INVITED PAPER

Random-based and Deep Graph Generators: Evolution and Future Prospects

Kohei WATABE[†a)], ***Member***

SUMMARY Graphs are highly flexible data structures that can model various data and relationships. By using graphs, we can abstract and represent various things in the real world. The technology of artificially generating graphs is important in various fields where graphs are applied to various fields in engineering, including communication networks, social networks, and so on. In this paper, we organize and introduce graph generation techniques from early random-based methods to the latest deep graph generators, focusing on the aspects of feature reproduction and specification. Techniques for reproducing and specifying graph features in graph generation may provide new research methods for classical graph theory and optimization problems on graphs. This paper also presents recent achievements that may lead to further exploration in these fields and discusses the future prospects of graph generation.

key words: *graph generation, generative model, deep graph generator, graph feature, conditional generation*

1. Introduction

The graph structure is an extremely versatile and fundamental data structure, and various things in the real world can be abstractly represented by graphs. A graph is a mathematical structure represented by a set of nodes (vertices) and a set of edges (links), and it can simply represent the relationships between objects. In the field of transportation networks (e.g., railway networks, bus route maps, road networks) and communication networks (e.g., the Internet and telephone networks), graphs have long been used as a means of mathematically handling objects. In addition, human relationships in social networks, electrical circuits in electrical engineering, and molecular structures in molecular engineering are also represented by graphs. A tree structure is also a special case of a graph structure, and it frequently appears in the structure of databases, program code, and so on. In the field of machine learning, which has made remarkable progress in recent years, the tree structure of decision trees and the network structure of neural networks are also representations by graphs.

The technology of artificially generating graphs is important in various fields such as communication networks, social networks, transportation networks, databases, molecular engineering, and epidemiology. Specifically, applications include understanding interaction dynamics in social networks [1], link prediction [2], community detection [3], recommendation systems [4], drug discovery [5, 6], code completion [7], and epidemic modeling [8]. In simulations on graphs in various fields, the effectiveness of a method can only be demonstrated by repeating simulations with a large number of graphs that have specific features, but it is not always possible to obtain a sufficient amount of real graph datasets. Additionally, it can be applied to prediction tasks by generating future graphs or non-existent graph structures, such as predicting future social network structures, developing new drugs through the generation of unknown molecular structures, and suggesting possible code completions from incomplete program code.

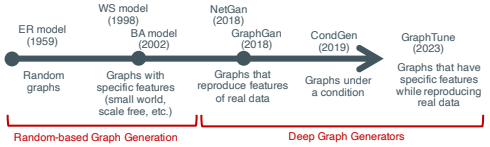


Fig. 1 The history of graph generation techniques.

The early graph generation techniques were based on random algorithms, aiming to generate graphs that reproduce a single aspect of features of real-world graphs. The first proposed graph generation model was the Erdős-Rényi (ER) model [9], proposed in 1959, which is a simple model that randomly connects pre-defined nodes with edges. Around 2000, models such as the Watts-Strogatz (WS) [10] model and the Barabási-Albert (BA) model [11] were proposed, aiming to reproduce features such as small-worldness and scale-free property, and they have inspired many following models (Fig. 1). These random-based graph generators have greatly contributed to our understanding of how these structures of small-worldness and scale-free property are created by reproducing the feature of real-world graphs with minimal algorithms.

With the development of deep learning-based generation techniques for image and natural language processings, a new type of graph generation model called deep graph generators, which learn and reproduce every aspect of features of real-world graph data, has emerged around 2018 [12, 13]. Random-based graph generators excel in generating graphs with lightweight algorithms guaranteeing the reproduction of a single aspect of features. However, there is no guarantee that they will reproduce features other than the ones they focus on. As a result, using graphs generated by random-based graph generators as substitutes for real-world graphs is challenging. In contrast to random-based graph genera-

Manuscript received January 1, 2015.
Manuscript revised January 1, 2015.

[†]The author is with the Graduate School of Science and Engineering, Saitama University, Saitama city, Saitama 338-8570, Japan.
a) E-mail: kwatabe@mail.saitama-u.ac.jp

DOI: 10.1587/trans.E0.??.1

tors designed to reproduce a specific feature such as small-worldness and scale-free property, deep graph generators aim to capture and reproduce the features of a set of graphs provided as a dataset in every aspect.

Recently, some studies have introduced generative models capable of conditionally generating graphs with specified features, alongside many deep graph generators that generate graphs with the same features as the given dataset. Some models including models proposed by Li *et al.* [14], Faez *et al.* [15], and Tseng *et al.* [16], allow specifying the number of subgraphs with specific structures, such as rings or hexagons. CondGen [17] is a conditional graph generator for general graphs that can categorize graphs in a dataset by arbitrary semantic information and generate graphs specified by the category. Our research group has proposed GraphTune, which can specify a feature continuously while reproducing the features of given graph data in every aspect [18, 19].

In this paper, we organize and introduce graph generation techniques from classical random-based generators to the latest deep graph generators, focusing on the aspects of feature reproduction and specification. We also discuss the new research fields that these techniques open up. While there are several surveys on graph generation techniques, they have typically focused on summarizing random-based generators [20, 21] and deep graph generators [12, 13], separately. However, the essence of the graph generation problem lies in sampling from the graph space, where all potential graphs exist. Developing random-based and deep graph generators is an attempt to explore methods to sample graphs with specific features from the graph space homogeneously. This paper differs from conventional surveys by comprehensively addressing both random-based and deep graph generators, focusing on how each generator samples graphs with a specific feature from the graph space. Additionally, the techniques that generate graphs with a specific feature may provide new research methods for classical graph theory and optimization problems on graphs. This paper also discusses the prospects of the new research methods.

The rest of this paper is structured as follows. First, in Section 2, we explain how the graph generation problem is formulated. Next, in Sections 3 and 4, we introduce representative random-based and deep graph generators, respectively. In Section 5, we describe the future challenges of the current graph generators and the direction of future research. Finally, in Section 6, we conclude this paper.

