy. GraphTune shifts the distribution of features in generated graphs continuously, utilizing a Long Short-Term Memory (LSTM)-based Conditional Variational AutoEncoder (CVAE). As demonstrated in Reference [20], GraphTune can effectively tune global-level structural features, such as average shortest path length, clustering coefficient, and the power-law exponent of the degree distribution. Although the distribution of features shifts continuously in the generated results, the mean of certain features may not precisely approach the specified target value. Moreover, due to the high variance in the feature distribution, accurately specifying a feature becomes difficult. This introduces the possibility that generated graphs may exhibit features significantly deviating from the specified target values.

In this paper, we propose *F-GraphTune*, an extension of the conventional GraphTune model that incorporates a *Feature Estimator* (FE). FE estimates the feature values of the generated graphs and calculates the error relative to the specified values. This error is then fed back into GraphTune through neural networks within FE, enhancing the accuracy of graph generation. The two models, GraphTune and FE, employ an alternate training algorithm that trains both models alternately, freezing the weight updates for one model at a time, to prevent information about the specified feature from leaking into the FE. F-GraphTune not only retains the conventional method's ability to reproduce graph features but also improves the tunability of these features through FE.

In summary, the main contributions of this paper are as follows:

- We propose a novel learning-based graph generative model, F-GraphTune, which extends GraphTune by integrating FE, a module that estimates the feature values of the generated graph. F-GraphTune achieves higher accuracy in specifying feature values compared to the conventional GraphTune model, as FE provides feedback on feature errors to GraphTune.
- We further propose an alternate training algorithm to effectively train the two models, GraphTune and FE, while preventing data leakage from GraphTune to FE. This algorithm ensures that FE accurately estimates values solely from the graphs generated by GraphTune, even when graph feature values are provided as inputs to GraphTune.
- We compare F-GraphTune with recent conditional graph generative models using a real graph dataset. The results show that the RMSE of the features of generated graphs in F-GraphTune is up to 53.7% lower than that in GraphTune.

The remainder of this paper is organized as follows. Section II reviews related work on graph generative models. Section III defines the problem of generating graphs with specified features. Section IV presents GraphTune, the basis of the proposed model. Section V details the model architecture and the associated training and generation algorithms. Section VI reports the empirical evaluations of F-GraphTune and baseline models. Section VII discusses the study's limitations and potential future work. Finally, Section VIII provides the conclusions.

II. RELATED WORKS

Graph generation has developed over several decades, resulting in an extensive body of literature contributed by numerous researchers. One of the earliest and most fundamental models is the Erdős-Rényi model, introduced in 1959, which generates random graphs based on simple probabilistic rules. In the late 1990s and early 2000s, two influential models [4], [5] were proposed that successfully reproduced key structural properties of graphs, including small-world networks and power-law degree distributions. These models attracted significant attention and established the foundation for further advances in the field. Since these pioneering works, numerous stochastic graph generation methods have been developed, many inspired by these early models [18], [24]–[26]. Simultaneously, efforts to quantify various structural features of graphs have increased, allowing for a more detailed understanding of their properties. A common characteristic of traditional stochastic models is their focus on reproducing one or a few specific graph features—such as small-worldness, power-law degree distributions, or local clustering—rather than attempting to capture the full complexity of graph features. As a result, these models are limited in their capacity to generate graphs that can serve as realistic substitutes for real-world graphs, which typically display a complex combination of multiple features.

On the other hand, learning-based models that aim to reproduce real-world graphs have been studied, and in particular, the recent trend has shifted toward conditional graph generation. Such models have been applied across various domains, including molecular design and social network modeling, and their development has been summarized in recent survey articles [16]–[18]. In the area of unconditional graph generation, sequence-based approaches that convert graphs into sequential representations have achieved notable success [7]–[9], [14]. For conditional graph generation, domain-specific models have been developed for tasks in cheminformatics [27], [28], evolutionary biology [29], and natural language processing [30], [31]. However, these models often depend on specialized knowledge and are not easily applicable to general-purpose graph generation. Models such as

DeepGMG [8], SCGG [21], and GraphGUIDE [22] overcome this limitation by supporting graph generation based on local structural patterns, including hexagons, n-rings, and n-cliques.

Meanwhile, CondGen [12] and GraphTune [20] focus on global-level structural features rather than local patterns. CondGen allows the specification of categories of global structural properties, such as average shortest path length and the Gini index, using a variational generative adversarial framework. The GraphTune model we have proposed enables continuous tuning of global-level features, including average shortest path length, clustering coefficient, and modularity, and demonstrates superior performance compared to CondGen [20].

III. PROBLEM FORMULATION

The mathematical notation and formulation used in this paper follow those of GraphTune [20]. We consider undirected, connected graphs without self-loops. By convention, a graph is denoted as $G = (V, E)$, where V and E represent the set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ and the set of edges $E = \{(x, y) \mid x, y \in V\}$, respectively. We let $\Omega = \{G_1, G_2, \ldots\}$ denote the universal set of graphs.

We consider a mapping $F : \Omega \to \boldsymbol{A}$, where a graph $G_i \in \Omega$ is mapped to a feature vector $A_{G_i} \in \boldsymbol{A}$, defined by $F(G_i) = A_{G_i} = [\alpha^1_{G_i}, \alpha^2_{G_i}, \cdots]^T$. The jth component $\alpha^j_{G_i}$ of the vector A_{G_i} represents a structural feature of the graph G_i. Each feature is expressed as a real number, i.e., $\alpha^j_{G_i} \in \mathbb{R}$. For example, the elements of A_{G_i} may correspond to features such as the average shortest path length.

According to the formulation of GraphTune [20], we define the task of generating a graph with specified features as the problem of inferring the inverse image from feature vectors back to graphs. We define the inverse mapping $F^{-1}(\cdot)$ of the function $F(\cdot)$, where a graph G_i can be obtained by evaluating $F^{-1}(A_{G_i}) = \{G_i, \ldots\}$. We address the inference problem of estimating an approximate inverse mapping $\hat{F}^{-1}(\cdot)$ using a subset $\mathcal{G}$ of the universal set Ω. Solving this problem enables the generation of a new graph G'_i from a feature vector $A_{G'_i}$, in which the values of the features are replaced with arbitrary ones. Since the complete set Ω of graphs is not accessible, the inference must rely on a subset $\mathcal{G}$ drawn from an available dataset.