2. Formulation of Graph Generation Problem

A graph is defined as a set of nodes and edges. As a notational convention, a graph is represented by $G = (V, E)$, where V and E denote a set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ and a set of edges $E = \{(u, v) \mid u, v \in V\}$, respectively.

As an approach to analyzing graphs, various quantified features representing properties of graphs have been proposed and widely used in many studies. The number of nodes $|V|$ and the number of edges $|E|$ can be treated as the most primitive quantified features of graphs. In addition to the features, numerous other quantified features such as average degree, average shortest path length, and clustering coefficient, have been proposed. Table 1 shows representative features. In applications such as social networks, there are cases where large graphs with a large number of nodes $|V|$ and edges $|E|$ are handled. In such cases, calculating the values of the features of the graph to capture the rough properties of the graph is an extremely effective approach. In other words, graph features are indicators that represent a single aspect of a graph, and the vector composed of the features can be considered as a summarization of the graph. There are many convenient tools for calculating features, and calculating representative features such as those listed in Table 1 can be easily done using graph analysis libraries such as NetworkX [22, 23] or igraph [24].

Table 1 Representative graph features.

Category	Quantified features
Graph size	Number of nodes
	Number of edges
	Largest component size
	Average shortest path length
	Diameter
Connectivity	Average degree
	Edge density
	Power-law exponent of degree distribution
	Assortativity
	Reciprocity
Community structure	Clustering coefficient
	Clieque
	Modularity
	Fraction of the giant component
Othres	Largest eigenvalue of an adjacency matrix

The graph generation problem can be understood as sampling graphs with specific features from a vast graph space. Here, the graph space refers to a set that includes all possible graphs, and it is possible to define distances between graphs based on their features. In the context of graph generation, we design a function $f(\theta)$ that takes as input parameters θ, as illustrated in Fig. 2. The function $f(\theta)$ corresponds to the algorithm or model for generating graphs, and its output is a graph. The input parameters θ define a subspace of the graph space corresponding to the population of graphs from which the output graph is sampled. The output graph of $f(\theta)$ can be regarded as a sample from the subspace defined by the parameter θ. Depending on an algorithm or model, the parameter θ can be given by the user or determined by machine learning technology based on a dataset. The subspace corresponding to θ depends on the design of $f(\theta)$, that is, the design of the generation algorithm or model. In an ideal $f(\theta)$, a subspace corresponds to a set of graphs with specific feature values, enabling to output only graphs with those specific features. This subspace is generally vast, and it is practically difficult to examine what kind of graphs are included in it, apart from trying to sample graphs. It is important to note that enumerating all graphs with specific features is

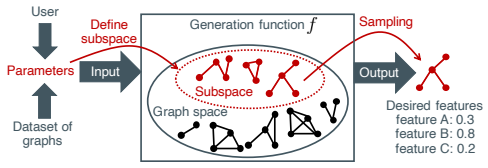


Fig. 2 Formulation of the graph generation problem.

generally impossible, except in cases where the graph space is significantly constrained, such as when the number of nodes is small. When nodes are distinguished by node ID, there are $2^{|V|(|V|-1)/2}$ possible graphs for a graph with $|V|$ nodes, leading to a combinatorial explosion concerning the number of nodes.

3. Random-based Graph Generators

The random-based graph generators output various graphs by determining the connections between nodes randomly, and many of them focus on reproducing a single aspect of features of real-world graphs. In this section, we introduce representative models of random-based graph generators and discuss the spatial properties of the generated graphs and issues of models.

3.1 Reproduction of the Number of Nodes and Edges

The ER model proposed in 1959 [9] is the most primitive graph generator with an extremely simple generation algorithm connecting nodes with a certain probability. The ER model has parameters for the number of nodes n and the probability p of connecting edges between nodes. In the algorithm, any pair of nodes is connected with probability p for a given set of nodes. Since edges are randomly connected, the node degrees (the number of edges connected to a node) are homogeneous, resulting in graphs without any special nodes. Examples of a graph generated by the ER model are shown on the left side of Fig. 3. Nodes are arranged in a circular pattern for visibility, but the links are randomly connected, showing no significant bias in the node degrees. The number of nodes $|V|$ and the number of edges $|E|$ are the most basic features of a graph as the ER model is designed to specify these features with n and $pn(n-1)/2$, respectively.

Inheriting the properties of the ER model, many subsequent generators allow for specifying the number of nodes $|V|$ and edges $|E|$. The stochastic Block (SB) model [25] is a model that can specify the number of nodes and edges inherited from the ER model, and it is commonly used in benchmark datasets for community detection algorithms.

3.2 Reproduction of Small-Worldness

In 1998, a random-based graph generation model was proposed to reproduce a property called small-worldness observed in real-world graphs [10]. Small-worldness refers to the property where the average shortest path length L between any two nodes in a graph is small relative to the number of nodes n. This feature is commonly found in real-world graphs such as co-authorship networks and power grids. Specifically, a small-world network is defined as a network in which the average path length L scales like $\log n$. Watts and Strogatz proposed the WS model to reproduce small-worldness with a very simple algorithm. They start with a ring lattice with an average degree of $2K$ and randomly rewire each edge with probability p to another node, generating a small-world network [10]. Examples of a graph generated by the WS model are shown in the middle of Fig. 3. Most neighboring nodes are connected by edges, indicating strong local connections, while some nodes are connected to distant nodes, contributing to a graph with a small average shortest path length. There are many models other than the WS model that reproduce small-worldness. A representative of such a model is the extension of the WS model to a D-dimensional lattice proposed by Biskup [26]. Additionally, models that reproduce scale-free properties, which will be introduced in the next section, mostly reproduce small-worldness.

3.3 Reproduction of Scale-Free Property

After the proposal of the WS model to reproduce small-worldness, a random-based generator focusing on a feature known as the scale-free property, which appears as a power-law distribution of node degrees, was proposed [11]. In graphs with scale-free property such as social networks and food web relationships, a few nodes are connected to a large number of nodes, while many nodes are connected to only a few nodes. More precisely, it refers to graphs where the distribution $f(k)$ of node degrees follows $f(k) \propto k^{-\gamma}$, where $2 \leq \gamma \leq 3$. Albert *et al.* proposed the BA model [11], starting with m complete graphs and repeatedly adding new nodes with m edges. By connecting a new node to existing nodes with a probability proportional to their node degree, the BA model enables the generation of scale-free graphs whose exponent γ of power-law distribution of node degrees is 3. An example of a graph generated by the BA model is shown on the right side of Fig. 3. In contrast to the ER model or the WS model, the connections of edges are concentrated on nodes located in the upper right, indicating a significant degree of heterogeneity in node degrees.

Since the proposal of the BA model, numerous models replicating scale-free properties have been suggested. Representative models include the R-MAT model [27] and RTG [28]. Furthermore, there are more generalized models such as BTER [29] and Darwini [30], which allow specifying distributions other than power-law distributions.

3.4 Spatial Properties of Graphs Generated by Random-Based Models

As random-based generators have evolved, various aspects of graph features, such as small-worldness and scale-free property, have been highlighted, and quantified features have been

studied to evaluate these aspects. Commonly used quantified features include average degree, clustering coefficient, and modularity, in addition to the average shortest path length used as a metric for small-worldness and the power-law exponent γ used to quantify scale-free property.

As mentioned above, there are numerous features to summarize a graph, but there is no random-based graph generator that faithfully reproduces or specifies all these features. As shown in Section 3.2, the WS model can reproduce a small average shortest path length relative to the network size, though it cannot necessarily tune other features independently. Even for tunable features (e.g., the average shortest path length in the WS model), it is necessary to manually tune parameters after deeply understanding the nature of the model, the graph generation process, and the dependencies between each feature. Therefore, it is still difficult even for experts to generate graphs with specific features. Fig. 5 calculates the features of 300 graphs generated by the WS model with $K = 2$ and displays scatter plots of the pairs of features indicated on the horizontal and vertical axes. Additionally, the plots on the diagonal of Fig. 5 show the distributions of each feature. When we depict the features of graphs generated by the WS model in the aspect of multiple features like Fig. 5, it can be seen that each feature has a high dependency specific to the model and independent tuning is not possible in the restriction of the model. Considering the nature of the features, it should be possible to determine the clustering coefficient with high freedom even when the average shortest path length is fixed. However, only graphs with specific values are generated in Fig. 5, indicating that homogeneous sampling from the graph space is not achieved.

4. Deep Graph Generators

Random-based graph generators cannot reproduce many features of real-world graphs in the multi-aspect of the features. They have significantly influenced numerous studies by focusing on a specific feature of real-world graphs and reproducing them with simple models. However, as mentioned earlier, real-world graphs contain many features, and the features used to represent graphs vary by domain. Even models like the WS or BA models, which can tune specific features with parameters, cannot tune all features simultaneously. Additionally, each feature depends on the model and cannot be specified independently.

Since around 2018, research on graph generation using deep learning, called deep graph generators, has been actively pursued and these models aim to reproduce features of graph datasets in every aspect. The background of this trend is the rapid development of generative models in deep learning, such as the proposal of Variational AutoEncoder (VAE) [31] and Generative Adversarial Network (GAN) [32] in 2014. These models, which were earlier developed for tasks such as image or language processing, have been extended to more complex data structures like graphs in the late 2010s. A key technical challenge in graph generation models is effectively inputting complex graph structures into deep learning frameworks, which are primarily designed for processing vectors or tensors. Various approaches have been attempted, such as converting adjacency matrices into images or treating walks on graphs as sequence data. Another challenge is the discrete nature of graphs, which complicates graph generation with deep learning. To address this, methods have been developed to convert graph structures into continuous vector representations.

4.1 Reproduction of Graphs in Datastes

Deep graph generators aim to reproduce graphs that inherit the features of a graph dataset used as input. From the perspective of reproducibility, the generated graphs should have the same features as those of the input graph dataset in every aspect. Therefore, the accuracy of generation is evaluated by calculating and comparing multiple features of the graphs in the dataset and the generated graphs. Naturally, the simplest way to match features is to output the graphs in the dataset as they are. However, this approach is not practically meaningful, so the model is required to generate graphs that reproduce statistical features while also generating graphs that do not exist in the dataset efficiently. Therefore, it is necessary to evaluate based on metrics such as novelty, which is defined as the ratio of generated graphs that are not present in the dataset, and uniqueness, which is defined as the ratio of unique graphs among the generated graphs. The superior generation model is expected to not generate graphs that are included in the dataset and the same graph multiple times, and novelty and uniqueness are used to evaluate these aspects. In an actual evaluation of novelty and uniqueness, it is not easy to calculate these metrics of the generated graphs since the representations of graphs are not unique (e.g., $G_1 = \{\{A, B, C\}, \{(A, B), (B, C)\}\}$ and $G_2 = \{\{A, B, C\}, \{(A, C), (B, C)\}\}$ are different representations of the same graph). Isomorphic graphs that have the same structure but different representations should be treated as the same graph in the evaluation of novelty and uniqueness. However, the graph isomorphism problem is known to be computationally difficult, thus, it is necessary to use approximate methods like the Weisfeiler-Lehman test [33].

Models that directly reproduce the features of graphs in the dataset have been proposed since the inception of deep graph generators [34–53]. Studies in the field of designing molecules have been particularly active, with various models have been proposed [34–40]. However, most of these models utilize domain-specific knowledge of molecule chemistry, making them difficult to apply to other fields. On the other hand, several domain-independent models have also been proposed, and they can be categorized based on how to convert graphs into vectors or tensors as inputs to neural networks. Conversion methods include 1) using the adjacency matrix of the graph [41–43], 2) converting graphs into edge lists [44–47], and 3) treating walks on graphs as node sequences [48–50] or edge sequences [51–53]. Representative models often used as benchmarks for new methods include GraphVAE [42], GraphRNN [48] and Graphite [46].