IV. GRAPHTUNE

F-GraphTune is an extension of GraphTune with the incorporation of FE. In this section, we explain the architecture of GraphTune, which serves as the foundation for F-GraphTune.

A. DFS CODE

To input graph data into neural networks, GraphTune transforms each graph into a DFS code, which is a sequential representation. The algorithm employs a depth-first search to assign timestamps to all nodes and generates a sequence of 5-tuples $(t_u, t_v, L(u), L(e), L(v))$ corresponding to the discovered edges $e = (u, v)$ in the order of traversal. In this representation, t_u denotes the timestamp of node u, and $L(\cdot)$ indicates the label of a node or edge, respectively. An End Of Sequence (EOS) token $(\mathrm{EOS}_t, \mathrm{EOS}_t, \mathrm{EOS}_L, 1, \mathrm{EOS}_L)$ is appended at the end of each graph sequence to handle cases where graph sizes vary.

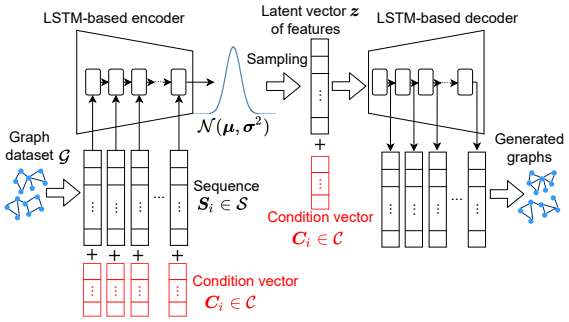


FIGURE 1. The architecture of GraphTune. GraphTune composed of CVAE with LSTM-based encoder and decoder. A graph G_i in the graph dataset $\mathcal{G}$ is converted to a sequence $S_i \in \mathcal{S}$ of 5-tuples by using DFS code. The sequence is processed by an LSTM-based encoder to map the graph to a latent space. The decoder generates a sequence of 5-tuples from a latent vector, and the sequence is converted to a generated graph. The condition vector is input to both the encoder and decoder.

B. MODEL ARCHITECTURE

GraphTune consists of a CVAE with LSTM-based encoder and decoder, as shown in Fig. 1. During training, a set $\mathcal{S}$ of sequence data $\boldsymbol{S}_i = \{\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots\}$ obtained by converting a graph $G_i \in \mathcal{G}$ into DFS code is input into the model. Concurrently, values of graph features that we intend to specify are calculated and then input into the model as a set $\mathcal{C}$ of condition vectors $\boldsymbol{C}_i$. Following the architecture of a standard VAE, the encoder $F_{\mathrm{enc}}(\boldsymbol{S}_i, \boldsymbol{C}_i)$ maps sequence data $\boldsymbol{S}_i$ to parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$. The latent vector $\boldsymbol{z}$ are sampled from a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ with dimension l. Namely, the encoder can be expressed as follows.

$$\boldsymbol{\mu} = f_\mu(\boldsymbol{S}_i, \boldsymbol{C}_i), \quad \boldsymbol{\sigma}^2 = f_{\sigma^2}(\boldsymbol{S}_i, \boldsymbol{C}_i), \quad \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2). \tag{1}$$

Here, we summarize these functions by the following single function:

$$\boldsymbol{z} = F_{\mathrm{enc}}(\boldsymbol{S}_i, \boldsymbol{C}_i). \tag{2}$$

The decoder $F_{\mathrm{dec}}(\boldsymbol{z}, \boldsymbol{C}_i)$ learns to reconstruct the original sequence $\boldsymbol{S}_i$ from a latent vector $\boldsymbol{z}$ and the condition vector $\boldsymbol{C}_i$, outputting a reconstructed sequence $\tilde{\boldsymbol{S}}_i$. Then, the decoder can be outlined as follows.

$$\tilde{\boldsymbol{S}}_i = F_{\mathrm{dec}}(\boldsymbol{z}, \boldsymbol{C}_i). \tag{3}$$

The loss function for the training is composed of two parts: a Kullback-Leibler divergence loss (KL loss) and a reconstruction loss. The KL loss regularizes the distribution of the latent vector to be the standard normal distribution, and can be written as

$$\mathrm{Loss}_{\mathrm{enc}}(\boldsymbol{\mu}, \boldsymbol{\sigma}) = \frac{1}{2} \sum_{l=1}^{L} (1 + \log(\sigma_l^2) - \mu_l^2 - \sigma_l^2). \tag{4}$$

The reconstruction loss ensures the predicted sequence $\tilde{\boldsymbol{S}}_i$ is similar to the input sequence $\boldsymbol{S}_i$ from the dataset, and is defined as

$$\mathrm{Loss}_{\mathrm{dec}}(\tilde{\boldsymbol{S}}_i, \boldsymbol{S}_i) = -\frac{1}{|\boldsymbol{S}_i|}\sum_{j=0}^{|\boldsymbol{S}_i|}\sum_{c} \boldsymbol{s}_j(c)\log(\tilde{\boldsymbol{s}}_j(c)), \quad (5)$$

where $\boldsymbol{s}_j(c)$ and $\tilde{\boldsymbol{s}}_j(c)$ represent component $c \in \{t_u, t_v, L(u), L(e), L(v)\}$ of $\boldsymbol{s}_i$ and $\tilde{\boldsymbol{s}}_j$, respectively. GraphTune's overall loss, denoted as $\mathrm{Loss}_{\mathrm{tune}}$, can be expressed as follows.

$$\mathrm{Loss}_{\mathrm{tune}}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \tilde{\boldsymbol{S}}_i, \boldsymbol{S}_i) = \beta \cdot \mathrm{Loss}_{\mathrm{enc}}(\boldsymbol{\mu}, \boldsymbol{\sigma}) + \mathrm{Loss}_{\mathrm{dec}}(\tilde{\boldsymbol{S}}_i, \boldsymbol{S}_i), \quad (6)$$

where β is a weight given as a hyperparameter.