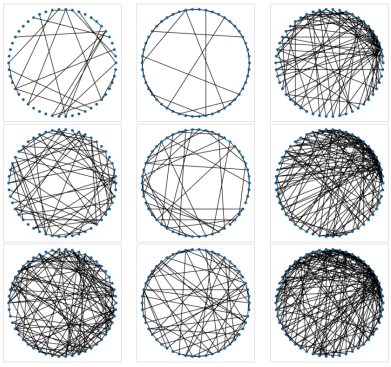


Fig. 3 Graphs generated by random-based generators.

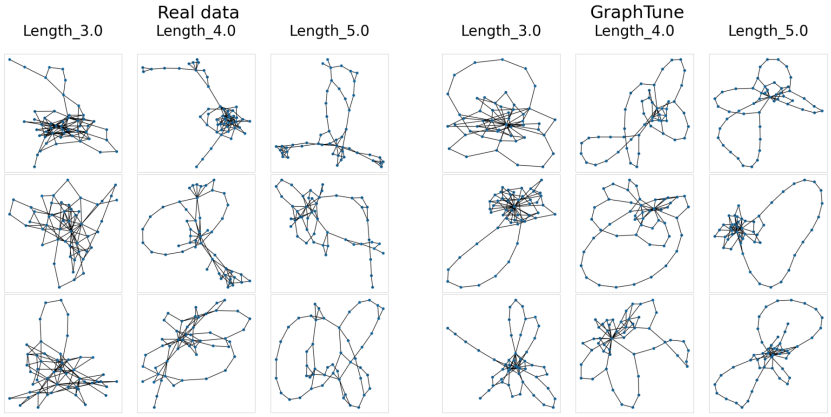


Fig. 4 Real graphs in training dataset and graphs generated by GraphTune.

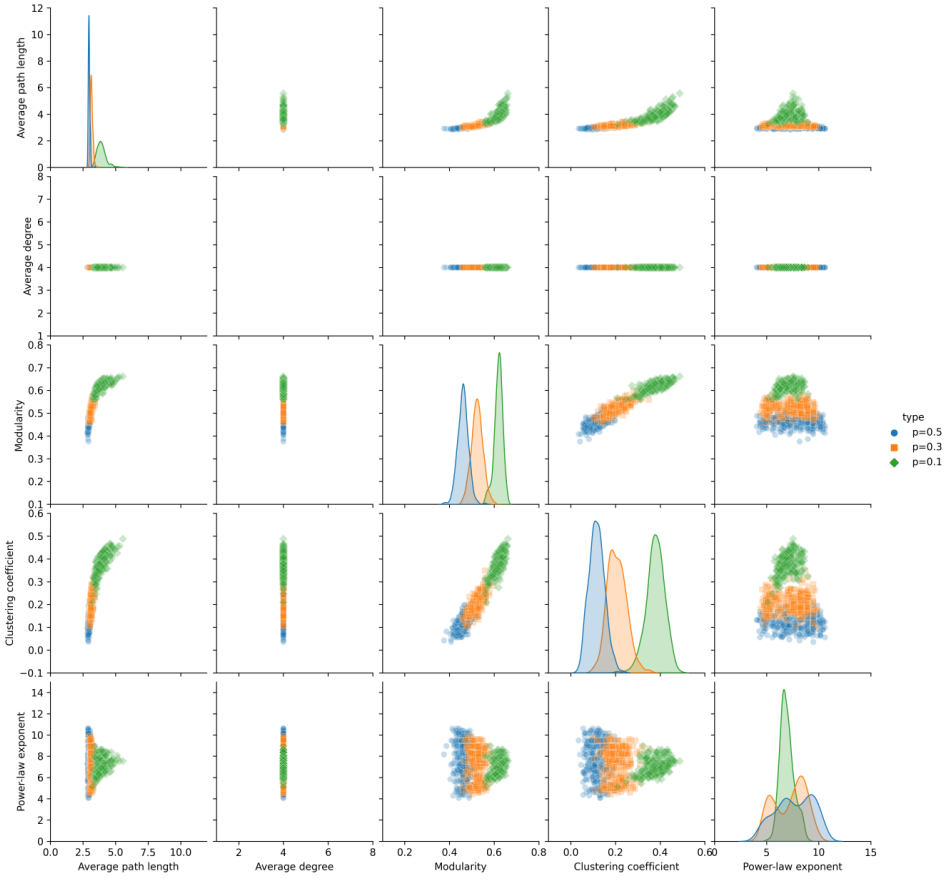


Fig. 5 Spatial properties of graphs generated by the WS model.

The model using edge sequences, a relatively recent approach, has achieved success in terms of accuracy and scalability, with GraphGen [53] being a representative model. In the edge sequence approach, walks on the graph are converted into sequences of edges, which are then input into a Recurrent Neural Network (RNN)-based deep learning model for time-series processing, enabling graph generation. One of the most successful graph generation models, GraphGen, converts sequences of edges with Depth-First Search (DFS) code [54], which is a sequence of edges ordered by depth-first search. Then the edge sequence is input into a Long Short Term Memory (LSTM) network for time-series processing. By solving the sequence prediction task with LSTM, it outputs the predicted sequence to achieve graph generation.

Though most of the models mentioned above are de-

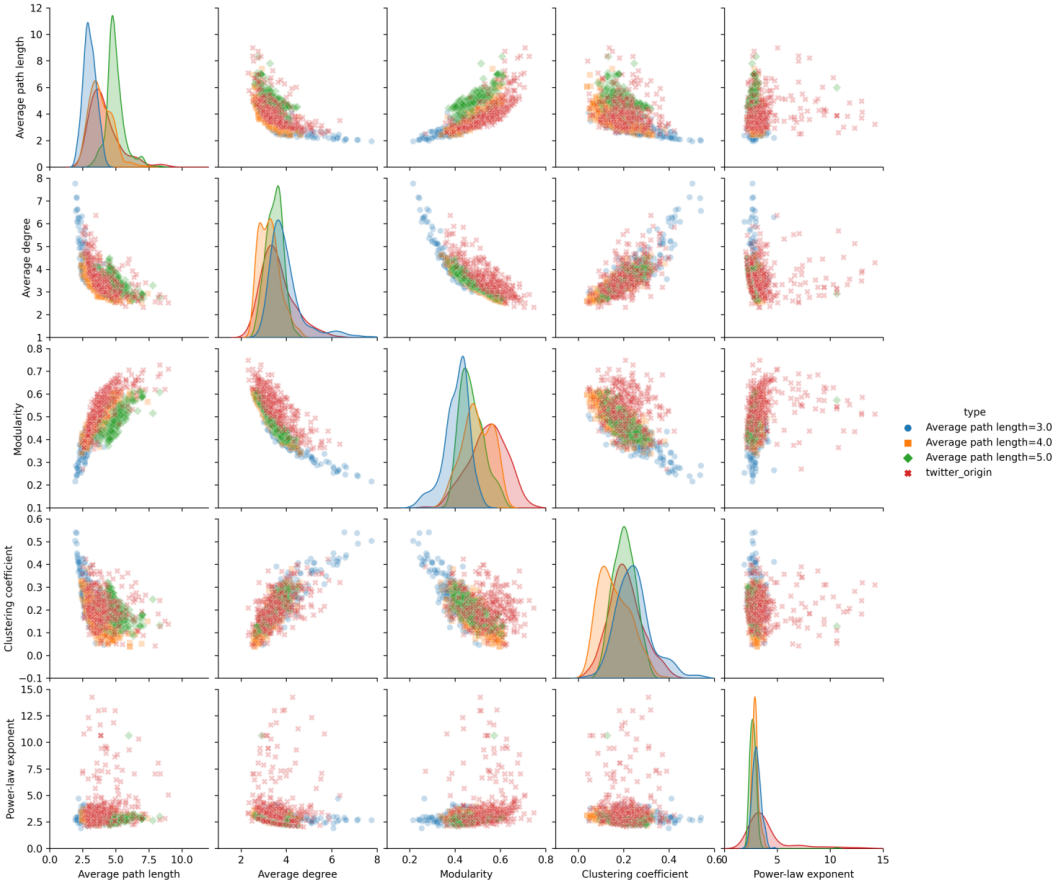


Fig. 6 Spatial properties of graphs generated by GraphTune.

signed for undirected graphs, a part of them can be applicable to directed graphs. D-VAE [50] is a representative model that explicitly supports directed graphs, and it can generate acyclic directed graphs by leveraging graph neural networks. TSGG-GAN [55] is also a model that supports directed graphs for temporal graph generation. In addition to the two models, the models that can generate graphs with edge labels (i.e., edge attributes like weights, types, and so on) can be easily extended for directed graphs. In these models supporting edge labels, we can generate directed graphs by adding edge directions as edge labels. GraphGen [53] and GraphRNN [48] are representative models that can generate graphs with edge labels, and You *et al.* [48] have mentioned that GraphRNN can be extended to directed graphs in the appendix of their paper. Moreover, adjacency matrix-based models [41–43] may be applicable to directed graphs by learning asymmetric adjacency matrices though they have not mentioned it in their papers. Unfortunately, though there are lots of models that have the potential to be extended to directed graphs, actual evaluations with real-world datasets are limited.

4.2 Specification of Graph Features

While many deep graph generators aim to reproduce the features of graphs in a given dataset, models that can tune and specify a feature within the dataset are more useful. Considering practical applications, models that can tune and specify features are more practical, such as when we evaluate the impact of specific features in graph simulations. Naturally, reproducing the features of real-world datasets is also important, so models are required that can tune specific features while reproducing the other features of graphs in the dataset in the multi-aspect of the features. Models that can tune features are known in the context of general generative models as conditional generative models, which generate data with a specific attribution that is given as a condition.

While there are models within deep graph generators that correspond to conditional generative models capable of tuning specific features, many of them focus on specifying the local structure of graphs. One of the few models that can generate general graphs regardless of the domain and achieve conditional generation is DeepGMG [14]. In the case of DeepGMG, it allows tuning the number of local structures such as triangles, squares, and hexagons. Moreover, models that adopt motif-based generation combine local structures (motifs) as subgraphs to generate graphs, enabling tuning of local structures similerly [56].

On the other hand, models that can tune global features of graphs, such as average shortest path length and clustering

coefficient, which are calculated from the entire graph, are currently limited in number. CondGen [17] is one of these models, that categorizes graphs in a dataset and assigns semantic labels to specify features. Additionally, Stoehr *et al.* [57] have successfully embedded graphs into a disentangled latent space. The generative model with a disentangled latent space can be considered that features of the generated graphs may be tuned by changing values of vectors in the latent space. However, unfortunately, the study by Stoehr *et al.* [57] does not provide an explicit method to specify features and its specification accuracy is not very high.

We have proposed GraphTune [18, 19], which can tune specific features of graphs continuously while maintaining their overall features. Like GraphGen [53], GraphTune takes sequences of edges converted from graphs as input and output of a neural network. GraphTune achieves diverse graph generation based on random numbers by encoding graph features into a latent space z using a VAE and then decoding them. The key characteristic of GraphTune is its ability to overwrite information related to a specific feature in the latent space z. By this characteristic, GraphTune is capable of generating graphs with a specific value of a feature while maintaining the values of the other features of graphs in a dataset.

The experimental results validating the generated graphs by GraphTune indicate that GraphTune can explicitly specify the specific feature while reproducing the features of the given dataset in every aspect. In the experiments, GraphTune was trained on induced subgraphs with 50 nodes sampled from the who-follows-whom graph on Twitter. Examples of graphs generated by tuning the average shortest path length as 3.0, 4.0, and 5.0 are shown in Fig. 4 alongside graphs used in the training. The distribution of each feature of the generated graphs is shown in Fig. 6. In Fig. 6, the plots in red indicate the distribution of graphs used in the training dataset, and it can be seen that most of the generated graphs fall within the range of the dataset's graph distribution. Additionally, it can be confirmed that the average shortest path length of the generated graphs is concentrated around the specified values of 3.0, 4.0, and 5.0, verifying that GraphTune generates graphs as specified.