In the generation process after training, we provide the decoder with the condition vector $\boldsymbol{C}^*$, whose elements are the specified feature values, and the latent vector $\boldsymbol{z}$, randomly sampled from the multivariate standard normal distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Then, the decoder generates a sequence data $\hat{\boldsymbol{S}}$ that corresponds to a graph with features according to $\boldsymbol{C}^*$. In summary, the behavior of the decoder is as follows:

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \qquad \tilde{\boldsymbol{S}} = F_{\mathrm{dec}}(\boldsymbol{z}, \boldsymbol{C}^*). \quad (7)$$

V. F-GRAPHTUNE: FEEDBACK-ENHANCED GRAPH GENERATIVE MODEL WITH TUNABLE FEATURES

In this paper, we extend GraphTune by incorporating an LSTM-based FE component, which estimates the value of the structural features from the graph reconstructed by GraphTune. This proposed model, which consists of two components, GraphTune and FE, is referred to as F-GraphTune: Feedback-enhanced graph generative model with tunable features. An alternate training algorithm is also proposed and applied to prevent the actual graph features that are input to the GraphTune component from leaking into the FE component when training F-GraphTune. In this section, we will start by explaining the preprocessing of the dataset, which is the same as that of GraphTune, in Section V-A. After that, we will describe the architecture of F-GraphTune in Section V-B. We will then provide a detailed account covering the training and generation processes from Section V-C to V-E.

A. PREPROCESS

The preprocessing of a dataset in F-GraphTune is the same as that of GraphTune. In the preprocessing, each graph $\boldsymbol{G}_i$ within a graph dataset $\mathcal{G}$ is converted into a sequence $\boldsymbol{S}_i$ with the DFS code, creating a set $\mathcal{S}$ of sequences $\boldsymbol{S}_i$. Palarelly, for each graph $\boldsymbol{G}_i$ within a graph dataset $\mathcal{G}$, we compute a vector $\boldsymbol{C}_i$ whose elements represent values of target features and create a set $\mathcal{C}$ of condition vectors $\boldsymbol{C}_i$. F-GraphTune learns from the set $\mathcal{S}$ of sequences and the set $\mathcal{C}$ of condition vectors to achieve a generative model.

B. MODEL ARCHITECTURE

The architecture of F-GraphTune involves two main components: a GraphTune component and an LSTM-based FE component as shown in Fig. 2. The GraphTune component,

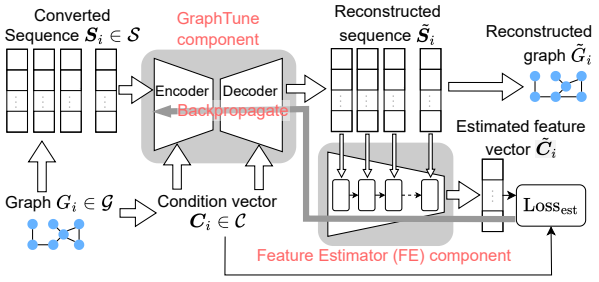


FIGURE 2. The architecture of F-GraphTune. The architecture of F-GraphTune involves two main components: GraphTune and an LSTM-based Feature Estimator (FE). The FE component takes sequence data generated by GraphTune as input and outputs an estimate for the feature vector of the generated graph corresponding to the provided sequence data. The FE loss $\mathrm{Loss}_{\mathrm{est}}$ is defined as the difference between the estimate and the condition vector C_i. Since the condition vector C_i represents the actual features of graph S_i, the FE component provides feedback on errors of features of generated graphs to the GraphTune component. Since the GraphTune component takes a condition vector as input, it may leak information regarding errors to the FE component, whereas the alternate training algorithm allows for appropriate error feedback avoiding the problem of the leakage.

identical to the original GraphTune described in Section IV, is responsible for learning graph features from a set $\mathcal{S}$ of sequences and generating graphs $\tilde{\mathcal{S}}$. On the other hand, the FE component computes the features of the generated graphs and provides feedback on the errors from the intended target values. By using the neural network-based FE to compute values of features, it becomes possible to backpropagate the error between the specified values of features and the values of features of generated graphs to the neural network of GraphTune.

The GraphTune component functions similarly to the original GraphTune, taking a set $\mathcal{S}$ of sequences converted from graphs and a set $\mathcal{C}$ of their condition vectors as input, and reconstructing a set $\tilde{\mathcal{S}}$ of sequences $\tilde{\boldsymbol{S}}_i$ as output. The processes in the GraphTune component is as shown in Eqs. (2) and (3).

A sequence $\tilde{\boldsymbol{S}}_i$ output from the GraphTune component is input to the FE component, and then the FE component outputs a vector $\tilde{\boldsymbol{C}}_i$ estimating the features of the graph $\tilde{\boldsymbol{G}}_i$ corresponding to the sequence $\tilde{\boldsymbol{S}}_i$. To treat sequence data, we employ a fully connected layer and a stacked LSTM as the FE component. The jth element $\tilde{\boldsymbol{s}}_j$ of the sequence $\tilde{\boldsymbol{S}}_i$ is embedded with a single fully connected layer f_{emb}. The embedded vector is then passed into each LSTM block represented by f_{est}. The initial hidden state vector $\boldsymbol{h}_0$ is initialized as a zero vector $\boldsymbol{0}$. The stacked LSTM with the embedding layer f_{emb} processes a sequence $\tilde{\boldsymbol{S}}_i$ of length $k = |\tilde{\boldsymbol{S}}_i|$ by recursively applying the LSTM block f_{est} to hidden state vector $\boldsymbol{h}_j$. The output $\boldsymbol{h}_k$ of the last LSTM block is fed to a single fully connected layer f_{feat}. Summarizing the above, the process of the FE component in F-GraphTune is as follows.

$$\boldsymbol{h}_0 = \boldsymbol{0}, \quad (8)$$

$$\boldsymbol{h}_{j+1} = f_{\mathrm{est}}(\boldsymbol{h}_j, f_{\mathrm{emb}}(\tilde{\boldsymbol{s}}_j)) \quad (j = 0, 1, \ldots, k-1), \quad (9)$$

$$\tilde{\boldsymbol{C}}_i = f_{\mathrm{feat}}(\boldsymbol{h}_k). \quad (10)$$

C. LOSS FUNCTIONS

In addition to the KL loss and reconstruction loss employed in the original GraphTune, F-GraphTune incorporates an estimation loss, $\mathrm{Loss}_{\mathrm{est}}$, which provides feedback on feature errors in the generated graphs, as computed by the FE component. The estimation loss $\mathrm{Loss}_{\mathrm{est}}$ is defined as follows.

$$\mathrm{Loss}_{\mathrm{est}}(\tilde{\boldsymbol{C}}_i, \boldsymbol{C}_i) = \|\tilde{\boldsymbol{C}}_i - \boldsymbol{C}_i\|_2^2. \tag{11}$$

$\mathrm{Loss}_{\mathrm{est}}$ signifies the errors between the estimated feature values $\tilde{\boldsymbol{C}}_i$ of the generated graph and the feature values $\boldsymbol{C}_i$ we intended to specify. Note that this error includes both the reconstruction error of graphs by the GraphTune component and the estimation error of graph feature values by the FE component. If the FE component can accurately estimate the feature values of generated graphs, backpropagating the estimation loss allows for appropriate feedback to be provided to the GraphTune component. The training process of F-GraphTune alternates between the GraphTune and FE components, with the estimation loss applied during the training of both components. A detailed explanation of the training process is provided in Section V-D.

The overall loss $\mathrm{Loss}_{\mathrm{all}}$ for the training of F-GraphTune is defined by the following.

$$\begin{aligned}&\mathrm{Loss}_{\mathrm{all}}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \tilde{\boldsymbol{S}}_i, \boldsymbol{S}_i, \tilde{\boldsymbol{C}}_i, \boldsymbol{C}_i)\\ &\quad = \mathrm{Loss}_{\mathrm{tune}}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \tilde{\boldsymbol{S}}_i, \boldsymbol{S}_i) + \gamma \cdot \mathrm{Loss}_{\mathrm{est}}(\tilde{\boldsymbol{C}}_i, \boldsymbol{C}_i),\end{aligned} \tag{12}$$

where $\mathrm{Loss}_{\mathrm{tune}}$ represents the loss for GraphTune explained as Eq. (6) in Section IV. The scale of $\mathrm{Loss}_{\mathrm{tune}}$ and $\mathrm{Loss}_{\mathrm{est}}$ is balanced by weight γ. For the training of the GraphTune component, Eq. (12) is used, while for the training of the FE component, Eq. (11) is utilized.

D. ALTERNATE TRAINING ALGORITHM

In F-GraphTune, an alternate training algorithm is adopted to address the issue of data leakage regarding condition vectors. This algorithm alternately trains the GraphTune component and the FE component. In our proposed approach, despite inputting a condition vector into the GraphTune component, the FE component attempts to estimate the condition vector. Consequently, standard training can lead to data leakage from the GraphTune component to the FE component, causing the FE component to simply output the input feature values. In the alternate training algorithm, during the training of the GraphTune component, the weight parameters of the FE component are frozen. Conversely, during the training of the FE component, the FE component is trained independently.

The alternate training algorithm consists of three steps: 1) training of the GraphTune component, 2) graph generation with the GraphTune component, and 3) training of the FE component. The details of the algorithm are shown in Algorithm 1. For a given set $\mathcal{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \dots\}$ of sequences and the corresponding set $\mathcal{C} = \{\boldsymbol{C}_1, \boldsymbol{C}_2, \dots\}$ of condition vectors, Algorithm 1 returns learned functions F_{enc}, F_{dec}, f_{enb}, f_{est}, and f_{feat}. The two components are alternately trained by iterating the above three steps N times (Lines 1-38).

Algorithm 1 Training of F-GraphTune

Require: Set of sequences $\mathcal{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \dots\}$,
Set of condition vectors $\mathcal{C} = \{\boldsymbol{C}_1, \boldsymbol{C}_2, \dots\}$
Ensure: Learned functions $F_{\mathrm{enc}}, F_{\mathrm{dec}}, f_{\mathrm{emb}}, f_{\mathrm{est}}$, and f_{feat}

1: **for** N times **do**
2: **for** M times **do**
3: $\mathrm{Loss}_{\mathrm{all}} \leftarrow 0$
4: **for** i from 1 to $|\mathcal{S}|$ **do**
5: $\hat{\boldsymbol{S}}_i \leftarrow F_{\mathrm{dec}}(F_{\mathrm{enc}}(\boldsymbol{S}_i, \boldsymbol{C}_i), \boldsymbol{C}_i)$
6: **if** $N = 1$ **then**
7: $\mathrm{Loss}_{\mathrm{all}} \leftarrow \mathrm{Loss}_{\mathrm{all}} + \mathrm{Loss}_{\mathrm{tune}}$
8: **else**
9: $\boldsymbol{h}_0 \leftarrow \boldsymbol{0}$
10: **for** j from 0 to $k-1$ **do**
11: $\boldsymbol{h}_{j+1} \leftarrow f_{\mathrm{est}}(\boldsymbol{h}_j, f_{\mathrm{emb}}(\boldsymbol{s}_j))$
12: **end for**
13: $\tilde{\boldsymbol{C}}_i \leftarrow f_{\mathrm{feat}}(\boldsymbol{h}_k)$
14: $\mathrm{Loss}_{\mathrm{all}} \leftarrow \mathrm{Loss}_{\mathrm{all}} + \mathrm{Loss}_{\mathrm{tune}} + \gamma \cdot \mathrm{Loss}_{\mathrm{est}}(\tilde{\boldsymbol{C}}_i, \boldsymbol{C}_i)$
15: **end if**
16: **end for**
17: Back-propagate Loss and update the weights of GraphTune component (i.e., F_{enc} and F_{dec})
18: **end for**
19: $\mathcal{S}' \leftarrow \mathcal{S}$
20: **for** n times **do**
21: $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}^2)$
22: $\boldsymbol{C} \leftarrow$ sample from $\mathcal{C}$
23: Add $F_{\mathrm{dec}}(\boldsymbol{z}, \boldsymbol{C})$ into $\mathcal{S}'$
24: **end for**
25: **for** L times **do**
26: $\mathrm{Loss}_{\mathrm{est}} \leftarrow 0$
27: **for** i from 1 to $|\mathcal{S}'|$ **do**
28: $\boldsymbol{C}'_i \leftarrow$ calculate features of S'_i
29: $\boldsymbol{h}_0 \leftarrow \boldsymbol{0}$
30: **for** j from 0 to $k-1$ **do**
31: $\boldsymbol{h}_{j+1} \leftarrow f_{\mathrm{est}}(\boldsymbol{h}_j, f_{\mathrm{emb}}(\boldsymbol{s}'_j))$
32: **end for**
33: $\tilde{\boldsymbol{C}}_i \leftarrow f_{\mathrm{feat}}(\boldsymbol{h}_k)$
34: $\mathrm{Loss}_{\mathrm{est}} \leftarrow \mathrm{Loss}_{\mathrm{est}} + \mathrm{Loss}_{\mathrm{est}}(\tilde{\boldsymbol{C}}_i, \boldsymbol{C}'_i)$
35: **end for**
36: Back-propagate Loss and update the weights of FE component (i.e., $f_{\mathrm{emb}}, f_{\mathrm{est}}$, and f_{feat})
37: **end for**
38: **end for**