5. Future Directions of Graph Generation Models

5.1 Advancement of Techniques for Tuning Features

Research on graph generation has evolved from random-based generators to deep graph generators, and it is expected that research on conditional generation will significantly advance in the future. There has been increasing research on techniques to generate graphs similar to those given in a dataset, with accurate models such as NetGan and GraphGen having emerged. On the other hand, there has been insufficient exploration of techniques for conditionally generating graphs (i.e., specifying a feature of a generated graph). As mentioned above, research on conditional generation is very limited, with CondGen [17] being one of the few approaches that categorize graphs in a dataset and aim for group-specific generation. Our proposed GraphTune establishes a technique that provides continuous values of a feature as a condition to the model in the context of conditional generation techniques of graphs.

While our proposed GraphTune is a pioneering model that aims to tune arbitrary features of graphs, the accuracy of feature specification is still not high, and specifying multiple features arbitrarily remains challenging. GraphTune allows tuning of mean values of various features, such as average shortest path length and clustering coefficient, but unfortunately, the variance of the features of the generated graphs is not always small. The small variance leads to accuracy in specifying the features, and the improvement of the accuracy is a current challenge. Even for specifying features such as the average shortest path length mentioned in the previous section, higher accuracy is required in practical applications. Besides of challenge in accuracy, while GraphTune can input any feature as a condition to the model without any modification of model architecture, a part of the features and a part of the combinations of features are difficult to specify. EEspecially, the specification of the power-law exponent of the degree distribution which is one of the most representative features of graphs is still difficult, so specifying it is an important challenge.

Improvements in specification accuracy can be achieved through several approaches, such as refining the model architecture and improving preprocessing of input data. Many researchers are currently exploring various approaches, and our research group has obtained promising results through several preliminary experiments. Regarding the model architecture, we are considering an approach that adds a neural network to evaluate and feedback on the specification accuracy of graph features generated by GraphTune [58]. By using this additional neural network to provide feedback, an improvement in specification accuracy is expected. On the other hand, from the perspective of preprocessing input data, it has been found that adding randomness to walks on graphs can improve the specification accuracy of generated graph features. In GraphGen and GraphTune, walks on graphs are converted into sequences of edges ordered by depth-first search.

Tables 2 and 3 show the results of experiments on graph generation using GraphTune, where the algorithm for converting graphs into sequences is replaced with 2nd-order random walk. The 2nd-order random walk is a graph sampling method initially proposed in node2vec [59], which is a representative model for node embeddings (not for graph generation). It is a random walk with a bias in the transition probability, and it effectively captures the graph structure. Let us consider a random walk currently at node v, assuming that it has traversed the edge (t, v) and is about to move to another node (Fig. 7). The bias $\alpha(t, x)$ of the transition probability from node v to another node x is defined as follows:

Table 2 The specification accuracy of graph feature when adding randomness to the sequence conversion (number of training data: 2000, parameters: $(p, q) = (1.0, 0.5)$).

Specified value	DFS	2nd-order RW
Averate shortest path length = 3	1.414	1.945
Averate shortest path length = 4	2.069	2.353
Averate shortest path length = 5	3.058	1.993

Table 3 The specification accuracy of graph feature when adding randomness to the sequence conversion (number of training data: 200, parameters: $(p, q) = (0.25, 4.0)$).

Specified value	DFS	2nd-order RW
Averate shortest path length = 3	1.088	0.648
Averate shortest path length = 4	1.580	0.831
Averate shortest path length = 5	2.835	1.403

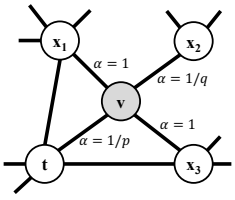


Fig. 7 The bias α of the transition probability of a 2nd-order random walk from node t through node v to node x.

$$\alpha(t,x) = \begin{cases} 1/p & \text{if } d_{tx} = 0, \\ 1 & \text{if } d_{tx} = 1, \\ 1/q & \text{if } d_{tx} = 2, \end{cases} \tag{1}$$

where d_{tx} represents the shortest path length between nodes t and x, and p and q are control parameters of the bias. The numbers in the tables represent the Root Mean Squared Error (RMSE), where a smaller value indicates higher accuracy. The definition of RMSE is as follows:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} (f_G - f_G^*)^2}, \tag{2}$$

where $\mathcal{G}$ is the set of generated graphs, and f_G and f_G^* are the feature values of the generated graph G and the specified value, respectively. It can be seen that using 2nd-order random walk improves the accuracy of generated graph features compared to the conventional DFS-based conversion when trained on a small dataset with 200 graphs. This result is thought to be due to the increased diversity of the latent space by adding randomness, especially in situations where the number of data is limited, and is a very interesting finding. For the result of the 2nd-order random walk, we varied p and q within the range of $[0.25, 4.0]$ and displayed the results with the smallest RMSE in Tables 2 and 3.

Developing methods to extrapolate graphs is another important research direction. The range of features that can be specified in GraphTune is limited to the range of features in the graphs included in the dataset. Specifying features that fall outside the range of features in the graphs in the dataset is currently challenging. One approach to address this could be to further utilize graphs generated by GraphTune as additional training data, thereby expanding the range of features that can be specified.

Besides the tunability of features, applicability to large-scale graphs is also crucial for practical applications. In the experiments using GraphTune in Section 4.2, the number of nodes of the graphs used in the training was limited to 50. The other conditional generation models for graphs, such as CondGen [17] and DeepGMG [14], also performed experiments on small-scale graphs with a limited number of nodes (e.g., order of 200 or less). However, in practical applications including social networks and communication networks, the number of nodes in the graph is often in the order of 1000 or more. Therefore, it is necessary to develop techniques that can tune features for large-scale graphs. Though current techniques for tuning features are not applicable to large-scale graphs, a part of the unconditional generation models, such as NetGAN [45], can generate large-scale graphs with over 10,000 nodes. We consider that techniques for tuning features can also have the potential to generate similar scale graphs, so it is expected to develop techniques for applying large-scale graphs by learning from these models.