As the first step, the GraphTune component is trained through M iterations (Lines 2-18). The value of overall loss (i.e. Eq. (12)) is initialized to 0 (Line 3), and then the total value of the loss is calculated for all sequences in $\mathcal{S}$ (Lines 4-16). The GraphTune component takes $\boldsymbol{S}_i$ and $\boldsymbol{C}_i$ as inputs and reconstructs $\tilde{\boldsymbol{S}}_i$ through the encoder and decoder (Line 5). For all sequences in the dataset, the value of $\mathrm{Loss}_{\mathrm{all}}$ is accumulated (Lines 6-15). However, in the initial iteration, we disregard the $\mathrm{Loss}_{\mathrm{est}}$ that is influenced by the accuracy of the FE component since the FE component hasn't been

trained yet. In the iterations beyond the second one, in order to obtain estimate $\tilde{\boldsymbol{C}}$, the LSTM block f_{est} in the FE component recursively calculates $\boldsymbol{h}_j$ by processing jth element $\boldsymbol{s}_j$ of ith sequence $\boldsymbol{S}_i = [\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_{k-1}]$ in $\mathcal{S}$ (Lines 9-13). The estimate $\tilde{\boldsymbol{C}}$ is calculated from the final output $\boldsymbol{h}_k$ using f_{feat} (Line 13). Estimation loss $\gamma \cdot \mathrm{Loss}_{\mathrm{est}}(\tilde{\boldsymbol{C}}_i, \boldsymbol{C}_i)$ is added to the overall loss (Line 14). The weights of the functions F_{enc} and F_{dec} of the GraphTune component are updated by backpropagating the loss (Line 17). Note that the weights of f_{emb}, f_{est}, and f_{feat} are frozen and not updated.

In the second step, n sequences of graphs are generated using the decoder F_{dec} that is trained in the first step (Lines 19-24). In the beginning, a set $\mathcal{S}'$ of sequences for the training of the FE component in the third step is initialized (Line 19). Using a latent vector $\boldsymbol{z}$ that follows a multivariate Gaussian distribution and a condition vector $\boldsymbol{C}$ randomly sampled from $\mathcal{C}$, n graphs are generated using the decoder F_{dec} and added to $\mathcal{S}'$ (Lines 20-24).

The third step involves training the FE component in L iterations (Lines 25-37). First, an estimation loss (i.e. Eq.(11)) is initialized to 0 (Line 26), and then the estimation loss for all sequences $\boldsymbol{S}_i$ in $\mathcal{S}'$ is calculated (Lines 27-35). Since $\mathcal{S}'$ includes sequences regarding newly generated graphs, the features of the graphs are calculated (Line 28). Following the same procedure as in the first step, the FE component estimates features, and the estimation loss is calculated and accumulated (Lines 29-34). Note that during the training of the FE component, only the estimation loss is considered, and losses related to the GraphTune component are not taken into account. The weights of the functions f_{emb}, f_{est}, and f_{feat} in the FE component are updated by backpropagating the estimation loss (Line 36). Note that the weights of F_{enc} and F_{dec} are not updated because the loss is not backpropagated to the GraphTune component.

E. GENERATION

When generating graphs with specified feature values after training, you can generate the desired graph by providing the condition vector $\boldsymbol{C}$, whose elements are the specified feature values, and the latent vector $\boldsymbol{z}$ to the decoder F_{dec} of the GraphTune component. The latent vector $\boldsymbol{z}$ is sampled from a multivariate Gaussian distribution. The process of generating graphs in F-GraphTune is exactly the same as the generation process in the original GraphTune shown in Eq. (7). Note that the FE component is utilized only during training and not during generation.

VI. EXPERIMENTS

We verify that F-GraphTune can tune values of structural features with high accuracy. In this section, we present a performance evaluation of F-GraphTune on a real graph dataset extracted from the Twitter who-follows-whom network. Through the evaluation, we show that F-GraphTune yields better performance than the conventional generative models including the original GraphTune.

A. BASELINES

To confirm the basic characteristics of F-GraphTune in a conditional graph generation task, we compare the performance of F-GraphTune with two baseline models: GraphTune [20] and CondGen [12].

GraphTune

The original GraphTune serves as the base model for F-GraphTune; therefore, by comparing it with GraphTune, we can accurately evaluate the effectiveness of feedback from the FE component. To the best of our knowledge, GraphTune is the only model that is oriented towards the continuous tuning of global-level structural features in human relationship graphs including social networks and citation networks. In the evaluations in Section VI, we use the hyperparameters recommended in the paper [20].

CondGen

CondGen employs conditional structure generation through graph variational generative adversarial nets and is one of the few models that achieves a conditional generation for general graphs that is not limited to a specific domain. Unlike the original GraphTune and F-GraphTune, which continuously tune feature values, CondGen can only specify feature values as categorical values using datasets that are pre-categorized by labels. In the evaluations in Section VI, we use the hyperparameters recommended in the paper [12].

B. SETTINGS

The hyperparameters of a model in F-GraphTune are set as follows. For the FE component, we use single-layer LSTM blocks for f_{est}, which has a hidden state vector of dimension 512. The rate of dropout on the LSTM layers is set to 0.5. we add a dropout layer with a dropout rate of 0.3 before the single fully connected layer f_{feat}. The hyperparameters related to the GraphTune component are set to the same values as those adopted in the original GraphTune [20].

We train the neural networks with a batch size of 37 using the Adam optimizer. The initial learning rate is set to 0.001. These configurations are the same as those of the original GraphTune. We set the number of training iterations for the GraphTune component M and the FE component L to 5000 each. The number of iterations for the alternate training algorithm N is set to 2. The number of graphs generated for training the FE component, denoted as n is set to be the same as the number of provided sequences, denoted as $|\mathcal{S}|$. Additionally, we configure the weight parameters: β is set to 3, and γ is set to 1000 for loss calculation.

To evaluate the performance of F-GraphTune on real graphs, we used the graph dataset that is utilized in the evaluation of the original GraphTune [20]. The graphs in the dataset are sampled from the Twitter who-follows-whom graph in the Higgs Twitter Dataset [32]. The detailed sampling method can be found in the original GraphTune paper. In the evaluations, we split the dataset into two parts: the training set

and the validation set. The size ratio of the training set and the validation set are 90% and 10%, respectively.

As structural features of graphs, we focus on the following 5 features, which were also used in the original GraphTune paper: average of shortest path length, average degree, modularity [33], clustering coefficient, and a power-law exponent of a degree distribution [34]. These global-level structural features are selected from survey papers [35], [36] on the measurement of complex network structures, and they have been widely used as graph features of human relationship graphs [37], [38]. For the creation of the dataset, the value of features is rounded to four decimal places.

C. METRICS

In this section, we demonstrate that F-GraphTune outperforms conventional models in accurately generating graphs with specific structural features. Performance comparison among three models, F-GraphTune, GraphTune, and CondGen, is provided.

We trained F-GraphTune, GraphTune, and CondGen using the training set described in Section VI-B, and generated graphs with specific conditions using these trained models. We individually trained these 3 models for each of the 5 focused features mentioned in Section VI-B, resulting in a total of 15 $(= 3 \times 5)$ trained models. After the training process, we generated 300 graphs for each model. For each feature, we selected 3 typical values from the range of values in the training set to use as the condition vector values. The typical values we selected are the same as those used in the original GraphTune paper [20]. In the generation process of both F-GraphTune and GraphTune, we input the condition vectors to the models to specify the feature values of the generated graphs. Since CondGen requires pre-categorized training sets with labels, the training sets are divided into 3 categories using thresholds at the midpoint between typical values.

The performance metrics used for evaluation are Root Mean Squared Error (RMSE). The RMSE metric is defined by the following:

$$\mathrm{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(c^* - c_i')^2}, \tag{13}$$

where c^* and c_i' represent the values specified by the condition vector and the values of the corresponding features for the graph generated in the i-th iteration, respectively. m represents the total number of generated graphs. Since we generate graphs for three typical values of the feature, we compare the average of RMSE for the three typical values.

D. PERFORMANCE EVALUATIONS

The summary of the results of generation for Twitter dataset are listed in TABLE 1. The values in the columns of F-GraphTune, GraphTune, CondGen represents RMSE of the features in graphs generated by each model. The column of Twitter dataset provides statistical information (mean and percentiles) about the distribution of features of graphs included in the dataset for reference. The best and second-best performances among the three models under each condition for a particular feature are highlighted in red and blue, respectively. For CondGen, since all generated graphs were disconnected graphs, we couldn't calculate the average shortest path length of the graphs. Instead, we depicted "–" in the table. Additionally, we calculated the average shortest path length of the largest connected component and presented it in parentheses in the TABLE 1 for reference.

According to the results in TABLE 1, we can confirm that F-GraphTune can accurately specify the feature values of the generated graphs. The RMSE of the graphs generated by F-GraphTune is smaller compared to the other models in most cases, and up to 53.7% reduction of error is achieved. Overall, these results show that feedback from the FE component plays a generally beneficial role in enhancing graph generation accuracy.

To validate the accuracy of the generated graphs, we also evaluated the accuracy of the FE component. The results are shown in Fig. 3. The horizontal axis indicates the feature values estimated by the FE component, while the vertical axis indicates the corresponding ground-truth feature values of the generated graphs. For features such as the clustering coefficient and the power-law exponent of the degree distribution—where the performance improvement achieved by F-GraphTune was limited—the accuracy of the FE component was found to be relatively low. In particular, the power-law exponent of the degree distribution is notably difficult to estimate accurately, as its value tends to be unstable when derived from small graphs used in the evaluation. These results suggest that the performance of F-GraphTune is influenced by the estimation accuracy of the FE component.

VII. LIMITATIONS AND FUTURE DIRECTIONS

Although F-GraphTune resolved a critical issue in GraphTune regarding the accuracy of graph generation, it still shares some limitations with GraphTune.

Accuracy of Feature Estimation

The accuracy of the feature estimation in the FE component is not sufficient to achieve high accuracy in graph generation. Despite its effectiveness in many cases, F-GraphTune does not consistently achieve sufficient accuracy improvement in feature estimation across all feature types. This limitation arises from the fact that the FE component does not consistently produce accurate feature values for the graphs generated by the GraphTune component. One possible way to address this issue is to employ a more sophisticated model for the FE component, such as a Transformer-based architecture.

Generation of Large Graphs

While the number of nodes in our dataset is relatively large compared to previous studies on learning-based conditional graph generation conducted before GraphTune, it is still considered small in the context of social networks. At present,

TABLE 1. RMSE for 5 features in graphs generated by F-GraphTune, GraphTune, and CondGen for the Twitter dataset. The best and second-best performances among the three models under each condition for a particular feature are highlighted in red and blue, respectively.

Global-level structural feature	F-GraphTune RMSE	GraphTune RMSE	CondGen RMSE	Twitter dataset average (25%ile / median / 75%ile)
Average shortest path length	**0.969**	2.093	– (1.64)	4.26 (3.40 / 4.09 / 4.84)
Average degree	**0.666**	0.822	0.868	3.59 (2.96 / 3.44 / 3.92)
Modularity	**0.154**	0.163	0.285	0.550 (0.509 / 0.563 / 0.617)
Clustering coefficient	0.178	0.194	**0.172**	0.203 (0.152 / 0.196 / 0.251)
Power-law exponent of a degree distribution	0.985	**0.944**	1.73	4.28 (2.91 / 3.48 / 4.23)

both GraphTune and F-GraphTune are unfortunately unable to generate graphs with more than 100 nodes. Overcoming this limitation will require further innovation. Hierarchical generation [15] and combining with traditional statistical random graph models are promising approaches in this context.

Generation of Extrapolation

Though F-GraphTune has enhanced the graph generation accuracy of GraphTune by incorporating the FE component, it did not improve the extrapolation performance of GraphTune. The FE component is trained on graphs from the original dataset and those generated by the GraphTune component. Therefore, it lacks the ability to provide accurate feedback for graphs that fall outside the range of feature values present in the original dataset. However, the technique of specifying graph features provided by F-GraphTune (or GraphTune) could be a crucial technology for addressing the challenge of extrapolation in graph generation tasks. When specifying values near the boundaries of the range of graph features in the original dataset, some graphs are generated that lie outside those boundaries. By extending the original dataset with graphs generated in this manner, it may be possible to train a graph generative model capable of performing extrapolative graph generation.