5.2 Extension of Features Specification Techniques to Other Fields

If conditional generation techniques, especially those that allow for specifying multiple features continuously, mature, a new research field that approaches graph analysis from generative techniques in addition to traditional graph theory will emerge. For example, physics has evolved through mutual feedback between theoretical and experimental physics, with theoretical hypotheses being tested in experimental physics and experimental results being explained in theoretical physics. Similar to the relation between theoretical and experimental physics, graph generation techniques are expected to become important tools for proving hypotheses and constructing new theories in graph theory. It will be possible to approach various problems in traditional graph theory, such as deriving and verifying approximations for various graph features and elucidating relationships between features, from a generative perspective.

In particular, techniques that allow for tuning arbitrary features provide a new perspective on optimization problems in traditional graph theory. For example, it becomes possible to approach problems such as deriving and verifying approximations for the relationship between average degree and modularity, which was difficult in traditional graph theory, from both theoretical and generative aspects. By generating graphs that have the highest modularity among graphs with an average degree of 4, for instance, it becomes possible to derive and verify approximations for the relationship between average degree and modularity. This ability to fix arbitrary conditions enables a new approach to problems that were previously challenging in traditional graph theory.

6. Conclusions

In this paper, we comprehensively reviewed graph generation techniques that can be applied to various problems, covering both classical random-based generative models and the recently popular deep graph generators. We particularly focused on graph features and provided a detailed explanation of techniques for tuning graph features in various graph generation models. For future prospects, I introduced several promising approaches to improve the accuracy when tuning graph features in graph generation techniques, such as refining the model architecture and improving preprocessing of input data. Additionally, I mentioned the potential of graph generation techniques that can tune features to provide experimental feedback to classical graph theory, contributing to the construction of new theories and the validation of existing theories.

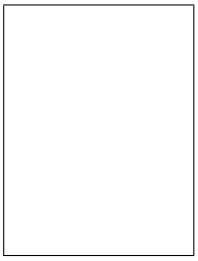
Acknowledgment

This work was partly supported by JSPS KAKENHI Grant Number JP23H03379.

References

[1] Q. Yan, L. Wu, and L. Zheng, "Social Network Based Microblog User Behavior Analysis," Physica A: Statistical Mechanics and its Applications, vol.392, no.7, 2013.
[2] M.W. Ahn and W.S. Jung, "Accuracy Test for Link Prediction in Terms of Similarity Index: The Case of WS and BA Models," Physica A: Statistical Mechanics and its Applications, vol.429, 2015.
[3] S. Fortunato and D. Hric, "Community Detection in Networks: A User Guide," Physics Reports, vol.659, 2016.
[4] Y. Jiang, C. Huang, and L. Huang, "Adaptive Graph Contrastive Learning for Recommendation," Proc. the 29th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2023), 2023.
[5] 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.
[6] 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, 2020.
[7] M. Brockschmidt, M. Allamanis, A.L. Gaunt, and O. Polozov, "Generative Code Modeling with Graphs," Proc. the 7th International Conference on Learning Representations (ICLR 2019), 2019.
[8] R. Pastor-Satorras and A. Vespignani, "Epidemic Dynamics and Endemic States in Complex Networks," Physical Review E, vol.63, no.6, 2001.
[9] P. Erdös and A. Rényi, "On Random Graphs I," Publicationes Mathematicae, vol.6, no.26, 1959.
[10] D.J. Watts and S.H. Strogatz, "Collective Dynamics of 'Small-World' Networks," Nature, vol.393, no.6684, 1998.
[11] R. Albert and A.L. Barabási, "Statistical Mechanics of Complex Networks," Reviews of Modern Physics, vol.74, no.1, 2002.
[12] X. Guo and L. Zhao, "A Systematic Survey on Deep Generative Models for Graph Generation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.45, no.5, 2023.
[13] F. Faez, Y. Ommi, M.S. Baghshah, and H.R. Rabiee, "Deep Graph Generators: A Survey," IEEE Access, vol.9, 2021.
[14] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, "Learning Deep Generative Models of Graphs," Proc. the 6th International Conference on Learning Representations (ICLR 2018) Workshop, 2018.
[15] 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.
[16] A.M. Tseng, N. Diamant, T. Biancalani, and G. Scalia, "GraphGUIDE: Interpretable and Controllable Conditional Graph Generation with Discrete Bernoulli Diffusion," arXiv, 2023.
[17] C. Yang, P. Zhuang, W. Shi, A. Luu, and P. Li, "Conditional Structure Generation through Graph Variational Generative Adversarial Nets," Proc. the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), 2019.
[18] S. Nakazawa, Y. Sato, K. Nakagawa, S. Tsugawa, and K. Watabe, "A Tunable Model for Graph Generation Using LSTM and Conditional VAE," Proc. the 41st IEEE International Conference on Distributed Computing Systems (ICDCS 2021) Poster Track, 2021.
[19] 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, 2023.
[20] 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, 2007.
[21] 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.
[22] A.A. Hagberg, D.A. Schult, and P.J. Swart, "Exploring Network Structure, Dynamics, and Function Using NetworkX," Proc. the 7th Python in Science Conference (SciPy 2008), 2008.
[23] "NetworkX - Network Analysis in Python." https://networkx.org/.
[24] "Igraph - The Network Analysis Package." https://igraph.org/.
[25] P.W. Holland, K.B. Laskey, and S. Leinhardt, "Stochastic Blockmodels: First Steps," Social Networks, vol.5, no.2, 1983.
[26] M. Biskup, "On the Scaling of the Chemical Distance in Long-Range Percolation Models," The Annals of Probability, vol.32, no.4, 2004.
[27] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A Recursive Model for Graph Mining," Proc. the 2004 SIAM International Conference on Data Mining (SDM 2004), 2004.
[28] L. Akoglu and C. Faloutsos, "RTG: A Recursive Realistic Graph Generator Using Random Typing," Lecture Notes in Artificial Intelligence, vol.5781, 2009.
[29] 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.
[30] S. Edunov, D. Logothetis, C. Wang, A. Ching, and M. Kabiljo, "Darwini: Generating Realistic Large-Scale Social Graphs," arXiv, 2016.
[31] D.P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Proc. the 2nd International Conference on Learning Representations (ICLR 2014), 2014.
[32] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," Proc. the 27th International Conference on Neural Information Processing Systems (NIPS 2014), 2014.
[33] N.T. Huang and S. Villar, "A Short Tutorial on The Weisfeiler-Lehman Test and Its Variants," Proc. 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2021), 2021.
[34] M.J. Kusner, B. Paige, and J.M. Hernández-Lobato, "Grammar Variational Autoencoder," Proc. the 34th International Conference on Machine Learning (ICML 2017), 2017.
[35] Q. Liu, M. Allamanis, M. Brockschmidt, and A. Gaunt, "Constrained Graph Variational Autoencoders for Molecule Design," Proc. the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), 2018.
[36] W. Jin, R. Barzilay, and T. Jaakkola, "Junction Tree Variational Autoencoder for Molecular Graph Generation," Proc. the 35th Inter-