VIII. CONCLUSIONS

In this paper, we proposed F-GraphTune, a Feedback-enhanced Graph generative model with Tunable features, which enables accurate specification of feature values in generated graphs. F-GraphTune extends GraphTune by incorporating the FE component, which estimates and provides feedback on feature values of graphs reconstructed by GraphTune. We have also developed the alternate training algorithm that allows the GraphTune component and the FE component to cooperate without leaking information regarding the feature values of input graphs.

We conducted experiments using a graph dataset from a real social network, and the results demonstrate that F-GraphTune can tune graph features more accurately than conventional models. In our experiments, among the five tuned features, F-GraphTune exhibited the best performance in 3 features, showing an improvement in accuracy of up to 53.7%.

Based on the experiments conducted in this paper, there are several promising prospects for F-GraphTune. It is highly intriguing to explore the boundaries of the space in which graphs can exist by generating extrapolated graphs. F-GraphTune can enable us to investigate this space by simultaneously specifying multiple interdependent features.

REFERENCES

[1] V. T. Hoang, H.-J. Jeon, E.-S. You, Y. Yoon, S. Jung, and O.-J. Lee, "Graph Representation Learning and Its Applications: A Survey," *Sensors*, vol. 23, no. 8, 2023.
[2] G. Dou, "Scalable Parallel and Distributed Simulation of an Epidemic on a Graph," *PLOS ONE*, vol. 18, no. 9, 2023.
[3] P. Erdös and A. Rényi, "On Random Graphs I," *Publicationes Mathematicae*, vol. 6, no. 26, pp. 290–297, 1959.
[4] D. J. Watts and S. H. Strogatz, "Collective Dynamics of 'Small-World' Networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
[5] R. Albert and A.-L. Barabási, "Statistical Mechanics of Complex Networks," *Reviews of Modern Physics*, vol. 74, no. 1, pp. 47–97, 2002.
[6] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic Blockmodels: First Steps," *Social Networks*, vol. 5, no. 2, pp. 109–137, 1983.
[7] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, "GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models," in *Proc. of the 35th International Conference on Machine Learning (ICML 2018)*, 2018.
[8] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, "Learning Deep Generative Model of Graphs," in *Proc. of 6th International Conference on Learning Representations (ICLR 2018) Workshop*, 2018.
[9] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating Graphs via Random Walks," in *Proc. of the 35th International Conference on Machine Learning (ICML 2018)*, 2018, pp. 609–618.
[10] M. Simonovsky and N. Komodakis, "GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders," in *Proc. of the 27th International Conference on Artificial Neural Networks (ICANN 2018)*, 2018.
[11] R. Assouel, M. Ahmed, M. H. Segler, A. Saffari, and Y. Bengio, "DEFactor: Differentiable Edge Factorization-Based Probabilistic Graph Generation," *arXiv*, 2018.
[12] C. Yang, P. Zhuang, W. Shi, A. Luu, and P. Li, "Conditional Structure Generation through Graph Variational Generative Adversarial Nets," in *Proc. of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019, pp. 1338–1349.
[13] J. Lim, S.-Y. Hwang, S. Moon, S. Kim, and W. Y. Kim, "Scaffold-Based Molecular Design with a Graph Generative Model," *Chemical Science*, vol. 2020, no. 4, pp. 1153–1164, 2020.
[14] N. Goyal, H. V. Jain, and S. Ranu, "GraphGen: A Scalable Approach to Domain-agnostic Labeled Graph Generation," in *Proc. of the Web Conference 2020 (WWW 2020)*, 2020, pp. 1253–1263.

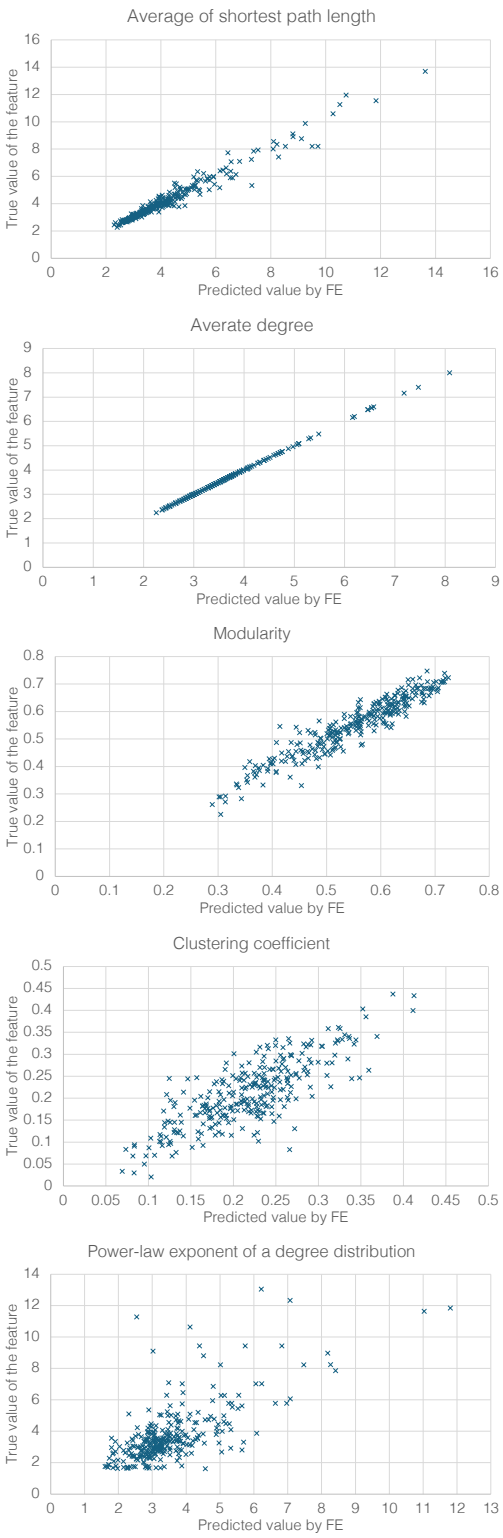


FIGURE 3. Accuracy of the FE component in F-GraphTune. The horizontal axis indicates the feature values estimated by the FE component, while the vertical axis indicates the corresponding ground-truth feature values of the generated graphs.

[15] W. Jin, R. Barzilay, and T. Jaakkola, "Hierarchical Generation of Molecular Graphs using Structural Motifs," in *Proc. of the 37th International Conference on Machine Learning (ICML 2020)*, 2020.

[16] X. Guo and L. Zhao, "A Systematic Survey on Deep Generative Models for Graph Generation," *arXiv*, 2020.

[17] F. Faez, Y. Ommi, M. S. Baghshah, and H. R. Rabiee, "Deep Graph Generators: A Survey," *IEEE Access*, vol. 9, pp. 106 675–106 702, 2021.

[18] A. Bonifati, I. Holubová, A. Prat-Pérez, and S. Sakr, "Graph Generators: State of the Art and Open Challenges," *ACM Computing Surveys*, vol. 53, no. 2, 2021.

[19] S. Nakazawa, Y. Sato, K. Nakagawa, S. Tsugawa, and K. Watabe, "A Tunable Model for Graph Generation Using LSTM and Conditional VAE," in *Proc. of the 41st IEEE International Conference on Distributed Computing Systems (ICDCS 2021) Poster Track*, 2021.

[20] K. Watabe, S. Nakazawa, Y. Sato, S. Tsugawa, and K. Nakagawa, "GraphTune: A Learning-Based Graph Generative Model With Tunable Structural Features," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 4, pp. 2226–2238, 2023.

[21] F. Faez, N. H. Dijujin, M. S. Baghshah, and H. R. Rabiee, "SCGG: A Deep Structure-Conditioned Graph Generative Model," *PLOS ONE*, vol. 17, no. 11, 2022.

[22] A. M. Tseng, N. Diamant, T. Biancalani, and G. Scalia, "GraphGUIDE: Interpretable and Controllable Conditional Graph Generation with Discrete Bernoulli Diffusion," *arXiv*, 2023.

[23] N. Stoehr, M. Brockschmidt, J. Stuehmer, and E. Yilmaz, "Disentangling Interpretable Generative Parameters of Random and Real-World Graphs," in *Proc. of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019) Workshop*, 2019.

[24] A. Vázquez, "Growing Network with Local Rules: Preferential Attachment, Clustering Hierarchy, and Degree Correlations," *Physical Review E*, vol. 67, no. 5, 2003.

[25] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A Recursive Model for Graph Mining," in *Proc. of the 2004 SIAM International Conference on Data Mining (SDM 2004)*, 2004.

[26] T. G. Kolda, A. Pinar, T. Plantenga, and C. Seshadhri, "A Scalable Generative Graph Model with Community Structure," *SIAM Journal on Scientific Computing*, vol. 36, no. 5, 2014.

[27] Y. Li, L. Zhang, and Z. Liu, "Multi-Objective De Novo Drug Design with Conditional Graph Generative Model," *Journal of Cheminformatics*, vol. 10, no. 33, 2018.

[28] E. Jonas, "Deep Imitation Learning for Molecular Inverse Problems," in *Proc. of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019.

[29] J. Liu, Y. Chi, and C. Zhu, "A Dynamic Multiagent Genetic Algorithm for Gene Regulatory Network Reconstruction Based on Fuzzy Cognitive Maps," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 2, pp. 419–431, 2016.

[30] Y. Wang, W. Che, J. Guo, and T. Liu, "A Neural Transition-Based Approach for Semantic Dependency Graph Parsing," in *Proc. of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*, 2018.

[31] B. Chen, L. Sun, and X. Han, "Sequence-to-Action: End-to-End Semantic Graph Generation for Semantic Parsing," in *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, 2018, pp. 766–777.

[32] M. D. Domenico, A. Lima, P. Mougel, and M. Musolesi, "The Anatomy of a Scientific Rumor," *Scientific Reports*, vol. 3, no. 2980, 2013.

[33] M. E. J. Newman and M. Girvan, "Finding and Evaluating Community Structure in Networks," *Physical Review E*, vol. E, no. 69, 2004.

[34] A. Jeff, B. Ed, and P. Dietmar, "powerlaw: A Python Package for Analysis of Heavy-Tailed Distributions," *PLOS ONE*, vol. 9, no. 1, 2014.

[35] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwanga, "Complex Networks: Structure and Dynamics," *Physics Reports*, vol. 424, no. 4-5, pp. 175–308, 2006.

[36] L. da F. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas, "Characterization of Complex Networks: A Survey of Measurements," *Advances in Physics*, vol. 56, no. 1, pp. 167–242, 2007.

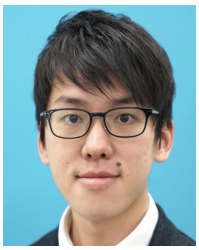
[37] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a Social Network or a News Media?" in *Proc. of the 19th International Conference on World Wide Web (WWW 2010)*, 2010, pp. 591–600.

[38] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the Evolution of User Interaction in Facebook," in *Proc. of the 2nd ACM Workshop on Online Social Networks (WOSN 2009)*, 2009, pp. 37–42.

TAKAHIRO YOKOYAMA received the B.E. and M.E. degrees in Engineering from Nagaoka University of Technology, Niigata, Japan, in 2022 and 2024, respectively.

YOSHIKI SATO received the B.E. and M.E. degrees in Engineering from Nagaoka University of Technology, Niigata, Japan, in 2021 and 2023, respectively.

SHO TSUGAWA received the M.E. and Ph.D. degrees from Osaka University, Japan, in 2009 and 2012, respectively. He is currently an Associate Professor of the Institute of Systems and Information Engineering, University of Tsukuba, Japan. His research interests include network science, social network analysis, and computational social science. He is a member of IEEE, ACM, IEICE, and IPSJ.



KOHEI WATABE received his B.E. and M.E. degrees in Engineering from Tokyo Metropolitan University, Tokyo, Japan, in 2009 and 2011, respectively. He also received the Ph.D. degree from Osaka University, Japan, in 2014. He was a JSPS research fellow (DC2) from April 2012 to March 2014. He was an Assistant Professor of the Graduate School of Engineering, Nagaoka University of Technology, from April 2014 to October 2019. He has been an Associate Professor of the Graduate School of Engineering, Nagaoka University of Technology, from November 2019 to March 2024. He has been an Associate Professor of the Graduate School of Engineering, Nagaoka University of Technology, since April 2024. He is a member of the IEEE and the IEICE.