national Conference on Machine Learning (ICML 2018), 2018.
[37] N.D. Cao and T. Kipf, "MolGAN: An Implicit Generative Model for Small Molecular Graphs," Proc. the 35th International Conference on Machine Learning (ICML 2018) Workshop, 2018.
[38] M. Popova, M. Shvets, J. Oliva, and O. Isayev, "MolecularRNN: Generating Realistic Molecular Graphs with Optimized Properties," arXiv, 2019.
[39] C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang, and J. Tang, "GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation," Proc. the 8th International Conference on Learning Representations (ICLR 2020), 2020.
[40] W. Jin, R. Barzilay, and T. Jaakkola, "Hierarchical Generation of Molecular Graphs using Structural Motifs," Proc. the 37th International Conference on Machine Learning (ICML 2020), 2020.
[41] T. Ma, J. Chen, and C. Xiao, "Constrained generation of semantically valid graphs via regularizing variational autoencoders," Proc. the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), 2018.
[42] M. Simonovsky and N. Komodakis, "GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders," Proc. the 27th International Conference on Artificial Neural Networks (ICANN 2018), 2018.
[43] S. Fan and B. Huang, "Conditional labeled graph generation with GANs," Proc. the 7th International Conference on Learning Representations (ICLR 2019) Workshop, 2019.
[44] T.N. Kipf and M. Welling, "Variational Graph Auto-Encoders," Proc. the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016) Workshop, 2016.
[45] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating Graphs via Random Walks," Proc. the 35th International Conference on Machine Learning (ICML 2018), 2018.
[46] A. Grover, A. Zweig, and S. Ermon, "Graphite: Iterative Generative Modeling of Graphs," Proc. the 36th International Conference on Machine Learning (ICML 2019), 2019.
[47] A. Gamage, E. Chien, J. Peng, and O. Milenkovic, "Multi-MotifGAN (MMGAN): Motif-Targeted Graph Generation and Prediction," Proc. 2020 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2020), 2020.
[48] J. You, R. Ying, X. Ren, W.L. Hamilton, and J. Leskovec, "GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models," Proc. the 35th International Conference on Machine Learning (ICML 2018), 2018.
[49] S.Y. Su and H. Hajimirsadeghi, "Graph Generation with Variational Recurrent Neural Network," Proc. the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019) Workshop, 2019.
[50] M. Zhang, S. Jiang, Z. Cui, R. Garnett, and Y. Chen, "D-VAE: A Variational Autoencoder for Directed Acyclic Graphs," Proc. the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), 2019.
[51] D. Bacciu, A. Micheli, and M. Podda, "Graph generation by sequential edge prediction," Proc. the 27th European Symposium on Artificial Neural Networks (ESANN 2019), 2019.
[52] D. Bacciu, A. Micheli, and M. Podda, "Edge-Based Sequential Graph Generation with Recurrent Neural Networks," Neurocomputing, vol.416, no.27, 2020.
[53] N. Goyal, H.V. Jain, and S. Ranu, "GraphGen: A Scalable Approach to Domain-agnostic Labeled Graph Generation," Proc. the Web Conference 2020 (WWW 2020), 2020.
[54] X. Yan and J. Han, "gSpan: graph-based substructure pattern mining," Proc. 2002 IEEE International Conference on Data Mining (ICDM 2002), 2002.
[55] S. Yang, J. Liu, K. Wu, and M. Li, "Learn to Generate Time Series Conditioned Graphs with Generative Adversarial Nets," arXiv, 2020.
[56] R. Liao, Y. Li, Y. Song, S. Wang, W. Hamilton, D.K. Duvenaud, R. Urtasun, and R. Zemel, "Efficient Graph Generation with Graph Recurrent Attention Networks," Proc. the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), 2019.
[57] N. Stoehr, M. Brockschmidt, J. Stuehmer, and E. Yilmaz, "Disentangling Interpretable Generative Parameters of Random and Real-World Graphs," Proc. the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019) Workshop, 2019.
[58] T. Yokoyama, Y. Sato, S. Tsugawa, and K. Watabe, "An Accurate Graph Generative Model with Tunable Features," Proc. the 32nd International Conference on Computer Communications and Networks (ICCCN 2023) Poster Session, 2023.
[59] A. Grover and J. Leskovec, "node2vec: Scalable Feature Learning for Networks," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016), 2016.



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 was an Associate Professor of the Graduate School of Engineering, Nagaoka University of Technology, since November 2019. He has been an Associate Professor of the Graduate School of Science and Engineering, Saitama University, since April 2024. He is a member of the IEEE and the IEICE.